ਕੰਪਿਊਟਰ ਐਪਲੀਕੇਸ਼ਨ

(ਗਿਆਰਵੀਂ ਸ਼੍ਰੇਣੀ ਲਈ)

ਪੰਜਾਬ ਸਕੂਲ ਸਿੱਖਿਆ ਬੋਰਡ

ਸਾਹਿਬਜ਼ਾਦਾ ਅਜੀਤ ਸਿੰਘ ਨਗਰ

© ਪੰਜਾਬ ਸਰਕਾਰ

ਐਡੀਸ਼ਨ 2025-26........................ .... 629 ਕਾਪੀਆਂ

All rights, including those of translation, reproduction
and annotation etc., are reserved by the
Punjab Government

ਚੇਤਾਵਨੀ

1. ਕੋਈ ਵੀ ਏਜੰਸੀ-ਹੋਲਡਰ ਵਾਧੂ ਪੈਸੇ ਵਸੂਲਣ ਦੇ ਮੰਤਵ ਨਾਲ ਪਾਠ-ਪੁਸਤਕਾਂ ਤੇ ਜਿਲਦ-ਸਾਜੀ ਨਹੀਂ ਕਰ ਸਕਦਾ।(ਏਜੰਸੀ-ਹੋਲਡਰਾਂ ਨਾਲ ਹੋਏ ਸਮਝੌਤੇ ਦੀ ਧਾਰਾ ਨੰ. 7 ਅਨੁਸਾਰ)
2. ਪੰਜਾਬ ਸਕੂਲ ਸਿੱਖਿਆ ਬੋਰਡ ਦੁਆਰਾ ਛਪਵਾਈਆਂ ਅਤੇ ਪ੍ਰਕਾਸ਼ਿਤ ਪਾਠ-ਪੁਸਤਕਾਂ ਦੇ ਜਾਅਲੀ ਨਕਲੀ ਪ੍ਰਕਾਸ਼ਨਾਂ (ਪਾਠ-ਪੁਸਤਕਾਂ) ਦੀ ਛਪਾਈ, ਪ੍ਰਕਾਸ਼ਨ, ਸਟਾਕ ਕਰਨਾ, ਜਮ੍ਹਾਂ-ਖੋਰੀ ਜਾਂ ਵਿਕਰੀ ਆਦਿ ਕਰਨਾ ਭਾਰਤੀ ਦੰਡ ਪ੍ਰਣਾਲੀ ਦੇ ਅੰਤਰਗਤ ਫ਼ੌਜਦਾਰੀ ਜੁਰਮ ਹੈ।
   (ਪੰਜਾਬ ਸਕੂਲ ਸਿੱਖਿਆ ਬੋਰਡ ਦੀਆਂ ਪਾਠ-ਪੁਸਤਕਾਂ ਬੋਰਡ ਦੇ 'ਵਾਟਰ ਮਾਰਕ' ਵਾਲੇ ਕਾਗਜ਼ ਉੱਪਰ ਹੀ ਛਪਵਾਈਆਂ ਜਾਂਦੀਆਂ ਹਨ।

ਇਹ ਪੁਸਤਕ ਵਿਕਰੀ ਲਈ ਨਹੀਂ ਹੈ।

ਸਕੱਤਰ, ਪੰਜਾਬ ਸਕੂਲ ਸਿੱਖਿਆ ਬੋਰਡ, ਵਿੱਦਿਆ ਭਵਨ, ਫੇਜ਼-8 ਸਾਹਿਬਜ਼ਾਦਾ ਅਜੀਤ ਸਿੰਘ ਨਗਰ-160002 ਰਾਹੀਂ ਪ੍ਰਕਾਸ਼ਿਤ ਅਤੇ ਮੈਸ: ਰਵਿੰਦਰਾ ਪੇਪਰ ਮਾਰਟ, ਜਲੰਧਰ ਦੁਆਰਾ ਛਾਪੀ ਗਈ।

ਸੂਚਨਾ ਤਕਨਾਲੋਜੀ ਦੇ ਮੁਢਲੇ ਸਿਧਾਂਤ
(Fundamentals of Information Technology)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

1.1 ਕੰਪਿਊਟਰ ਸਿਸਟਮ
1.2 ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਹਿੱਸੇ
1.3 ਸਾਫਟਵੇਅਰ ਅਤੇ ਇਸ ਦੀਆਂ ਕਿਸਮਾਂ
1.4 IT ਵਿਚ ਰੁਝਾਨ

ਜਾਣ-ਪਛਾਣ (Introduction)

ਜਦੋਂ ਅਸੀਂ IT ਬਾਰੇ ਗੱਲ ਕਰਦੇ ਹਾਂ, ਸਭ ਤੋਂ ਪਹਿਲਾਂ ਜੋ ਸਾਡੇ ਦਿਮਾਗ ਵਿੱਚ ਆਉਂਦਾ ਹੈ ਉਹ ਹਨ: ਕੰਪਿਊਟਰ, ਨੈਟਵਰਕ ਅਤੇ ਇੰਟਰਨੈੱਟ। IT ਜਾਂ ਇਨਫੋਰਮੇਸ਼ਨ ਟੈਕਨੋਲੋਜੀ ਇਲੈਕਟ੍ਰੋਨਿਕ ਉਪਕਰਨਾਂ ਦਾ ਅਧਿਐਨ ਜਾਂ ਵਰਤੋਂ ਹੈ, ਜਿਵੇਂ ਕਿ ਕੰਪਿਊਟਰ, ਇਲੈਕਟ੍ਰੋਨਿਕ ਡਾਟਾ ਜਾਂ ਜਾਣਕਾਰੀ ਨੂੰ ਇੱਕਠਾ ਕਰਨ, ਸਟੋਰ ਕਰਨ, ਪ੍ਰੋਸੈਸ ਕਰਨ ਅਤੇ ਇਲੈਕਟ੍ਰੋਨਿਕ ਡਾਟਾ ਜਾਂ ਸੂਚਨਾ ਨੂੰ ਅਦਾਨ-ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

1.1 ਕੰਪਿਊਟਰ ਸਿਸਟਮ (Computer System)

IT ਦਾ ਸਭ ਤੋਂ ਮਹਤਵਪੂਰਨ ਹਿੱਸਾ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਹੈ। ਕੰਪਿਊਟਰ ਇਕ ਇਲੈਕਟ੍ਰੋਨਿਕ ਡਿਵਾਇਸ ਹੈ ਜੋ ਕਿ ਡਾਟਾ ਨੂੰ ਲੈਂਦਾ ਹੈ, ਸਟੋਰ ਕਰਦਾ ਹੈ ਅਤੇ ਪ੍ਰੋਸੈਸ ਕਰਕੇ ਉਪਯੋਗੀ ਜਾਣਕਾਰੀ ਵਿਚ ਬਦਲਦਾ ਹੈ।

ਚਿੱਤਰ 1.1 : ਕੰਪਿਊਟਰ ਸਿਸਟਮ

1.1.1 ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਮੁੱਖ ਕਾਰਜ : (Main functions of a Computer System) :

ਚਿੱਤਰ 1.2 : ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੀਆਂ ਚਾਰ ਮੁੱਖ ਪ੍ਰਕਿਰਿਆਵਾਂ

1. ਇਨਪੁੱਟ : ਡਾਟਾ ਅਤੇ ਕੱਚੇ ਤੱਥ, ਜਿਸਨੂੰ ਅਸੀਂ ਸਿਸਟਮ ਦੁਆਰਾ ਪ੍ਰੋਸੈਸ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ, ਅਸੀਂ ਕੰਪਿਊਟਰ ਵਿੱਚ ਦਾਖਲ ਕਰਦੇ ਹਾਂ, ਨੂੰ ਇਨਪੁੱਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਤੋਂ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
2. ਪ੍ਰੋਸੈਸਿੰਗ : ਇਹ ਜਾਣਕਾਰੀ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਡਾਟਾ ਤੇ ਪ੍ਰਕਿਰਿਆ ਕਰਦਾ ਹੈ।
3. ਸਟੋਰ : ਇਹ ਡਾਟਾ ਅਤੇ ਸੂਚਨਾ ਨੂੰ ਸਟੋਰ ਕਰਦਾ ਹੈ।
4. ਆਉਟਪੁੱਟ : ਪ੍ਰੋਸੈਸਡ ਡਾਟਾ ਸਾਨੂੰ ਰਿਜਲਟ ਵਜੋਂ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਆਉਟਪੁੱਟ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।

1.1.2 ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (Characteristics of a Computer System) : ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾ ਹੇਠਾਂ ਦਿੱਤੀਆਂ ਗਈਆਂ ਹਨ:

1. **ਸਪੀਡ (Speed)** : ਇੱਕ ਕੰਪਿਊਟਰ ਗਣਿਤ ਦੀਆਂ ਗਣਨਾਵਾਂ ਕਰਦੇ ਸਮੇਂ ਆਮ ਮਨੁੱਖਾਂ ਦੇ ਮੁਕਾਬਲੇ ਬਹੁਤ ਤੇਜ਼ ਰਫ਼ਤਾਰ ਨਾਲ ਕੰਮ ਕਰਦਾ ਹੈ। ਕੰਪਿਊਟਰ ਇੱਕ ਸਕਿੰਟ ਦੇ ਇੱਕ ਹਿੱਸੇ ਵਿਚ ਇੱਕ ਹਦਾਇਤ ਦੀ ਪ੍ਰਕਿਰਿਆ ਕਰ ਸਕਦਾ ਹੈ। ਕੰਪਿਊਟਰ ਦੀ ਗਤੀ ਮਾਈਕ੍ਰੋਸਕਿੰਟ,ਮਿਲੀਸਕਿੰਟ, ਨੈਨੋ ਸਕਿੰਟ ਅਤੇ ਇੱਥੋਂ ਤੱਕ ਕਿ ਪਿਕੋਸਕਿੰਡ ਵਿੱਚ ਵੀ ਮਾਪੀ ਜਾਂਦੀ ਹੈ।
2. **ਸ਼ੁੱਧਤਾ (Accuracy)** : ਕੰਪਿਊਟਰ ਹਰੇਕ ਗਣਨਾ ਨੂੰ ਬਹੁਤ ਉੱਚੀ ਅਤੇ ਇੱਕੋ ਜਿਹੀ ਸ਼ੁੱਧਤਾ ਨਾਲ ਕਰਦਾ ਹੈ। ਇਸ ਨੂੰ ਦਿੱਤੇ ਗਏ ਅਸੰਗਤ ਜਾਂ ਗਲਤ ਡਾਟਾ ਦੇ ਕਾਰਨ ਗਲਤੀਆਂ ਹੋ ਸਕਦੀਆਂ ਹਨ।
3. **ਮਿਹਨਤੀ (Diligence)** : ਇੱਕ ਕੰਪਿਊਟਰ ਲੱਖਾਂ ਕੰਮ ਜਾਂ ਗਣਨਾਵਾਂ ਇੱਕੋਸਾਰਤਾ ਅਤੇ ਸ਼ੁੱਧਤਾ ਨਾਲ ਕਰ ਸਕਦਾ ਹੈ। ਇਹ ਬੋਰੀਅਤ, ਥਕਾਵਟ ਅਤੇ ਇਕਾਗਰਤਾ ਵਿੱਚ ਕਮੀ ਆਦਿ ਭਾਵਨਾਵਾਂ ਤੋਂ ਮੁਕਤ ਹੈ ਅਤੇ ਇਸ ਲਈ ਬਿਨਾਂ ਕਿਸੇ ਗਲਤੀ ਦੇ ਘੰਟਿਆਂ ਬੱਧੀ ਇੱਕਠੇ ਕੰਮ ਕਰ ਸਕਦਾ ਹੈ।
4. **ਬਹੁਪੱਖੀਤਾ (Versatility)** : ਬਹੁਪੱਖੀਤਾ ਕੰਪਿਊਟਰ ਬਾਰੇ ਸਭ ਤੋਂ ਸ਼ਾਨਦਾਰ ਚੀਜਾਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਕੰਪਿਊਟਰ ਦੀ ਇੱਕੋ ਜਿਹੀ ਸ਼ੁੱਧਤਾ ਅਤੇ ਕੁਸ਼ਲਤਾ ਨਾਲ ਵੱਖ-ਵੱਖ ਤਰ੍ਹਾਂ ਦੇ ਕੰਮ ਕਰਨ ਦੀ ਸਮੱਰਥਾ ਹੈ। ਇਕ ਪਲ ਵਿੱਚ ਇਹ ਕੋਈ ਵੀ ਇੱਕ ਆਪਰੇਸ਼ਨ ਕਰ ਸਕਦਾ ਹੈ ਅਤੇ ਅਗਲੇ ਹੀ ਪਲ ਇਹ ਕੋਈ ਹੋਰ ਆਪਰੇਸ਼ਨ ਕਰ ਸਕਦਾ ਹੈ। ਇੱਕ ਕੰਪਿਊਟਰ ਦਿੱਤੀਆਂ ਹਦਾਇਤਾਂ ਅਨੁਸਾਰ ਲਗਭਗ ਕਿਸੇ ਵੀ ਕੰਮ ਨੂੰ ਕਰਨ ਦੇ ਸਮੱਰਥ ਹੁੰਦਾ ਹੈ।
5. **ਭਰੋਸੇਯੋਗਤਾ (Reliability)** : ਇਹ 100% ਸ਼ੁੱਧਤਾ ਅਤੇ ਭਰੋਸੇਯੋਗਤਾ ਦੇ ਨਾਲ ਸਾਰੇ ਕੰਮ ਕਰਦਾ ਹੈ। ਇੱਕ ਕੰਪਿਊਟਰ ਇੱਕ ਤਰ੍ਹਾਂ ਦੇ ਡਾਟਾ ਲਈ ਇੱਕੋ ਵਰਗਾ ਨਤੀਜਾ ਦਿੰਦਾ ਹੈ, ਜਿਵੇਂ ਕਿ ਜੇਕਰ ਅਸੀਂ ਇੱਨਪੁੱਟ ਦਾ ਇੱਕੋ ਸੈੱਟ ਕਈ ਵਾਰ ਦਿੰਦੇ ਹਾਂ, ਤਾਂ ਸਾਨੂੰ ਉਹੀ ਨਤੀਜਾ ਬਾਰ-ਬਾਰ ਮਿਲੇਗਾ। ਭਰੋਸੇਯੋਗਤਾ ਕੇਵਲ ਮਨੁੱਖ ਦੁਆਰਾ ਕੀਤੀਆਂ ਗਈਆਂ ਗਲਤੀਆਂ ਦੁਆਰਾ ਪ੍ਰਭਾਵਿਤ ਹੋ ਸਕਦੀ ਹੈ।
6. **ਆਟੋਮੇਸ਼ਨ (Automation)** : ਕੰਪਿਊਟਰ ਸਾਰੇ ਕੰਮ ਆਪਣੇ ਆਪ ਕਰਦਾ ਹੈ ਭਾਵ ਇਹ ਮਨੁੱਖੀ ਦਖਲ ਤੋਂ ਬਿਨਾਂ ਕੰਮ ਕਰ ਸਕਦਾ ਹੈ।
7. **ਮੈਮਰੀ (Memory)** : ਇੱਕ ਕੰਪਿਊਟਰ ਵਿੱਚ ਬਿਲਟ-ਇਨ ਮੈਮਰੀ ਹੁੰਦੀ ਹੈ ਜਿਸਨੂੰ ਪ੍ਰਾਈਮਰੀ ਮੈਮਰੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਸੈਕੰਡਰੀ ਸਟੋਰੇਜ ਡਿਵਾਇਸ ਜਿਵੇਂ ਕਿ HDD, CDs, ਪੈੱਨ ਡਰਾਈਵ ਆਦਿ ਵਿੱਚ ਇਹ ਵੱਡੀ ਮਾਤਰਾ ਅਤੇ ਬਹੁਤ ਉੱਚ ਕੁਸ਼ਲਤਾ ਨਾਲ ਡਾਟਾ ਅਤੇ ਹਦਾਇਤਾਂ ਨੂੰ ਸਟੋਰ ਕਰ ਸਕਦਾ ਹੈ।

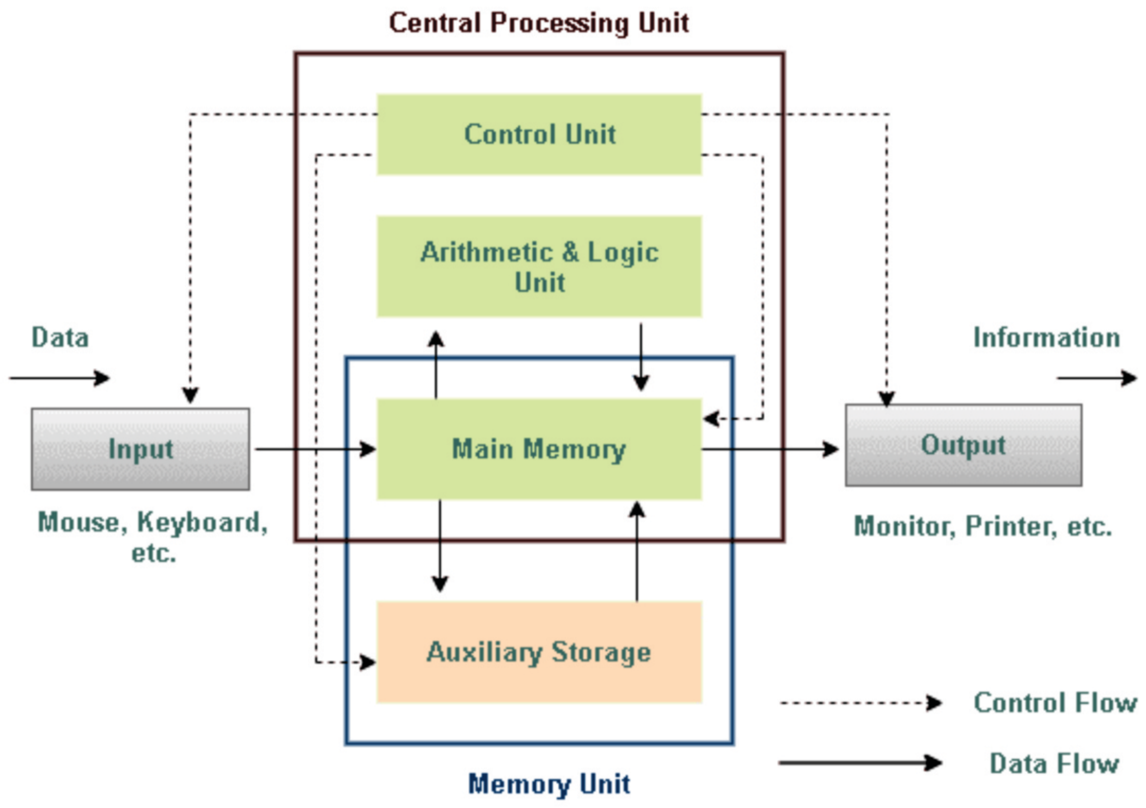
1.1.3 ਕੰਪਿਊਟਰ ਦੀਆਂ ਸੀਮਾਵਾਂ ਜਾਂ ਕਮੀਆਂ (Limitations or Drawbacks of Computer) :

1. **No I.Q (I.Q ਨਹੀਂ ਹੈ):** ਕੰਪਿਊਟਰ ਕੋਈ ਜਾਦੂਈ ਡਿਵਾਇਸ ਨਹੀਂ ਹੈ। ਇਸ ਵਿੱਚ ਕੋਈ ਅਕਲ ਜਾਂ ਸੋਚਣ ਸ਼ਕਤੀ ਨਹੀਂ ਹੁੰਦੀ। ਇਹ ਉਹ ਕੰਮ ਕਰ ਸਕਦਾ ਹੈ ਜੋ ਕਿ ਇਕ ਮਨੁੱਖ ਕਰ ਸਕਦਾ ਹੈ ਪਰ ਮੁੱਖ ਅੰਤਰ ਇਹ ਹੈ ਕਿ ਕੰਪਿਊਟਰ ਬਹੁਤ ਤੇਜ਼ ਗਤੀ ਅਤੇ ਭਰੋਸੇਯੋਗ ਸ਼ੁੱਧਤਾ ਨਾਲ ਕੰਮ ਕਰਦਾ ਹੈ।
2. **No Feeling (ਕੋਈ ਭਾਵਨਾ ਨਹੀਂ ਹੈ):** ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਸਾਰੇ ਜਾਣਦੇ ਹਾਂ ਕਿ ਕੰਪਿਊਟਰ ਸਿਰਫ ਇੱਕ ਮਸ਼ੀਨ ਹੈ, ਇਸ ਵਿੱਚ ਮਨੁੱਖਾਂ ਵਰਗੀਆਂ ਕੋਈ ਭਾਵਨਾਵਾਂ ਨਹੀਂ ਹੁੰਦੀਆਂ। ਇਸ ਵਿੱਚ ਮਨੁੱਖ ਦੀ ਤਰ੍ਹਾਂ ਸੋਚਣ ਲਈ ਕੋਈ ਦਿਮਾਗ ਨਹੀਂ ਹੁੰਦਾ। ਮਨੁੱਖ ਨੇ ਆਪਣੇ ਦਿਮਾਗ ਨਾਲ ਇਸ ਮਸ਼ੀਨ ਅਤੇ ਮੈਮਰੀ ਨੂੰ ਬਣਾਇਆ ਹੈ ਪਰ ਮਨੁੱਖ ਦਿਲ ਨਹੀਂ ਬਣਾ ਸਕਿਆ।
3. ਸਿਸਟਮ ਖਰਾਬ ਹੋਣ ਕਾਰਨ ਸਮੱਸਿਆਵਾਂ ਪੈਦਾ ਹੋ ਸਕਦੀਆਂ ਹਨ।

1.2 ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਹਿੱਸੇ (Components of Computer System) :

ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਇੱਕ ਇੱਕਲਾ ਡਿਵਾਇਸ ਨਹੀਂ ਹੈ। ਆਈਟਮਾਂ (Items) ਜੋ ਇੱਕ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਬਣਾਉਂਦੀਆਂ ਹਨ ਉਹਨਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਕੰਪੋਨੈਂਟਸ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਕਈ ਕੰਪੋਨੈਂਟਸ ਤੋਂ ਮਿਲ ਕੇ ਬਣਦਾ ਹੈ, ਜਿਹਨਾਂ ਦੇ ਕੁਝ ਖਾਸ ਕੰਮ ਅਤੇ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹੁੰਦੀਆਂ ਹਨ, ਤਾਂ ਕਿ ਉਹ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਸਪੋਰਟ ਕਰ ਸਕਣ।

ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਤਿੰਨ ਮੁੱਖ ਭਾਗ ਹਨ: ਇਨਪੁੱਟ ਯੂਨਿਟ, ਆਉਟਪੁੱਟ ਯੂਨਿਟ, ਅਤੇ ਸੀ.ਪੀ.ਯੂ. (CPU -ਸੈਂਟਰਲ ਪ੍ਰੋਸੈਸਿੰਗ ਯੂਨਿਟ)।



ਚਿੱਤਰ 1.3 : ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਹਿੱਸੇ

1.2.1 ਇਨਪੁੱਟ ਯੂਨਿਟ (Input Unit)–ਉਹ ਡਿਵਾਇਸ ਜੋ ਯੂਜ਼ਰ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਹਦਾਇਤਾਂ ਦੇਣ ਜਾਂ ਡਾਟਾ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦੇ ਹਨ, ਉਹਨਾਂ ਨੂੰ ਇਨਪੁੱਟ ਯੂਨਿਟ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਯੂਜ਼ਰ ਨੂੰ ਕੰਪਿਊਟਰ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ।

- ਆਮ ਤੌਰ ਤੇ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਇਨਪੁੱਟ ਡਿਵਾਇਸ ਹੇਠਾਂ ਦਿੱਤੇ ਗਏ ਹਨ:

1. **ਕੀ-ਬੋਰਡ (Keyboard) ਇਨਪੁੱਟ ਯੂਨਿਟ :** ਕੀ ਬੋਰਡ ਕੰਪਿਊਟਰ ਵਿੱਚ ਡਾਟਾ ਜਾਂ ਹਦਾਇਤਾਂ ਨੂੰ ਇਨਪੁੱਟ ਕਰਨ ਲਈ ਸਭ ਤੋਂ ਆਮ ਡਿਵਾਇਸਾਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ। ਕੀ ਬੋਰਡ ਦੀਆਂ ਕੀਜ਼ ਬਹੁਤ ਸਾਰੇ ਅਲਫਾ–ਨਿਊਮੈਰਿਕ ਅੱਖਰ, ਸਪੈਸ਼ਲ ਅੱਖਰ ਅਤੇ ਬਹੁਤ ਸਾਰੀਆਂ ਕਮਾਂਡਾਂ (ਹਦਾਇਤਾਂ), ਫੰਕਸ਼ਨ ਕੀਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਕੇ, ਵਰਤਣ ਦੀ ਆਗਿਆ ਦਿੰਦੀਆਂ ਹਨ। ਇਹ ਖਾਸ ਕੰਮਾਂ ਲਈ ਸ਼ਾਰਟਕੱਟ ਵੀ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਕਈ ਵਾਰ ਕੀ ਬੋਰਡ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਡਾਟਾ ਦਾਖਲ ਕਰਨਾ ਸੰਭਵ ਜਾਂ ਉਚਿਤ ਨਹੀਂ ਹੁੰਦਾ, ਅਤੇ ਇਸ ਲਈ ਹੋਰ ਡਿਵਾਇਸ ਵੀ ਉਪਲਬਧ ਹਨ।

ਚਿੱਤਰ 1.4 : ਕੀ–ਬੋਰਡ

2. **ਮਾਊਸ :** ਮਾਊਸ ਇੱਕ ਪੁਆਇੰਟਿੰਗ ਅਤੇ ਕਲਿੱਕ ਕਰਨ ਵਾਲਾ ਡਿਵਾਇਸ ਹੈ। ਇਹ ਯੂਜ਼ਰ ਨੂੰ ਸਕਰੀਨ ਤੇ ਕਿਸੇ ਖਾਸ ਪੁਆਇੰਟ ਤੇ ਕਰਸਰ ਰੱਖ ਕੇ ਕੰਪਿਊਟਰ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ। ਕੰਪਿਊਟਰ ਸਕਰੀਨ ਮਾਊਸ ਦੀ ਗਤੀ ਨੂੰ ਟ੍ਰੈਕ ਕਰਦੀ ਹੈ ਅਤੇ ਸਕਰੀਨ ਉੱਤੇ ਇਸ ਦੀ ਲੋਕੇਸ਼ਨ ਨੂੰ ਕਰਸਰ ਦੇ ਰੂਪ ਵਿੱਚ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਦੀ ਹੈ।

ਚਿੱਤਰ 1.5 : ਮਾਊਸ

3. **ਜੌਏਸਟਿਕ:** ਜੌਏਸਟਿਕ ਇੱਕ ਇਨਪੁੱਟ ਡਿਵਾਇਸ ਹੈ ਜਿਸਦੀ ਵਰਤੋਂ ਕਰਸਰ ਜਾਂ ਪੁਆਇੰਟਰ ਦੀ ਗਤੀ ਨੂੰ ਨਿਯੰਤਰਿਤ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਪੁਆਇੰਟਰ/ਕਰਸਰ ਦੀ ਸਪੀਡ (ਗਤੀ) ਨੂੰ ਜੌਏਸਟਿਕ ਉੱਤੇ ਇਕ ਲੀਵਰ ਦੁਆਰਾ ਨਿਯੰਤਰਿਤ (ਕਾਬੂ) ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਇਨਪੁੱਟ ਡਿਵਾਇਸ ਜ਼ਿਆਦਾਤਰ ਗੇਮਿੰਗ ਐਪਲੀਕੇਸ਼ਨਾਂ ਲਈ ਵਰਤੀ ਜਾਂਦੀ ਹੈ ਅਤੇ ਕਈ ਵਾਰ ਇਸਦੀ ਵਰਤੋਂ ਗ੍ਰਾਫਿਕਸ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਵੀ ਹੁੰਦੀ ਹੈ। ਇੱਕ ਜੌਏਸਟਿਕ ਹਿੱਲਣ ਵਿਚ ਅਸਮਰਥ ਲੋਕਾਂ ਲਈ ਇੱਕ ਇੰਨਪੁੱਟ ਡਿਵਾਇਸ ਵਜੋਂ ਮਦਦਗਾਰ ਹੋ ਸਕਦੀ ਹੈ।

ਚਿੱਤਰ 1.6 : ਜੌਇਸਟਿਕ

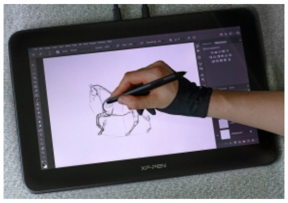
4. **ਸਕੈਨਰ:** ਸਕੈਨਕ ਇੱਕ ਡਾਕੂਮੈਂਟ (ਦਸਤਾਵੇਜ਼) ਜਾਂ ਫੋਟੋ ਨੂੰ ਇੱਕ ਡਿਜੀਟਲ ਫਾਈਲ ਵਿੱਚ ਬਦਲਦਾ ਹੈ, ਜਿਸਨੂੰ ਕੰਪਿਊਟਰ ਪੜ੍ਹ ਜਾਂ ਡਿਸਪਲੇ ਕਰ ਸਕਦਾ ਹੈ। ਇਸਦਾ ਮੁੱਖ ਕੰਮ ਸਕੈਨ ਕਰਨਾ ਜਾਂ ਡਾਕੂਮੈਂਟ ਦੀ ਫੋਟੋ ਖਿੱਚ ਕੇ, ਸੂਚਨਾ ਨੂੰ ਡਿਜਿਟਾਈਜ਼ ਕਰਨਾ ਅਤੇ ਉਸਨੂੰ ਕੰਪਿਊਟਰ ਦੀ ਸਕਰੀਨ ਤੇ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਨਾ ਹੈ। ਇੱਕ ਸਕੈਨਰ ਫਲੈਟ ਬੈਡ ਜਾਂ ਹੈਂਡ–ਹੈਲਡ ਸਕੈਨਰ ਹੋ ਸਕਦਾ ਹੈ।

ਚਿੱਤਰ 1.7 : ਸਕੈਨਰ

5. **ਟੱਚ-ਡਿਸਪਲੇ:** ਇੱਕ ਟੱਚ–ਡਿਸਪਲੇ ਸਕਰੀਨ ਇਕ ਡਿਸਪਲੇ ਡਿਵਾਇਸ ਹੈ ਜੋ ਕਿ ਯੂਜ਼ਰ ਨੂੰ ਉਹਨਾਂ ਦੀ ਉਂਗਲੀ ਜਾਂ ਸਟਾਈਲਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇੱਕ ਕੰਪਿਊਟਰ ਨਾਲ ਇੰਟਰੈਕਟ ਕਰਨ ਦੀ ਇਜ਼ਾਜ਼ਤ ਦਿੰਦਾ ਹੈ। ਇਹ ਇੱਕ GUI (ਗਰਾਫੀਕਲ ਯੂਜ਼ਰ ਇੰਟਰਫੇਸ) ਨੈਵੀਗੇਟ ਕਰਨ ਲਈ ਮਾਊਸ ਜਾਂ ਕੀ–ਬੋਰਡ ਦਾ ਇੱਕ ਉਪਯੋਗੀ ਵਿਕਲਪ ਹੈ। ਟੱਚ-ਸਕਰੀਨਾਂ ਦੀ ਵਰਤੋਂ ਵੱਖ-ਵੱਖ ਡਿਵਾਈਸਾਂ, ਜਿਵੇਂ ਕਿ ਕੰਪਿਊਟਰ ਅਤੇ ਲੈਪਟਾਪ ਡਿਸਪਲੇ, ਸਮਾਰਟ ਫੋਨ, ਟੈਬਲੇਟ ਆਦਿ ਵਿਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ਚਿੱਤਰ 1.8 : ਟੱਚ ਡਿਸਪਲੇ

6. **ਗ੍ਰਾਫਿਕਸ ਟੈਬਲੈਟ:** ਇਸ ਨੂੰ ਕੀ ਬੋਰਡ ਦੀ ਤਰ੍ਹਾਂ ਹੀ, ਆਈਕਨਾਂ (ਕੰਪਿਊਟਰ ਸਕਰੀਨ ਤੇ ਚਿੰਨ੍ਹ ਜਾਂ ਚਿੱਤਰ) ਤੇ ਟੈਪ ਕਰਕੇ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਡਾਟਾ ਦਾਖਲ ਕਰਨ ਵਿੱਚ ਮਦਦਗਾਰ ਹੁੰਦਾ ਹੈ, ਜਿਵੇਂ ਕਿ ਹੱਥ-ਲਿਖਤਾਂ ਜਾਂ ਵਸਤੂਆਂ ਦੀਆਂ ਡਰਾਇੰਗਾਂ ਜਿਨ੍ਹਾਂ ਨੂੰ ਹੋਰ ਤਰੀਕਿਆਂ ਨਾਲ ਅਸਾਨੀ ਨਾਲ ਕੈਪਚਰ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ।



ਚਿੱਤਰ 1.9 : ਟੈਬਲੇਟ

7. **ਮਾਈਕ੍ਰੋਫੋਨ:** ਮਾਈਕ੍ਰੋਫੋਨ ਯੂਜ਼ਰ ਤੋਂ ਹਦਾਇਤਾਂ ਨੂੰ ਜਾਂ ਹੋਰ ਅਵਾਜਾਂ ਜਿਨ੍ਹਾਂ ਨੂੰ ਯੂਜ਼ਰ ਇੱਕਠਾ ਕਰਨਾ ਚਾਹੁੰਦਾ ਹੈ, ਨੂੰ ਰਿਕਾਰਡ ਕਰਦਾ ਹੈ। ਉਹ ਯੂਜ਼ਰ ਜਿਹੜੇ ਕੀ-ਬੋਰਡ ਜਾਂ ਮਾਊਸ ਦੀ ਵਰਤੋਂ ਨਹੀਂ ਕਰ ਸਕਦੇ, ਉਹਨਾਂ ਲਈ ਡਾਟਾ ਇਨਪੁੱਟ ਕਰਨ ਦਾ ਇਹ ਇੱਕ ਬਹੁਤ ਉਪਯੋਗੀ ਡਿਵਾਇਸ ਹੈ।

ਚਿੱਤਰ 1.10 ਮਾਈਕਰੋਫੋਨ

8. **ਬਾਰਕੋਡ ਰੀਡਰ:** ਬਾਰਕੋਡ ਰੀਡਰ ਹੈਂਡ ਹੈਲਡ ਜਾਂ ਸਟੈਟਿਕ ਹੋ ਸਕਦਾ ਹੈ ਅਤੇ ਇਹ ਜਾਣਕਾਰੀ ਨੂੰ ਬਾਰਕੋਡ ਤੋਂ ਲਾਈਟ ਸੋਰਸ, ਸਕੈਨਰ ਅਤੇ ਡਿਕੋਡਰ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਕੈਪਚਰ ਕਰਦਾ ਹੈ। ਇਹ ਮੈਨੂਅਲ ਇਨਪੁੱਟ ਦੁਆਰਾ ਹੋਈਆਂ ਗਲਤੀਆ ਨੂੰ ਘਟਾਉਂਦੇ ਹਨ।

ਚਿੱਤਰ 1.11 ਬਾਰਕੋਡ ਰੀਡਰ

9. **ਬਾਇਓ-ਮੈਟ੍ਰਿਕ :** ਬਾਇਓਮੈਟ੍ਰਿਕ ਡਿਵਾਈਸ ਇੱਕ ਸੁਰੱਖਿਅਤ ਪਛਾਣ ਅਤੇ ਪ੍ਰਮਾਣਿਕਤਾ ਦੇਣ ਲਈ (Security identification and authentication) ਡਿਵਾਇਸ ਹੈ। ਅਜਿਹੇ ਡਿਵਾਇਸ ਸਰੀਰਕ ਜਾਂ ਵਿਵਹਾਰਕ (Physiological or behavioral) ਵਿਸ਼ੇਸ਼ਤਾ ਦੇ ਆਧਾਰ ਤੇ ਜੀਵਿਤ ਵਿਅਕਤੀ ਦੀ ਪੁਸ਼ਟੀ ਕਰਨ ਜਾਂ ਪਛਾਣ ਕਰਨ ਦੇ ਸਵੈਚਾਲਿਤ ਤਰੀਕਿਆਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ। ਇਹਨਾਂ ਵਿੱਚ ਫਿੰਗਰਪ੍ਰਿੰਟ, ਚਿਹਰੇ ਦੀਆਂ ਫੋਟੋਆਂ, ਆਇਰਿਸ ਅਤੇ ਆਵਾਜ਼ ਦੀ ਪਛਾਣ ਵਰਗੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਸ਼ਾਮਲ ਹਨ।

ਚਿੱਤਰ 1.12 ਬਾਇਓ ਮੈਟ੍ਰਿਕ

1.2.2 ਆਊਟਪੁੱਟ ਡਿਵਾਇਸ (Output devices) :

ਉਹ ਡਿਵਾਇਸ ਜੋ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਯੂਜ਼ਰ ਨੂੰ ਜਾਣਕਾਰੀ, ਡਾਟਾ ਜਾਂ ਹਦਾਇਤਾਂ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦੇ ਹਨ, ਉਹਨਾਂ ਨੂੰ ਆਊਟਪੁੱਟ ਯੂਨਿਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਕੰਪਿਊਟਰ ਨੂੰ ਯੂਜ਼ਰ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ।

- ਆਮ ਤੌਰ ਤੇ ਵਰਤੇ ਜਾਂਦੇ ਆਊਟਪੁੱਟ ਡਿਵਾਇਸ ਹੇਠਾਂ ਦਿੱਤੇ ਗਏ ਹਨ:

1. **ਪ੍ਰਿੰਟਰ :** ਇੱਕ ਪ੍ਰਿੰਟਰ ਦਾ ਮੁੱਖ ਉਦੇਸ਼ ਡਾਕੂਮੈਂਟਾਂ, ਡਾਟਾ, ਸੂਚਨਾ, ਚਿੱਤਰਾਂ ਅਤੇ ਫੋਟੋਆਂ ਦੀ ਹਾਰਡ ਕਾਪੀ ਤਿਆਰ ਕਰਨਾ ਹੈ। ਪ੍ਰਿੰਟਰਾਂ ਦੀਆਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਅਤੇ ਆਕਾਰ ਉਪਲਬਧ ਹਨ ਜਿਵੇਂ ਕਿ ਇੰਕਜੈੱਟ, ਲੇਜ਼ਰ, ਫੋਟੋ ਪ੍ਰਿੰਟਰ, ਲਾਈਨ ਪ੍ਰਿੰਟਰ, ਥਰਮਲ ਪ੍ਰਿੰਟਰ ਅਤੇ 3D ਪ੍ਰਿੰਟਰ ਆਦਿ।

ਚਿੱਤਰ 1.13 ਪ੍ਰਿੰਟਰ

2. **ਪਲੋਟਰ:** ਪਲੋਟਰ ਉਹ ਡਿਵਾਇਸ ਹਨ, ਜੋ ਕੰਪਿਊਟਰ ਏਡਿਡ ਡਿਜ਼ਾਇਨ (CAD) ਵਰਗੇ ਖੇਤਰਾਂ ਵਿੱਚ ਆਰਕੀਟੈਕਚਰਲ ਡਿਜ਼ਾਇਨ ਤੋਂ ਲੈ ਕੇ ਨਵੇਂ ਸਪੇਸ ਪ੍ਰੋਬ ਤੱਕ ਵੱਡੀਆਂ ਅਤੇ ਸਹੀ ਡਰਾਇੰਗਾਂ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਅਸਲ ਵਿੱਚ ਪਲੋਟਰਾਂ ਨੇ ਕਾਗਜ਼ ਤੇ ਲਾਈਨ ਖਿੱਚਣ ਲਈ ਇੱਕ ਪੈੱਨ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਅਤੇ ਬਾਅਦ ਵਿੱਚ ਕਈ ਪੈੱਨਾਂ ਨੇ ਪ੍ਰਕਿਰਿਆ ਵਿੱਚ ਰੰਗ ਲਿਆਂਦਾ। ਹਾਲਾਂਕਿ ਪੈੱਨ ਨੂੰ ਲਗਭਗ ਪੂਰੀ ਤਰ੍ਹਾਂ ਇੰਕਜੈਟ ਅਤੇ ਲੇਜ਼ਰ ਪ੍ਰਿੰਟ ਪ੍ਰਣਾਲੀਆਂ ਦੁਆਰਾ ਬਦਲ ਦਿੱਤਾ ਗਿਆ ਹੈ। ਪਰ ਪਲੋਟਰ ਪੈਨ ਤੇ ਅਧਾਰਤ ਹਨ ਇਸ ਨੂੰ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਯਾਦ ਰੱਖਣ ਵਿੱਚ ਇਹ ਸਾਡੀ ਮਦਦ ਕਰਦਾ ਹੈ। ਇਸਦੀਆਂ ਦੋ ਮੁੱਖ ਕਿਸਮਾਂ ਫਲੈਟਬੈੱਡ ਅਤੇ ਡਰੱਮ ਪਲਾਟਰ ਹਨ ॥

3. **ਸਪੀਕਰ ਅਤੇ ਹੈੱਡਫੋਨ:** ਇਹ ਯੂਜ਼ਰ ਨੂੰ ਆਡੀਓ ਆਊਟਪੁੱਟ ਪ੍ਰਦਾਨ ਕਰਦੇ

ਚਿੱਤਰ 1.14 ਪਲੋਟਰ

ਹਨ। ਸਪੀਕਰ ਸਾਊਂਡ ਕਾਰਡ ਵਿੱਚ ਇਲੈਕਟ੍ਰਾਨਿਕ ਸਿਗਨਲਾਂ ਨੂੰ ਅਵਾਜ਼ ਤਰੰਗਾਂ ਵਿੱਚ ਬਦਲਦੇ ਹਨ ਜੋ ਕਿ ਯੂਜ਼ਰ ਵਲੋਂ ਸੁਣੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ। ਸਪੀਕਰ ਦੀ ਆਵਾਜ਼ ਸਾਂਝੇ ਤੌਰ ਤੇ ਸਾਰੀਆਂ ਨੂੰ ਸੁਣਦੀ ਹੈ। ਹੈੱਡਫੋਨ ਇਹ ਯਕੀਨੀ ਬਣਾਉਂਦੇ ਹਨ ਕਿ ਗੋਪਨੀਯਤਾ (Privacy) ਬਣਾਈ ਗਈ ਹੈ। ਹਾਲਾਂ ਕਿ ਜੇਕਰ ਆਉਟਪੁੱਟ ਦੀ ਮਾਤਰਾ ਨੂੰ ਬਹੁਤ ਜ਼ਿਆਦਾ ਸੈਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਯੂਜ਼ਰ ਦੇ ਸੁਨਣ ਦੀ ਸਮਰੱਥਾ ਨੂੰ ਅਸਥਾਈ ਜਾਂ ਸਥਾਈ ਤੌਰ ਤੇ ਨੁਕਸਾਨ ਹੋ ਸਕਦਾ ਹੈ।

ਚਿੱਤਰ 1.15 : ਸਪੀਕਰ ਅਤੇ ਹੈੱਡਫੋਨ

4. **ਵਿਜ਼ੁਅਲ ਡਿਸਪਲੇ ਯੂਨਿਟ (VDU) ਜਾਂ ਸਕਰੀਨ:** ਇਸ ਨੂੰ ਇਨਪੁਟ ਡਿਵਾਇਸ ਅਤੇ ਆਉਟਪੁੱਟ ਡਿਵਾਇਸ ਦੋਵਾਂ ਦੀ ਤਰ੍ਹਾਂ ਮੰਨਿਆ ਜਾ ਸਕਦਾ ਹੈ ਕਿਉਂਕਿ ਇਹ ਕੰਪਿਊਟਰ ਲਈ ਹਦਾਇਤਾਂ ਜਾਂ ਪ੍ਰਾਪਤ ਸੂਚਨਾ ਦੋਨਾਂ ਨੂੰ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਦਾ ਹੈ। ਇਹ ਯੂਜ਼ਰ, ਜੋ ਕਿ ਡਾਟਾ ਅਤੇ ਹਦਾਇਤਾਂ ਨੂੰ ਇਨਪੁੱਟ ਕਰ ਰਿਹਾ ਹੈ ਦੀ ਮਦਦ ਕਰਦਾ ਹੈ ਕਿਉਂਕਿ ਇਹ ਯੂਜ਼ਰ ਨੂੰ ਆਪਣੇ ਕੀਤੇ ਕੰਮ ਨੂੰ ਦੇਖ ਕੇ ਚੈੱਕ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ, ਉਹਨਾਂ ਨੂੰ ਆਪਣੇ ਫੈਸਲੇ ਜਾਂ ਹਦਾਇਤਾਂ ਨੂੰ ਕਨਫਰਮ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ ਗਲਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ।

ਚਿੱਤਰ 1.16 : VDU

1.2.3 ਸਿਸਟਮ ਯੂਨਿਟ (System Unit) – CPU ਜਿਸਨੂੰ ਸੈਂਟਰਲ ਪ੍ਰੋਸੈਸਿੰਗ ਯੂਨਿਟ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ, ਹਦਾਇਤਾਂ ਅਤੇ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਲੋੜੀਂਦੀਆਂ ਸਾਰੀਆਂ ਹਾਰਡਵੇਅਰ ਗਤੀਵਿਧੀਆਂ ਦਾ ਰਿਕਾਰਡ ਰੱਖਦਾ ਹੈ, ਸਾਰੇ ਗਣਿਤਕ ਅਤੇ ਲਾਜੀਕਲ ਆਪਰੇਸ਼ਨ ਕਰਦਾ ਹੈ ਅਤੇ ਨਤੀਜੇ ਵਜੋਂ ਆਉਟਪੁੱਟ ਪੇਸ਼ ਕਰਦਾ ਹੈ। ਇਹ ਵੱਖ-ਵੱਖ ਸਟੋਰੇਜ ਡਿਵਾਇਸਾਂ ਦਾ ਅੰਦਰੂਨੀ (Internal) ਅਤੇ ਬਾਹਰੀ (External) ਰਿਕਾਰਡ ਰੱਖਦਾ ਹੈ ਕਿ ਸਾਰੇ ਪ੍ਰੋਗਰਾਮ ਅਤੇ ਹਦਾਇਤਾਂ ਕਿਥੇ ਪਏ ਹਨ ਤਾਂ ਕਿ ਭਵਿੱਖ ਵਿਚ ਮੁੜ ਪ੍ਰਾਪਤ ਕੀਤੇ ਜਾ ਸਕਣ।

1.2.3.1 CPU ਦੇ ਕੰਪੋਨੈਂਟਸ – CPU ਨੂੰ ਅੱਗੇ ਕੰਟਰੋਲ ਯੂਨਿਟ, ਆਰਥਮੈਟ੍ਰਿਕ ਅਤੇ ਲਾਜੀਕਲ ਯੂਨਿਟ ਅਤੇ ਮੈਮੋਰੀ ਯੂਨਿਟ ਵਿਚ ਵੰਡਿਆ ਗਿਆ ਹੈ। ਸਾਰੇ ਸਿਸਟਮ ਵਿਚ ਹਰੇਕ ਯੂਨਿਟ ਦੀ ਆਪਣੀ ਖਾਸ ਮਹੱਤਤਾ ਹੈ।

ੳ) **ਕੰਟਰੋਲ ਯੂਨਿਟ Control unit :** ਕੰਟਰੋਲ ਯੂਨਿਟ ਕੰਪਿਊਟਰ ਨਾਲ ਜੁੜੇ ਸਾਰੇ ਆਪਰੇਸ਼ਨਾਂ ਲਈ ਉਚਿਤ ਸਮਾਂ ਅਤੇ ਕੰਟਰੋਲ ਸਿਗਨਲ ਭੇਜਦਾ ਹੈ। ਕੰਟਰੋਲ ਯੂਨਿਟ ਦਾ ਮੁੱਖ ਕੰਮ ਮੇਨ ਮੈਮੋਰੀ ਤੋਂ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕਰਨਾ, ਡਿਵਾਇਸਾਂ ਅਤੇ ਇਸ ਵਿੱਚ ਸ਼ਾਮਲ ਉਪਰੇਸ਼ਨਾਂ ਨੂੰ ਨਿਰਧਾਰਤ ਕਰਨਾ ਅਤੇ ਓਪਰੇਸ਼ਨਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਕੰਟਰੋਲ ਸਿਗਨਲ ਪੈਦਾ ਕਰਨਾ ਹੈ। ਪ੍ਰੋਸੈਸਰ, ਮੈਮਰੀ ਅਤੇ ਹੋਰ ਪੈਰੀਫਿਰਲ ਡਿਵਾਇਸਾਂ ਦੇ ਦਰਮਿਆਨ ਡਾਟਾ ਦੇ ਪ੍ਰਵਾਹ ਨੂੰ ਕੰਟਰੋਲ ਯੂਨਿਟ ਦੇ ਟਾਈਮਿੰਗ ਸਿਗਨਲਾਂ ਰਾਹੀਂ ਕੰਟਰੋਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਕੰਪਿਊਟਰ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੀਆਂ ਹਦਾਇਤਾਂ ਨੂੰ ਮੁੜ ਪ੍ਰਾਪਤ ਕਰਕੇ ਅਤੇ ਸਹੀ ਕ੍ਰਮ ਵਿੱਚ ਇਨਟਰਪ੍ਰੇਟ (Interpret) ਕਰਨਾ ਯਕੀਨੀ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ।

ਅ) **ALU– :** ਇਹ ਸੈਂਟਰਲ ਪ੍ਰੋਸੈਸਿੰਗ ਯੂਨਿਟ ਦਾ ਇੱਕ ਮੁੱਖ ਹਿੱਸਾ ਹੈ, ਜਿਸਨੂੰ ਅਰਥਮੈਟਿਕ ਲੋਜਿਕ ਯੂਨਿਟ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਇਹ ਅਰਥਮੈਟਿਕ ਅਤੇ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਕਰਦਾ ਹੈ। ਇੱਥੇ ਕੰਪਿਊਟਰ ਹਦਾਇਤਾਂ ਨੂੰ ਪੂਰਾ ਕਰਨ ਲਈ ਲੋੜੀਂਦੇ ਗਣਤਿਕ ਅਤੇ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਕਰਦਾ ਹੈ। ਡਾਟਾ ਨੂੰ ਇਕ ਵਿਸ਼ੇਸ਼ ਰਜਿਸਟਰ ਵਿੱਚ ਰਖਿਆ ਜਾਂਦਾ ਹੈ ਜਿਸਨੂੰ ਐਕਯੁਲੇਟਰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਗਣਤਿਕ (ਜੋੜ ਅਤੇ ਘਟਾਓ) ਅਤੇ ਅਰਥਮੈਟਿਕ ਅਤੇ ਲਾਜੀਕਲ ਓਪਰੇਸ਼ਨ ਜਿਵੇਂ ਕਿ 'and' ਅਤੇ 'not' ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

ੲ) **ਰਜਿਸਟਰਜ਼ (Registers) :** ਰਜਿਸ਼ਟਰਜ਼ ਸਭ ਤੋਂ ਛੋਟੇ ਅਤੇ ਸਭ ਤੋਂ ਤੇਜ਼ ਹੁੰਦੇ ਹਨ। ਇਹਨਾਂ ਨੂੰ CPU ਦੁਆਰਾ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇੱਕ ਰਜਿਸਟਰ ਵਿਚ ਇੱਕ ਹਦਾਇਤ, ਇੱਕ ਸਟੋਰੇਜ ਐਡਰੈਸ ਜਾਂ ਕਿਸੇ ਪ੍ਰਕਾਰ ਦਾ ਡਾਟਾ (ਜਿਵੇਂ ਕਿ ਇੱਕ ਬਿਟ-ਕ੍ਰਮ ਜਾਂ ਇੱਕ ਅੱਖਰ) ਹੋ ਸਕਦਾ ਹੈ।

1.2.3.2 ਮੈਮੋਰੀ ਯੂਨਿਟ Memory Unit : ਯੂਨਿਟ, ਜਿੱਥੇ ਸਾਰੇ ਪ੍ਰੋਗਰਾਮ ਅਤੇ ਡਾਟਾ ਸਟੋਰ ਕੀਤੇ ਜਾਂਦੇ ਹਨ, ਨੂੰ ਮੈਮੋਰੀ ਯੂਨਿਟ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਨੂੰ ਅੱਗੇ ਦੋ ਵੱਖ-ਵੱਖ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਵੰਡਿਆ ਗਿਆ ਹੈ:

- ਪ੍ਰਾਇਮਰੀ ਸਟੋਰੇਜ਼ ਅਤੇ ਸੈਕੰਡਰੀ ਸਟੋਰੇਜ
- ਪ੍ਰਾਇਮਰੀ ਸਟੋਰੇਜ-ਕੰਪਿਊਟਰ ਦੀ ਪ੍ਰਾਇਮਰੀ ਸਟੋਰੇਜ ਨੂੰ ਅੱਗੇ ਹੇਠ ਲਿਖੀਆਂ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ:
  - ਰੈਂਡਮ ਐਕਸੈਸ ਮੈਮੋਰੀ (Random Access Memory(RAM) : ਇਹ ਤੇਜ਼ ਅਤੇ ਆਕਾਰ ਵਿੱਚ ਵੱਡੀ ਹੁੰਦੀ ਹੈ, ਪਰ ਇਹ CPU ਦੇ ਵਿੱਚ ਮੌਜੂਦ ਮੈਮੋਰੀ ਨਾਲੋਂ ਹੌਲੀ ਹੁੰਦੀ ਹੈ। ਇਹ ਲਗਭਗ ਸਾਰੇ ਹੋਰ ਡਾਟਾ ਅਤੇ ਹਦਾਇਤਾਂ ਨੂੰ ਰਖਦੀ ਹੈ ਜੋ ਕੰਪਿਊਟਰ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਲਈ ਲੋੜੀਂਦੇ ਹੋ ਸਕਦੇ ਹਨ, ਜੋ ਕਿ ਉਸ ਸਮੇਂ ਕੰਪਿਊਟਰ ਤੇ ਚੱਲ ਰਿਹਾ ਹੁੰਦਾ ਹੈ। ਜੇਕਰ ਪ੍ਰੋਗਰਾਮ ਦੀਆਂ ਲੋੜਾਂ ਬਦਲਦੀਆਂ ਹਨ ਜਾਂ ਪ੍ਰੋਗਰਾਮ ਕਿਸੇ ਦੂਸਰੇ ਪ੍ਰੋਗਰਾਮ ਨਾਲ ਬਦਲਿਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਮੈਮਰੀ ਤੇ ਨਵੇ ਡਾਟਾ ਅਤੇ ਹਦਾਇਤਾਂ ਨੂੰ ਓਵਰਰਾਈਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
  - ਰੀਡ ਓਨਲੀ ਮੈਮੋਰੀ Read Only Memory (ROM) : ਇਸ ਵਿੱਚ ਪਹਿਲਾਂ ਤੋਂ ਪ੍ਰੋਗਰਾਮ ਕੀਤੀਆਂ ਸਥਾਈ ਹਦਾਇਤਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜਿਵੇਂ ਕਿ ਬੇਸਿਕ ਇਨਪੁਟ/ਆਉਟਪੁੱਟ (BIOS) ਪ੍ਰੋਗਰਾਮ, ਜੋ ਕਿ ਜਦੋਂ ਤੁਸੀਂ ਕੰਪਿਊਟਰ ਦਾ ਸਵਿਚ ਆਨ ਕਰਦੇ ਹੋ, ਤਾਂ ਇਹ ਜਾਂਚ ਕਰਦੇ ਹਨ ਕਿ ਸਾਰੇ ਕਨੈਕਟ ਕੀਤੇ ਡਿਵਾਇਸ ਸਹੀ ਥਾਂ ਤੇ ਹਨ, ਇਹ ਜਾਂਚ ਕਰਦੇ ਹਨ ਕਿ ਉਹ ਸਾਰੇ ਡਿਵਾਇਸ ਸਹੀ ਕੰਮ ਕਰ ਰਹੇ ਹਨ ਅਤੇ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਨੂੰ ਕੰਪਿਊਟਰ ਵਿਚ ਲੋਡ ਕਰਦੇ ਹਨ। BIOS ਕੰਪਿਊਟਰ ਦਾ ਇੱਕ ਅਭਿੰਨ ਅੰਗ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਉਸਨੂੰ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ, ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਵਾਂਗ ਨਹੀਂ ਹੁੰਦਾ ਜੋ ਕਿ ਬਾਅਦ ਵਿੱਚ ਵੀ ਜੋੜਿਆ ਜਾ ਸਕਦਾ ਹੈ।
- ਸਕੈਂਡਰੀ ਸਟੋਰੇਜ—ਸਕੈਂਡਰੀ ਸਟੋਰੇਜ ਡਿਵਾਈਸ ਇਕ ਨਾਨ-ਵੋਲੇਟਾਈਨ ਸਟੋਰੇਜ਼ ਡਿਵਾਈਸ ਹੁੰਦੀ ਹੈ ਜੋ ਕਿ ਅੰਦਰੂਨੀ (Internal) ਜਾਂ ਬਾਹਰੀ (External) ਹੋ ਸਕਦੀ ਹੈ। ਇਸਨੂੰ ਪ੍ਰੋਗਰਾਮਾਂ ਅਤੇ ਡਾਟਾ ਨੂੰ ਸਥਾਈ ਤੌਰ ਤੇ ਸਟੋਰ ਕਰਨ ਲਈ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨਾਲ ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਜੋ ਭਵਿੱਖ ਵਿੱਚ ਲੋੜ ਪੈਣ ਤੇ ਉਹਨਾਂ ਨੂੰ ਮੁੜ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾ ਸਕੇ। ਜਿਵੇਂ ਕਿ : ਫਲਾਪੀ ਡਿਸਕ, ਹਾਰਡ ਡਿਸਕ, ਸੀ-ਡੀ., ਡੀ.ਵੀ.ਡੀ, ਆਮ ਟਿਕਲ ਡਿਸਕ, ਯੂ.ਐੱਸ.ਬੀ. ਫਲੈਸ਼ ਡਰਾਈਵ ਆਦਿ।

1.3 ਸਾਫਟਵੇਅਰ ਅਤੇ ਇਸ ਦੀਆਂ ਕਿਸਮਾਂ (Software & its types) :

ਸਾਫਟਵੇਅਰ ਪ੍ਰੋਗਰਾਮ ਜਾਂ ਹਦਾਇਤਾਂ ਹੁੰਦੇ ਹਨ ਜੋ ਕੰਪਿਊਟਰ ਨੂੰ ਦੱਸਦੇ ਹਨ ਕਿ ਕੀ ਕਰਨਾ ਹੈ ਜਾਂ ਹਾਰਡਵੇਅਰ ਨੂੰ ਚਲਾਉਂਦੇ ਹਨ। ਸਾਫਟਵੇਅਰ ਵਿੱਚ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਸੰਚਾਲਨ ਨਾਲ ਜੁੜੇ ਪ੍ਰੋਗਰਾਮਾਂ, ਪ੍ਰੋਸੈਸਾਂ ਅਤੇ ਰੁਟੀਨਾਂ ਦਾ ਸਮੂਹ ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ। ਯੂਜ਼ਰ ਨੂੰ ਆਪਸ ਵਿੱਚ ਜੁੜੇ ਕੰਮਾਂ ਨੂੰ ਕਰਨ ਲਈ ਦਿੱਤੇ ਗਏ ਪ੍ਰੋਗਰਾਮਾਂ ਦੇ ਸਮੂਹ ਨੂੰ ਸਾਫਟਵੇਅਰ ਪੈਕੇਜ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਕੰਪਿਊਟਰ ਸਾਫਟਵੇਅਰ ਨੂੰ ਕਈ ਤਰੀਕਿਆਂ ਨਾਲ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ:

1.3.1 ਵਿਕਾਸ ਜਾਂ ਡਵੈਲਪਮੈਂਟ ਦੇ ਆਧਾਰ ਤੇ ਸਾਫਟਵੇਅਰ ਦੀਆਂ ਕਿਸਮਾਂ:

- ਇਕ ਤਰੀਕਾ ਜਿਸ ਵਿੱਚ ਸਾਫਟਵੇਅਰ ਦੇ ਵਿਕਾਸ ਜਾਂ ਡਵੈਲਮੈਂਟ ਨੂੰ ਦੇਖਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਹੇਠਾਂ ਦਿੱਤੇ ਮੁੱਖ ਨੁਕਤਿਆਂ ਨੂੰ ਧਿਆਨ ਵਿੱਚ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ:
  1. ਕੀ ਇਹ ਕਾਪੀ -ਰਾਈਟ ਸੁਰੱਖਿਅਤ ਹੈ ਜਾਂ ਸਾਰਿਆਂ ਲਈ ਉਪਲਬਧ ਹੈ ?
  2. ਕੀ ਯੂਜ਼ਰ ਸੋਰਸ ਕੋਡ ਵਿੱਚ ਬਦਲਾਅ ਕਰ ਸਕਦਾ ਹੈ ?
  3. ਕੀ ਇਹ ਕਿਸੇ ਵਿਸ਼ੇਸ਼ ਸੰਸਥਾ ਲਈ ਜਾਂ ਆਮ ਤੌਰ ਤੇ ਬਹੁਤ ਸਾਰੇ ਵਿਅਕਤੀਆਂ ਅਤੇ ਸੰਸਥਾਵਾਂ ਦੀ ਵਰਤੋਂ ਲਈ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਹੈ ?

ਇਹਨਾਂ ਅੰਤਰਾਂ ਦਾ ਵਰਣਨ ਹੇਠਾਂ ਦਿੱਤਾ ਗਿਆ ਹੈ:

- **ਓਪਨ ਸੋਰਸ ਸਾਫਟਵੇਅਰ:** ਓਪਨ ਸੋਰਸ ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮਰ ਦੁਆਰਾ ਤਿਆਰ ਕੀਤਾ ਸੋਰਸ ਕੋਡ ਕਿਸੇ ਵੀ ਵਿਅਕਤੀ ਲਈ ਉਪਲਬਧ ਹੈ ਜੋ ਕਿ ਇਸ ਨੂੰ ਡੀਬੱਗ, ਸੋਧ ਜਾਂ ਲੋੜ ਅਨੁਸਾਰ ਬਦਲ ਸਕਦਾ ਹੈ। ਮੂਲ ਸੋਰਸ ਕੋਡ ਦੇ ਕੁਝ ਉਤਪਾਦਕ (Producers) ਸਾਰੇ ਜਾਂ ਕੁਝ ਕੋਡ ਨੂੰ ਐਕਸੈਸ ਜਾਂ ਮੋਡੀਫਾਈ ਕਰਨ ਨੂੰ ਪ੍ਰਤੀਬੰਧਿਤ ਕਰ ਸਕਦੇ ਹਨ, ਪਰ ਉਸਨੂੰ ਐਕਸੈਸ ਜਾਂ ਵਰਤੋਂ ਹਰ ਕੋਈ ਕਰ ਸਕਦਾ ਹੈ।
- **ਕਲੋਜ਼ਡ ਸੋਰਸ ਸਾਫਟਵੇਅਰ:** ਕਲੋਜ਼ਡ ਸੋਰਸ ਸਾਫਟਵੇਅਰ ਨੂੰ ਸੰਪਤੀ ਦੇ ਅਧਿਕਾਰਾਂ ਅਤੇ ਕਾਪੀ ਰਾਈਟ ਦੁਆਰਾ ਸੁਰੱਖਿਅਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਪਾਬੰਦੀਆਂ ਨੂੰ ਤੋੜਨ ਵਾਲਿਆਂ ਤੇ ਕਾਨੂੰਨੀ ਕਾਰਵਾਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਅਜਿਹੇ ਸਾਫਟਵੇਅਰ ਨੂੰ ਖਰੀਦਣ ਵੇਲੇ ਤੁਸੀਂ ਇਸਦੀ ਵਰਤੋਂ ਕਰਨ ਦਾ ਸਿਰਫ ਇੱਕ ਅਧਿਕਾਰ ਹੀ ਖਰੀਦ ਰਹੇ ਹੋ ਨਾ ਕਿ ਉਸਦੇ ਮਾਲਿਕਾਨਾ ਹੱਕ।
- **ਆਫ਼ ਦੀ ਸ਼ੈਲਫ ਅਤੇ ਬੀਸਪੋਕ ਸਾਫਟਵੇਅਰ :** ਇੱਕ ਕਾਰੋਬਾਰ ਜਾਂ ਸੰਸਥਾ ਜੋ ਵਿੱਤੀ-ਯੋਜਨਾ ਜਾਂ ਇਨਵੈਨਟਰੀ ਸਿਸਟਮ ਵਰਗੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਬਦਲਣਾ ਜਾਂ ਅਪਣਾਉਦਾ ਚਾਹੁੰਦੀ ਹੈ, ਇੱਕ ਆਮ ਪੈਕੇਜ ਖਰੀਦ ਸਕਦੀ ਹੈ ਅਤੇ ਫਿਰ ਉਸਦੇ ਕੁਝ ਲੋੜੀਂਦੇ ਹਿੱਸੇ ਵਰਤ ਸਕਦੀ ਹੈ, ਜਾਂ ਜੇਕਰ ਉਹਨਾਂ ਦਾ ਕੰਮ ਕੁਝ ਵੱਖਰਾ ਹੈ, ਤਾਂ ਉਹ ਉਹਨਾਂ ਲਈ ਲਿਖਿਆ ਖਾਸ ਪੈਕੇਜ ਵੀ ਲੈ ਸਕਦੇ ਹਨ। ਪਹਿਲੇ ਵਿਕਲਪ ਨੂੰ ਅਕਸਰ ਆਫ ਦੀ ਸ਼ੈਲਫ ਕਿਹਾ ਜਾਦਾ ਹੈ। ਕਿਉਂਕਿ ਪੈਕੇਜ ਪੂਰਵ-ਡਿਜਾਇਨ ਕੀਤਾ ਗਿਆ ਹੈ ਅਤੇ ਇੱਕ ਬਕਸੇ ਵਿੱਚ ਆਇਆ ਹੈ। ਜਿਸਨੂੰ ਕਿਸੇ ਵੀ ਵਿਅਕਤੀ ਨੇ ਸ਼ਾਬਦਿਕ ਤੌਰ ਤੇ ਸ਼ੈਲਫ ਤੋਂ ਚੁੱਕ ਲਿਆ ਹੈ। ਕੋਈ ਵੀ ਇਸ ਵਿੱਚ ਛੋਟੀਆਂ ਤਬਦੀਲੀਆਂ ਕਰ ਸਕਦਾ ਹੈ, ਜਿਵੇਂ ਕਿ ਰੰਗ ਜਾਂ ਲੇ-ਆਉਟ, ਫਾਰਮਾਂ ਅਤੇ ਰਿਪੋਰਟਾਂ ਦਾ ਡਿਜਾਇਨ ਆਦਿ। ਸਾਫਟਵੇਅਰ ਵਿੱਚ ਕੰਮ ਕਰਨ ਲਈ ਕਿਸੇ ਨੂੰ ਆਪਣੇ ਸਿਸਟਮ ਅਤੇ ਕੰਮ ਕਰਨ ਦੇ ਤਰੀਕਿਆਂ ਨੂੰ ਬਦਲਣਾ ਪੈ ਸਕਦਾ ਹੈ।
- **ਬੀਸਪੋਕ ਸਾਫਟਵੇਅਰ:** ਬੀਸਪੋਕ ਸੂਟ ਜਾਂ ਸ਼ਰਟ (ਕਮੀਜ਼) ਦੀ ਤਰ੍ਹਾਂ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿ ਕਿਸੇ ਦਾ ਸਹੀ ਮਾਪ ਲੈ ਕੇ ਬਣਾਏ ਜਾਂਦੇ ਹਨ। ਇਹ ਮਹਿੰਗੇ ਹੁੰਦੇ ਹਨ ਅਤੇ ਡਿਜ਼ਾਇਨਰ ਨੂੰ ਕਿਸੇ ਦੀਆਂ ਜ਼ਰੂਰਤਾਂ ਨੂੰ ਸਮਝਣ ਵਿੱਚ ਸਮਾਂ ਲਗਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਬਣਾ ਕੇ ਦੇਣ ਵਿੱਚ ਲੱਗਣ ਵਾਲੇ ਸਮੇਂ ਨੂੰ ਵੀ ਵਿਚਾਰਿਆ ਜਾਂਦਾ ਹੈ। ਜੋ ਇਸ ਤਰ੍ਹਾਂ ਦੇ ਸਾਫਟਵੇਅਰ ਬਣਵਾਉਂਦੇ ਹਨ ਉਹਨਾਂ ਦੇ ਉਸ ਉੱਤੇ ਸੰਪਤੀ ਦੇ ਮਾਲਿਕਾਨਾ ਹੱਕ ਹੁੰਦੇ ਹਨ।
- **ਸ਼ੇਅਰ ਵੇਅਰ :** ਜੇਕਰ ਯੂਜ਼ਰ ਨੂੰ ਇਹ ਨਿਸ਼ਚਿਤ ਨਹੀਂ ਹੈ ਕਿ ਉਹਨਾਂ ਨੂੰ ਕੀ ਚਾਹੀਦਾ ਹੈ, ਤਾਂ ਉਹ 30-60 ਦਿਨਾਂ ਤੱਕ ਪ੍ਰੋਡੱਕਟ ਖਰੀਦਣ ਤੋਂ ਪਹਿਲਾਂ ਸ਼ੇਅਰਵੇਅਰ ਨੂੰ ਵਰਤ ਕੇ ਦੇਖ ਸਕਦੇ ਹਨ। ਸਮੇਂ ਦੇ ਪੂਰੇ ਹੋਣ ਤੇ ਜਾਂ ਤਾਂ ਖਰੀਦ ਮੁੱਲ ਦਾ ਭੁਗਤਾਨ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜਾਂ ਸਾਫਟਵੇਅਰ ਉਹਨਾਂ ਲਈ ਬੰਦ ਹੋ ਜਾਂਦਾ ਹੈ। ਇਸਦੇ ਮਾਲਿਕਾਨਾਂ ਹੱਕ ਅਤੇ ਕਾਪੀਰਾਈਟ ਸ਼ੇਅਰਵੇਅਰ ਲਿਖਣ ਵਾਲੇ ਕੋਲ ਹੀ ਰਹਿੰਦੇ ਹਨ।
- **ਫ੍ਰੀਵੇਅਰ:** ਫ੍ਰੀਵੇਅਰ ਨੂੰ ਮੁਫ਼ਤ ਵੰਡਿਆ ਜਾਂਦਾ ਹੈ, ਹਾਲਾਂਕਿ ਅੱਪਗ੍ਰੇਡਸ ਤੇ ਪਾਬੰਦੀ ਹੋ ਸਕਦੀ ਹੈ ਜਾਂ ਕੋਈ ਵਾਰੰਟੀ ਨਹੀਂ ਹੋ ਸਕਦੀ। ਸਾਰੇ ਅਧਿਕਾਰ ਲੇਖਕ ਜਾਂ ਪਬਲੀਸ਼ਰ ਕੋਲ ਰਹਿੰਦੇ ਹਨ। ਫ੍ਰੀਵੇਅਰ ਉਤਪਾਦਾਂ ਦੀ ਰੇਜ਼ ਛੋਟੇ, ਵਿਸ਼ੇਸ਼ ਯੁਟਿਲਿਟੀ ਪ੍ਰੋਗਰਾਮਾਂ ਤੋਂ ਲੈ ਕੇ ਜਾਣ-ਪਛਾਣ ਪ੍ਰੋਗਰਾਮਾਂ ਜਿਵੇਂ ਕਿ ਐਕਰੋਬੇਟ ਰੀਡਰ, MSN ਮੈਸੇਜਰ ਜੋ ਕਿ ਸਾਰਿਆਂ ਲਈ ਉਪਲਬਧ ਹਨ, ਤੱਕ ਹੈ।
- **ਏਮਬੈੱਡਡ ਸਾਫਟਵੇਅਰ:** ਇਸ ਕਿਸਮ ਦੇ ਸਾਫਟਵੇਅਰ ਇੱਕ ਆਮ PC ਜਾਂ ਡਿਵਾਈਸਾਂ, ਜਿਨ੍ਹਾਂ ਨੂੰ ਅਸੀਂ ਕੰਪਿਊਟਰ ਨਹੀਂ ਸਮਝਦੇ, ਵਿਚ ਏਮਬੈਡ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ ਮਾਈਕ੍ਰੋਵੇਵ ਓਵਨ, GPS ਸਿਸਟਮ, ਸਮਾਰਟ ਘੜੀਆਂ ਅਤੇ ਗਾਈਡਿਡ ਮਿਸਾਈਲ ਟੈਕਨੋਲੋਜੀ ਆਦਿ॥

1.3.2 ਕੰਮ ਦੇ ਅਧਾਰ ਤੇ ਸਾਫਟਵੇਅਰ ਦੀਆਂ ਕਿਸਮਾਂ

ਸਾਫਟਵੇਅਰ ਦੇ ਵਰਗੀਕਰਨ ਦਾ ਦੂਜਾ ਤਰੀਕਾ ਇਸ ਦੇ ਕੰਮ ਕਰਨ ਤੇ ਨਿਰਭਰ ਕਰਦਾ ਹੈ। ਇਸ ਲਈ ਕੰਪਿਊਟਰ ਸਾਫਟਵੇਅਰ ਨੂੰ ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ, ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ ਅਤੇ ਯੂਟਿਲਿਟੀ ਸਾਫਟਵੇਅਰ ਦੇ ਰੂਪ ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

- **ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ:** ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਕੰਪਿਊਟਰ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੇ ਪ੍ਰੋਗਰਾਮ ਹੁੰਦੇ ਹਨ ਅਤੇ ਇਹ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ ਲਈ ਪਲੇਟਫਾਰਮ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ। ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਇੱਕ ਜਾਂ ਇੱਕ ਤੋਂ ਵੱਧ ਪ੍ਰੋਗਰਾਮਾਂ ਦਾ ਸੰਗ੍ਰਹਿ ਹੈ ਜੋ ਕਿ ਇੱਕ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਸਮੁੱਚੇ ਸੰਚਾਲਨ ਅਤੇ ਕਾਰਜਸ਼ੀਲਤਾ ਨੂੰ ਕੰਟਰੋਲ ਅਤੇ ਪ੍ਰਬੰਧਿਤ ਕਰਦਾ ਹੈ। ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਤੋਂ ਬਿਨਾਂ, ਕੰਪਿਊਟਰ ਹਾਰਡਵੇਅਰ ਆਪਣੇ ਕੰਮ ਨੂੰ ਸਹੀ ਢੰਗ ਨਾਲ ਕਰਨ ਦੇ ਯੋਗ ਨਹੀਂ ਹੁੰਦਾ। ਇਸਨੂੰ ਹੇਠ ਲਿਖੀਆਂ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ, ਲੈਂਗੂਏਜ ਪ੍ਰੋਸੈਸਰ, ਡਿਵਾਈਸ ਡਾਈਵਰ, ਅਤੇ ਯੂਟਿਲਿਟੀ ਸਾਫਟਵੇਅਰ।
    - **ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ-**ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਯੂਜ਼ਰ ਅਤੇ ਹਾਰਡਵੇਅਰ ਵਿਚਕਾਰ ਇੱਕ ਇੰਟਰਫੇਸ ਵਜੋਂ ਕੰਮ ਕਰਦਾ ਹੈ। ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ (OS) ਇੱਕ ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿ ਕੰਪਿਊਟਰ ਹਾਰਡਵੇਅਰ, ਸਾਫਟਵੇਅਰ ਸਰੋਤਾਂ ਦਾ ਪ੍ਰਬੰਧ ਕਰਦਾ ਹੈ ਅਤੇ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਾਂ ਲਈ ਸਾਂਝੀਆਂ ਸੇਵਾਵਾਂ ਉਪਲਬਧ ਕਰਵਾਉਂਦਾ ਹੈ। ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਕੰਪਿਊਟਰ ਦੇ ਸਾਰੇ ਸਾਫਟਵੇਅਰ ਅਤੇ ਹਾਰਡਵੇਅਰ ਦਾ ਪ੍ਰਬੰਧ ਕਰਦਾ ਹੈ। ਬਹੁਤਵਾਰ, ਇੱਕੋ ਸਮੇਂ ਕਈ ਵੱਖ-ਵੱਖ ਪ੍ਰੋਗਰਾਮ ਚਲਦੇ ਹਨ ਅਤੇ ਉਹਨਾਂ ਸਾਰੀਆਂ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਦੇ CPU, ਮੈਮੋਰੀ ਅਤੇ ਅਤੇ ਸਟੋਰੇਜ ਨੂੰ ਅਸੈਸ ਕਰਨ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ । ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਹਰੇਕ ਪ੍ਰੋਗਰਾਮ ਦੀ ਜ਼ਰੂਰਤ ਮੁਤਾਬਕ ਤਾਲਮੇਲ ਬਠਾਉਂਦਾ ਹੈ ਤਾਂ ਜੋ ਸਭ ਦੀ ਜਰੂਰਤ ਪੂਰੀ ਹੋ ਸਕੇ।

      ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਆਮ ਤੌਰ ਤੇ ਕੰਪਿਊਟਰ ਵਿਚ ਪਹਿਲਾਂ ਤੋਂ ਹੀ ਲੋਡ ਹੋਏ ਹੁੰਦੇ ਹਨ। ਜ਼ਿਆਦਾਤਰ ਲੋਕ ਉਹੀ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ ਜੋ ਕਿ ਕੰਪਿਊਟਰ ਦੇ ਨਾਲ ਹੀ ਆਉਂਦਾ ਹੈ, ਪਰ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਨੂੰ ਅਪਗ੍ਰੇਡ ਕਰਨਾ ਜਾਂ ਬਦਲਣਾ ਵੀ ਸੰਭਵ ਹੈ। ਪਰਸਨਲ ਕੰਪਿਊਟਰ ਲਈ ਤਿੰਨ ਸਭ ਤੋਂ ਆਮ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਹਨ; ਮਾਈਕ੍ਰੋਸਾਫਟ ਵਿੰਡੋਜ਼ **(Microsoft Windows)**, **Mac OS**, and **Linux**.
    - **ਲੈਂਗੂਏਜ ਪ੍ਰੋਸੈਸਰ:** ਲੈਂਗੂਏਜ਼ ਪ੍ਰੋਸੈਸਰ ਇਕ ਸਾਫਟਵੇਅਰ ਪ੍ਰੋਗਰਾਮ ਹੈ ਜੋ ਉੱਚ ਪੱਧਰੀ ਭਾਸ਼ਾ ਵਿੱਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਕੋਡ ਨੂੰ ਮਸ਼ੀਨ ਕੋਡ ਵਿੱਚ ਬਦਲਣ ਲਈ ਡਿਜ਼ਾਇਨ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਲੈਂਗੂਏਜ ਪ੍ਰੋਸੈਸਰ FORTRAN ਅਤੇ COBOL ਵਰਗੀਆਂ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਪਾਏ ਜਾਂਦੇ ਹਨ। ਲੈਂਗੂਏਜ ਪ੍ਰੋਸੈਸਰਾਂ ਦੀਆਂ ਤਿੰਨ ਕਿਸਮਾਂ ਹਨ:
        - ਅਸੈਂਬਲਰ
        - ਇੰਨਟਰਪ੍ਰੇਟਰ
        - ਕੰਪਾਈਲਰ
    - **ਡਿਵਾਇਸ ਡਰਾਈਵਰ-**ਇੱਕ ਡਿਵਾਇਸ ਡ੍ਰਾਈਵਰ ਸਾਫਟਵੇਅਰ ਪ੍ਰੋਗਰਾਮ ਦਾ ਖਾਸ ਰੂਪ ਹੈ ਜੋ ਸਿਸਟਮ ਅਤੇ ਹਾਰਡਵੇਅਰ ਡਿਵਾਈਸ ਵਿਚਕਾਰ ਤਾਲਮੇਲ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ ਜਾਂ ਇੱਕ ਖਾਸ ਹਾਰਡਵੇਅਰ ਡਿਵਾਈਸ ਨੂੰ ਕੰਟਰੋਲ ਕਰਦਾ ਹੈ ਜਿਵੇਂ ਕਿ ਕੰਪਿਊਟਰ ਨਾਲ ਜੁੜਿਆ ਪ੍ਰਿੰਟਰ। ਕੰਪਿਊਟਰ ਦੇ ਸਹੀ ਢੰਗ ਨਾਲ ਕੰਮ ਕਰਨ ਲਈ ਡਿਵਾਈਸ ਡਰਾਈਵਰ ਜ਼ਰੂਰੀ ਹਨ।
    - **ਯੂਟਿਲਿਟੀ ਸਾਫਟਵੇਅਰ:** ਕੰਪਿਊਟਰ ਦਾ ਕੰਮ ਕਈ ਵਾਰ ਪ੍ਰਭਾਵਿਤ ਹੋ ਸਕਦਾ ਹੈ। ਜਿਵੇਂ ਕਿ ਮੈਮੋਰੀ ਖਤਮ ਹੋਣਾ ਜਾਂ ਕਈ ਵਾਰ ਹੌਲੀ ਕੰਮ ਕਰਨਾ, ਜੋ ਕਿ ਵਾਈਰਸ ਦੇ ਕਾਰਨ ਹੋ ਸਕਦਾ ਹੈ। ਯੂਟਿਲਟੀ ਪ੍ਰੋਗਰਾਮ ਛੋਟੇ ਪ੍ਰੋਗਰਾਮ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿ ਕੰਪਿਊਟਰ ਦੀ ਕਾਰਜ ਕੁਸ਼ਲਤਾ ਵਧਾਉਂਦੇ ਹਨ। ਉਦਾਹਰਨ ਦੇ ਤੌਰ ਤੇ ਬੈੱਕਅਪ ਲੈਣਾ, ਜੋ ਕਿ ਤੁਹਾਨੂੰ ਹਾਰਡਵੇਅਰ ਫੇਲੀਅਰ ਦੀ ਸੂਰਤ ਵਿੱਚ ਜਾਂ ਚੋਰੀ ਹੋਣ ਦੀ ਸੂਰਤ ਵਿੱਚ ਬੈਕਅਪ ਲੈਣ ਦੀ ਯਾਦ ਦਵਾਉਂਦੇ ਹਨ। ਐਟੀਵਾਈਰਸ ਯੂਟਿਲਟੀ ਲਗਾਤਾਰ ਕੰਪਿਊਟਰ ਨੂੰ ਮਾਲਵੇਅਰ ਦੀ ਗਤੀਵਿਧੀ ਜਾਂ ਮੌਜੂਦਗੀ ਲਈ ਨਿਯਮਿਤ ਤੌਰ ਤੇ ਸਕੈਨ ਕਰਦੇ ਹਨ, ਇਹ ਨਵੇਂ ਖਤਰਿਆਂ ਦੀ ਜਾਂਚ ਵੀ ਕਰਦੇ ਹਨ ਅਤੇ ਉਹਨਾਂ ਨੂੰ ਕੁਆਰੰਟੀਨ ਲੋਕੇਸ਼ਨ ਤੇ ਰੱਖ ਦਿੰਦੇ ਹਨ, ਜਦੋਂ ਤੱਕ ਕਿ ਯੂਜ਼ਰ ਉਹਨਾਂ ਨੂੰ ਡਿਲੀਟ ਕਰਨ ਦੀ ਪੁਸ਼ਟੀ ਨਹੀਂ ਕਰਦਾ।

- **ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਅਰ-** ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ ਉਹ ਸਾਫਟਵੇਅਰ ਹੈ ਜੋ ਯੂਜ਼ਰ ਨੂੰ ਖਾਸ ਕੰਮ ਕਰਨ ਵਿਚ ਮਦਦ ਕਰਦਾ ਹੈ। ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ ਨੂੰ ਕਈ ਤਰ੍ਹਾਂ ਦੇ ਕੰਮ ਕਰਨ ਲਈ ਡਿਜ਼ਾਇਨ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜਿਵੇਂ ਕਿ ਅਕਾਂਉਟਸ ਮੈਨੇਜਮੈਂਟ, ਟੈਕਸਟ ਬਣਾਉਣਾ ਅਤੇ ਐਡਿਟ ਕਰਨਾ, ਪ੍ਰੈਜਨਟੈਸ਼ਨ ਬਣਾਉਣਾ ਅਤੇ ਅਤੇ ਡਰਾਇੰਗ ਡਿਜਾਈਨਿੰਗ ਆਦਿ। ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ ਦੀਆਂ ਉਦਾਹਰਨਾਂ ਹਨ; ਵੈਬ ਬ੍ਰਾਊਜ਼ਰ ਜੋ ਕਿ ਯੂਜ਼ਰ ਨੂੰ ਇੰਟਰਨੈੱਟ ਅਸੈਸ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ, ਅਡੋਬ ਫੋਟੋਸ਼ਾਪ ਚਿੱਤਰਾਂ ਅਤੇ ਫੋਟੋਆਂ ਨੂੰ ਬਣਾਉਣ ਅਤੇ ਐਡਿਟ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

1.4 IT ਵਿੱਚ ਰੁਝਾਨ (Trends in IT) :

ਟੈਕਨੋਲੋਜੀ ਵਿੱਚ ਤਰੱਕੀ ਅਤੇ ਸੂਚਨਾ ਪ੍ਰਣਾਲੀਆਂ ਵਿਚ ਵਿਕਾਸ ਦੇ ਨਾਲ, ਹਰ ਖੇਤਰ ਵਿੱਚ ਇਨਫੋਰਮੇਸ਼ਨ ਟੈਕਨੋਲੋਜੀ ਨੂੰ ਅਪਣਾਉਣ ਦੀ ਲੋੜ ਹੈ। ਇਨਫੋਰਮੇਸ਼ਨ ਟੈਕਨੋਲੋਜੀ ਦੀ ਵਰਤੋਂ ਕੰਪਿਊਟਰਾਂ ਅਤੇ ਦੂਰ ਸੰਚਾਰ ਉਪਕਰਨਾਂ ਦੇ ਸੰਦਰਭ ਵਿੱਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। IT ਉਦਯੋਗ ਨੂੰ ਇੱਕ ਉੱਭਰਦਾ ਉਦਯੋਗ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ ਅਸੀਂ ਸਾਰੇ ਹੀ ਇਸ ਉਦਯੋਗ ਵਿੱਚ ਇਸਦੀ ਸ਼ੁਰੂਆਤ ਤੋਂ ਹੀ ਕਈ ਟਰੈਂਡਜ਼ (ਰੁਝਾਨਾਂ) ਦਾ ਸਾਹਮਣਾ ਕਰਦੇ ਆਏ ਹਾਂ। ਇਸ ਉਦਯੋਗ ਵਿੱਚ ਹਰ ਸਾਲ ਨਵੇਂ ਟਰੈਂਡ ਪੈਦਾ ਹੁੰਦੇ ਹਨ, ਅਤੇ ਪੇਸ਼ੇਵਰਾਂ/ਪ੍ਰੋਫੈਸ਼ਨਲਾਂ ਲਈ ਇਹਨਾਂ ਵੱਖ-ਵੱਖ ਟਰੈਂਡਜ਼ ਤੋਂ ਜਾਣੂ ਹੋਣਾ ਮਹੱਤਵਪੂਰਨ ਹੋ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਟਰੈਂਡਜ਼ ਵਿੱਚ ਕੰਪਿਊਟਰ ਤੇ ਅਧਾਰਤ ਇਨਫੋਰਮੇਸ਼ਨ ਸਿਸਟਮ ਦਾ ਅਧਿਐਨ, ਡਿਜ਼ਾਇਨ, ਵਿਕਾਸ, ਐਪਲੀਕੇਸ਼ਨ, ਇੰਪਲੀਮੈਨਟੇਸ਼ਨ ਸਪੋਰਟ ਜਾਂ ਪ੍ਰਬੰਧਨ ਸ਼ਾਮਲ ਹਨ। ਆਓ ਇਹਨਾਂ ਵਿੱਚੋਂ ਕੁਝ ਦੀ ਚਰਚਾ ਕਰੀਏ:

1.4.1 GPS : ਗਲੋਬਲ ਪੋਜੀਸ਼ਨਿੰਗ ਸਿਸਟਮ (GPS) U.S. ਯੁਟੀਲਿਟੀ ਪ੍ਰੋਗਰਾਮ ਹੈ ਜੋ ਕਿ ਯੂਜ਼ਰ ਨੂੰ (PNT) ਪੋਜੀਸ਼ਨ, ਨੈਵੀਗੇਸ਼ਨ ਅਤੇ ਟਾਈਮਿੰਗ ਸੇਵਾਵਾਂ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ। ਇਸ ਸਿਸਟਮ ਵਿੱਚ ਤਿੰਨ ਹਿੱਸੇ ਹੁੰਦੇ ਹਨ; ਸਪੇਸ ਸੈਗਮੈਂਟ, ਕੰਟਰੋਲ ਸੈਗਮੈਂਟ ਅਤੇ ਯੂਜ਼ਰ ਸੈਗਮੈਂਟ।

1.4.2 Bluetooth: (ਬਲੂਟੁੱਥ) : ਬਲੂਟੁੱਥ ਰੇਡੀਓ ਤਰੰਗਾਂ ਵਰਤ ਕੇ ਦੋ ਡਿਵਾਈਸਾਂ ਦਰਮਿਆਨ ਸੂਚਨਾ ਭੇਜਦਾ ਹੈ। ਬਲੂਟੁੱਥ ਦੁਆਰਾ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਰੇਡੀਓ ਤਰੰਗਾਂ ਵਾਈ-ਫਾਈ ਜਾਂ ਸੈਲੂਲਰ ਸਿਗਨਲ ਦੋ ਹੋਰ ਤਰੀਕੇ ਜੋ ਕਿ ਡਿਵਾਈਸਾਂ ਨੂੰ ਜੋੜਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ, ਨਾਲੋਂ ਬਹੁਤ ਕਮਜ਼ੋਰ ਹੁੰਦੀਆਂ ਹਨ।

1.4.3 Wi-Fi: (ਵਾਈ-ਫਾਈ) : ਇਹ ਇੱਕ ਵਾਇਰਲੈੱਸ ਟੈਕਨੋਲੋਜੀ ਹੈ ਜੋ ਕਿ ਕੰਪਿਊਟਰਾਂ, ਟੈਬਲੈੱਟਸ, ਸਮਾਰਟ ਫੋਨ ਅਤੇ ਹੋਰ ਡਿਵਾਈਸਾਂ ਨੂੰ ਇੰਟਰਨੈੱਟ ਨਾਲ ਜੋੜਨ ਲਈ ਵਰਤੀ ਜਾਂਦੀ ਹੈ। ਵਾਈ-ਫਾਈ ਇਕ ਵਾਇਰਲੈੱਸ ਰਾਊਟਰ ਤੋਂ ਨੇੜਲੇ ਡਿਵਾਈਸ ਨੂੰ ਭੇਜਿਆ ਜਾਣ ਵਾਲਾ ਰੇਡੀਓ ਸਿਗਨਲ ਹੈ, ਜੋ ਕਿ ਸਿਗਨਲ ਦਾ ਉਸ ਡਾਟਾ ਵਿੱਚ ਅਨੁਵਾਦ ਕਰਦਾ ਜਿਸਨੂੰ ਤੁਸੀਂ ਦੇਖ ਸਕਦੇ ਹੋ ਅਤੇ ਵਰਤ ਸਕਦੇ ਹੋ। ਡਿਵਾਈਸ ਇੱਕ ਰੇਡੀਓ ਸਿਗਨਲ ਨੂੰ ਰਾਊਟਰ ਵਿੱਚ ਵਾਪਸ ਭੇਜਦੀ ਹੈ, ਜੋ ਕਿ ਤਾਰ ਜਾਂ ਕੇਬਲ ਦੁਆਰਾ ਇੰਟਰਨੈੱਟ ਨਾਲ ਜੁੜਿਆ ਹੁੰਦਾ ਹੈ।

ਇੱਕ ਵਾਈ-ਫਾਈ ਨੈਟਵਰਕ ਸਿਰਫ਼ ਇੱਕ ਇੰਟਰਨੈੱਟ ਕਨੈਕਸ਼ਨ ਹੈ ਜੋ ਕਿ ਇੱਕ ਵਾਇਰਸ ਰਾਊਟਰ ਰਾਹੀਂ ਘਰ ਜਾਂ ਕਾਰੋਬਾਰ ਵਿੱਚ ਕਈ ਡਿਵਾਈਸਾਂ ਨਾਲ ਸਾਂਝਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਰਾਊਟਰ ਸਿੱਧਾ ਤੁਹਾਡੇ ਇੰਟਰਨੈੱਟ ਮਾਡਮ ਨਾਲ ਜੁੜਿਆ ਹੁੰਦਾ ਹੈ ਅਤੇ ਇੱਕ ਹੱਬ ਵਾਂਗ ਕੰਮ ਕਰਦਾ ਹੈ ਅਤੇ ਇੰਟਰਨੈੱਟ ਸਿਗਨਲ ਨੂੰ ਤੁਹਾਡੇ ਸਾਰੇ Wifi ਇਨਏਬਲਡ ਡਿਵਾਇਸਾਂ ਤੇ ਪ੍ਰਸਾਰਿਤ (broadcast) ਕਰਦਾ ਹੈ। ਇਹ ਤੁਹਾਨੂੰ ਉਦੋਂ ਤੱਕ ਇੰਟਰਨੈੱਟ ਨਾਲ ਜੋੜੇ ਰੱਖਦਾ ਹੈ ਜਦੋਂ ਤੱਕ ਤੁਸੀਂ ਆਪਣੇ ਨੈੱਟਵਰਕ ਦੇ ਕਵਰੇਜ ਖੇਤਰ ਵਿੱਚ ਰਹਿੰਦੇ ਹੋ।

1.4.4 ਕਲਾਊਡ ਕੰਪਿਊਟਿੰਗ (Cloud Computing) : ਸਭ ਤੋਂ ਵੱਡੇ ਰੁਝਾਨਾਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ ਕਲਾਊਡ ਕੰਪਿਊਟਿੰਗ। ਵੱਧ ਤੋਂ ਵੱਧ ਯੂਜ਼ਰਾਂ ਨੰ ਇਹ ਅਹਿਸਾਸ ਹੋ ਰਿਹਾ ਹੈ ਕਿ ਇੱਕ ਕੰਪਨੀ ਲਈ ਉਹਨਾਂ ਦੀ ਸਾਰੀ ਡਿਜੀਟਲ ਜਾਣਕਾਰੀ ਲਈ ਅਤੇ ਸਰੋਤਾਂ (resources) ਲਈ ਇੱਕ ਨਿਸ਼ਚਿਤ ਜਗਾ ਦਾ ਹੋਣਾ ਮਹੱਤਵਪੂਰਨ ਹੈ, ਅਤੇ ਇੱਕ ਚੰਗੀ ਤਰ੍ਹਾਂ ਸੁਰੱਖਿਅਤ ਜਗ੍ਹਾ ਹੋਣੀ ਜੋ ਹਰ

ਚੀਜ਼ ਦੀ ਦੇਖਭਾਲ ਅਤੇ ਸੂਚਨਾ ਸੁਰੱਖਿਅਤ ਰੱਖ ਸਕੇ। ਕਲਾਊਡ ਕੰਪਿਊਟਿੰਗ ਉਹਨਾਂ ਲਈ ਹੈ, ਜੋ ਆਪਣੇ ਕੰਮ ਨੂੰ ਬਿਹਤਰ ਬਣਾਉਣਾ ਚਾਹੁੰਦੇ ਹਨ ਅਤੇ ਇਸਨੂੰ ਡਿਜੀਟਲ ਸਪੇਸ ਵਿੱਚ ਵਧੇਰੇ ਕੁਸ਼ਲ ਬਣਾਉਣਾ ਚਾਹੁੰਦੇ ਹਨ। ਕਲਾਊਡ ਮਾਈਗ੍ਰੇਸ਼ਨ ਉਹਨਾਂ ਕਾਰੋਬਾਰਾਂ ਲਈ ਵੀ ਅਵਿਸ਼ਵਾਸਯੋਗ ਤੌਰ ਤੇ ਲਾਭਕਾਰੀ ਸਾਬਤ ਹੋਇਆ ਹੈ ਜੋ ਡਿਜੀਟਲ ਹੋਣ ਵੱਲ ਵਧਣਾ ਚਾਹੁੰਦੇ ਹਨ ਅਤੇ ਜੋ ਆਪਣੇ ਡਿਜੀਟਲ ਡਾਟਾ ਦੇ ਬਿਹਤਰ ਰਿਕਾਰਡਾਂ ਨੂੰ ਕਾਇਮ ਰੱਖਣਾ ਚਾਹੁੰਦੇ ਹਨ।

1.4.5 **ਫਾਇਰਵਾਲ:** ਫਾਇਰਵਾਲ ਇੱਕ ਕਿਸਮ ਦਾ ਬੈਰੀਅਰ ਹੈ ਜੋ ਕਿ ਨਿੱਜੀ ਨੈਟਵਰਕ ਅਤੇ ਪਬਲਿਕ ਇੰਟਰਨੈੱਟ ਦੇ ਵਿਚਕਾਰ ਹੁੰਦਾ ਹੈ। ਫਾਇਰਵਾਲ ਇੱਕ ਨੈੱਟਵਰਕ ਸੁਰੱਖਿਆ ਡਿਵਾਈਸ ਹੈ ਜੋ ਸੁਰੱਖਿਆ ਨਿਯਮਾਂ ਦੇ ਆਧਾਰ ਤੇ ਇਨਕਮਿੰਗ ਅਤੇ ਆਊਟਗੋਇੰਗ ਨੈਟਵਰਕ ਟ੍ਰੈਫਿਕ ਦੀ ਨਿਗਰਾਨੀ ਅਤੇ ਫਿਲਟਰ ਕਰਦਾ ਹੈ। ਇਸਦਾ ਮੁੱਖ ਉਦੇਸ਼ ਸੁਰੱਖਿਅਤ ਟ੍ਰੈਫਿਕ ਨੂੰ ਅੰਦਰ ਆਉਣ ਦੇਣਾ ਅਤੇ ਖਤਰਨਾਕ ਟ੍ਰੈਫਿਕ ਨੂੰ ਬਾਹਰ ਰੱਖਣਾ ਹੈ।

1.4.6 **ਈ -ਕਾਮਰਸ:** ਈ -ਕਾਮਰਸ ਵਸਤੂਆਂ ਅਤੇ ਸੇਵਾਵਾਂ ਨੂੰ ਖਰੀਦਣ ਅਤੇ ਵੇਚਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ((Process) ਹੈ, ਜਾਂ ਇੱਕ ਇਲੈਕਟ੍ਰਾਨਿਕ ਨੈਟਵਰਕ ਤੇ ਇੰਟਰਨੈਟ ਰਾਹੀਂ ਫੰਡਾਂ ਜਾਂ ਡਾਟਾ ਦਾ ਸੰਚਾਰ ਕਰਨਾ ਹੈ। ਈ-ਕਾਮਰਸ ਕਿਸੇ ਕਾਰੋਬਾਰ ਨੂੰ ਆਨ-ਲਾਈਨ ਚਲਾਉਣ ਦੇ ਸਾਰੇ ਪਹਿਲੂਆਂ ਦਾ ਹਵਾਲਾ ਦਿੰਦਾ ਹੈ। ਈ-ਕਾਮਰਸ ਨੇ ਮਾਰਕੀਟ ਕਿਵੇਂ ਕੰਮ ਕਰਦੀ ਹੈ, ਨੂੰ ਪੂਰੀ ਤਰ੍ਹਾਂ ਬਦਲ ਦਿੱਤਾ ਹੈ, ਜਿਸ ਨਾਲ ਦੁਨੀਆ ਵਿੱਚ ਕਿਤੇ ਵੀ ਚੀਜਾਂ/ਉਤਪਾਦਾਂ(ਪ੍ਰੋਡਕਟ) ਨੂੰ ਖੋਜਣਾ ਅਤੇ ਖਰੀਦਣਾ ਆਸਾਨ ਹੋ ਜਾਂਦਾ ਹੈ। ਅਸਲ ਵਿੱਚ ਈ-ਕਾਮਰਸ ਨੇ ਇੱਕ ਵਿਸ਼ਵਵਿਆਪੀ ਬਾਜ਼ਾਰ ਬਣਾਇਆ ਹੈ ਜੋ ਕਿ ਪੁਰਾਣੇ ਰਿਟੇਲ ਬਾਜ਼ਾਰ ਵਿੱਚ ਸੰਭਵ ਨਹੀਂ ਸੀ।

1.4.7 **ਆਨ-ਲਾਈਨ ਸ਼ਾਪਿੰਗ:** ਆਨ-ਲਾਈਨ ਸ਼ਾਪਿੰਗ ਦਾ ਮਤਲਬ ਹੈ ਵਸਤੂਆਂ ਅਤੇ ਸੇਵਾਵਾਂ ਦੀ ਆਨ-ਲਾਈਨ ਵਿਕਰੀ ਅਤੇ ਖਰੀਦਦਾਰੀ। ਆਨ-ਲਾਈਨ ਸ਼ਾਪਿੰਗ ਇੱਕ ਈ-ਕਾਮਰਸ ਦਾ ਹਿੱਸਾ ਹੈ ਜਿਸ ਵਿੱਚ ਕ੍ਰੈਡਿਟ ਕਾਰਡ ਦੁਆਰਾ ਵਿਕਰੇਤਾ ਦੀ ਵੈਬਸਾਈਟ ਤੇ ਆਈਟਮਾਂ ਨੂੰ ਖਰੀਦਣਾ ਅਤੇ ਆਈਟਮ ਨੂੰ ਤੁਹਾਡੇ ਘਰ ਪਹੁੰਚਾਉਣਾ ਸ਼ਾਮਲ ਹੈ। ਆਨ-ਲਾਈਨ ਸ਼ਾਪਿੰਗ ਵਿਚ ਆਨ-ਲਾਈਨ ਰਿਸਰਚ ਰਾਹੀਂ ਆਨ-ਲਾਈਨ ਵਸਤੂਆਂ ਦੀ ਖੋਜ ਕਰਨਾ ਵੀ ਸ਼ਾਮਲ ਹੈ।

1.4.8 **ਮੋਬਾਈਲ ਐਪਸ:** ਮੋਬਾਈਲ ਐਪਲੀਕੇਸ਼ਨਾਂ ਦੀ ਪ੍ਰਸਿੱਧੀ ਵਿੱਚ ਪਿਛਲੇ ਕੁਝ ਸਾਲਾਂ ਵਿੱਚ ਵਾਧਾ ਹੋਇਆ ਹੈ। ਦੁਨਿਆ ਭਰ ਵਿੱਚ ਬਰੈਂਡ ਅਤੇ ਕੰਪਨੀਆਂ ਅਜਿਹੇ ਤਰੀਕੇ ਲੱਭਣ ਦੀ ਕੋਸ਼ਿਸ਼ ਕਰ ਰਹੇ ਹਨ ਜਿਸ ਵਿੱਚ ਕੋਈ ਵੀ ਮੋਬਾਈਲ ਐਪ ਦੁਆਰਾ ਆਪਣੇ ਕੰਮ ਵਿੱਚ ਸੁਧਾਰ ਕਰ ਸਕਣ ਅਤੇ ਨਵੇਂ ਸਰੋਤਾਂ ਨੂੰ ਲਾਗੂ ਕਰਕੇ ਆਪਣੇ ਚਲਦੇ ਕੰਮਾਂ ਹੋਰ ਕੁਸ਼ਲਤਾ ਨਾਲ ਕਰ ਸਕਣ।

1.4.9 **ਬਿਗ-ਡਾਟਾ ਐਨਾਲਿਟਿਕਸ:** ਬਿਗ ਡਾਟਾ ਐਨਾਲਿਟਿਕਸ ਟਰੈਂਡ ਹੈ ਜੋ ਪਿਛਲੇ ਕੁਝ ਸਾਲਾ ਵਿੱਚ ਵਧਿਆ ਹੈ, ਅਤੇ ਇਹ ਉਹ ਚੀਜ਼ ਹੈ ਜੋ ਹੁਣ ਲਗਭਗ ਹਰ ਕਿਸਮ ਦੇ ਉਦਯੋਗ ਵਿਚ ਲਾਗੂ ਕੀਤੀ ਜਾ ਰਹੀ ਹੈ, ਜੋ ਵੱਡੇ ਪੈਮਾਨੇ ਤੇ ਉਤਪਾਦਨ ਅਤੇ ਨਿਰਮਾਣ ਅਤੇ ਸਪਲਾਈ ਕਰਦੇ ਹਨ। ਬਿਗ ਡਾਟਾ ਐਨਾਲਿਟਿਕਸ ਜਾਣਕਾਰੀ ਨੂੰ ਬਿਹਤਰ ਢੰਗ ਨਾਲ ਪ੍ਰੋਸੈਸ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ ਅਤੇ ਯੂਜਰਾਂ ਨੂੰ ਉਹਨਾਂ ਦੇ ਖੇਤਰਾਂ, ਜਿਹਨਾਂ ਨੂੰ ਉਹਨਾਂ ਨੇ ਵਿਕਸਤ ਕਰਨਾ ਹੈ, ਨੂੰ ਬਿਹਤਰ ਸਮਝਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ।

1.4.10 **ਆਟੋਮੇਸ਼ਨ:** ਆਟੋਮੇਸ਼ਨ ਇੱਕ ਅਜਿਹਾ ਟਰੈਂਡ ਹੈ ਜਿਸ ਨੇ ਵੱਡੇ ਪੱਧਰ ਤੇ ਨਿਰਮਾਣ ਅਤੇ ਉਤਪਾਦਨ ਯੂਨਿਟਾਂ ਨੂੰ ਪ੍ਰਭਾਵਿਤ ਕੀਤਾ ਹੈ ਅਤੇ ਇਸਦਾ ਆਉਣ ਵਾਲੇ ਸਾਲਾਂ ਵਿੱਚ ਸਿਰਫ ਹੋਰ ਵਧਣ ਦਾ ਅਨੁਮਾਨ ਹੈ। ਆਟੋਮੇਸ਼ਨ ਨੇ ਪ੍ਰਕਿਰਿਆਵਾਂ (Process) ਨੂੰ ਤੇਜ਼ ਰਫਤਾਰ ਨਾਲ ਕੰਮ ਕਰਨ ਦੇ ਯੋਗ ਬਣਾਇਆ ਹੈ ਅਤੇ ਕੰਪਨੀਆਂ ਨੂੰ ਵਧੇਰੇ ਕੁਸ਼ਲ ਤਰੀਕੇ ਨਾਲ ਆਪਣੇ ਟੀਚਿਆਂ ਤੱਕ ਪਹੁੰਚਣ ਦੇ ਯੋਗ ਬਣਾਇਆ ਹੈ।

1.4.11 **ਆਰਟੀਫਿਸ਼ੀਅਲ ਇੰਟੈਲੀਜੈਂਸ:** ਜਿੱਥੇ ਆਟੋਮੇਸ਼ਨ ਵੱਧ ਰਹੀ ਹੈ ਉੱਥੇ ਆਰਟੀਫਿਸ਼ੀਅਲ ਇੰਟੈਲੀਜੈਂਸ (AI) ਵੀ ਵੱਧ ਰਹੀ ਹੈ। ਸਮਾਰਟ ਟੈਕਨੋਲੋਜੀ ਜਿਸ ਵਿੱਚ ਸਮਾਰਟ ਮਸ਼ੀਨਾਂ ਸ਼ਾਮਲ ਹਨ ਜੋ ਆਰਟੀਫਿਸ਼ੀਅਲ ਇੰਟੈਲੀਜੈਂਸ ਜਾਂ ਆਟੋਮੇਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕਰਦੀਆਂ ਹਨ, ਵੱਧ ਰਹੀ ਹੈ, ਇੱਥੋਂ ਤੱਕ ਕਿ ਸਮਾਲ ਸਕੇਲ ਯੂਨਿਟਾਂ ਅਤੇ ਛੋਟੀਆਂ ਜਗਾਵਾਂ ਤੇ ਵੀ। ਸਮਾਰਟ ਟੈਕਨੋਲੋਜੀ ਦੀ ਵਰਤੋਂ ਦੇ ਨਤੀਜੇ ਵਜੋਂ, ਹੁਣ ਘਰ ਬਹੁਤ ਸਾਰੇ ਇੰਟਗ੍ਰੇਟਿਡ ਡਿਵਾਇਸਾਂ ਨਾਲ ਸਮਾਰਟ ਬਣ ਰਹੇ ਹਨ ਜੋ ਸਾਡੀ ਜ਼ਿੰਦਗੀ ਨੂੰ ਆਸਾਨ ਬਣਾਉਣ ਲਈ ਕੰਮ ਕਰਦੇ ਹਨ। ਅਲੈਕਸਾ (Alexa) ਵਰਗੇ ਸਧਾਰਨ ਟੂਲ ਘਰਾਂ ਦਾ ਜ਼ਰੂਰੀ ਹਿੱਸਾ ਬਣ ਗਏ ਹਨ ਅਤੇ ਇਸ ਵਿੱਚ ਵਾਧਾ ਹੋ ਰਿਹਾ ਹੈ। ਵਰਚੁਅਲ ਰਿਐਲਿਟੀ ਜਿਵੇਂ ਕਿ ਗੇਮਿੰਗ ਇੰਡਸਟਰੀ, ਵਿੰਚ ਨਵੀਂ ਟੈਕਨੋਲੋਜੀ ਦੇ ਕਾਰਨ,

ਪਹਿਲਾਂ ਹੀ ਪ੍ਰਸਿੱਧ ਹੋਣਾ ਸ਼ੁਰੂ ਹੋ ਗਿਆ ਹੈ, ਜੋ ਕਿ ਯੂਜ਼ਰ ਲਈ ਗੇਮਿੰਗ ਅਨੁਭਵ ਨੂੰ ਬਿਹਤਰ ਬਣਾਉਂਦਾ ਹੈ। ਆਗੁਮੈਂਟੇਂਡ ਰਿਐਲਿਟੀ ਇੱਕ ਹੋਰ ਪਹੁੰਚ ਹੈ। ਆਰਟੀਫਿਸ਼ੀਅਲ ਐਕਸਪੀਰੀਐਂਸ ਲਈ ਜੋ ਕਿ ਕਿਸੇ ਦੁਆਰਾ ਵੀ ਹੁਣ ਐਕਸੈਸ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਆਗੁਮੈਂਟੇਂਡ ਰਿਐਲਿਟੀ ਇਕ ਅਜਿਹੀ ਇਮੇਜ਼ ਨੂੰ ਯੂਜ਼ਰ ਦੇ ਅਸਲੀ ਦੁਨੀਆ ਦੇ ਵਿਊ ਤੇ ਸੂਪਰ ਇੰਮਪੋਜ਼ ਕਰਕੇ ਇੱਕ ਕੰਪੋਜ਼ਿਟ (ਸੰਯੁਕਤ) ਦ੍ਰਿਸ਼ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ।

1.4.12 IoT ਨੈਟਵਰਕਾਂ ਦਾ ਵਿਕਾਸ: ਇੰਟਰਨੈੱਟ ਆਫ ਥਿੰਗਜ਼ ਇੱਕ ਕੰਨਸੈਪਟ ਹੈ ਕਿ ਸਾਰੇ ਡਿਜੀਟਲ ਡਿਵਾਈਸਾਂ ਨੂੰ ਇੱਕ ਮਾਧਿਅਮ ਰਾਹੀਂ ਜੋ ਕੇ ਉਹਨਾਂ ਨੂੰ ਕੋਈ ਵੀ ਆਪਣੇ ਘਰ ਦੇ ਅੰਦਰ ਕੰਟਰੋਲ ਕਰਨ ਦੇ ਯੋਗ ਹੋਵੇਗਾ। ਇਸ ਕੰਨਸੈਪਟ ਦਾ ਵਰਨਣ ਕਰਨ ਲਈ ਚੀਜਾਂ ਜਿਹਨਾਂ ਤੇ ਸੈਂਸਰ ਲਗੇ ਹੋਣ, ਸਾਫਟਵੇਅਰ ਅਤੇ ਹੋਰ ਟੈਕਨੋਲੋਜੀਆਂ ਜਿਸ ਨਾਲ ਇਹ ਸਾਰੇ ਆਪਸ ਵਿੱਚ ਜੁੜ ਜਾਣ (connect) ਅਤੇ ਡਾਟਾ ਦਾ ਆਦਾਨ-ਪ੍ਰਦਾਨ ਹੋਰ ਡਿਵਾਈਸਾਂ ਅਤੇ ਸਿਸਟਮਾਂ ਤੇ ਇੰਟਰਨੈੱਟ ਦੇ ਜਰੀਏ ਕਰ ਸਕਣ।

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਕੰਪਿਊਟਰ ਇਕ ਇਲੈਕਟ੍ਰਾਨਿਕ ਡਿਵਾਈਸ ਹੈ ਜੋ ਕਿ ਡਾਟਾ ਨੂੰ ਲੈਂਦਾ ਹੈ, ਸਟੋਰ ਕਰਦਾ ਹੈ ਅਤੇ ਪ੍ਰੋਸੈਸ ਕਰਕੇ ਉਪਯੋਗੀ ਜਾਣਕਾਰੀ ਵਿਚ ਬਦਲਦਾ ਹੈ।
2. ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਕਈ ਕੰਪੋਨੇਂਟਸ ਤੋਂ ਮਿਲ ਕੇ ਬਣਦਾ ਹੈ, ਜਿਨ੍ਹਾਂ ਦੇ ਕੁਝ ਖਾਸ ਕੰਮ ਅਤੇ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹੁੰਦੀਆਂ ਹਨ ਤਾਂ ਕਿ ਉਹ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਸਪੋਰਟ ਕਰ ਸਕਣ।
3. ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਤਿੰਨ ਮੁੱਖ ਭਾਗ ਹਨ-ਇਨਪੁਟ ਯੂਨਿਟ, ਆਉਟਪੁੱਟ ਯੂਨਿਟ ਅਤੇ CPU ਸੈਂਟਰਲ ਪ੍ਰੋਸੈਸਿੰਗ ਯੂਨਿਟ।
4. ਉਹ ਡਿਵਾਈਸ ਜੋ ਯੂਜ਼ਰ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਹਦਾਇਤਾਂ ਦੇਣ ਜਾਂ ਡਾਟਾ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦੇ ਹਨ, ਉਹਨਾਂ ਨੂੰ ਇਨਪੁੱਟ ਯੂਨਿਟ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
5. ਉਹ ਡਿਵਾਇਸ ਜੋ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਯੂਜ਼ਰ ਨੂੰ ਜਾਣਕਾਰੀ, ਡਾਟਾ ਜਾਂ ਹਦਾਇਤਾਂ ਪ੍ਰਦਾਨ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦੇ ਹਨ ਉਨਾਂ ਨੂੰ ਆਉਟਪੁੱਟ ਯੂਨਿਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
6. CPU ਨੂੰ ਅੱਗੇ ਕੰਟਰੋਲ ਯੂਨਿਟ, ਅਰਥਮੈਟਿਕ ਅਤੇ ਲੋਜੀਕਲ ਯੂਨਿਟ ਅਤੇ ਮੈਮੋਰੀ ਯੂਨਿਟ ਵਿੱਚ ਵੰਡਿਆ ਗਿਆ ਹੈ।
7. ਸਾਫਟਵੇਅਰ ਨੂੰ ਸ਼੍ਰੇਣੀਬੱਧ ਕਰਨ ਲਈ ਇਕ ਤਰੀਕਾ ਹੈ ਜਿਸ ਵਿਚ ਸਾਫਟਵੇਅਰ ਦੇ ਵਿਕਾਸ ਜਾਂ ਡਵੈਲਪਮੈਂਟ ਨੂੰ ਦੇਖਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਦੂਜੀ ਸ਼੍ਰੇਣੀ ਲਈ ਸਾਫਟਵੇਅਰ ਦੇ ਕੰਮ ਕਰਨ ਦੇ ਤਰੀਕੇ ਤੇ ਨਿਰਭਰ ਕਰਦੀ ਹੈ।
8. ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਕੰਪਿਊਟਰ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੇ ਪ੍ਰੋਗਰਾਮ ਹੁੰਦੇ ਹਨ ਅਤੇ ਇਹ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ ਲਈ ਪਲੇਟਫਾਰਮ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।
9. ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ ਉਹ ਸਾਫਟਵੇਅਰ ਹਨ ਜੋ ਯੂਜ਼ਰ ਨੂੰ ਖਾਸ ਕੰਮ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਦੇ ਹਨ।
10. IT ਦੇ ਟਰੈਂਡ ਵਿਚ ਕੰਪਿਊਟਰ ਤੇ ਅਧਾਰਤ ਇਨਫੋਰਮੇਸ਼ਨ ਸਿਸਟਮ ਦਾ ਅਧਿਐਨ, ਡਿਜ਼ਾਇਨ, ਵਿਕਾਸ, ਐਪਲੀਕੇਸ਼ਨ, ਇੰਪਲੀਮੈਨਟੇਸ਼ਨ, ਸਪੋਰਟ ਜਾਂ ਪ੍ਰਬੰਧਨ ਸ਼ਾਮਲ ਹਨ।

ਅਭਿਆਸ

1. ਬਹੁਪੰਸਦੀ ਪ੍ਰਸ਼ਨ:

1. ਇੱਕ ਕੰਪਿਊਟਰ ਗਣਿਤਿਕ ਗਣਨਾਵਾਂ ਕਰਦੇ ਸਮੇਂ ਆਮ ਮਨੁੱਖਾਂ ਦੇ ਮੁਕਾਬਲੇ ਬਹੁਤ ਤੇਜ਼ ....ਨਾਲ ਕੰਮ ਕਰਦਾ ਹੈ।

   a. ਮੈਮੋਰੀ   b. ਰਫ਼ਤਾਰ   c. ਪ੍ਰੋਸੈਸਿੰਗ   d. ਕੰਟਰੋਲ

2. ਇੱਕ ਜੋਏਸਟਿਕ ਇੱਕ ....ਪੁਆਇੰਟਿੰਗ ਡਿਵਾਈਸ ਹੈ।

   a. ਇਨ-ਲਾਈਨ   b. ਇਨਸਿਕਿਉਰ   c. ਇਨਪੁੱਟ   d. ਆਊਟਪੁੱਟ

3. ਸਕੈਨਰ ਇਕ ਡਾਕੂਮੈਂਟ ਜਾਂ ਫੋਟੋ ਨੂੰ ਇੱਕ ....ਫ਼ਾਈਲ ਵਿਚ ਬਦਲਦਾ ਹੈ।

   a. ਪਰੋਪਰ   b. ਕੋਡਿਡ   c. ਡਿਜ਼ੀਟਲ   d. ਯੂਜਲੈਸ

4. ਇੱਕ ....ਇੱਕ ਡਿਸਪਲੇ ਡਿਵਾਈਸ ਹੈ ਜੋ ਯੂਜ਼ਰ ਨੂੰ ਉਹਨਾਂ ਦੀ ਉਂਗਲੀ ਜਾਂ ਸਟਾਈਲਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇੱਕ ਕੰਪਿਊਟਰ ਨਾਲ ਇੰਟਰੈਕਟ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ।

   a. ਟਚ ਸਕਰੀਨ   b. ਜੋਏ-ਸਟਿਕ   c. ਕੀਬੋਰਡ   d. ਪ੍ਰਿੰਟਰ

5. ਇੱਕ ਪ੍ਰਿੰਟਰ ਦਾ ਉਦੇਸ਼ ਡਾਕੂਮੈਂਟ ਦੀ ....ਤਿਆਰ ਕਰਨਾ ਹੈ।

   a. ਹਾਰਡ ਕੱਾਪੀ   b. ਸਾਫਟ ਕੱਾਪੀ   c. ਦੋਵੇਂ a ਅਤੇ b   d. ਇਨ੍ਹਾਂ ਵਿੱਚੋਂ ਕੋਈ ਨਹੀਂ

2. ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ:

1. ਡਾਟਾ ਜੋ ਅਸੀਂ ਕੰਪਿਊਟਰ ਵਿਚ ਦਾਖਲ ਕਰਦੇ ਹਾਂ ਉਸਨੂੰ ....ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
2. ਕੰਪਿਊਟਰ ਇੱਕ ਸੈਕਿੰਡ ਦੇ ....ਵਿੱਚ ਇੱਕ ਹਦਾਇਤ ਦੀ ਪ੍ਰਕਿਰਿਆ ਕਰ ਸਕਦਾ ਹੈ।
3. ਸਪੀਕਰ ਅਤੇ ਹੈੱਡਫੋਨ ਯੂਜ਼ਰ ਨੂੰ ....ਆਉਟਪੁੱਟ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।
4. ਇਕ ਸੈਕੰਡਰੀ ਸਟੋਰੇਜ ਡਿਵਾਈਸ ਇੱਕ ....ਸਟੋਰੇਜ ਡਿਵਾਈਸ ਹੈ।
5. ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਯੂਜ਼ਰ ਅਤੇ ਹਾਰਡਵੇਅਰ ਵਿਚਕਾਰ ਇੱਕ ....ਦੇ ਤੌਰ ਤੇ ਕੰਮ ਕਰਦਾ ਹੈ।

3. ਪੂਰੇ ਰੂਪ ਲਿਖੋ:

1. Wi-Fi
2. CPU
3. ROM
4. RAM
5. GPS
6. OS
7. GUI
8. CU
9. ALU
10. IoT
11. USB
12. BIOS
13. VDU

4. ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ:

1. ਕੰਪਿਊਟਰ ਕੀ ਹੈ ? ਇਸਦੇ ਚਾਰ ਮੁੱਖ ਕੰਮ ਕੀ ਹਨ ?
2. ਕੰਪਿਊਟਰ ਦੀਆਂ ਸੀਮਾਵਾਂ ਜਾਂ ਕਮੀਆਂ ਕੀ ਹਨ ?
3. ਪ੍ਰਾਈਮਰੀ ਸਟੋਰੇਜ ਕੀ ਹੈ ? ਵਿਆਖਿਆ ਕਰੋ।
4. ਸੈਕੰਡਰੀ ਸਟੋਰੇਜ ਕੀ ਹੈ ? ਵਿਆਖਿਆ ਕਰੋ।
5. ਯੂਟਿਲਿਟੀ ਸਾਫਟਵੇਅਰ ਕੀ ਹੈ ? ਵਿਆਖਿਆ ਕਰੋ ?
6. GPS ਦੀ ਪਰਿਭਾਸ਼ਾ ਦਿਓ।
7. ਬਲੂਟੁੱਥ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
8. Wi-Fi ਕੀ ਹੈ ?
9. ਕਲਾਊਡ ਕੰਪਿਊਟਿੰਗ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
10. ਆਰਟੀਫੀਸ਼ੀਅਲ ਇੰਟੈਲੀਜੈਂਸ ਕੀ ਹੈ ?
11. ਈ-ਕਾਮਰਸ ਅਤੇ ਆਨ-ਲਾਈਨ ਸ਼ਾਪਿੰਗ ਕੀ ਹੈ ?
12. ਫਾਈਰਵਾਲ ਕੀ ਹੈ ?

5. ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ:

1. ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
2. ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਕੰਪੋਨੇਂਟਸ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
3. ਇਨਪੁੱਟ ਯੂਨਿਟ ਕੀ ਹੈ ? ਕਿਸੇ ਚਾਰ ਇਨਪੁੱਟ ਡਿਵਾਈਸਾਂ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
4. ਆਉਟਪੁੱਟ ਡਿਵਾਈਸ ਕੀ ਹੈ ? ਕਿਸੇ ਚਾਰ ਆਉਟਪੁੱਟ ਡਿਵਾਈਸਾਂ ਬਾਰੇ ਦੱਸੋ।
5. ਸਾਫਟਵੇਅਰ ਕੀ ਹੈ ਅਤੇ ਇਹਨਾਂ ਨੂੰ ਕਿਵੇਂ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ? ਵਿਆਖਿਆ ਕਰੋ।
6. IT ਵਿੱਚ ਟਰੈਂਡ (ਰੁਝਾਨ) ਕੀ ਹਨ ? ਇਹਨਾਂ ਵਿੱਚੋਂ ਕਿਸੇ ਪੰਜ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
7. ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਕੀ ਹੈ ? ਵਿਆਖਿਆ ਕਰੋ।
8. ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦਾ ਬਲਾਕ ਡਾਇਗ੍ਰਾਮ ਬਣਾਓ ਅਤੇ ਇਸਦੇ ਵੱਖ-ਵੱਖ ਹਿੱਸਿਆ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।

ਨੰਬਰ ਸਿਸਟਮ ਅਤੇ ਲੌਜਿਕ ਗੇਟਸ
(Number System and Logic Gates)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

2.1 ਨੰਬਰ ਸਿਸਟਮ ਨਾਲ ਜਾਣ-ਪਛਾਣ

2.2 ਨੰਬਰ ਸਿਸਟਮ ਦੀਆਂ ਕਿਸਮਾਂ

2.3 ਨੰਬਰ ਸਿਸਟਮ ਦੇ ਵਿਚਕਾਰ ਬਦਲਾਅ

2.4 ਲੌਜਿਕ ਗੇਟਸ

2.5 NAND ਅਤੇ NOR ਗੇਟਸ ਨੂੰ ਯੂਨਿਵਰਸਲ ਗੇਟਸ ਦੇ ਤੌਰ ਤੇ ਸਿੱਧ ਕਰਨਾ

2.6 ਡਿਮੋਰਗਨ ਲਾਅ (Demorgan's law)

2.1 ਨੰਬਰ ਸਿਸਟਮ ਨਾਲ ਜਾਣ-ਪਛਾਣ (Introduction to Number System)

ਇੱਕ ਕੰਪਿਊਟਰ ਸਿਰਫ਼ ਨੰਬਰਾਂ ਨੂੰ ਸਮਝ ਸਕਦਾ ਹੈ ਜੋ ਅਸੀਂ ਲਿਖਦੇ ਹਾਂ ਭਾਵੇਂ ਅੱਖਰ ਜਾਂ ਸ਼ਬਦ, ਕੰਪਿਊਟਰ ਪਹਿਲਾਂ ਉਹਨਾਂ ਨੂੰ ਸੰਖਿਆਵਾਂ (numbers) ਵਿੱਚ ਅਨੁਵਾਦ ਕਰਦਾ ਹੈ। ਇੱਕ ਕੰਪਿਊਟਰ ਇੱਕ ਅੰਕ ਦੀ ਸਥਿਤੀ (positions) ਦੁਆਰਾ ਨੰਬਰ ਸਿਸਟਮ ਨੂੰ ਸਮਝ ਸਕਦਾ ਹੈ। ਨੰਬਰਾਂ ਨੂੰ ਦਰਸਾਉਣ ਅਤੇ ਕੰਮ ਕਰਨ ਦੀ ਤਕਨੀਕ ਨੂੰ ਨੰਬਰ ਸਿਸਟਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਨੰਬਰ ਸਿਸਟਮ ਦੀ ਵਰਤੋਂ ਕਿਸੇ ਚੀਜ਼ ਨੂੰ ਗਿਣਨ ਜਾਂ ਮਾਪਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਰੋਜ਼ਾਨਾ ਜੀਵਨ ਵਿੱਚ, ਅਸੀਂ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ (Decimal Number System) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਨੰਬਰ ਸਿਸਟਮ ਦੀਆਂ ਦੋ ਕਿਸਮਾਂ ਹਨ:

* ਨੋਨ-ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ (Non-positional number systems.)
* ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ (Positional number system.)

(ੳ) ਨੋਨ-ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ: ਇਸ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ, ਇੱਕ ਸੰਖਿਆ ਚਿੰਨ੍ਹ ਦਾ ਮੁੱਲ ਇਸਦੀ ਸਥਿਤੀ 'ਤੇ ਨਿਰਭਰ ਨਹੀਂ ਕਰਦਾ ਹੈ। ਇੱਕ ਨੋਨ-ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਸੀਮਤ ਸੰਖਿਆ ਵਿੱਚ ਪ੍ਰਤੀਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ । ਜਿਸ ਵਿੱਚ ਹਰੇਕ ਚਿੰਨ੍ਹ ਦਾ ਇੱਕ ਮੁੱਲ ਹੁੰਦਾ ਹੈ। ਹਾਲਾਂਕਿ, ਸੰਖਿਆ ਵਿੱਚ ਪ੍ਰਤੀਕ ਦੀ ਸਥਿਤੀ ਦਾ ਆਮ ਤੌਰ 'ਤੇ ਇਸਦੇ ਮੁੱਲ ਨਾਲ ਕੋਈ ਸਬੰਧ ਨਹੀਂ ਹੁੰਦਾ-ਹਰੇਕ ਚਿੰਨ੍ਹ ਦਾ ਮੁੱਲ ਨਿਸ਼ਚਿਤ ਹੁੰਦਾ ਹੈ ਰੋਮਨ ਨੰਬਰ ਸਿਸਟਮ ਨੋਨ-ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਦਾ ਇੱਕ ਵਧੀਆ ਉਦਾਹਰਣ ਹੈ ਇਸ ਨੰਬਰ ਪ੍ਰਣਾਲੀ ਵਿੱਚ ਚਿੰਨ੍ਹਾਂ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦਾ ਹੈ S={I,V,X,L,C,D,M}.

ਰੋਮਨ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਵੱਖ-ਵੱਖ ਚਿੰਨ੍ਹ ਦੇ ਮੁੱਲ

ਚਿੰਨ੍ਹ	I	V	X	L	C	D	M
ਮੁੱਲ	1	5	10	50	100	500	1000

ਚਿੱਤਰ 2.1

(ਅ) ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ : ਇਹਨਾਂ ਸੰਖਿਆ ਪ੍ਰਣਾਲੀਆਂ ਵਿੱਚ, ਇੱਕ ਸੰਖਿਆ ਚਿੰਨ੍ਹ ਦਾ ਮੁੱਲ ਉਸਦੀ ਸਥਿਤੀ 'ਤੇ ਨਿਰਭਰ ਕਰਦਾ ਹੈ। ਇੱਕ ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ, ਹਰੇਕ ਅੰਕ ਦਾ ਮੁੱਲ ਇਹ ਨਿਰਧਾਰਤ ਕਰਦਾ ਹੈ ਕਿ ਇਹ ਪੂਰੀ ਸੰਖਿਆ ਵਿੱਚ ਕਿਸ ਸਥਾਨ ਤੇ ਦਿਖਾਈ ਦਿੰਦਾ ਹੈ ਸਭ ਤੋਂ ਨੀਵਾਂ ਸਥਾਨ ਮੁੱਲ ਸਭ ਤੋਂ ਸੱਜੇ ਸਥਿਤੀ ਹੈ, ਅਤੇ ਖੱਬੇ ਪਾਸੇ ਦੀ ਹਰੇਕ ਲਗਾਤਾਰ ਸਥਿਤੀ ਦਾ ਉੱਚ ਸਥਾਨ ਮੁੱਲ ਹੈ। ਹਰੇਕ ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਦਾ ਇੱਕ ਅਧਾਰ (Base) ਜਾਂ ਰੇਡਿਕਸ (Radix) ਹੁੰਦਾ ਹੈ ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਦੀ ਉਦਾਹਰਨ ਹੈ: ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ

Decimal Position Notation

← Continue					
10	10	10	10	10	Radix
4	3	2	1	0	Position
10^4	10^3	10^2	10^1	10^0	Calculation
10,000	1,000	100	10	1	Positional Value

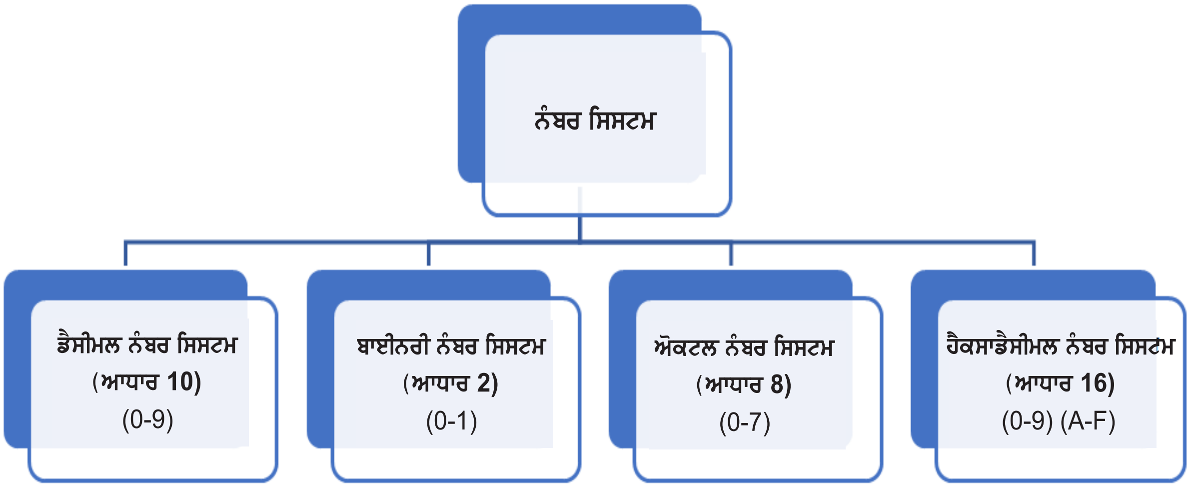
ਚਿੱਤਰ 2.2

ਹੋਰ ਉਦਾਹਰਣਾਂ ਹਨ ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ (ਰੇਡੀਕਸ 2 ਦੇ ਨਾਲ), ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ (ਰੇਡੀਕਸ 8 ਦੇ ਨਾਲ), ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ (ਰੈਡੀਕਸ 16 ਦੇ ਨਾਲ)

2.2 ਨੰਬਰ ਸਿਸਟਮ ਦੀਆਂ ਕਿਸਮਾਂ

ਚਾਰ ਕਿਸਮਾਂ ਦੇ ਨੰਬਰ ਸਿਸਟਮ ਹਨ ਜੋ ਇੱਕ ਕੰਪਿਊਟਰ ਦਾ ਸਮਰਥਨ ਕਰਦਾ ਹੈ। ਉਹ-

1. ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ (Binary Number System)
2. ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ (Octal Number System)
3. ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ (Decimal Number System)
4. ਹੈਕਸਾਡੈਸੀਮਲ (Hexadecimal Number System)



ਚਿੱਤਰ 2.3

ਅਧਾਰ/ਰੇਡੀਕਸ : ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਅੰਕਾਂ ਦੀ ਕੁੱਲ ਸੰਖਿਆ ਨੂੰ ਅਧਾਰ ਜਾਂ ਰੇਡੀਕਸ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ਨੰ .	ਨੰਬਰ ਸਿਸਟਮ	ਆਧਾਰ	ਵਰਣਨ	ਉਦਾਹਰਨ
1	ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ	10	• ਵਰਤੇ ਗਏ ਅੰਕ: 0 ਤੋਂ 9	27
2	ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ	2	• ਵਰਤੇ ਗਏ ਅੰਕ: 0, 1	11011
3	ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ	8	• ਵਰਤੇ ਗਏ ਅੰਕ: 0 ਤੋਂ 7	33
4	ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ	16	• ਵਰਤੇ ਗਏ ਅੰਕ: 0 ਤੋਂ 9 • ਵਰਤੇ ਗਏ ਅੱਖਰ: A- F	1 A

ਚਿੱਤਰ 2.4

ਵੱਖ-ਵੱਖ ਸੰਖਿਆ ਪ੍ਰਣਾਲੀ ਵਿੱਚੋਂ ਸੰਖਿਆਵਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ

ਡੈਸੀਮਲ	ਬਾਈਨਰੀ	ਔਕਟਲ	ਹੈਕਸਾਡੈਸੀਮਲ
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

ਚਿੱਤਰ 2.5

2.2.1 ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ (DECIMAL NUMBER SYSTEM)

ਇੱਕ ਨੰਬਰ ਸਿਸਟਮ ਜਿਸ ਵਿੱਚ 10 ਵੱਖ-ਵੱਖ ਚਿੰਨ੍ਹ 0,1,2,3,4,5,6,7,8 ਅਤੇ 9 ਹੁੰਦੇ ਹਨ, ਨੂੰ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ (Decimal number system) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਨੰਬਰ ਸਿਸਟਮ ਦਾ ਅਧਾਰ ਜਾਂ ਰੈਡਿਕਸ 10 ਹੈ। ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਕਿਸੇ ਵੀ ਸੰਖਿਆ ਨੂੰ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ (Decimal number system) ਵਿੱਚ ਵਾਪਸ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਨੰਬਰ ਸਿਸਟਮ ਮਨੁੱਖ ਦੁਆਰਾ ਵਰਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਇੱਕ ਪੋਜ਼ੀਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਹੈ।

ਹਰ ਅੰਕ ਦਾ ਉਹਨਾਂ ਦੀ ਸਥਿਤੀ ਅਨੁਸਾਰ ਆਪਣਾ ਸਥਾਨ ਮੁੱਲ ਹੁੰਦਾ ਹੈ। ਜਦੋਂ ਅਸੀਂ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਪਾਸੇ ਜਾਂਦੇ ਹਾਂ, ਤਾਂ ਸੰਖਿਆ ਦਾ ਮੁੱਲ 10 ਵਧ ਜਾਂਦਾ ਹੈ। ਦਸ਼ਮਲਵ ਦੇ ਖੱਬੇ ਪਾਸੇ ਦੀ ਸਥਿਤੀ ਇੱਕ, ਦਸਾਂ, ਸੈਂਕੜੇ, ਹਜ਼ਾਰਾਂ, ਅਤੇ ਇਸ ਤਰ੍ਹਾਂ ਦੀਆਂ ਇਕਾਈਆਂ ਲਈ ਹੁੰਦੀ ਹੈ।

ਉਦਾਹਰਨ ਲਈ - ਦਸ਼ਮਲਵ ਸੰਖਿਆ 8432 ਵਿੱਚ ਯੂਨਿਟ ਪੋਜੀਸ਼ਨ ਵਿੱਚ 2, ਦਸ ਦੀ ਸਥਿਤੀ ਵਿੱਚ 3, ਸੈਂਕੜਿਆਂ ਦੀ ਸਥਿਤੀ ਵਿੱਚ 4 ਅਤੇ ਹਜ਼ਾਰਾਂ ਦੀ ਸਥਿਤੀ ਵਿੱਚ 8 ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ।

$$(8x10^3)+(4x10^2)+(3x10^1)+(2x10^0)$$

$$(8x1000)+(4x100)+(3x10)+(2x1)$$

$$8000+400+30+2$$

$$8432$$

2.2.2 ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ (BINARY NUMBER SYSTEM)

ਇੱਕ ਨੰਬਰ ਸਿਸਟਮ ਜਿਸ ਵਿੱਚ ਦੋ ਵੱਖ-ਵੱਖ ਚਿੰਨ੍ਹ 0 ਅਤੇ 1 ਹੁੰਦੇ ਹਨ, ਨੂੰ ਬਾਇਨਰੀ ਨੰਬਰ ਸਿਸਟਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਨੰਬਰ ਸਿਸਟਮ ਦਾ ਅਧਾਰ ਜਾਂ ਰੈਡਿਕਸ 2 ਹੈ। ਇੱਕ ਡੈਸੀਮਲ ਨੰਬਰ ਨੂੰ ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਨੰਬਰ ਸਿਸਟਮ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੁਆਰਾ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਇੱਕ ਪੁਜ਼ੀਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਹੈ।

ਬਿੱਟ ਦਾ ਅਰਥ ਬਾਈਨਰੀ ਅੰਕ।ਹੈ। ਇਹ ਅੰਕ 0 ਅਤੇ 1 ਹਨ। ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ ਇਹਨਾਂ ਅੰਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ। ਇੱਕ ਡਿਜੀਟਲ ਕੰਪਿਊਟਰ ਹਰ ਚੀਜ਼ ਨੂੰ ਬਾਈਨਰੀ ਅੰਕਾਂ ਦੇ ਰੂਪ ਵਿੱਚ ਸਟੋਰ ਕਰਦਾ ਹੈ। 8 ਬਿੱਟਾਂ ਦੇ ਸੁਮੇਲ ਨੂੰ ਬਾਈਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਇੱਕ ਬਾਈਟ 8 ਬਿੱਟ ਦੇ ਬਰਾਬਰ ਹੈ। ਇਹ ਮੈਮਰੀ ਦੀ ਸਭ ਤੋਂ ਛੋਟੀ ਇਕਾਈ ਹੈ।

ਇੱਕ ਡਿਵਾਈਸ ਵਿੱਚ ਸਟੋਰੇਜ ਦੀ ਸਭ ਤੋਂ ਬੁਨਿਆਦੀ ਇਕਾਈ ਨੂੰ ਇੱਕ ਬਿੱਟ ਦੁਆਰਾ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ। ਕੰਪਿਊਟਰ ਕਿਸੇ ਵੀ ਕਿਸਮ ਦੀ ਜਾਣਕਾਰੀ ਦਿਖਾਉਣ ਲਈ ਬਿੱਟ ਨੂੰ ਵਰਤਦਾ ਹੈ।

0 ਅਤੇ 1 ਇਹ ਦੋ ਚਿੰਨ੍ਹ ਹਰ ਸੰਖਿਆ ਨੂੰ ਦਰਸਾਉਂਦੇ ਹਨ 0 ਘੱਟ ਦਰ ਲਈ ਹੈ ਜਦੋਂ ਕਿ 1 ਉੱਚ ਦਰ ਲਈ ਹੈ। ਸੰਖਿਆਵਾਂ ਇੱਥੇ ਇੱਕ ਵਿਅਕਤੀਗਤ ਇਕਾਈ ਦੇ ਰੂਪ ਵਿੱਚ ਨਹੀਂ ਹਨ, ਪਰ 1 ਅਤੇ 0 ਦੇ ਸਮੂਹਾਂ ਦੇ ਬਣੇ ਹੋਏ ਹਨ। ਹਰੇਕ ਬਾਈਨਰੀ ਅੰਕ ਇੱਕ ਬਿੱਟ ਹੁੰਦਾ ਹੈ ਅਤੇ ਸਥਾਨ ਮੁੱਲ ਖੱਬੇ ਤੋਂ ਸੱਜੇ ਦੋ ਦੀਆਂ ਵਧਦੀਆਂ ਸ਼ਕਤੀਆਂ ਹਨ। ਸਭ ਤੋਂ ਘੱਟ ਮਹੱਤਵਪੂਰਨ ਬਿੱਟ (LSB) ਸੱਜੇ ਪਾਸੇ ਹੈ ਜਦੋਂ ਕਿ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਬਿੱਟ (MSB) ਖੱਬੇ ਪਾਸੇ ਹੈ। ਉਦਾਹਰਨ ਲਈ - 11010

2.2.3 ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ (OCTAL NUMBER SYSTEM)

ਔਕਟਲ ਸ਼ਬਦ ਲਾਤੀਨੀ ਸ਼ਬਦ OCT ਤੋਂ ਆਇਆ ਹੈ ਜਿਸਦਾ ਅਰਥ ਅੱਠ ਹੈ। ਇਹ 0 ਤੋਂ 7 ਤੱਕ ਅੰਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ। ਇੱਕ ਨੰਬਰ ਸਿਸਟਮ ਜਿਸ ਵਿੱਚ 8 ਵੱਖ ਵੱਖ ਚਿੰਨ੍ਹ 0, 1,2,3,4,5,6 ਅਤੇ 7 ਹੁੰਦੇ ਹਨ, ਨੂੰ ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਨੰਬਰ ਸਿਸਟਮ ਦਾ ਅਧਾਰ ਜਾਂ ਰੈਡਿਕਸ 8 ਹੈ। ਡੈਸੀਮਲ ਨੰਬਰ ਨੂੰ ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ ਇਹ ਨੰਬਰ ਸਿਸਟਮ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੁਆਰਾ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਇੱਕ ਪੁਜ਼ੀਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਹੈ।

ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ ਲਈ ਫਾਰਮੂਲਾ 2^3=8 ਹੈ। ਇਹ ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ ਨੂੰ ਦਰਸਾਉਣ ਲਈ 3 ਬਾਈਨਰੀ ਅੰਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ। ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ, ਜਾਂ ਸੰਖੇਪ ਵਿੱਚ OCT, ਅਧਾਰ-8 ਨੰਬਰ ਸਿਸਟਮ ਹੈ, ਅਤੇ 0 ਤੋਂ 7 ਅੰਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ, ਮਤਲਬ ਕਿ 10 ਔਕਟਲ ਵਿੱਚ ਅੱਠ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ ਅਤੇ 100 ਔਕਾਲ ਵਿੱਚ 64 ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

ਕੰਪਿਊਟਿੰਗ ਵਾਤਾਵਰਨ ਵਿੱਚ, ਇਹ ਆਮ ਤੌਰ ਤੇ ਬਾਈਨਰੀ ਅੰਕਾਂ ਨੂੰ ਤਿੰਨਾਂ ਵਿੱਚ ਸਮੂਹਿਕ ਕਰਕੇ ਬਾਈਨਰੀ ਸੰਖਿਆਵਾਂ ਦੀ ਇੱਕ ਛੋਟੀ ਪ੍ਰਤੀਨਿਧਤਾ ਵਜੋਂ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਲੀਨਕਸ ਜਾਂ UNIX ਵਿੱਚ Chmod ਕਮਾਂਡ ਫਾਈਲ ਪਰਮਿਸ਼ਨ (Permission) ਨਿਰਧਾਰਤ ਕਰਨ ਲਈ octal ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ

2.2.4 ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ (HEXADECIMAL NUMBER SYSTEM)

ਹੈਕਸਾਡੈਸੀਮਲ ਨੂੰ ਦੋ ਸ਼ਬਦਾਂ ਹੈਕਸਾ (HEXA) ਅਤੇ ਡੇਸੀ (DECI) ਵਿੱਚ ਵੰਡਿਆ ਗਿਆ ਹੈ, ਜਿੱਥੇ 'ਹੈਕਸਾ' ਦਾ ਅਰਥ ਹੈ 6 ਅਤੇ 'ਡੇਸੀ' ਦਾ ਅਰਥ ਹੈ 10 ।ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਨੂੰ 0 - 9 ਅਤੇ A-F ਅੰਕਾਂ ਵਿੱਚ ਦਰਸਾਇਆ ਗਿਆ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਪਹਿਲੇ 9 ਅੰਕਾਂ ਨੂੰ (0-9) ਸੰਖਿਆਵਾਂ ਦੇ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਕਿ ਅਗਲੇ 6 ਅੰਕਾਂ ਨੂੰ A-F ਤੋਂ ਚਿੰਨ੍ਹਾਂ ਵਜੋਂ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਲਈ, 16 ਅੰਕ 0, 1, 2, 3, 4, 5, 6, 7, 8,9 ਅਤੇ A, B, C, D, E, F ਹਨ। ਉਦਾਹਰਨ ਲਈ: 8A316, 7F16 ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਹਨ।

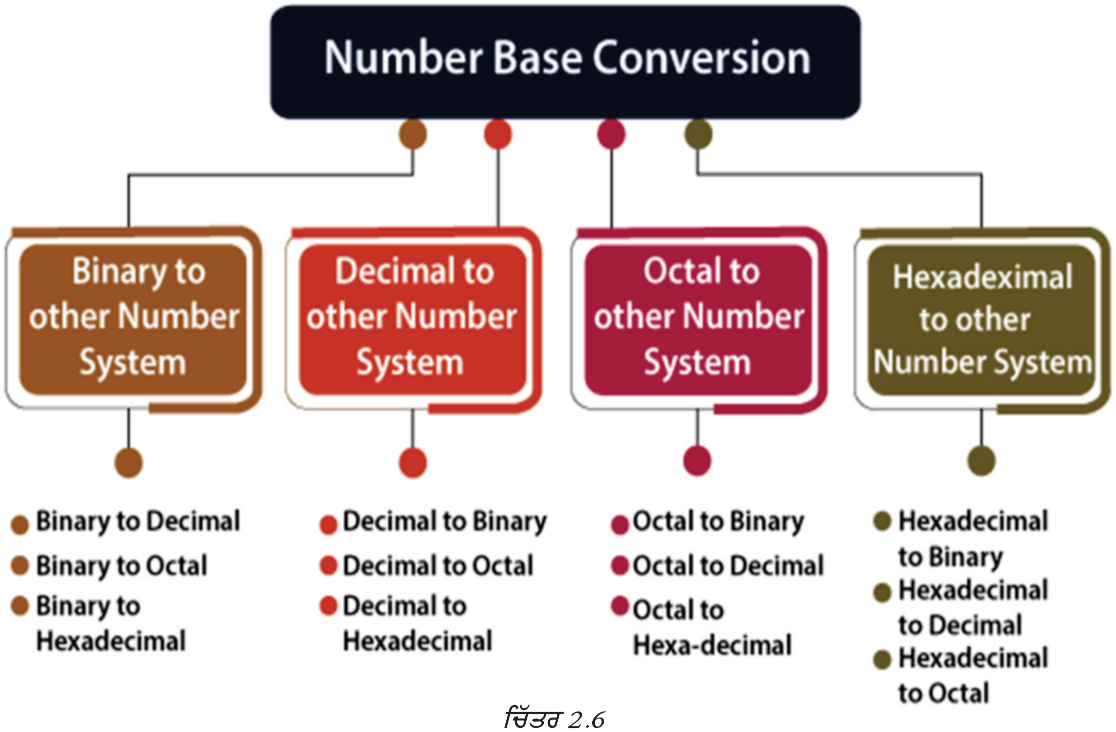
ਇੱਕ ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਨੂੰ ਇੱਕ ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਹਰੇਕ ਅੰਕ ਦਾ ਭਾਰ 16 ਹੁੰਦਾ ਹੈ।

2.3 ਨੰਬਰ ਸਿਸਟਮ ਦੇ ਵਿੱਚਕਾਰ ਬਦਲਾਅ (CONVERSION OF NUMBER SYSTEM):

ਬਾਈਨਰੀ ਸਿੱਖਿਆ ਲਈ ਤਿੰਨ ਰੂਪਾਂਤਰ ਸੰਭਵ ਹਨ, ਜਿਵੇਂ ਕਿ, ਬਾਈਨਰੀ ਤੋਂ ਦਸ਼ਮਲਵ, ਬਾਈਨਰੀ ਤੋਂ ਔਕਟਲ, ਅਤੇ ਬਾਈਨਰੀ ਤੋਂ ਹੈਕਸਾਡੈਸੀਮਲ ਹਨ। ਸਾਡੇ ਪਿਛਲੇ ਭਾਗ ਵਿੱਚ, ਅਸੀਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਨੰਬਰ ਸਿਸਟਮ ਜਿਵੇਂ ਕਿ ਬਾਇਨਰੀ, ਡੈਸੀਮਲ, ਔਕਟਲ ਅਤੇ ਹੈਕਸਾਡੈਸੀਮਲ ਬਾਰੇ ਸਿੱਖਿਆ।

ਜਿਵੇਂ ਕਿ ਸਾਡੇ ਕੋਲ ਚਾਰ ਕਿਸਮ ਦੇ ਨੰਬਰ ਸਿਸਟਮ ਹਨ ਇਸਲਈ ਹਰ ਇੱਕ ਨੂੰ ਬਾਕੀ ਤਿੰਨ ਪ੍ਰਣਾਲੀਆਂ ਵਿੱਚ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਹੇਠਾਂ ਦਿੱਤੇ ਪਰਿਵਰਤਨ ਸੰਭਵ ਹਨ

1. ਬਾਈਨਰੀ ਤੋਂ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ (Binary to other Number Systems.)
2. ਡੈਸੀਮਲ ਤੋਂ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ (Decimal to other Number Systems.)
3. ਔਕਟਲ ਤੋਂ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ (Octal to other Number Systems.)
4. ਹੈਕਸਾਡੈਸੀਮਲ ਤੋਂ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ (Hexadecimal to other Number Systems.)



ਚਿੱਤਰ 2.6

1. ਬਾਈਨਰੀ ਤੋਂ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ (Binary to other Number Systems)

ਬਾਈਨਰੀ ਸੰਖਿਆ ਲਈ ਤਿੰਨ ਰੂਪਾਂਤਰ ਸੰਭਵ ਹਨ, ਜਿਵੇਂ ਕਿ, ਬਾਈਨਰੀ ਤੋਂ ਡੈਸੀਮਲ, ਬਾਈਨਰੀ ਤੋਂ ਔਕਟਲ, ਅਤੇ ਬਾਈਨਰੀ ਤੋਂ ਹੈਕਸਾਡੈਸੀਮਲ/ਬਾਈਨਰੀ ਸੰਖਿਆ ਨੂੰ ਡੈਸੀਮਲ ਵਿੱਚ ਬਦਲਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਬਾਕੀ ਬਚੀਆਂ ਸੰਖਿਆਵਾਂ ਤੋਂ ਵੱਖਰੀ ਹੁੰਦੀ ਹੈ।

(ੳ) ਬਾਈਨਰੀ ਤੋਂ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Binary to Decimal Conversion)

ਇਸ ਪ੍ਰਕਿਰਿਆ ਵਿੱਚ ਅਸੀਂ ਬਾਈਨਰੀ ਸੰਖਿਆ ਦੇ ਬਿੱਟਾਂ ਨੂੰ ਇਸਦੇ ਪੋਜੀਸ਼ਨਲ ਵੇਟਸ ਅਨੁਸਾਰ ਗੁਣਾ ਕਰਨਾ ਸ਼ੁਰੂ ਕਰਦੇ ਹਾਂ, ਅਤੇ ਅੰਤ ਵਿੱਚ, ਅਸੀਂ ਉਹਨਾਂ ਸਾਰੇ ਉਤਪਾਦਾਂ (Products) ਨੂੰ ਜੋੜਦੇ ਹਾਂ।

ਆਉ ਇਹ ਸਮਝਣ ਲਈ ਇੱਕ ਉਦਾਹਰਣ ਲੈਂਦੇ ਹਾਂ ਕਿ ਬਾਈਨਰੀ ਤੋਂ ਦਸ਼ਮਲਵ ਤੱਕ ਪਰਿਵਰਤਨ ਕਿਵੇਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

Example 1: $(1101.001)_2$

ਅਸੀਂ (10110.001), ਦੇ ਹਰੇਕ ਬਿੱਟ ਨੂੰ ਇਸਦੇ ਸੰਬੰਧਿਤ ਸਥਿਤੀ ਦੇ ਭਾਰ ਨਾਲ ਗੁਣਾ ਕੀਤਾ, ਅਤੇ ਅੰਤ ਵਿੱਚ ਅਸੀਂ ਸਾਰੇ ਬਿੱਟਾਂ ਦੇ ਉਤਪਾਦ (Product) ਨੂੰ ਇਸਦੇ ਭਾਰ ਨਾਲ ਜੋੜਦੇ ਹਾਂ।

$$\mathbf{(1101.001)_2} = (1\times2^3)+(1\times2^2)+(0\times2^1)+(1\times2^0)+(0\times2^{-1})+(0\times2^{-2})+(1\times2^{-3})$$

$$= (1\times8) + (1\times4) + (0\times2) + (1\times1) + \left(0\times\frac{1}{2}\right) + \left(0\times\frac{1}{4}\right) + \left(1\times\frac{1}{8}\right)$$

$$=8+4+0+1+0.125$$

$$= (13.125)_{10}$$

(ਅ) ਬਾਈਨਰੀ ਤੋਂ ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Binary to Octal Conversion)

ਬਾਈਨਰੀ ਦਾ ਅਧਾਰ ਸੰਖਿਆ 2 ਹੈ ਅਤੇ ਔਕਟਲ 8 ਹੈ। ਇੱਕ ਬਾਈਨਰੀ ਸੰਖਿਆ ਵਿੱਚ, ਤਿੰਨ ਬਿੱਟਾਂ ਦਾ ਜੋੜਾ ਇੱਕ ਔਕਟਲ ਅੰਕ ਦੇ ਬਰਾਬਰ ਹੁੰਦਾ ਹੈ। ਇੱਕ ਬਾਈਨਰੀ ਸੰਖਿਆ ਨੂੰ ਇੱਕ ਔਕਟਲ ਸੰਖਿਆ ਵਿੱਚ ਬਦਲਣ ਲਈ ਸਿਰਫ ਦੋ ਕਦਮ ਹਨ ਜੋ ਕਿ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹਨ:

ਪਹਿਲੇ ਪੜਾਅ ਵਿੱਚ, ਸਾਨੂੰ ਬਾਈਨਰੀ ਬਿੰਦੂ ਦੇ ਦੋਵੇਂ ਪਾਸੇ ਤਿੰਨ ਬਿੱਟਾਂ ਦੇ ਜੋੜੇ ਬਣਾਉਣੇ ਪੈਣਗੇ। ਜੇਕਰ ਤਿੰਨ ਬਿੱਟਾਂ ਦੇ ਜੋੜੇ ਵਿੱਚ ਇੱਕ ਜਾਂ ਦੋ ਬਿੱਟ ਘੱਟ ਹੋਣਗੇ, ਤਾਂ ਅਸੀਂ ਦੋਨਾਂ ਸਾਈਡਾਂ ਦੇ ਅੰਤ 'ਤੇ ਲੋੜੀਦੇ ਜ਼ੀਰੋ ਜੋੜਦੇ ਹਾਂ। ਦੂਜੇ ਪੜਾਅ ਵਿੱਚ, ਅਸੀਂ ਹਰੇਕ ਜੋੜੇ ਦੇ ਅਨੁਸਾਰ ਔਕਟਲ ਅੰਕਾਂ ਨੂੰ ਲਿਖਦੇ ਹਾਂ।

Example 1: $(10101.0011)_2$

1. ਪਹਿਲਾਂ, ਅਸੀਂ ਬਾਈਨਰੀ ਬਿੰਦੂ ਦੇ ਦੋਵੇਂ ਪਾਸੇ ਤਿੰਨ ਬਿੱਟਾਂ ਦੇ ਜੋੜੇ ਬਣਾਉਂਦੇ ਹਾਂ।

10 101.001 1

ਬਾਈਨਰੀ ਪੁਆਇੰਟ ਦੇ ਸੱਜੇ ਪਾਸੇ, ਆਖਰੀ ਜੜੇ ਵਿੱਚ ਸਿਰਫ਼ ਇੱਕ ਬਿੱਟ ਹੈ, ਇਸ ਨੂੰ ਤਿੰਨ ਬਿੱਟਾਂ ਦਾ ਪੂਰਾ ਜੋੜਾ ਬਣਾਉਣ ਲਈ, ਅਸੀਂ ਸਿਰੇ ਵਾਲੇ ਪਾਸੇ ਦੋ ਜ਼ੀਰੋ ਜੋੜਦੇ ਹਾਂ, ਅਤੇ ਬਾਈਨਰੀ ਪਵਾਇੰਟ ਦੇ ਖੱਬੇ ਵਾਲੇ ਪਾਸੇ, ਆਖਰੀ ਜੋੜੇ ਵਿੱਚ ਦੋ ਬਿੱਟ ਹੁੰਦੇ ਹਨ ਇਸਲਈ ਅਸੀਂ ਸਿਰੇ ਵਾਲੇ ਪਾਸੇ ਇੱਕ ਜ਼ੀਰੋ ਜੋੜਾਂਗੇ।

010 101.001 100

2. ਫਿਰ ਔਕਟਲ ਅੰਕ ਲਿਖੋ, ਜੋ ਹਰੇਕ ਜੋੜੇ ਨਾਲ ਮੇਲ ਖਾਂਦੇ ਹਨ।

$\mathbf{(10101.0011)_2=(25.14)_8}$

(ਦ) ਬਾਈਨਰੀ ਤੋਂ ਹੈਕਸਾ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿਚ ਬਦਲਾਅ (Binary to Hexadecimal Conversion)

ਬਾਈਨਰੀ ਦਾ ਅਧਾਰ 2 ਹੈ ਅਤੇ ਹੈਕਸਾਡੈਸੀਮਲ ਦਾ 16 ਹੈ। ਇੱਕ ਬਾਈਨਰੀ ਸੰਖਿਆ ਵਿੱਚ, ਚਾਰ ਬਿੱਟਾਂ ਦਾ ਜੋੜਾ ਇੱਕ ਹੈਕਸਾਡੈਸੀਮਲ ਅੰਕ ਦੇ ਬਰਾਬਰ ਹੁੰਦਾ ਹੈ। ਇੱਕ ਬਾਈਨਰੀ ਨੰਬਰ ਨੂੰ ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਵਿੱਚ ਬਦਲਣ ਲਈ ਸਿਰਫ ਦੋ ਕਦਮ ਹਨ।

ਪਹਿਲੇ ਪੜਾਅ ਵਿੱਚ, ਸਾਨੂੰ ਬਾਈਨਰੀ ਬਿੰਦੂ ਦੇ ਦੋਵੇਂ ਪਾਸੇ ਚਾਰ ਬਿੱਟਾਂ ਦੇ ਜੜੇ ਬਣਾਉਣੇ ਪੈਣਗੇ। ਜੇਕਰ ਚਾਰ ਬਿੱਟਾਂ ਦੇ ਗਰੁੱਪ ਵਿੱਚ ਇੱਕ, ਦੋ, ਜਾਂ ਤਿੰਨ ਬਿੱਟ ਬਚੇ ਹਨ, ਤਾਂ ਦੋਨਾਂ ਪਾਸਿਆਂ ਦੇ ਅੰਤ 'ਤੇ ਲੋੜੀਂਦੇ ਜ਼ੀਰੋ ਜੋੜਦੇ ਹਾਂ।

ਦੂਜੇ ਪੜਾਅ ਵਿੱਚ, ਅਸੀਂ ਹਰੇਕ ਜੋੜੇ ਦੇ ਅਨੁਸਾਰ ਹੈਕਸਾਡੈਸੀਮਲ ਅੰਕਾਂ ਨੂੰ ਲਿਖਦੇ ਹਾਂ।

Example 1: $(110101011.0011)_2$

1. ਪਹਿਲਾਂ, ਅਸੀਂ ਬਾਈਨਰੀ ਬਿੰਦੂ ਦੇ ਦੋਵੇਂ ਪਾਸੇ ਚਾਰ ਬਿੱਟਾਂ ਦੇ ਜੜੇ ਬਣਾਉਂਦੇ ਹਾਂ।

1 1010 1011.0011

ਬਾਈਨਰੀ ਪੁਆਇੰਟ ਦੇ ਖੱਬੇ ਪਾਸੇ, ਪਹਿਲੇ ਗਰੁੱਪ ਵਿੱਚ ਇੱਕ ਬਿੱਟ ਹੈ। ਇਸ ਨੂੰ ਚਾਰ ਬਿੱਟਾਂ ਦਾ ਪੂਰਾ ਜੋੜਾ ਬਣਾਉਣ ਲਈ, ਸਿਰੇ ਵਾਲੇ ਪਾਸੇ ਤਿੰਨ ਜ਼ੀਰੋ ਜੋੜੋ।

0001 1010 1011.0011

2. ਫਿਰ, ਅਸੀਂ ਹੈਕਸਾਡੈਸੀਮਲ ਅੰਕ ਲਿਖਦੇ ਹਾਂ, ਜੋ ਹਰੇਕ ਜੋੜੇ ਨਾਲ ਮੇਲ ਖਾਂਦੇ ਹਨ।

$\mathbf{(110101011.0011)_2=(1AB.3)_{16}}$

2. ਡੈਸੀਮਲ ਤੋਂ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ (Decimal to other Number System)

ਡੈਸੀਮਲ ਨੰਬਰ ਇੱਕ ਪੂਰਨ ਅੰਕ ਜਾਂ ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਪੂਰਨ ਅੰਕ ਹੋ ਸਕਦਾ ਹੈ। ਜਦੋਂ ਡੈਸੀਮਲ ਨੰਬਰ ਇੱਕ ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਪੂਰਨ ਅੰਕ ਹੁੰਦਾ ਹੈ, ਤਾਂ ਅਸੀਂ ਡੈਸੀਮਲ ਨੰਬਰ ਦੇ ਦੋਨਾਂ ਭਾਗਾਂ (ਪੂਰਨ ਅੰਕ ਅਤੇ ਅੰਸ਼ਿਕ) ਨੂੰ ਵੱਖਰੇ ਰੂਪ ਵਿੱਚ ਬਦਲਦੇ ਹਾਂ। ਹੇਠਾਂ ਦਿੱਤੇ ਪੜਾਅ ਹਨ ਜੋ ਡੈਸੀਮਲ ਨੰਬਰ ਨੂੰ ਕਿਸੇ ਵੀ ਅਧਾਰ **'r'** ਦੀ ਸਮਾਨ ਸੰਖਿਆ ਵਿੱਚ ਬਦਲਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

ਪਹਿਲੇ ਪੜਾਅ ਵਿੱਚ, ਅਸੀਂ ਬੇਸ 'r' ਦੇ ਨਾਲ ਪੂਰਨ ਅੰਕ ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਾਗ 'ਤੇ ਵੰਡ ਦੀ ਕਾਰਵਾਈ ਕਰਦੇ ਹਾਂ। ਅਸੀਂ ਬਾਕੀ ਬਚੇ ਹੋਏ ਭਾਗਾਂ ਨੂੰ ਜ਼ੀਰੋ ਹੋਣ ਤੱਕ ਸੂਚੀਬੱਧ ਕਰਾਂਗੇ। ਫਿਰ ਅਸੀਂ ਅਧਾਰ 'r' ਦੇ ਬਰਾਬਰ ਸੰਖਿਆ ਦੇ ਪੂਰਨ ਅੰਕ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਉਲਟ ਕ੍ਰਮ ਵਿੱਚ ਬਾਕੀ ਬਚੇ (remainders) ਲੱਭਦੇ ਹਾਂ। ਇਸ ਵਿੱਚ, ਸਭ ਤੋਂ ਘੱਟ ਅਤੇ ਸਭ ਤੋਂ ਵੱਧ ਮਹੱਤਵਪੂਰਨ ਅੰਕਾਂ ਨੂੰ ਕ੍ਰਮਵਾਰ ਪਹਿਲੇ ਅਤੇ ਅਖੀਰਲੇ ਬਾਕੀ (last remainders) ਦੁਆਰਾ ਦਰਸਾਇਆ ਗਿਆ ਹੈ।

ਅਗਲੇ ਪੜਾਅ ਵਿੱਚ, ਗੁਣਾ ਦੀ ਕਾਰਵਾਈ ਨੂੰ ਫ੍ਰੈਕਸ਼ਨਲ ਅਤੇ ਲਗਾਤਾਰ ਪ੍ਰਾਪਤ ਹੋਏ ਫਰੈਕਸ਼ਨ ਦੇ ਅਧਾਰ 'r' ਨਾਲ ਕੀਤਾ ਜਾਂਦੀ ਹੈ। ਕੈਰੀਜ਼ ਉਦੋਂ ਤੱਕ ਨੋਟ ਕੀਤੇ ਜਾਂਦੇ ਹਨ ਜਦੋਂ ਤੱਕ ਨਤੀਜਾ ਜ਼ੀਰੋ ਨਹੀਂ ਹੁੰਦਾ ਜਾਂ ਜਦੋਂ ਬਰਾਬਰ ਅੰਕ ਦੀ ਲੋੜੀਂਦੀ ਸੰਖਿਆ ਪ੍ਰਾਪਤ ਨਹੀਂ ਹੁੰਦੀ ਹੈ। ਆਧਾਰ 'r' ਦੀ ਬਰਾਬਰ ਸੰਖਿਆ ਦਾ ਭਿੰਨਾਤਮਕ ਹਿੱਸਾ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ, ਕੈਰਇੰਗ ਦਾ ਆਮ ਕ੍ਰਮ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ।

(ੳ) ਡੈਸੀਮਲ ਤੋਂ ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Decimal to Binary Conversion)

ਡੈਸੀਮਲ ਨੂੰ ਬਾਈਨਰੀ ਵਿੱਚ ਬਦਲਣ ਲਈ, ਦੋ ਕਦਮਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜੋ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹਨ:

ਪਹਿਲੇ ਪੜਾਅ ਵਿੱਚ, ਅਸੀਂ ਬਾਈਨਰੀ(2) ਦੇ ਅਧਾਰ ਦੇ ਨਾਲ ਪੂਰਨ ਅੰਕ ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਾਗਾਂ (Successive Quotient) ਉੱਤੇ ਵੰਡ ਕਾਰਵਾਈ ਕਰਦੇ ਹਾਂ।

ਅੱਗੇ, ਅਸੀਂ ਬਾਈਨਰੀ(2) ਦੇ ਅਧਾਰ ਨਾਲ ਪੂਰਨ ਅੰਕ ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਾਗਾਂ (Successive Quotient) ਉੱਤੇ ਗੁਣਾ ਕਰਦੇ ਹਾਂ।

Example 1: $(68.25)_{10}$

Step 1:

ਨੰਬਰ 68 ਅਤੇ ਇਸਦੇ ਲਗਾਤਾਰ ਭਾਗਾਂ ਨੂੰ ਅਧਾਰ 2 ਨਾਲ ਵੰਡੋ।

Radix(ਰੈਡੀਕਸ)	Number/Quotient	Remainder
2	68	0 (LSB)
2	34	0
2	17	1
2	8	0
2	4	0
2	2	0
2	1	1(MSB)
	0	

ਚਿੱਤਰ 2.7

$(68)_{10}=(1000100)_2$

Step 2:

ਹੁਣ, ਬੇਸ 2 ਦੇ ਨਾਲ 0.25 ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਿੰਨਾਂ ਦਾ ਗੁਣਾ ਕਰੋ।

Operation	Result	carry
0.25×2	0.50	0
0.50×2	0	1

ਚਿੱਤਰ 2.8

$(0.25)_{10}$=$(.01)_2$

$(68.25)_{10}$=$(1000100.01)_2$

(ਅ) ਡੈਸੀਮਲ ਤੋਂ ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Decimal to Octal Conversion)

ਡੈਸੀਮਲ ਨੂੰ ਔਕਟਲ ਵਿੱਚ ਬਦਲਣ ਲਈ, ਦੋ ਪੜਾਅ ਕਰਨੇ ਜ਼ਰੂਰੀ ਹਨ, ਜੋ ਕਿ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹਨ:

ਪਹਿਲੇ ਪੜਾਅ ਵਿੱਚ, ਅਸੀਂ ਪੂਰਨ ਅੰਕ (8) ਦੇ ਅਧਾਰ ਦੇ ਨਾਲ ਪੂਰਨ ਅੰਕ ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਾਗਾਂਕ ਉੱਤੇ ਵੰਡ ਕਾਰਵਾਈ ਕਰਦੇ ਹਾਂ।

ਅਗਲਾ, ਅਸੀਂ ਪੂਰਨ ਅੰਕ ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਾਗਾਂ(Successive Quotient) ਨੂੰ ਔਕਟਲ(8) ਦੇ ਅਧਾਰ ਨਾਲ ਗੁਣਾ ਕਰਦੇ ਹਾਂ।

Example 1: $(68.25)_{10}$

Step 1:

ਨੰਬਰ 68 ਅਤੇ ਇਸਦੇ ਲਗਾਤਾਰ ਭਾਗਾਂ (Successive Quotient) ਨੂੰ ਅਧਾਰ 8 ਨਾਲ ਵੰਡੋ।

Radix	Number/Quotient	Remainder
8	68	4 (LSB)
8	8	0
8	1	1(MSB)
	0	

$(68)_{10}$=$(104)_8$

ਚਿੱਤਰ 2.9

Step 2:

ਹੁਣ ਅਧਾਰ 8 ਦੇ ਨਾਲ 0.25 ਅਤੇ ਲਗਾਤਾਰ ਭਿੰਨਾਂ ਦਾ ਗੁਣਾ ਕਰੋ।

Operation	Result	carry
0.25×8	0	2

$(0.25)_{10}$=$(2)_8$

ਚਿੱਤਰ 2.10

ਇਸ ਲਈ, ਡੈਸੀਮਲ ਨੰਬਰ 68.25 ਦੀ ਔਕਟਲ ਸੰਖਿਆ $(104.2)_8$ ਹੈ

(ੲ) ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਤੋਂ ਹੈਕਸਾ ਡੈਸੀਮਲ ਵਿੱਚ ਬਦਲਾਅ (Decimal to hexadecimal conversion)

ਡੈਸੀਮਲ ਨੂੰ ਹੈਕਸਾਡੈਸੀਮਲ ਵਿੱਚ ਬਦਲਣ ਲਈ, ਦੋ ਪੜਾਅ ਕਰਨੇ ਜ਼ਰੂਰੀ ਹਨ, ਜੋ ਕਿ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹਨ:

ਪਹਿਲੇ ਪੜਾਅ ਵਿੱਚ, ਅਸੀਂ ਹੈਕਸਾਡੈਸੀਮਲ (16) ਦੇ ਅਧਾਰ ਦੇ ਨਾਲ ਪੂਰਨ ਅੰਕ ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਾਗਾਂਕ (Successive Quotient) ਉੱਤੇ ਵੰਡ ਕਾਰਵਾਈ ਕਰਦੇ ਹਾਂ।

ਅੱਗੇ, ਅਸੀਂ ਹੈਕਸਾਡੈਸੀਮਲ (16) ਦੇ ਅਧਾਰ ਨਾਲ ਪੂਰਨ ਅੰਕ ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਾਗਾਂਕ (Successive Quotient) ਉੱਤੇ ਗੁਣਾ ਕਰਦੇ ਹਾਂ।

Example 1: $(68.25)_{10}$

Step 1:

ਨੰਬਰ 68 ਅਤੇ ਇਸਦੇ ਲਗਾਤਾਰ ਭਾਗਾਂ(Successive Quotient) ਨੂੰ ਅਧਾਰ 16 ਨਾਲ ਵੰਡੋ।

Radix	Number/Quotient	Remainder
16	68	4 (LSB)
16	4	4(MSB)
	0	

ਚਿੱਤਰ 2.11

$(68)_{10}=(44)_{16}$

Step 2:

ਹੁਣ ਅਧਾਰ 16 ਦੇ ਨਾਲ 0.25 ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਿੰਨਾਂ ਦਾ ਗੁਣਾ ਕਰੋ।

ਅੱਗੇ, ਅਸੀਂ ਹੈਕਸਾਡੈਸੀਮਲ (16) ਦੇ ਅਧਾਰ ਨਾਲ ਪੂਰਨ ਅੰਕ ਅਤੇ ਕ੍ਰਮਵਾਰ ਭਾਗਾਂਕ (Successive Quotient) ਉੱਤੇ ਗੁਣਾ ਕਰਦੇ ਹਾਂ।

Operation	Result	carry
0.25×16	0	4

ਚਿੱਤਰ 2.12

$(0.25)_{10}=(4)_{16}$

ਇਸ ਲਈ, ਡੈਸੀਮਲ ਨੰਬਰ $(68.25)_{10}$ ਦੀ ਹੈਕਸਾਡੈਸੀਮਲ ਸੰਖਿਆ $(44.4)_{16}$ ਹੈ।

3. ਔਕਟਲ ਤੋਂ ਹੋਰ ਸੰਖਿਆ ਪ੍ਰਣਾਲੀ ਵਿੱਚ (Octal to other Number System)

ਬਾਈਨਰੀ ਅਤੇ ਡੈਸੀਮਲ ਦੀ ਤਰ੍ਹਾਂ, ਔਕਟਲ ਸੰਖਿਆ ਨੂੰ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਵੀ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਔਕਟਲ ਨੰਬਰ ਡੈਸੀਮਲ ਵਿੱਚ ਬਦਲਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਬਾਕੀਆਂ ਨਾਲੋ ਵੱਖਰੀ ਹੈ। ਆਓ ਸਮਝਣਾ ਸ਼ੁਰੂ ਕਰੀਏ ਕਿ ਪਰਿਵਰਤਨ ਕਿਵੇਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

(ੳ) ਔਕਟਲ ਤੋਂ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ (Octal to Decimal Conversion)

ਔਕਟਲ ਨੂੰ ਡੈਸੀਮਲ ਵਿੱਚ ਬਦਲਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਬਾਇਨਰੀ ਤੋਂ ਡੈਸੀਮਲ ਦੇ ਵਰਗੀ ਹੀ ਹੈ। ਇਹ ਪ੍ਰਕਿਰਿਆ ਔਕਟਲ ਸੰਖਿਆ ਦੇ ਅੰਕਾਂ ਨੂੰ ਇਸਦੇ ਅਨੁਸਾਰੀ ਸਥਿਤੀ ਦੇ ਭਾਰਾਂ(Positional weights) ਨਾਲ ਗੁਣਾ ਕਰਨ ਤੋਂ ਸ਼ੁਰੂ ਹੁੰਦੀ ਹੈ। ਅਤੇ ਅੰਤ ਵਿੱਚ, ਅਸੀਂ ਉਹਨਾਂ ਸਾਰੇ ਉਤਪਾਦਾਂ (Products) ਨੂੰ ਜੋੜਦੇ ਹਾਂ.

ਆਉ ਇਹ ਸਮਝਣ ਲਈ ਇੱਕ ਉਦਾਹਰਨ ਲਈਏ ਕਿ ਔਕਟਲ ਤੋਂ ਦਸ਼ਮਲਵ ਤੱਕ ਪਰਿਵਰਤਨ ਕਿਵੇਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

Example 1: $(62.25)_8$

Step 1:

ਅਸੀਂ 62.25 ਦੇ ਹਰੇਕ ਅੰਕ ਨੂੰ ਇਸਦੇ ਸੰਬੰਧਿਤ ਸਥਿਤੀ ਦੇ ਭਾਰ ਨਾਲ ਗੁਣਾ ਕਰਦੇ ਹਾਂ, ਅਤੇ ਅੰਤ ਵਿੱਚ ਅਸੀਂ ਸਾਰੇ ਬਿੱਟਾਂ ਦੇ ਗੁਣਨਫਲ (Product) ਨੂੰ ਇਸਦੇ ਭਾਰ ਨਾਲ ਜੋੜਦੇ ਹਾਂ।

$$\mathbf{(62.25)_8} = (6 \times 8^1)+(2 \times 8^0)+(2\times8^{-1})+(5\times8^{-2})$$

$$\mathbf{(62.25)_8} = 48 + 2 +\left(2 \times \frac{1}{8}\right) + \left(5 \times \frac{1}{64}\right)$$

$$\mathbf{(62.25)_8} = 48 + 2 +0.25 + 0.078125$$

$$\mathbf{(62.25)_8} = 50.328125$$

ਇਸ ਲਈ, ਔਕਟਲ ਸੰਖਿਆ $(62.25)_8$ ਦੀ ਡੈਸੀਮਲ ਨੰਬਰ $(50.328125)_{10}$ ਹੈ

(ਅ) ਔਕਟਲ ਤੋਂ ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Octal to Binary Conversion)

ਔਕਟਲ ਨੂੰ ਬਾਈਨਰੀ ਵਿੱਚ ਬਦਲਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਬਾਇਨਰੀ ਦੀ ਔਕਟਲ ਵਿੱਚ ਉਲਟ ਪ੍ਰਕਿਰਿਆ ਹੈ। ਅਸੀਂ ਹਰੇਕ ਔਕਟਲ ਸੰਖਿਆ ਅੰਕ ਦੇ ਤਿੰਨ ਬਿੱਟ ਬਾਈਨਰੀ ਕੋਡ ਲਿਖਦੇ ਹਾਂ।

Example 1: $(62.25)_8$

ਅਸੀਂ 6, 2 ਅਤੇ 5 ਲਈ ਤਿੰਨ-ਬਿੱਟ ਬਾਈਨਰੀ ਅੰਕ ਲਿਖਦੇ ਹਾਂ।

$\mathbf{(62.25)_8}=(110010.010101)_2$

ਇਸ ਲਈ, ਔਕਟਲ ਸੰਖਿਆ $(62.25)_8$ ਦੀ ਬਾਈਨਰੀ ਸੰਖਿਆ $(110010.010101)_2$ ਹੈ।

(ੲ) ਔਕਟਲ ਤੋਂ ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Octal to hexadecimal conversion)

ਔਕਟਲ ਨੂੰ ਹੈਕਸਾਡੈਸੀਮਲ ਵਿੱਚ ਬਦਲਣ ਲਈ, ਦੋ ਪੜਾਵਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜੋ ਕਿ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ:

ਪਹਿਲੇ ਪੜਾਅ ਵਿੱਚ, ਅਸੀਂ ਨੰਬਰ 62.25 ਦੇ ਬਰਾਬਰ ਬਾਈਨਰੀ ਲੱਭਾਂਗੇ।

ਅੱਗੇ, ਸਾਨੂੰ ਬਾਈਨਰੀ ਬਿੰਦੂ ਦੇ ਦੋਵੇਂ ਪਾਸੇ ਚਾਰ ਬਿੱਟਾਂ ਦੇ ਗਰੁੱਪ ਬਣਾਉਣੇ ਪੈਣਗੇ। ਜੇਕਰ ਚਾਰ ਬਿੱਟਾਂ ਦੇ ਗਰੁੱਪ ਵਿੱਚ ਇੱਕ, ਦੋ, ਜਾਂ ਤਿੰਨ ਬਿੱਟ ਬਚੇ ਹਨ, ਤਾਂ ਅਸੀਂ ਦੋਨੋਂ ਸਾਈਡਾਂ ਦੇ ਅੰਤ 'ਤੇ ਜ਼ੀਰੋ ਦੀ ਲੋੜੀਂਦੀ ਸੰਖਿਆ ਜੋੜਦੇ ਹਾਂ ਅਤੇ ਹਰੇਕ ਗਰੁੱਪ ਦੇ ਅਨੁਸਾਰੀ ਹੈਕਸਾਡੈਸੀਮਲ ਅੰਕ ਲਿਖਦੇ ਹਾਂ।

Example 1: $(62.25)_8$

Step 1:

ਅਸੀਂ 6, 2 ਅਤੇ 5 ਲਈ ਤਿੰਨ-ਬਿੱਟ ਬਾਈਨਰੀ ਅੰਕ ਲਿਖਦੇ ਹਾਂ।

$(62.25)_8=(110010.010101)_2$

ਇਸ ਲਈ, ਔਕਟਲ 62.25 ਦੀ ਬਾਈਨਰੀ ਸੰਖਿਆ ਹੈ $(110010.010101)_2$

Step 2:

1. ਹੁਣ, ਅਸੀਂ ਬਾਈਨਰੀ ਪੁਆਇੰਟ ਦੇ ਦੋਵੇਂ ਪਾਸੇ ਚਾਰ ਬਿੱਟਾਂ ਦੇ ਗਰੁੱਪ ਬਣਾਉਂਦੇ ਹਾਂ।

11   0010.0101 01

ਬਾਈਨਰੀ ਬਿੰਦੂ ਦੇ ਖੱਬੇ ਪਾਸੇ, ਪਹਿਲੇ ਗਰੁੱਪ ਦੇ ਦੋ ਅੰਕ ਹੁੰਦੇ ਹਨ, ਅਤੇ ਸੱਜੇ ਪਾਸੇ, ਆਖਰੀ ਗਰੁੱਪ ਦੇ ਵੀ ਦੋ-ਅੰਕ ਹੁੰਦੇ ਹਨ। ਉਹਨਾਂ ਨੂੰ ਚਾਰ ਬਿੱਟਾਂ ਦੇ ਪੂਰੇ ਗਰੁੱਪ ਬਣਾਉਣ ਲਈ, ਦੋਨਾਂ ਸਿਰੇ ਦੇ ਪਾਸਿਆਂ 'ਤੇ ਵਾਧੂ ਜ਼ੀਰੋ ਜੋੜੋ।

0011   0010.0101   0100

2. **ਹੁਣ, ਅਸੀਂ ਹੈਕਸ ਾਡੈਸੀਮਲ ਅੰਕ ਲਿਖਦੇ ਹਾਂ, ਜੋ ਹਰੇਕ ਗਰੁੱਪ ਨਾਲ ਮੇਲ ਖਾਂਦੇ ਹਨ।**

$(0011 \quad 0010.0101 \quad 0100)_2 = (32.54)_{10}$

4. ਹੈਕਸਾ ਡੈਸੀਮਲ ਤੋਂ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ (Hexa-decimal to other Number System)

ਬਾਈਨਰੀ, ਡੈਸੀਮਲ, ਅਤੇ ਔਕਟਲ ਦੀ ਤਰ੍ਹਾਂ, ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰਾਂ ਨੂੰ ਵੀ ਹੋਰ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਹੈਕਸਾਡੈਸੀਮਲ ਨੂੰ ਦਸ਼ਮਲਵ ਵਿੱਚ ਬਦਲਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਬਾਕੀਆਂ ਤੋਂ ਵੱਖਰੀ ਹੈ। ਆਓ ਸਮਝਣਾ ਸ਼ੁਰੂ ਕਰੀਏ ਕਿ ਪਰਿਵਰਤਨ ਕਿਵੇਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

(ੳ) ਹੈਕਸਾ ਡੈਸੀਮਲ ਤੋਂ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Hexa-decimal to Decimal Conversion)

ਹੈਕਸਾਡੈਸੀਮਲ ਨੂੰ ਡੈਸੀਮਲ ਵਿੱਚ ਬਦਲਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਬਾਇਨਰੀ ਤੋਂ ਡੈਸੀਮਲ ਦੇ ਸਮਾਨ ਹੈ। ਇਹ ਪ੍ਰਕਿਰਿਆ ਹੈਕਸਾਡੈਸੀਮਲ ਸੰਖਿਆਵਾਂ ਦੇ ਅੰਕਾਂ ਨੂੰ ਇਸਦੇ ਅਨੁਸਾਰੀ ਸਥਿਤੀ ਦੇ ਭਾਰਾਂ ਨਾਲ ਗੁਣਾ ਕਰਨ ਤੋਂ ਸ਼ੁਰੂ ਹੁੰਦੀ ਹੈ। ਅਤੇ ਅੰਤ ਵਿੱਚ, ਅਸੀਂ ਉਹਨਾਂ ਸਾਰੇ ਗੁਣਨਫਲਾਂ (Products) ਨੂੰ ਜੋੜਦੇ ਹਾਂ.

ਆਉ ਇਹ ਸਮਝਣ ਲਈ ਇੱਕ ਉਦਾਹਰਣ ਲੈਂਦੇ ਹਾਂ ਕਿ ਹੈਕਸਾਡੈਸੀਮਲ ਤੋਂ ਡੈਸੀਮਲ ਤੱਕ ਪਰਿਵਰਤਨ ਕਿਵੇਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

Example 1: $(62A.25)_{16}$

Step 1:

ਅਸੀਂ 62A.25 ਦੇ ਹਰੇਕ ਅੰਕ ਨੂੰ ਇਸਦੇ ਸੰਬੰਧਿਤ ਸਥਿਤੀ ਦੇ ਭਾਰ ਨਾਲ ਗੁਣਾ ਕਰਦੇ ਹਾਂ, ਅਤੇ ਅੰਤ ਵਿੱਚ ਅਸੀਂ ਇਸਦੇ ਭਾਰ ਨਾਲ ਸਾਰੇ ਬਿੱਟਾਂ ਦੇ ਗੁਣਨਫਲਾਂ (Products) ਨੂੰ ਜੋੜਦੇ ਹਾਂ।

$$\mathbf{(62A.25)_{16}} = (6\times16^2)+(2\times16^1)+(A\times16^0)+(2\times16^{-1})+(5\times16^{-2})$$
$$= (6\times256)+(2\times16)+(10\times1)+(2\times16)+(5\times16)$$
$$=1536+32+10+\left(2\times\frac{1}{16}\right)+\left(5\times\frac{1}{256}\right)$$
$$=1578+0.125+0.125$$
$$=1578.14453125$$

ਇਸ ਲਈ, ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ 62A.25 ਦੀ ਦਸ਼ਮਲਵ ਸੰਖਿਆ ਹੈ $(1578.14453125)_{10}$

(ਅ) ਹੈਕਸਾ ਡੈਸੀਮਲ ਤੋਂ ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Hexadecimal to Binary Conversion)

ਹੈਕਸਾਡੈਸੀਮਲ ਨੂੰ ਬਾਈਨਰੀ ਵਿੱਚ ਬਦਲਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਬਾਈਨਰੀ ਤੋਂ ਹੈਕਸਾਡੈਸੀਮਲ ਦੀ ਉਲਟ ਪ੍ਰਕਿਰਿਆ ਹੈ। ਅਸੀਂ ਹਰੇਕ ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਦੇ ਅੰਕ ਦੇ ਚਾਰ ਬਿੱਟ ਬਾਈਨਰੀ ਕੋਡ ਲਿਖਦੇ ਹਾਂ।

Example 1: $(62A.25)_{16}$

ਅਸੀਂ , 6, A, 2, ਅਤੇ 5 ਲਈ ਚਾਰ-ਬਿੱਟ ਬਾਈਨਰੀ ਅੰਕ ਲਿਖਦੇ ਹਾਂ।

$(62A.25)_{16}$=(0110 0010 1010.0010 0 101)$_2$

ਇਸ ਲਈ, ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ 62A.25 ਦਾ ਬਾਈਨਰੀ ਨੰਬਰ ਹੈ

(0110 0010 1010.0010 0101)$_2$

(ੲ) ਹੈਕਸਾ ਡੈਸੀਮਲ ਤੋਂ ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਬਦਲਾਅ (Hexadecimal to Octal Conversion)

ਹੈਕਸਾਡੈਸੀਮਲ ਨੂੰ ਔਕਟਲ ਵਿੱਚ ਬਦਲਣ ਲਈ, ਦੋ ਪੜਾਵਾਂ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ, ਜੋ ਕਿ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ:

ਪਹਿਲੇ ਪੜਾਅ ਵਿੱਚ, ਅਸੀਂ ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਦੇ ਬਾਈਨਰੀ ਬਰਾਬਰ ਲੱਭਾਂਗੇ।

ਅੱਗੇ, ਸਾਨੂੰ ਬਾਈਨਰੀ ਬਿੰਦੂ ਦੇ ਦੋਵੇਂ ਪਾਸੇ ਤਿੰਨ ਬਿੱਟਾਂ ਦੇ ਗਰੁੱਪ ਬਣਾਉਣੇ ਪੈਣਗੇ। ਜੇਕਰ ਤਿੰਨ ਬਿੱਟਾਂ ਦੇ ਗਰੁੱਪ ਵਿੱਚ ਇੱਕ ਜਾਂ ਦੋ ਬਿੱਟ ਘੱਟ ਹਨ, ਤਾਂ ਅਸੀਂ ਸਿਰੇ ਦੇ ਪਾਸਿਆਂ ਤੇ ਜ਼ੀਰੋ ਦੀ ਲੋੜੀਂਦੀ ਸੰਖਿਆ ਜੋੜਦੇ ਹਾਂ ਅਤੇ ਹਰੇਕ ਗਰੁੱਪ ਦੇ ਅਨੁਸਾਰੀ ਔਕਟਲ ਅੰਕਾਂ ਨੂੰ ਲਿਖਦੇ ਹਾਂ।

Example 1: $(62A.25)_{16}$

Step 1:

ਅਸੀਂ 6, 2, A, ਅਤੇ 5 ਲਈ ਚਾਰ-ਬਿੱਟ ਬਾਈਨਰੀ ਅੰਕ ਲਿਖਦੇ ਹਾਂ।

$(62A.25)_{16}$=(0110 0010 1010.0010 0101)$_2$

ਇਸ ਲਈ, ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ 62A.25 ਦੀ ਬਾਈਨਰੀ ਸੰਖਿਆ ਹੈ

(0110 0010 1010.0010 0101)$_2$

Step 2:

ਫਿਰ, ਅਸੀਂ ਬਾਈਨਰੀ ਬਿੰਦੂ ਦੇ ਦੋਵੇਂ ਪਾਸੇ ਤਿੰਨ ਬਿੱਟਾਂ ਦੇ ਗਰੁੱਪ ਬਣਾਉਂਦੇ ਹਾਂ।

011 000 101 010.001 001 010

ਫਿਰ ਅਸੀ ਐਕਟਲ ਅੰਕ ਲਿਖਦੇ ਹਾਂ, ਜੋ ਹੇਰਕ ਗਰੁੱਪ ਨਾ ਮੇਲ ਖਾਂਦਾ ਹੈ।

(0110 0010 1010.0010 0101)$_2$=(3052.112)$_8$

ਇਸ ਲਈ, ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ 62A.25 ਦੀ ਔਕਟਲ ਸੰਖਿਆ 3052.112 ਹੈ

2.4 ਲੌਜਿਕ ਗੇਟਸ (LOGIC GATES)

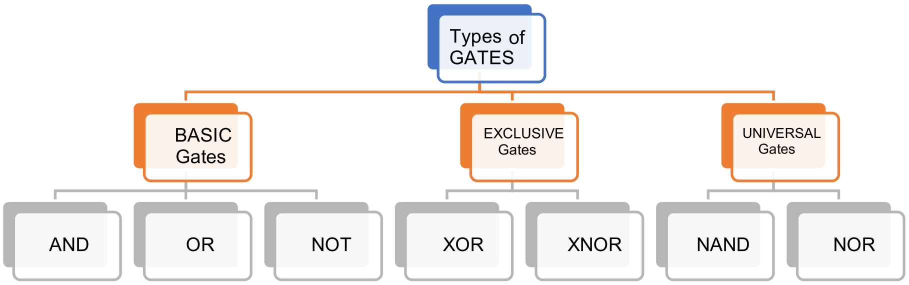
ਡਿਜੀਟਲ ਸਰਕਟ ਉਹ ਇਲੈਕਟ੍ਰੋਨਿਕਸ ਹਨ ਜੋ ਡਿਜੀਟਲ ਸਿਗਨਲਾਂ ਦੀ ਸਾਂਭ ਸੰਭਾਲ ਕਰਦੇ ਹਨ। ਇਹ ਸਰਕਟ ਐਨਾਲਾਗ ਪੱਧਰ (Levels) ਦੀਆਂ ਵੱਖਰੀਆਂ ਅਵਸਥਾਵਾਂ ਨਾਲ ਕੰਮ ਕਰਦੇ ਹਨ। ਡਿਜੀਟਲ ਇਲੈਕਟ੍ਰਾਨਿਕ ਸਰਕਟ ਆਮ ਤੌਰ ਤੇ ਲੌਜਿਕ ਗੇਟਸ ਤੋਂ ਬਣਾਏ ਜਾਂਦੇ ਹਨ।

ਲੌਜਿਕ ਗੇਟਸ ਕਿਸੇ ਵੀ ਡਿਜੀਟਲ ਸਿਸਟਮ ਦੇ ਬੁਨਿਆਦੀ ਬਿਲਡਿੰਗ ਬਲਾਕ ਹੁੰਦੇ ਹਨ। ਇਹ ਇੱਕ ਅਜਿਹਾ ਇਲੈਕਟ੍ਰਾਨਿਕ ਸਰਕਟ ਹੁੰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਇੱਕ ਜਾਂ ਇੱਕ ਤੋਂ ਵੱਧ ਇਨਪੁਟ ਹੁੰਦੇ ਹਨ ਅਤੇ ਕੇਵਲ ਇੱਕ ਹੀ ਆਉਟਪੁੱਟ ਹੁੰਦਾ ਹੈ ॥

ਲੌਜਿਕ ਗੇਟਸ ਡਾਇਓਡਸ (Diodes) ਅਤੇ ਟਰਾਂਜ਼ਿਸਟਰਾਂ ਤੋਂ ਬਣੇ ਹੁੰਦੇ ਹਨ। ਇਸਦੀ ਵਰਤੋਂ ਇੱਕ ਡਿਜੀਟਲ ਸਿਗਨਲ ਦੀ ਆਗਿਆ ਦੇਣ ਅਤੇ ਨਾ ਦੇਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਨਪੁਟ ਅਤੇ ਆਉਟਪੁੱਟ ਵਿਚਕਾਰ ਸਬੰਧ ਇੱਕ ਖਾਸ ਲੌਜਿਕ ਤੇ ਅਧਾਰਤ ਹੁੰਦਾ ਹੈ।

ਆਮ ਬੁਨਿਆਦੀ ਲੌਜਿਕ ਗੇਟਸ AND ਗੇਟ, OR ਗੇਟ, ਅਤੇ NOT ਗੇਟ ਹਨ, ਜਦੋਂ ਕਿ NAND ਅਤੇ NOR ਗੇਟਾਂ ਨੂੰ

ਯੂਨੀਵਰਸਲ ਗੇਟਸ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਗੇਟਾਂ ਦੀ ਵਿਆਖਿਆ ਹੇਠਾਂ ਦਿੱਤੀ ਗਈ ਹੈ:



ਚਿੱਤਰ 2.13

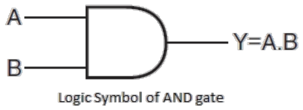
2.4.1 ਬੇਸਿਕ ਗੇਟ (BASIC GATES) :

(ੳ) AND GATE :

AND ਗੇਟ ਡਿਜੀਟਲ ਲੋਜਿਕ ਸਰਕਟ ਵਿੱਚ ਇੱਕ ਮਹੱਤਵਪੂਰਣ ਭੂਮਿਕਾ ਅਦਾ ਕਰਦਾ ਹੈ। AND ਗੇਟ ਦਾ ਆਉਟਪੁੱਟ ਮੁੱਲ ਹਮੇਸ਼ਾਂ 0 ਹੋਵੇਗਾ ਜਦੋਂ ਕੋਈ ਵੀ ਇੱਕ ਇਨਪੁਟ ਅਵਸਥਾ 0 ਹੁੰਦੀ ਹੈ। ਭਾਵ ਜੇਕਰ AND ਗੇਟ ਵਿੱਚ ਕੋਈ ਇਨਪੁਟ ਅਵਸਥਾ 0 ਹੁੰਦੀ ਹੈ, ਤਾਂ ਇਹ ਹਮੇਸ਼ਾ 0 ਆਉਟਪੁੱਟ ਵਾਪਸ ਕਰੇਗਾ ॥

AND ਗੇਟ ਲਈ ਲੋਜਿਕ ਜਾਂ ਬੂਲੀਅਨ ਸਮੀਕਰਨ, ਇੱਕ ਫੁਲ ਸਟਾਪ ਜਾਂ ਇੱਕ ਬਿੰਦੂ ਦੁਆਰਾ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ। ਜਿਵੇਂ ਕਿ : A.B = Y

Y ਦਾ ਮੁੱਲ ਉਦੋਂ '1' ਹੋਵੇਗਾ ਜਦੋਂ ਦੋਵੇਂ ਇਨਪੁਟਸ A ਅਤੇ B, '1' ਤੇ ਸੈੱਟ ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਦੋ ਲੋਜਿਕ ਵੇਰੀਏਬਲ A ਅਤੇ B ਲਈ AND ਗੇਟ ਦਾ ਲੋਜਿਕਲ ਚਿੰਨ੍ਹ ਅਤੇ ਟਰੁੱਥ ਟੇਬਲ ਹੇਠਾਂ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:



Logic Symbol of AND gate

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table for AND Gate

Figure Two-input AND gate

ਚਿੱਤਰ 2.14

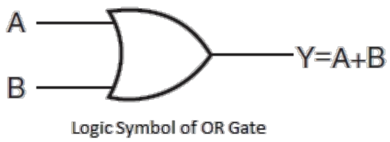
(b) OR GATE : ਇਹ ਦੋ ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਲੋਜਿਕ ਵੇਰੀਏਬਲਾਂ 'ਤੇ ਇੱਕ Oring Operation ਕਰਦਾ ਹੈ। ਇਸ ਗੇਟ ਵਿੱਚ ਦੋ ਜਾਂ ਵੱਧ ਇਨਪੁਟ ਅਤੇ ਇੱਕ ਆਉਟਪੁੱਟ ਹੋ ਸਕਦਾ ਹੈ। ਇੱਕ OR ਗੇਟ ਦਾ ਆਉਟਪੁੱਟ LOW (0) ਕੇਵਲ ਉਦੋਂ ਹੁੰਦਾ ਹੈ। ਜਦੋਂ ਇਸਦੇ ਸਾਰੇ ਇਨਪੁੱਟ Low (0) ਹੁੰਦੇ ਹਨ। ਨਹੀਂ ਤਾਂ ਆਉਟਪੁੱਟ High (1) ਹੁੰਦੀ ਹੈ। ਦੋ ਲੋਜਿਕ ਵੇਰੀਏਬਲ A ਅਤੇ B 'ਤੇ OR ਓਪਰੇਸ਼ਨ ਨੂੰ Y = A+B ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ, ਅਤੇ Y ਦੇ ਬਰਾਬਰ A ਜਾਂ B ਦੇ ਤੌਰ 'ਤੇ ਪੜ੍ਹਿਆ ਜਾਂਦਾ ਹੈ। ਇੱਥੇ, A ਅਤੇ B ਇੱਨਪੁਟ ਲੋਜਿਕ ਵੇਰੀਏਬਲ ਹਨ ਅਤੇ Y ਆਉਟਪੁੱਟ ਹੈ। OR ਗੇਟ ਦਾ ਆਉਟਪੁੱਟ ਮੁੱਲ ਹਮੇਸ਼ਾਂ 1 ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਕੋਈ ਵੀ ਇੱਕ ਇਨਪੁਟ ਅਵਸਥਾ 1 ਹੁੰਦੀ ਹੈ। ਜੇਕਰ OR ਗੇਟ ਵਿੱਚ ਸਾਰੀਆਂ ਇਨਪੁਟ ਦੀ ਸਥਿਤੀ (Low) (0) ਹੋਵੇ ਤਾਂ ਇਹ ਹਮੇਸ਼ਾ Low ਆਉਟਪੁੱਟ ਵਾਪਿਸ ਕਰੇਗਾ ਜਿਵੇਂ ਕਿ 0

OR ਗੇਟ ਲਈ ਲੋਜਿਕ ਜਾਂ ਬੂਲੀਅਨ ਸਮੀਕਰਨ A+B ਦੁਆਰਾ ਦਰਸਾਏ ਗਏ ਇਨਪੁਟਸ ਦਾ ਲਾਜ਼ੀਕਲ ਜੋੜ ਹੈ।

X+Y=Z ਮੁੱਲ ਉਦੋਂ '1' ਹੋਵੇਗਾ ਜਦੋਂ ਕੋਈ ਵੀ ਇਨਪੁਟ X ਅਤੇ Y '1' ਤੇ ਸੈੱਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਆਉਟਪੁੱਟ '1' ਹੈ ਜੇਕਰ ਕੋਈ ਜਾਂ ਦੋਵੇਂ ਇਨਪੁਟ '1' ਹਨ। ਜੇਕਰ ਦੋਵੇਂ ਇੰਪੁੱਟ '0' ਹਨ, ਤਾਂ ਆਉਟਪੁੱਟ '0' ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ,

ਆਊਟਪੁੱਟ 1 ਹੋਣ ਲਈ, ਘੱਟੋ-ਘੱਟ ਇੱਕ ਜਾਂ ਦੋ ਇਨਪੁੱਟ 1 ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ। ਦੋ ਲੋਜਿਕ ਵੇਰੀਏਬਲ A ਅਤੇ B ਲਈ OR ਗੇਟ ਦਾ ਤਰਕ ਚਿੰਨ੍ਹ ਅਤੇ ਟਰੁੱਥ ਟੇਬਲ ਹੇਠਾਂ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:



Logic Symbol of OR Gate

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table for OR Gate

ਚਿੱਤਰ 2.15

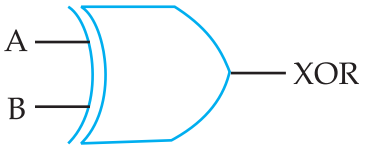
NOT GATE : NOT ਗੇਟ ਨੂੰ "ਇਨਵਰਟ" ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ, ਕਿਉਂਕਿ NOT (A) ਗੇਟ ਦਿੱਤੀ ਗਈ ਇਨਪੁੱਟ ਤੇ "ਲਾਜ਼ੀਕਲ ਨੈਗੇਸ਼ਨ" ਦਾ ਕਾਰਜ ਕਰਦਾ ਹੈ, ਭਾਵ ਕਿ NOT ਗੇਟ ਦਾ ਨਤੀਜਾ ਹਮੇਸ਼ਾ ਦਿੱਤੀ ਗਈ ਇਨਪੁੱਟ ਤੋਂ ਉਲਟਾ ਪ੍ਰਾਪਤ ਹੁੰਦਾ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ ਕਹੀਏ ਤਾਂ, ਜੇਕਰ NOT ਗੇਟ ਨੂੰ ਦਿੱਤੀ ਜਾਣ ਵਾਲੀ ਇਨਪੁੱਟ "ਸਹੀ" ਭਾਵ 1 ਹੋਵੇਗੀ ਤਾਂ ਇਸਦੀ "ਆਊਟਪੁੱਟ ਗਲਤ" ਭਾਵ 0 ਹੋਵੇਗੀ ਅਤੇ ਇਸਦੇ ਉਲਟ ਜੇਕਰ NOT ਗੇਟ ਨੂੰ ਦਿੱਤੀ ਜਾਣ ਵਾਲੀ ਇਨਪੁੱਟ "ਗਲਤ" ਭਾਵ 0 ਹੋਵੇਗੀ ਤਾਂ ਇਸਦੀ ਆਊਟਪੁੱਟ "ਸਹੀ" ਭਾਵ 1 ਪ੍ਰਾਪਤ ਹੋਵੇਗੀ। ਸਟੈਂਡਰਡ NOT ਗੇਟ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਇੱਕ ਪ੍ਰਤੀਕ ਦਿੱਤਾ ਗਿਆ ਹੈ ਜੋ "ਸੱਜੇ ਪਾਸੇ ਵੱਲ ਇਸ਼ਾਰਾ ਕਰਦੇ ਤਿਕੋਣ ਦੇ ਸਿਰੇ ਤੇ ਚੱਕਰ) ਬਣਾ ਕੇ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਸਰਕਲ ਭਾਵ ਚੱਕਰ ਨੂੰ "ਇਨਵਰਸ਼ਨ ਬੱਬਲ" ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ, ਜਿਸਦੀ ਵਰਤੋਂ NOT, NAND ਅਤੇ NOR ਗੇਟਾਂ ਦੀ ਆਊਟਪੁੱਟ ਤੇ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਜੋ ਕਿ NOT ਫੰਕਸ਼ਨ ਦੇ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ।

ਚਿੱਤਰ 2.16

2.4.2. Exclusive Gates :

(ੳ) XOR: XOR ਗੇਟ ਨੂੰ "ਐਕਸਕਲੂਸਿਵ OR "ਗੇਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।ਸਭ ਤੋਂ ਸਰਲ XOR ਗੇਟ ਉਹ ਡਿਜ਼ੀਟਲ ਸਰਕਟ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿ ਦੋ-ਇਨਪੁੱਟਸ ਲੈਂਦਾ ਹੈ ਅਤੇ ਕੇਵਲ ਉਦੋਂ ਹੀ ਲਾਜ਼ੀਕਲ 1 ਆਉਟਪੁੱਟ ਕਰਦਾ ਹੈ, ਜਦੋਂ ਦਿੱਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਦੋਨੋਂ ਇਨਪੁੱਟਸ ਇੱਕ ਦੂਜੇ ਤੋਂ ਭਿੰਨ ਹੋਣ ਭਾਵ, ਕਿ ਆਪਸ ਵਿੱਚ ਮੇਲ ਨਾ ਖਾਂਦੀਆਂ ਹੋਣ। ਜਿਵੇਂ ਕਿ ਇਸਦੀ ਆਊਟਪੁੱਟ ਲਾਜ਼ੀਕਲ 1 ਹੋਵੇਗੀ, ਜੇਕਰ ਦੋਨਾਂ ਵਿੱਚੋਂ ਕੋਈ ਇਨਪੁੱਟ 1 ਹੋਵੇ ਦੂਜੀ 0।ਜੇਕਰ ਦੋਨੋਂ ਇਨਪੁੱਟਸ 1 ਜਾਂ 0 ਹੋਈਆਂ ਤਾਂ ਇਸਦਾ ਨਤੀਜਾ ਹਮੇਸ਼ਾ 0 ਹੀ ਆਵੇਗਾ।ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ ਇੱਕ XOR ਗੇਟ ਦੀ ਬੂਲੀਅਨ ਸਮੀਕਰਨ ਦਿੱਤੀ ਗਈ ਹੈ, ਜਿਸਦੀਆਂ ਦੋ ਇਨਪੁੱਟਸ A ਅਤੇ B ਹਨ ਅਤੇ ਆਉਟਪੁੱਟ X ਹੈ:

$$X = A \oplus B$$

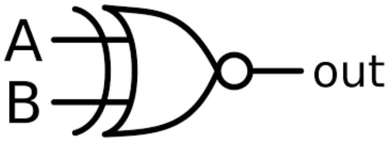


A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

ਚਿੱਤਰ 2.17 XOR ਗੇਟ ਦਾ ਚਿੰਨ ਅਤੇ ਟ੍ਰਥ ਟੇਬਲ

(ਅ) XNOR: XNOR ਗੇਟ ਇੱਕ ਡਿਜੀਟਲ ਲੌਜਿਕ ਗੇਟ ਹੈ ਜਿਸਦਾ ਫੰਕਸ਼ਨ ਐਕਸਕਲੂਸਿਵ OR ਗੇਟ ਦਾ ਲਾਜ਼ੀਕਲ ਪੂਰਕ (Complement) ਹੈ, ਭਾਵ ਕਿ ਉਸ ਤੋਂ ਉਲਟਾ ਹੈ।

XNOR ਗੇਟਾਂ ਦੀ ਵਰਤੋਂ ਮੁੱਖ ਤੌਰ ’ਤੇ ਉਨ੍ਹਾਂ ਇਲੈਕਟ੍ਰਾਨਿਕ ਸਰਕਟਾਂ ਵਿੱਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜੋ ਅੰਕਗਣਿਤ ਓਪਰੇਸ਼ਨ (Arithmatic Operations) ਦੀਆਂ ਕਾਰਵਾਈਆਂ ਅਤੇ ਡੇਟਾ ਦੀ ਜਾਂਚ ਕਰਦੇ ਹਨ ਜਿਵੇਂ ਕਿ ਐਡਰ, ਸਬਟਰੈਕਟਰ ਜਾਂ ਪੈਰਿਟੀ ਚੈਕਰਸ, ਆਦਿ।



XNOR ਗੇਟ ਦਾ ਚਿੰਨ ਅਤੇ ਟਰੂਥ ਟੇਬਲ

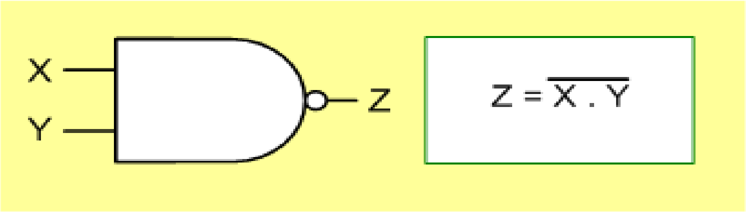
Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

ਚਿੱਤਰ 2.18

2.4.3 Universal Gate : (ਯੂਨੀਵਰਸਲ ਗੇਟ)

(a) NAND GATE : NAND : ਗੇਟ ਦੋ ਗੇਟਾਂ ਦਾ ਸੁਮੇਲ ਹੈ AND ਅਤੇ NOT, ਇਸਦਾ ਅਰਥ ਹੈ AND ਗੇਟ ਦੇ ਉਲਟ। ਇਸਨੂੰ ਬੁਲੀਅਨ ਅਲਜੇਬਰਾ ਵਿੱਚ ਇੱਕ “ਯੂਨੀਵਰਸਲ” ਗੇਟ ਵੀ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਇਸ ਗੇਟ ਰਾਹੀ ਹੋਰ ਸਾਰੇ ਲੋਜਿਕ ਗੇਟ ਬਣਾਏ ਜਾ ਸਕਦੇ ਹਨ। NAND ਗੇਟ ਲਈ ਗ੍ਰਾਫਿਕ ਚਿੰਨ੍ਹ ਦੀ ਆਉਟਪੁੱਟ ’ਤੇ ਇੱਕ ਬੁਲਬੁਲੇ ਦਾ ਨਿਸ਼ਾਨ AND ਚਿੰਨ੍ਹ ਹੁੰਦਾ ਹੈ, ਇਹ ਦਰਸਾਉਂਦਾ ਹੈ ਕਿ AND ਗੇਟ ਦੇ ਆਉਟਪੁੱਟ ਤੇ Complement ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਟਰੂਥ ਟੇਬਲ ਅਤੇ NAND ਗੇਟ ਦਾ ਗ੍ਰਾਫਿਕ ਚਿੰਨ੍ਹ ਚਿੱਤਰ ਵਿੱਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ।

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0



ਚਿੱਤਰ 2.19

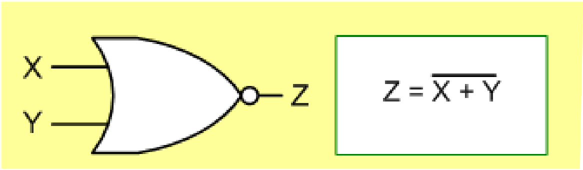
ਆਉਟਪੁਟ ਦਾ ਮੁੱਲ ਉਦੋਂ ਸਹੀ ਹੋਵੇਗਾ ਜਦੋਂ ਕੋਈ ਇੱਕ ਇਨਪੁਟ 0 ਤੇ ਸੈੱਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

(ਅ) NOR Gate : NOR ਗੇਟ ਦੋ ਗੇਟਾਂ NOT ਅਤੇ OR ਦਾ ਸੁਮੇਲ ਹੈ। ਇਸਦਾ ਅਰਥ ਹੈ ਇੱਕ OR ਲੋਜਿਕ ਗੇਟ ਦਾ ਉਲਟ। ਇਸਨੂੰ ਵੀ ਬੁਲੀਅਨ ਅਲਜੇਬਰਾ ਵਿੱਚ ਇੱਕ “ਯੂਨੀਵਰਮਲ’ ਗੇਟ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਇਸ ਗੇਟ ਰਾਹੀਂ ਵੀ ਹੋਰ ਸਾਰੇ ਲੋਜਿਕ ਗੇਟ ਬਣਾਏ ਜਾ ਸਕਦੇ ਹਨ।

NOR ਗੇਟ ਲਈ ਗ੍ਰਾਫਿਕ ਚਿੰਨ੍ਹ ਵਿੱਚ ਇੱਕ OR ਚਿੰਨ੍ਹ ਹੁੰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਆਉਟਪੁੱਟ ਉੱਤੇ ਇੱਕ ਬੁਲਬੁਲਾ ਹੁੰਦਾ ਹੈ, ਇਹ ਦਰਸਾਉਂਦਾ ਹੈ ਕਿ OR ਗੇਟ ਦੇ ਆਉਟਪੁੱਟ ਉੱਤੇ Complement ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਟਰੁਥ ਟੇਬਲ ਅਤੇ NOR ਗੇਟ ਦਾ ਗ੍ਰਾਫਿਕ ਚਿੰਨ੍ਹ ਚਿੱਤਰ ਵਿੱਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ।

X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



ਚਿੱਤਰ 2.20

ਟਰੁਥ ਟੇਬਲ ਸਪੱਸ਼ਟ ਤੌਰ 'ਤੇ ਦਰਸਾਉਂਦਾ ਹੈ ਕਿ NOR ਓਪਰੇਸ਼ਨ OR ਦਾ complement ਹੈ।

2.5 NAND and NOR ਗੇਟਸ ਨੂੰ ਯੂਨਿਵਰਸਲ ਗੇਟਸ ਦੇ ਤੌਰ ਤੇ ਸਿੱਧ ਕਰਨਾ:

ਇੱਕ ਯੂਨੀਵਰਸਲ ਉਹ ਗੇਟ ਇੱਕ ਗੇਟ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿਸੇ ਵੀ ਬੂਲੀਅਨ ਫੰਕਸ਼ਨ ਨੂੰ ਬਿਨਾਂ ਕਿਸੇ ਹੋਰ ਕਿਸਮ ਦੇ ਗੇਟ ਦੀ ਵਰਤੋਂ ਕੀਤੇ ਬਿਨਾਂ ਲਾਗੂ ਕਰ ਸਕਦਾ ਹੈ। NAND ਅਤੇ NOR ਗੇਟ ਯੂਨੀਵਰਸਲ ਗੇਟ ਹਨ।

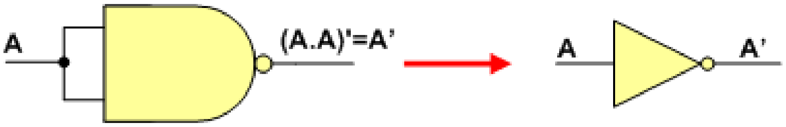
NAND ਅਤੇ NOR ਗੇਟ ਬਣਾਉਣ ਵਿੱਚ ਆਸਾਨ ਅਤੇ ਕਿਫਾਇਤੀ ਹਨ ਅਤੇ ਸਾਰੇ IC ਡਿਜੀਟਲ ਲੌਜਿਕ ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਬੁਨਿਆਦੀ ਗੇਟ ਹਨ।

2.5.1 NAND ਗੇਟ ਇੱਕ ਯੂਨਿਵਰਸਲ ਗੇਟ ਹੈ : NAND ਗੇਟ ਇੱਕ ਯੂਨੀਵਰਸਲ ਗੇਟ ਹੈ ਕਿਉਂਕਿ ਇਹ AND, OR ਅਤੇ NOT ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਲਾਗੂ ਕਰ ਸਕਦਾ ਹੈ ਇਹ ਸਾਬਤ ਕਰਨ ਲਈ ਕਿ ਕੋਈ ਵੀ ਬੂਲੀਅਨ ਫੰਕਸ਼ਨ ਸਿਰਫ NAND ਗੇਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਲਾਗੂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਅਸੀਂ ਦਿਖਾਵਾਂਗੇ ਕਿ ਸਿਰਫ ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ AND, OR, ਅਤੇ NOT ਓਪਰੇਸ਼ਨ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ।

(ੳ) ਸਿਰਫ਼ NAND ਗੇਟ ਦੀ ਵਰਤੋਂ ਕਰਕੇ NOT ਗੇਟ ਨੂੰ ਲਾਗੂ ਕਰਨਾ

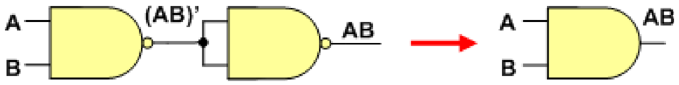
ਦੋ ਤਰੀਕੇ ਦਿਖਾਏ ਗਏ ਹਨ ਜਿਸ ਵਿੱਚ ਇੱਕ NAND ਗੇਟ ਨੂੰ ਇੱਕ ਇਨਵਰਟਰ (NOT ਗੇਟ) ਵਜੋਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

ਜੇਕਰ NAND ਗੇਟ ਤੱਕ ਇੱਕ ਸਿੰਗਲ ਇਨਪੁੱਟ A ਹੈ ਤਾਂ ਇਹ (A.A)' ਨੂੰ ਆਉਟਪੁੱਟ ਦੇ ਤੌਰ ਤੇ ਦਿੰਦਾ ਹੈ। ਜੇਕਰ A=0 ਇਹ ਆਉਟਪੁੱਟ 1 ਦੇਵੇਗਾ ਅਤੇ ਜੇਕਰ A=1 ਇਹ ਆਉਟਪੁੱਟ 0 ਦੇਵੇਗਾ।



ਚਿੱਤਰ 2.21

(ਅ) ਸਿਰਫ਼ NAND ਗੇਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ AND ਗੇਟ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ।

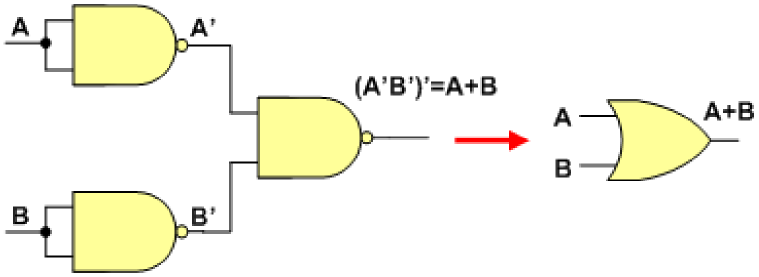


ਚਿੱਤਰ 2.22

ਅਸੀਂ NAND ਗੇਟ ਦੀ ਵਰਤੋਂ ਕਰਕੇ AND ਗੇਟ ਬਣਾ ਸਕਦੇ ਹਾਂ। । ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ AND ਨੂੰ ਇੱਕਂ NAND ਗੇਟ ਦੁਆਰਾ ਬਦਲਿਆ ਗਿਆ ਹੈ ਅਤੇ ਇਸਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਇੱਕ ਹੋਰ ਗੇਟ ਇਨਵਰਟਰ ਦੁਆਰਾ ਪੂਰਕ ਕੀਤਾ ਗਿਆ ਹੈ।

(ੲ) ਸਿਰਫ਼ NAND ਗੇਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ OR ਗੇਟ ਨੂੰ ਲਾਗੂ ਕਰਨਾ

ਅਸੀਂ NAND ਗੇਟ ਦੀ ਵਰਤੋਂ ਕਰਕੇ OR ਗੇਟ ਬਣਾ ਸਕਦੇ ਹਾਂ। ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿਚ OR ਗੇਟ ਨੂੰ ' NAND ਗੇਟ ਦੁਆਰਾ ਬਦਲਿਆ ਗਿਆ ਹੈ ਅਤੇ ਇਸਦੇ ਸਾਰੇ ਇਨਪੁਟਸ ਨੂੰ NAND ਗੇਟ ਇਨਵਰਟਰਾਂ ਦੁਆਰਾ ਪੂਰਕ ਕੀਤਾ ਗਿਆ ਹੈ। ਇਸ ਲਈ, 'NAND ਗੇਟ ਇੱਕ ਯੂਨੀਵਰਸਲ ਗੇਟ ਹੈ ਕਿਉਂਕਿ ਇਹAND, OR ਅਤੇ NOT ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਲਾਗੂ ਕਰ ਸਕਦਾ ਹੈ।



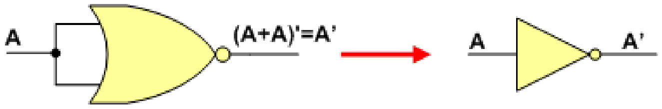
ਚਿੱਤਰ 2.23

2.5.2 NOR ਗੇਟ ਇੱਕ ਯੂਨੀਵਰਸਲ ਗੇਟ ਹੈ : ਇਹ ਸਾਬਤ ਕਰਨ ਲਈ ਕਿ NOR ਗੇਟ ਇੱਕ ਯੂਨੀਵਰਸਲ ਗੇਟ ਹੈ, ਅਸੀਂ ਦਿਖਾਵਾਂਗੇ ਕਿ ਇਹਨਾਂ ਗੇਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ AND, OR, ਅਤੇ NOR ਓਪਰੇਸ਼ਨ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ।

(ੳ) ਸਿਰਫ਼ NOR ਗੇਟ ਦੀ ਵਰਤੋਂ ਕਰਕੇ NOT ਗੇਟ ਨੂੰ ਲਾਗੂ ਕਰਨਾ।

ਚਿੱਤਰ ਦੋ ਤਰੀਕੇ ਦਿਖਾਉਂਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਇੱਕ NOR ਗੇਟ ਨੂੰ ਇੱਕ ਇਨਵਰਟਰ (NOT ਗੇਟ) ਵਜੋਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

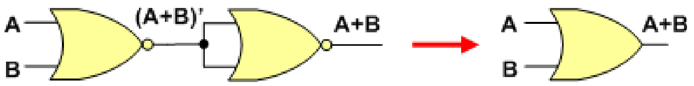
ਸਾਰੇ NOR ਇਨਪੁਟ ਪਿੰਨ ਇਨਪੁਟ ਸਿਗਨਲ A ਨਾਲ ਜੁੜਦੇ ਹਨ, ਇੱਕ ਆਉਟਪੁੱਟ A' ਦਿੰਦਾ ਹੈ।



ਚਿੱਤਰ 2.24

(ਅ) ਸਿਰਫ਼ NOR ਗੇਟ ਦੀ ਵਰਤੋਂ ਕਰਕੇ OR ਗੇਟ ਨੂੰ ਲਾਗੂ ਕਰਨਾ

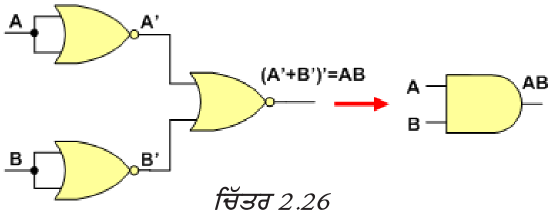
ਇੱਕ OR ਗੇਟ ਨੂੰ NOR ਗੇਟਾਂ ਦੁਆਰਾ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ਜਿਵੇਂ ਕਿ ਚਿੱਤਰ ਵਿੱਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ (OR ਗੇਟ ਨੂੰ NOR ਗੇਟ ਦੁਆਰਾ ਬਦਲਿਆ ਗਿਆ ਹੈ ਅਤੇ ਇਸਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਇੱਕ NOR ਗੇਟ ਇਨਵਰਟਰਾਂ ਦੁਆਰਾ ਪੂਰਕ ਕੀਤਾ ਗਿਆ ਹੈ)



ਚਿੱਤਰ 2.25

(ੲ) ਸਿਰਫ NOR ਗੇਟ ਦੀ ਵਰਤੋਂ ਕਰਕੇ AND ਗੇਟ ਨੂੰ ਲਾਗੂ ਕਰਨਾ :-

ਇੱਕ AND ਗਟ ਨੂੰ NOR ਗੇਟਾਂ ਦੁਆਰਾ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ਜਿਵੇਂ ਕਿ ਚਿੱਤਰ ਵਿੱਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ (AND ਗੇਟ ਨੂੰ NOR ਗੇਟ ਦੁਆਰਾ ਬਦਲਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਇਸਦੇ ਸਾਰੇ ਇਨਪੁਟਸ NOR ਗੇਟ ਇਨਵਰਟਰਾਂ ਦੁਆਰਾ ਪੂਰਕ ਹੁੰਦੇ ਹਨ)



ਚਿੱਤਰ 2.26

ਇਸ ਤਰ੍ਹਾਂ, NOR ਗੇਟ ਇੱਕ ਯੂਨੀਵਰਸਲ ਗੇਟ ਹੈ ਕਿਉਂਕਿ ਇਹ AND, OR ਅਤੇ NOT ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਲਾਗੂ ਕਰ ਸਕਦਾ ਹੈ।

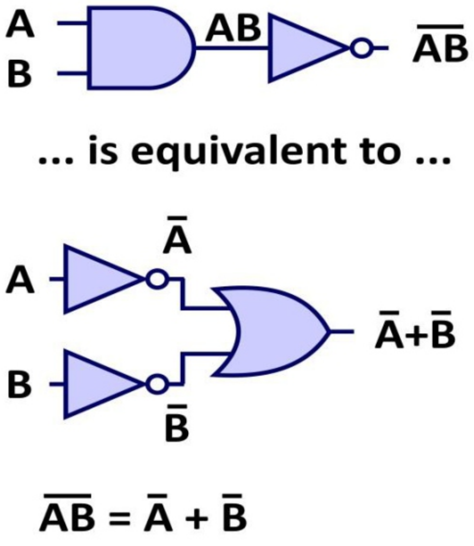
2.6 ਡੀਮੋਰਗਨ ਥਿਓਰਮ (DEMORGAN'S THEOREM)

ਇਹ ਥਿਓਰਮ ਡਿਜੀਟਲ ਸਰਕਟਾਂ ਨੂੰ ਡਿਜ਼ਾਈਨ ਕਰਨ ਲਈ ਅਤੇ ਵੱਖ-ਵੱਖ ਬੁਲੀਅਨ ਅਲਜੇਬਰਾ ਸਮੀਕਰਨਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਵਰਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ NAND ਗੇਟ ਅਤੇ NOR ਗੇਟ ਵਰਗੇ ਬੁਨਿਆਦੀ ਗੇਟ ਓਪਰੇਸ਼ਨ ਨੂੰ ਹੱਲ ਕਰਦਾ ਹੈ। ਇਹ ਲਾਈਨਾਂ ਨੂੰ ਤੋੜ ਕੇ ਅਤੇ ਜੋੜ ਨੂੰ ਉਤਪਾਦ (Product) ਵਿੱਚ ਅਤੇ ਉਤਪਾਦ (Product) ਨੂੰ ਜੋੜ ਵਿੱਚ ਬਦਲਣ ਦੇ ਸਿਧਾਂਤ ਤੇ ਅਧਾਰਤ ਹੈ। ਇਹ ਥਿਓਰਮ ਦੱਸਦੀ ਹੈ ਕਿ ਸਾਰੀਆਂ ਟਰਮਜ਼ (Terms) ਦੇ ਗੁਣਨਫਲ ਦਾ ਪੂਰਕ (Complement) ਹਰੇਕ ਟਰਮਜ਼ (Terms) ਦੇ ਪੂਰਕ ਦੇ ਜੋੜ ਦੇ ਬਰਾਬਰ ਹੁੰਦਾ ਹੈ। ਇਸੇ ਤਰ੍ਹਾਂ, ਸਾਰੀਆਂ ਟਰਮਜ਼ (Terms) ਦੇ ਜੋੜ ਦਾ ਪੂਰਕ ਹਰੇਕ ਟਰਮਜ਼ (Terms) ਦੇ ਪੂਰਕ ਦੇ ਗੁਣਨਫਲ ਦੇ ਬਰਾਬਰ ਹੈ। ਇਸ ਲਈ, ਇਸਦੇ ਦੋ ਨਿਯਮ ਹਨ।

ਥਿਓਰੇਮ 1: ਇਹ ਸਾਬਤ ਕਰਦਾ ਹੈ ਕਿ ਅਜਿਹੀਆਂ ਸਥਿਤੀਆਂ ਵਿੱਚ ਜਿੱਥੇ ਦੋ (ਜਾਂ ਵੱਧ) ਇਨਪੁਟ ਵੇਰੀਏਬਲ ਗੁਣਾ ਅਤੇ ਪੂਰਕ (Complement) ਹੁੰਦੇ ਹਨ, ਉਹ ਵੱਖਰੇ ਵੇਰੀਏਬਲਾਂ ਦੇ ਪੂਰਕ ਦੇ OR ਦੇ ਬਰਾਬਰ ਹੁੰਦੇ ਹਨ। ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ ਇਹ ਦੱਸਦਾ ਹੈ ਕਿ ਇੱਕ ਗੁਣਨਫੱਲ (Product) ਦਾ ਪੂਰਕ, (Complement) ਦੇ ਜੋੜ ਦੇ ਬਰਾਬਰ ਹੈ।

$(AB)' = A' + B'$

A	B	$\overline{AB}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0



ਚਿੱਤਰ 2.27

ਥਿਓਰੇਮ 2: ਇਹ ਸਾਬਤ ਕਰਦਾ ਹੈ ਕਿ ਅਜਿਹੀਆਂ ਸਥਿਤੀਆਂ ਵਿੱਚ ਜਿੱਥੇ ਦੋ (ਜਾਂ ਵੱਧ) ਇਨੱਪੁਟ ਵੇਰੀਏਬਲ ਜੋੜੇ ਜਾਂਦੇ ਹਨ ਅਤੇ ਪੂਰਕ (Complement) ਕੀਤੇ ਜਾਂਦੇ ਹਨ, ਉਹ ਵੱਖਰੇ ਵੇਰੀਏਬਲਾਂ ਦੇ ਪੂਰਕਾਂ (Complements) ਦੇ AND (ਗੁਣਾਂਕ) ਦੇ ਬਰਾਬਰ ਹੁੰਦੇ ਹਨ। ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ ਇਹ ਦੱਸਦਾ ਹੈ ਕਿ ਇੱਕ ਜੋੜ ਦਾ ਪੂਰਕ (Complements) ਦੇ ਗੁਣਾਂਕ ਬਰਾਬਰ ਹੈ।

$(A+B)'=A'.B'$

A	B	$\bar{A}$	$\bar{B}$	$A+B$	$\overline{A+B}$	$\bar{A} \bullet \bar{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

ਚਿੱਤਰ 2.28

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

- ਨੰਬਰ ਪ੍ਰਣਾਲੀ ਦੀਆਂ ਦੋ ਬੁਨਿਆਦੀ ਕਿਸਮਾਂ ਹਨ ਜਿਵੇਂ ਕਿ ਨੋਨ ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ (Non Positional Number System) ਅਤੇ ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ (Positional Number System) ।
- ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਅੰਕਾਂ ਦੀ ਕੁੱਲ ਸੰਖਿਆ ਨੂੰ ਅਧਾਰ (Base) ਜਾਂ ਰੇਡਿਕਸ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
- NAND ਅਤੇ NOR ਨੂੰ ਯੂਨੀਵਰਸਲ ਗੇਟਸ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
- ਚਾਰ ਕਿਸਮਾਂ ਦੇ ਨੰਬਰ ਸਿਸਟਮ ਹਨ ਜੋ ਕੰਪਿਊਟਰ ਦਾ ਸਮਰਥਨ ਕਰਦਾ ਹੈ ਜਿਵੇਂ ਕਿ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ, ਬਾਈਨਰੀ ਨੰਬਰ ਸਿਸਟਮ, ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ, ਹੈਕਸਾਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ।
- XOR GATE ਨੂੰ Exclusive OR GATE ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਅਤੇ XNOR GATE ਨੂੰ Exclusive NOR GATE ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
- XNOR GATE ਇੱਕ ਇਲੈਕਟ੍ਰੋਨਿਕ ਸਰਕਟ ਹੈ ਜੋ ਡਾਟਾ ਚੈਕਿੰਗ ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ ਜਿਵੇਂ ਕਿ ਐਡਰ, ਸਬਟਰੈਕਟਰ ਜਾਂ ਪੈਰਿਟੀ ਚੈਕਰ।

ਅਭਿਆਸ

ਪ੍ਰਸ਼ਨ 1. ਬਹੁਪੰਸਦੀ ਪ੍ਰਸ਼ਨ

1. ਇੱਕ ਨੰਬਰ ਸਿਸਟਮ ਜੋ 16 ਵੱਖ-ਵੱਖ ਚਿੰਨ੍ਹਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ ਨੂੰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
   a. ਡੈਸਿਮਲ ਨੰਬਰ ਸਿਸਟਮ   b. ਬਾਇਨਰੀ ਨੰਬਰ ਸਿਸਟਮ
   c. ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ   d. ਹੈਕਸਾ ਡੈਸਿਮਲ ਨੰਬਰ ਸਿਸਟਮ
2. NAND GATE ਕਿਨ੍ਹਾਂ ਗੇਟ ਦਾ ਸੁਮੇਲ ਹੈ।
   a. AND ,NOT   b. OR , NOT
   c. AND,OR   d. NOT,XOR
3. OCTAL ਨੰਬਰ ਸਿਸਟਮ ਦਾ ਰੈਡਿਕਸ ਕੀ ਹੈ ?
   a. 2   b. 5
   c. 8   d. 16

4. $(38CB)_{16}$ ਕਿਸ ਨੰਬਰ ਸਿਸਟਮ ਦੀ ਉਦਾਹਰਨ ਹੈ।

   a. ਡੈਸਿਮਲ ਨੰਬਰ ਸਿਸਟਮ b. ਬਾਇਨਰੀ ਨੰਬਰ ਸਿਸਟਮ

   c. ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ d. ਹੈਕਸਾ ਡੈਸਿਮਲ ਨੰਬਰ ਸਿਸਟਮ

5. ਇਸ ਸੰਖਿਆ ਦਾ ਮੁੱਲ $(111)_2$ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਵਿੱਚ ਕੀ ਹੈ ?

   a. 7 b. 3

   c. 6 d. 8

ਪ੍ਰਸ਼ਨ 2.ਖ਼ਾਲੀ ਥਾਵਾਂ ਭਰੋ ।

I. ਲਾਜਿਕ ਗੇਟਸ ____________________ ਅਤੇ _______________ਦੇ ਬਣੇ ਹੁੰਦੇ ਹਨ।.

II. NOT ਗੇਟ ਨੂੰ _________________ ਗੇਟ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

III. ________________ ਅਤੇ __________________ ਯੂਨੀਵਰਸਲ ਗੇਟ ਹਨ।

IV. ਬੇਸਿਕ ਨੰਬਰ ਸਿਸਟਮ ਦੀਆਂ ਦੋ ਕਿਸਮਾਂ ਹਨ_________________ ਅਤੇ _____________.

V. ਰੋਮਨ ਨੰਬਰ ਸਿਸਟਮ_________________ਨੰਬਰ ਸਿਸਟਮ ਦੀ ਇੱਕ ਉਦਾਹਰਨ ਹੈ।

ਪ੍ਰਸ਼ਨ 3. ਸਹੀ ਅਤੇ ਗਲਤ

I. AND ਗੇਟ ਲਈ ਲੌਜਿਕ ਇੱਕ ਲਾਜ਼ੀਕਲ ਜੋੜ ਹੈ।

II. ਔਕਟਲ ਸ਼ਬਦ ਲਾਤੀਨੀ ਸ਼ਬਦ oct ਤੋਂ ਲਿਆ ਗਿਆ ਹੈ ਜਿਸਦਾ ਅਰਥ 8 ਹੈ ।

III. ਯੂਨਿਕਸ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਵਿੱਚ chmod ਕਮਾਂਡ ਫਾਈਲ ਦੀ ਪਰਮਿਸ਼ਨ (Permission) ਦੇਣ ਲਈ ਔਕਟਲ ਨੰਬਰ ਸਿਸਟਮ ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ।

IV. XNOR ਦਾ ਅਰਥ ਹੈ Exclusive NOR

V. ਹੈਕਸਾ ਦਾ ਅਰਥ 6 ਹੈ ।

ਪ੍ਰਸ਼ਨ 4. ਬਹੁਤ ਛੋਟੇ ਉਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ

I. ਲੌਜਿਕ ਗੇਟ ਕੀ ਹੈ ?

II. ਸੰਖਿਆ ਪ੍ਰਣਾਲੀ (Number system) ਦੀਆਂ ਦੋ ਬੁਨਿਆਦੀ ਕਿਸਮਾਂ ਕੀ ਹਨ ?

III. ਪੋਜਿਸ਼ਨਲ ਨੰਬਰ ਸਿਸਟਮ ਦੀਆਂ ਘੱਟੋ-ਘੱਟ 3 ਉਦਾਹਰਨਾਂ ਦਿਓ

IV. ਹੈਕਸਾ ਡੈਸੀਮਲ ਨੰਬਰ ਸਿਸਟਮ ਕੀ ਹੈ ?

V. AND ਗੇਟ ਨੂੰ ਟਰੁੱਥ ਟੇਬਲ ਨਾਲ ਸਮਝਾਓ ।

VI. NAND ਗੇਟ ਨੂੰ ਯੂਨੀਵਰਸਲ ਗੇਟ ਕਿਉਂ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ?

VII. XOR ਗੇਟ ਨੂੰ ਟਰੁੱਥ ਟੇਬਲ ਨਾਲ ਸਮਝਾਓ ?

VIII. NAND ਗੇਟ ਨੂੰ ਟਰੁੱਥ ਟੇਬਲ ਨਾਲ ਸਮਝਾਓ ।

ਪ੍ਰਸ਼ਨ 5. ਵੱਡੇ ਉਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

I. ਯੂਨੀਵਰਸਲ ਗੇਟਸ ਕੀ ਹਨ ? ਕਿਸੇ ਇੱਕ ਨੂੰ ਵਿਸਥਾਰ ਨਾਲ ਸਮਝਾਓ।

II. ਚਾਰ ਕਿਸਮਾਂ ਦੇ ਨੰਬਰ ਸਿਸਟਮ ਦੀ ਵਿਆਖਿਆ ਕਰੋ ਜੋ ਕਿ ਕੰਪਿਊਟਰ ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਹਨ ।

III. ਡੀ-ਮੋਰਗਨ ਥਿਓਰਮ ਦੀ ਵਿਸਥਾਰ ਵਿੱਚ ਵਿਆਖਿਆ ਕਰੋ। ।

IV. ਲੌਜ਼ਿਕ ਗੇਟਸ ਦੀਆਂ ਕਿਸਮਾਂ ਦੀ ਵਿਆਖਿਆ ਵਿਸਥਾਰ ਵਿੱਚ ਕਰੋ।

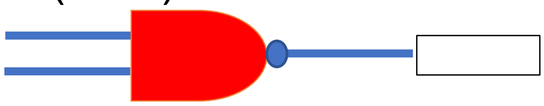
ਲੈਬ-ਐਕਟਿਵੀਟੀ

ਡੈਸੀਮਲ	ਬਾਈਨਰੀ	ਔਕਟਲ	ਹੈਕਸਾਡੈਸੀਮਲ
0	0000	0	0
1	0001	---	1
2	0010	2	---
3	---	3	3
4	0100	---	4
5	0101	---	5
6	0110	6	---
7	---	7	
8	1000	10	8
9	---	11	9
10	1010	12	---
11	1011	---	B
12	---	14	C
13	1101	15	---
14	1110	---	E
15	1111	17	---

ਹੱਲ ਕਰੋ :

ਡੈਸੀਮਲ	ਬਾਈਨਰੀ	ਔਕਟਲ	ਹੈਕਸਾਡੈਸੀਮਲ
23			
	111101		
		27	
			38C
89			
	10001		
			8F
		37	
			9A
99			

ਪੂਰਾ ਕਰੋ

1. $\overline{(A+B)}+C=$
2. $A.B+\overline{C}\ =$
3. $(A+B)\ +C=$
4. $A+\overline{\overline{(B+C)}}=$
5. 

ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਨਾਲ ਜਾਣ ਪਛਾਣ
(Introduction to Programming Language)

3

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

3.1 ਜਾਣ ਪਛਾਣ

3.2 ਪ੍ਰੋਗਰਾਮ, ਪ੍ਰੋਗਰਾਮਿੰਗ ਅਤੇ ਪ੍ਰੋਗਰਾਮਰ ਦੀ ਧਾਰਣਾ (Concept)

3.3 ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ

3.4 ਭਾਸ਼ਾ ਟ੍ਰਾਂਸਲੇਟਰਜ਼ (Translators)

3.5 ਐਰਰਜ਼ (Errors) ਦੀਆਂ ਕਿਸਮਾਂ

3.1 ਜਾਣ-ਪਛਾਣ (INTRODUCTION) :

ਇਸ ਪਾਠ ਵਿਚ ਅਸੀਂ ਮੁੱਢਲੇ ਤੌਰ ਤੇ ਪ੍ਰੋਗਰਾਮ, ਪ੍ਰੋਗਰਾਮਿੰਗ, ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰੋਸੈਸ ਅਤੇ ਕੰਪਿਊਟਰ ਸਿਸਟਮਾਂ ਵਿੱਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਵੱਖ-ਵੱਖ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਬਾਰੇ ਪੜ੍ਹਾਂਗੇ। ਪ੍ਰੋਗਰਾਮ ਮੁੱਢਲੇ ਤੌਰ ਤੇ ਕਿਸੇ ਕੰਮ ਨੂੰ ਕਰਨ ਲਈ ਕੰਪਿਊਟਰ ਦੁਆਰਾ ਸਮਝੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਹਦਾਇਤਾਂ ਦਾ ਸਮੂਹ ਹੁੰਦੇ ਹਨ। ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਲਿਖਣ ਦਾ ਪ੍ਰੋਸੈਸ ਪ੍ਰੋਗਰਾਮਿੰਗ ਕਹਾਉਂਦਾ ਹੈ। ਉਹ ਵਿਅਕਤੀ, ਜੋ ਪ੍ਰੋਗਰਾਮ ਲਿਖਦਾ ਹੈ ਉਸ ਨੂੰ ਪ੍ਰੋਗਰਾਮਰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਜਦੋਂ ਪ੍ਰੋਗਰਾਮਰ ਕਿਸੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਲਿਖਦਾ ਹੈ ਉਸ ਸਮੇਂ ਉਹ ਇੱਕ ਖਾਸ ਪ੍ਰੋਸੈਸ ਵਿਚੋਂ ਲੰਘਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਦਾ ਇਹ ਪ੍ਰੋਸੈਸ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰੋਸੈਸ ਅਖਵਾਉਂਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮਰ ਕਈ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵਿਚੋਂ ਕੋਈ ਇਕ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਦੀ ਚੋਣ ਕਰਕੇ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰਦਾ ਹੈ। ਇਸ ਪਾਠ ਵਿਚ ਇਹਨਾਂ ਸਾਰੀਆਂ ਧਾਰਣਾਵਾਂ ਦਾ ਵਿਸਥਾਰ ਵਿਚ ਵਰਨਣ ਕੀਤਾ ਜਾਵੇਗਾ।

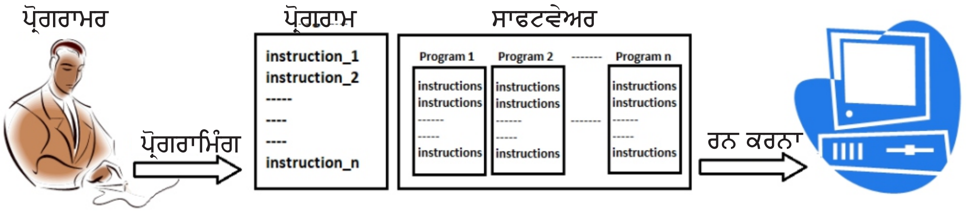
3.2 ਪ੍ਰੋਗਰਾਮ ਅਤੇ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀ ਧਾਰਣਾ (CONCEPT OF PROGRAM, PROGRAMMING AND PROGRAMMER)

ਅਸੀਂ ਜਾਣਦੇ ਹਾਂ ਕਿ ਇੱਕ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਮੁੱਢਲੇ ਤੌਰ ਤੇ ਦੋ ਭਾਗਾਂ ਦਾ ਬਣਿਆ ਹੁੰਦਾ ਹੈ: ਹਾਰਡਵੇਅਰ ਅਤੇ ਸਾਫਟਵੇਅਰ। ਸਾਫਟਵੇਅਰ ਤੋਂ ਬਿਨਾਂ ਕੰਪਿਊਟਰ ਹਾਰਡਵੇਅਰ ਕੋਈ ਵੀ ਕੰਮ ਨਹੀਂ ਕਰ ਸਕਦਾ। ਸਾਫਟਵੇਅਰ ਤੋਂ ਬਿਨਾਂ ਕੰਪਿਊਟਰ ਸਿਰਫ ਧਾਤੂ ਦਾ ਇੱਕ ਟੁੱਕੜਾ ਬਣ ਕੇ ਰਹਿ ਜਾਵੇਗਾ। ਕੰਪਿਊਟਰ ਹਾਰਡਵੇਅਰ ਨੂੰ ਕੋਈ ਕੰਮ ਕਰਨ ਯੋਗ ਬਨਾਉਣ ਲਈ ਸਾਨੂੰ ਇਸ ਵਿਚ ਸਾਫਟਵੇਅਰ ਭਰਨਾ ਪੈਂਦਾ ਹੈ। ਹੁਣ ਪ੍ਰਸ਼ਨ ਇਹ ਹੈ ਕਿ ਆਖਿਰ ਇਹ ਸਾਫਟਵੇਅਰ ਹੁੰਦਾ ਕੀ ਹੈ ?

ਸਾਫਟਵੇਅਰ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਾਂ ਦਾ ਸਮੂਹ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿਸੇ ਕੰਮ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਸਾਫਟਵੇਅਰ ਕੰਪਿਊਟਰ ਨੂੰ ਕੰਮ ਕਰਨ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ। ਇਹ ਸਾਫਟਵੇਅਰ ਹੀ ਹੁੰਦੇ ਹਨ। ਜੋ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਡਾਟਾ ਪ੍ਰੋਸੈਸ, ਸਟੋਰ ਅਤੇ ਨਤੀਜਾ ਵਾਪਿਸ ਹਾਸਿਲ ਕਰਨ ਯੋਗ ਬਣਾਉਂਦੇ ਹਨ। ਇਹ ਸਾਫਟਵੇਅਰ ਦੋ ਕਿਸਮਾਂ ਦੇ ਹੁੰਦੇ ਹਨ: ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ। ਸਿਸਟਮ ਸਾਫਟਵੇਅਰਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਅੰਦਰੂਨੀ ਕੰਮਾਂ ਕਾਰਾਂ ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਲਈ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ, ਜਦੋਂ ਕਿ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੁਆਰਾ ਕਿਸੇ ਖਾਸ ਕੰਮ ਨੂੰ ਕਰਨ ਲਈ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ।

ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰਾਂ ਦੇ ਮੁਕਾਬਲੇ ਜਿਆਦਾ ਗੁੰਝਲਦਾਰ ਹੁੰਦੇ ਹਨ। ਸਿਸਟਮ ਸਾਫਟਵੇਅਰਾਂ ਨੂੰ ਬਣਾਉਣ ਲਈ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰਾਂ ਦੀ ਤੁਲਨਾ ਵਿਚ ਜਿਆਦਾ ਕੁਸ਼ਲਤਾ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ।

ਕਿਸੇ ਵੀ ਕਿਸਮ ਦਾ ਸਾਫਟਵੇਅਰ ਆਮ ਤੌਰ ਤੇ ਇੱਕ ਇਕਾਈ (Single Entity) ਨਹੀਂ ਹੁੰਦਾ। ਇਹ ਪ੍ਰੋਗਰਾਮਾਂ ਦਾ ਸਮੂਹ ਹੁੰਦੇ ਹਨ। ਕਿਸੇ ਵੀ ਕਿਸਮ ਦੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਲਿਖਣ ਲਈ ਪ੍ਰੋਗਰਾਮਰ ਵੱਲੋਂ ਹਦਾਇਤਾਂ ਨੂੰ ਇਕ ਖਾਸ ਲੜੀ ਵਿਚ ਲਿਖਣਾ ਪੈਂਦਾ ਹੈ ਤਾਂ ਜੋ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਸਾਡੇ ਕੰਮ ਨੂੰ ਸਫਲਤਾਪੂਰਵਕ ਕਰ ਸਕੇ। ਇਸ ਤਰਾਂ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਕ ਪ੍ਰੋਗਰਾਮ ਉਹਨਾਂ ਹਦਾਇਤਾਂ ਦਾ ਸਮੂਹ ਹੁੰਦਾ ਹੈ ਜਿਹਨਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਸਮਝ ਸਕਦਾ ਹੈ।ਜਦੋਂ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਦਿੱਤੀਆਂ ਹਦਾਇਤਾਂ ਦੀ ਲੜੀ ਅਨੁਸਾਰ ਕੰਮ ਕਰਦਾ ਹੈ ਤਾਂ ਸਾਨੂੰ ਆਪਣੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਨਤੀਜਾ ਮਿਲ ਜਾਂਦਾ ਹੈ। ਜਦੋਂ ਕਿ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰ ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਮਾਂ ਦਾ



ਚਿੱਤਰ 3.1 ਪ੍ਰੋਗਰਾਮ, ਪ੍ਰੋਗਰਾਮਰ, ਪ੍ਰੋਗਰਾਮਿੰਗ ਅਤੇ ਸਾਫਟਵੇਅਰ

ਸਮੂਹ ਹੁੰਦੇ ਹਨ। ਕਿਸੇ ਵੀ ਕਿਸਮ ਦੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਲਿਖਣ ਦਾ ਪ੍ਰੋਸੈਸ ਪ੍ਰੋਗਰਾਮਿੰਗ ਅਖਵਾਉਂਦਾ ਹੈ। ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਲਿਖਣਾ ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮਿੰਗ ਅਖਵਾਉਂਦਾ ਹੈ ਜਦੋਂ ਕਿ ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਲਿਖਣਾ ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਾਮਿੰਗ ਅਖਵਾਉਂਦਾ ਹੈ। ਉਹ ਵਿਅਕਤੀ ਜੋ ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮ ਲਿਖਦਾ ਹੈ ਉਸਨੂੰ ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮਰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਉਹ ਵਿਅਕਤੀ ਜੋ ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਾਮ ਲਿਖਦਾ ਹੈ ਉਸਨੂੰ ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਾਮਰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

3.3 ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ (PROGRAMMING LANGUAGES)

ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਹਦਾਇਤਾਂ ਲਿਖਣ ਲਈ ਪ੍ਰੋਗਰਾਮਰ ਕੋਈ ਵੀ ਉਹ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦਾ ਹੈ ਜੋ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਸਮਝ ਸਕਦਾ ਹੋਵੇ। ਇਹ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਉਹਨਾਂ ਕੁਦਰਤੀ ਭਾਸ਼ਾਵਾਂ ਦੀ ਤਰ੍ਹਾਂ ਹੀ ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਅਸੀਂ ਰੋਜ਼ਾਨਾ ਜਿੰਦਗੀ ਵਿਚ ਵਰਤਦੇ ਹਾਂ, ਜਿਵੇਂ ਕਿ: ਪੰਜਾਬੀ, ਹਿੰਦੀ ਅਤੇ ਅੰਗਰੇਜ਼ੀ ਆਦਿ। ਜਿਸ ਤਰ੍ਹਾਂ ਅਸੀਂ ਕੁਦਰਤੀ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਰੋਜਾਨਾ ਜਿੰਦਗੀ ਵਿਚ ਆਪਸੀ ਸੰਚਾਰ ਕਰਨ ਲਈ ਵਰਤਦੇ ਹਾਂ ਉਸੇ ਤਰ੍ਹਾਂ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਅਸੀਂ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਲਈ ਵਰਤਦੇ ਹਾਂ ਅਤੇ ਸਾਫਟਵੇਅਰਾਂ ਦੁਆਰਾ ਕੰਪਿਊਟਰ ਨੂੰ ਯੂਜ਼ਰ ਦੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਕੰਮ ਕਰਨ ਯੋਗ ਬਣਾ ਸਕਦੇ ਹਾਂ। ।

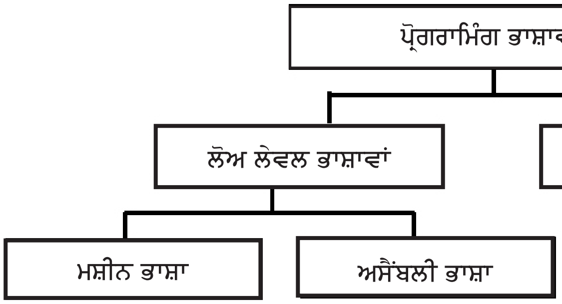
ਬਜ਼ਾਰ ਵਿਚ ਬਹੁਤ ਸਾਰੀਆਂ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਮੌਜੂਦ ਹਨ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਕੁੱਝ ਖਾਸ ਐਪਲੀਕੇਸ਼ਨ ਖੇਤਰਾਂ ਅਨੁਸਾਰ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ। ਇਹਨਾਂ ਵਿਚੋਂ ਕੁੱਝ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਤਿਆਰ ਕਰਨ ਲਈ ਜਿਆਦਾ ਢੁਕਵੀਆਂ ਹਨ ਅਤੇ ਕੁੱਝ ਭਾਸ਼ਾਵਾਂ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰਾਂ ਲਈ ਜਿਆਦਾ ਢੁਕਵੀਆਂ ਹਨ।

ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮ ਕੰਪਿਊਟਰਾਂ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਵਰਤਣ ਯੋਗ ਬਨਾਉਣ ਲਈ ਤਿਆਰ ਕੀਤੇ ਗਏ ਹਨ। ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਦੀ ਇਕ ਵਧੀਆ ਉਦਾਹਰਣ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਹੈ ਜਿਸਦੀ ਵਰਤੋਂ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਓਪਰੇਟ ਕਰਨ, ਇਨਪੁੱਟ ਆਊਟਪੁੱਟ ਡਿਵਾਈਸਾਂ, ਪ੍ਰੋਸੈਸਰ ਅਤੇ ਮੈਮਰੀ ਆਦਿ ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਲਈ ਬਣਾਇਆ ਗਿਆ ਹੈ। ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮ, ਜਿਵੇਂ ਕਿ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ, ਨੂੰ ਤਿਆਰ ਕਰਨ ਲਈ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਹਾਰਡਵੇਅਰ ਭਾਗਾਂ ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਸੰਬੰਧੀ ਲੋੜੀਂਦੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਕਰਨੀ ਪਵੇਗੀ। ਇਹ ਸਿਰਫ ਤਾਂ ਹੀ ਸੰਭਵ ਹੈ ਜੇ ਪ੍ਰੋਗਰਾਮਰ ਇਹਨਾਂ ਹਾਰਡਵੇਅਰ ਭਾਗਾਂ ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ ਬਾਰੇ ਜਾਣਕਾਰੀ ਰੱਖਦਾ ਹੋਵੇ। ਇਸ ਲਈ ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮਿੰਗ ਉਹਨਾਂ ਕੁਸ਼ਲ ਪ੍ਰੋਗਰਾਮਰਾਂ ਲਈ ਸੰਭਵ ਹੈ ਜੋ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਹਾਰਡਵੇਅਰ ਭਾਗਾਂ ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ ਸੰਬੰਧੀ ਵਿਸਥਾਰਤ ਜਾਣਕਾਰੀ ਰੱਖਦੇ ਹੋਣ। ਮਸ਼ੀਨ ਭਾਸ਼ਾ, ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਅਤੇ ਸੀ ਭਾਸ਼ਾ ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰਨ ਲਈ ਸਭ ਤੋਂ ਵੱਧ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਭਾਸ਼ਾਵਾਂ ਹਨ।

ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਾਮ ਕੋਈ ਖਾਸ ਕੰਮ ਕਰਨ ਲਈ ਜਾਂ ਖਾਸ ਸਮੱਸਿਆਵਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਬਣਾਏ ਜਾਂਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ ਸਟੂਡੈਂਟ ਮੈਨੇਜਮੈਂਟ ਸਿਸਟਮ (student management system), ਲਾਇਬ੍ਰੇਰੀ ਮੈਨੇਜਮੈਂਟ ਸਿਸਟਮ (library managment system), ਪੇਅਰੋਲ ਸਿਸਟਮ (payrolls system), ਇਨਵੈਂਟਰੀ ਕੰਟਰੋਲ ਸਿਸਟਮ (inventory control system), ਵਰਡ ਪ੍ਰੋਸੈਸਰ (word processor), ਸਪੈਡਸ਼ੀਟ ਸਾਫਟਵੇਅਰ (Spread sheet), ਗ੍ਰਾਫਿਕਸ ਸਾਫਟਵੇਅਰ (graphics software) ਆਦਿ। ਇਸ ਤਰ੍ਹਾਂ ਦੇ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਬਨਾਉਣ ਲਈ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਹਾਰਡਵੇਅਰ ਭਾਗਾਂ ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਵਾਲੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੀ ਜਰੂਰਤ ਨਹੀਂ ਪੈਂਦੀ। ਇਸ ਲਈ ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਾਮਰਾਂ ਨੂੰ ਇਹਨਾਂ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਤਿਆਰ ਕਰਨ ਲਈ ਕਿਸੇ ਤਰ੍ਹਾਂ ਦੇ ਹਾਰਵੇਅਰ ਭਾਗਾਂ ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ ਦੀ ਜਾਣਕਾਰੀ ਹੋਣਾ ਜਰੂਰੀ ਨਹੀਂ ਹੁੰਦਾ। ਇਸ ਤਰ੍ਹਾਂ ਦੇ ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਤਿਆਰ ਕਰਨ ਲਈ PYTHON, COBOL, FORTRAN, BASIC, PASCAL, C, C++, JAVA ਆਦਿ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵਰਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ

3.3.1 ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਕਿਸਮਾਂ (Types of Programming Languages:)

ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਜਾਣਦੇ ਹਾਂ ਕਿ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਸਾਫਟਵੇਅਰ ਤਿਆਰ ਕਰਨ ਲਈ ਬਹੁਤ ਕਿਸਮਾਂ ਦੀਆਂ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਬਾਜਾਰ ਵਿੱਚ ਉਪਲੱਬਧ ਹਨ। ਅਸੀਂ ਆਪਣੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਇਹਨਾਂ ਵਿਚੋਂ ਕੋਈ ਵੀ ਭਾਸ਼ਾ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰਨ ਲਈ ਚੁਣ ਸਕਦੇ ਹਾਂ। ਇਹਨਾਂ ਸਾਰੀਆਂ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਆਮ ਤੌਰ ਤੇ ਦੋ ਵੱਖ-ਵੱਖ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ, ਜਿਵੇਂ ਕਿ ਹੇਠਾਂ ਚਿੱਤਰ ਵਿਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ, ਵੰਡੀਆ ਜਾ ਸਕਦਾ ਹੈ:



ਚਿੱਤਰ : 3.2 ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਵਾਂ ਦੀਆਂ ਕਿਸਮਾਂ

ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਤਿਆਰ ਕਰਨ ਲਈ ਲੋਅ-ਲੇਵਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ (Low level Programming Languages) ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਕਿ ਐਪਲੀਕੇਸ਼ਨ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਤਿਆਰ ਕਰਨ ਲਈ ਕਈ ਤਰ੍ਹਾਂ ਦੀਆਂ ਹਾਈ ਲੇਵਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ। ਇਹਨਾਂ ਸਾਰੀਆਂ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦਾ ਵਿਸਥਾਰਿਤ ਵਰਨਣ ਹੇਠਾਂ ਕੀਤਾ ਗਿਆ ਹੈ:

ੳ. ਲੋਅ-ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ (Low Level Languages):

ਮਸ਼ੀਨ ਅਤੇ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਲੋਅ-ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਲੋਅ-ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਕਹਿਣ ਦਾ ਕਾਰਨ ਇਹ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਸਿਸਟਮ ਸਾਫਟਵੇਅਰ ਤਿਆਰ ਕਰਨ ਲਈ ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ ਕੰਪਿਊਟਰ ਹਾਰਡਵੇਅਰ ਦੀ ਨਿਚਲੇ ਪੱਧਰ (ਲੋਅ ਲੈਵਲ) ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ ਸੰਬੰਧੀ ਜਾਣਕਾਰੀ ਹੋਣਾ ਲਾਜ਼ਮੀ ਹੁੰਦਾ ਹੈ। ਲੋਅ-ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਮਸ਼ੀਨ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਹੱਲ ਕੀਤੀ ਜਾਣ ਵਾਲੀ ਸਮੱਸਿਆ ਉੱਪਰ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰਨ ਦੀ ਬਜਾਏ ਮਸ਼ੀਨ (ਕੰਪਿਊਟਰ ਸਿਸਟਮ) ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ ਉੱਪਰ ਜਿਆਦਾ ਧਿਆਨ ਕੇਂਦਰਿਤ ਕਰਕੇ ਰੱਖਣ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਲੋਅ-ਲੇਵਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀ ਵਿਸਥਾਰਿਤ ਜਾਣਕਾਰੀ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹੈ:

i. ਮਸ਼ੀਨ ਭਾਸ਼ਾ (Machine Language) : ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਨੂੰ ਬਾਈਨਰੀ ਭਾਸ਼ਾ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਭਾਸ਼ਾ ਨੂੰ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀ ਪਹਿਲੀ ਜੈਨਰੇਸ਼ਨ ਦੀ ਭਾਸ਼ਾ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਕੰਪਿਊਟਰ ਸਿਸਟਮਾਂ ਦੀ ਮੁੱਢਲੀ (fundamental) ਭਾਸ਼ਾ ਹੈ ਕਿਉਂਕਿ ਇਸ ਭਾਸ਼ਾ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਸਿੱਧੇ ਤੌਰ ਤੇ ਸਮਝਦਾ ਹੈ। ਇਸ ਭਾਸ਼ਾ ਨੂੰ ਸਮਝਣ ਲਈ ਕੰਪਿਊਟਰ ਨੂੰ ਕਿਸੇ ਤਰ੍ਹਾਂ ਦੀ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਦੀ ਜਰੂਰਤ ਨਹੀਂ ਪੈਂਦੀ। ਇਹ ਭਾਸ਼ਾ ਸਿਰਫ ਦੋ ਬਾਈਨਰੀ ਅੰਕਾਂ 0 ਅਤੇ 1 ਤੋਂ ਮਿਲ ਕੇ ਬਣੀ ਹੈ। ਮਸ਼ੀਨੀ ਭਾਸ਼ਾ ਵਿਚ ਹਰ ਇੱਕ ਹਦਾਇਤ ਸਿਰਫ ਇਹਨਾਂ ਦੋ ਅੰਕਾਂ ਤੋਂ ਹੀ ਮਿਲਕੇ ਬਣਦੀ ਹੈ।

ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਦੀ ਹਰ ਇੱਕ ਹਦਾਇਤ ਦੇ ਦੋ ਭਾਗ ਹੁੰਦੇ ਹਨ: Opcode **ਅਤੇ** operand, ਜਿਵੇਂ ਕਿ ਹੇਠਾਂ ਚਿੱਤਰ ਵਿਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:

OPCODE	OPERAND

ਚਿੱਤਰ-3.3 ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਵਿੱਚ ਹਦਾਇਤਾਂ ਦਾ ਫਾਰਮੈਟ

ਇਸ ਵਿੱਚ Opcode ਦਾ ਮਤਲਬ ਹੈ Operation Code ਅਤੇ Operand ਦਾ ਮਤਲਬ ਹੈ Operation Address. ਪਹਿਲਾ ਭਾਗ- **Operation code** ਹੁੰਦਾ ਹੈ ਜੋ ਇਕ ਕਮਾਂਡ ਹੁੰਦੀ ਹੈ। ਇਹ ਕੰਪਿਊਟਰ ਨੂੰ ਇਹ ਦੱਸਦੀ ਹੈ ਕਿ ਹਦਾਇਤ ਵੱਲੋਂ ਕਿਹੜਾ ਕੰਮ ਦੂਸਰਾ ਭਾਗ - **Operation Address** ਹੁੰਦਾ ਹੈ ਜੋ ਮੈਮਰੀ ਐਡਰੈਸ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ ਜੋ ਕੰਪਿਊਟਰ ਨੂੰ ਇਹ ਦੱਸਦਾ ਹੈ ਡਾਟਾ, ਜਿਸ ਉੱਪਰ ਕੰਮ ਕੀਤਾ ਜਾਣਾ ਹੈ, ਉਹ ਮੈਮਰੀ ਵਿਚ ਕਿੱਥੇ ਪਿਆ ਹੈ ਕਿਉਕਿ ਹਦਾਇਤਾਂ ਨੂੰ ਸਿਰਫ ਬਾਈਨਰੀ ਅੰਕਾਂ 0 ਅਤੇ 1 ਵਿੱਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ, ਇਸ ਲਈ ਇਹਨਾਂ ਮਸ਼ੀਨ ਹਦਾਇਤਾਂ ਨੂੰ ਬਾਈਨਰੀ ਰੂਪ ਵਿਚ ਯਾਦ ਰੱਖਣਾ ਬਹੁਤ ਔਖਾ ਹੁੰਦਾ ਹੈ। ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਨੂੰ ਵਰਤਣ ਦੇ ਕਈ ਲਾਭ ਵੀ ਹੁੰਦੇ ਹਨ ਅਤੇ ਹਾਨੀਆਂ ਵੀ ਹਨ। ਜਿਨ੍ਹਾਂ ਵਿਚੋਂ ਕੁੱਝ ਲਾਭ ਅਤੇ ਹਾਨੀਆਂ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ:

ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਦੇ ਲਾਭ (Advantages of Machine Language) :

- ਬਾਈਨਰੀ ਰੂਪ ਵਿਚ ਹਦਾਇਤਾਂ ਨੂੰ ਬਿਨਾਂ ਕਿਸੇ ਟਾਂਸਲੇਸ਼ਨ ਦੇ ਸਿੱਧੇ ਤੌਰ ਤੇ ਕੰਪਿਊਟਰ ਦੁਆਰਾ ਸਮਝਿਆ ਜਾ ਸਕਦੇ ਹੈ।
- ਕਿਉਕਿ ਮਸ਼ੀਨ ਹਦਾਇਤਾਂ ਨੂੰ ਬਿਨਾਂ ਕਿਸੇ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਦੇ ਕੰਪਿਊਟਰ ਦੁਆਰਾ ਸਿੱਧੇ ਹੀ ਸਮਝਿਆ ਜਾਂਦਾ ਹੈ ਇਸ ਲਈ ਇਹਨਾਂ ਉੱਪਰ ਬਿਨ੍ਹਾਂ ਕਿਸੇ ਦੇਰੀ ਦੇ ਕੰਪਿਊਟਰ ਵੱਲੋਂ ਤੇਜ਼ੀ ਨਾਲ ਕੰਮ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਦੀਆਂ ਹਾਨੀਆਂ (Disadvantages of Machine Language) :

- ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਦੀਆਂ ਹਦਾਇਤਾਂ ਨੂੰ ਯਾਦ ਰੱਖਣਾ ਬਹੁਤ ਔਖਾ ਹੁੰਦਾ ਹੈ ਕਿਉਕਿ ਇਹ ਹਦਾਇਤਾਂ ਸਿਰਫ ਬਾਈਨਰੀ ਅੰਕਾਂ 0 ਅਤੇ 1 ਤੋਂ ਮਿਲ ਕੇ ਬਣਦੀਆਂ ਹਨ।
- ਮਸ਼ੀਨ ਹਦਾਇਤਾਂ ਵਿਚ ਗਲਤੀਆਂ ਲੱਭਣਾ ਬਹੁਤ ਔਖਾ ਹੁੰਦਾ ਹੈ ਅਤੇ ਇਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਲੱਭਣ ਵਿਚ ਸਮਾਂ ਵੀ ਬਹੁਤ ਲੱਗਦਾ ਹੈ ਕਿਉਂਕਿ ਸਾਰੀਆਂ ਹਦਾਇਤਾਂ ਬਾਈਨਰੀ ਅੰਕਾਂ 0 ਅਤੇ 1 ਵਿੱਚ ਹੁੰਦੀਆਂ ਹਨ।
- ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ ਲਈ ਹਾਰਡਵੇਅਰ ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ ਦੀ ਜਾਣਕਾਰੀ ਹੋਣਾ ਜ਼ਰੂਰੀ ਹੁੰਦਾ ਹੈ। ਇਸ ਲਈ ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ ਲਈ ਪ੍ਰੋਗਰਾਮਰ ਦਾ ਹਾਰਡਵੇਅਰ ਵਿਚ ਤਕਨੀਕੀ ਤੌਰ ਤੇ ਕੁਸ਼ਲ ਹੋਣਾ ਲਾਜ਼ਮੀ ਹੈ।
- ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਵਿਚ ਬਣਾਇਆ ਗਿਆ ਪ੍ਰੋਗਰਾਮ ਕੇਵਲ ਉਸੇ ਤਰ੍ਹਾਂ ਦੀ ਮਸ਼ੀਨ ਉੱਪਰ ਹੀ ਚਲਾਏ ਜਾ ਸਕਦੇ ਹਨ ਜਿਸ ਤਰ੍ਹਾਂ ਦੀ ਮਸ਼ੀਨ ਉੱਪਰ ਉਸਨੂੰ ਬਣਾਇਆ ਗਿਆ ਹੈ। ਇਹਨਾਂ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਭਿੰਨ ਬਣਤਰ ਵਾਲੇ ਕੰਪਿਊਟਰਾਂ ਉੱਪਰ ਨਹੀਂ ਚਲਾਇਆ ਜਾ ਸਕਦਾ। ਇਸ ਤਰ੍ਹਾਂ ਦੇ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਮਸ਼ੀਨ ਡਿਪੈਂਡੈਂਟ (machine dependent) ਪ੍ਰੋਗਰਾਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

(ii) ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ (Assembly Language) : ਇਸ ਭਾਸ਼ਾ ਨੂੰ ਚਿੰਨ੍ਹਾਤਮਕ ਭਾਸ਼ਾ (**symbolic Language**) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਕਿਉਕਿ ਇਸ ਭਾਸ਼ਾ ਵਿਚ ਬਾਈਨਰੀ ਕੋਡ ਦੀ ਬਜਾਏ ਹਦਾਇਤਾਂ ਦੇ ਚਿੰਨ੍ਹਾਤਮਕ ਨਾਂਵਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਸ ਭਾਸ਼ਾ ਨੂੰ ਕੰਪਿਊਟਰ ਦੀਆਂ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀ ਦੂਜੀ ਜੈਨਰੇਸ਼ਨ ਦੀ ਭਾਸ਼ਾ ਵੀ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਦੀ ਤੁਲਨਾ ਵਿਚ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦਾ ਮੁੱਖ ਲਾਭ ਇਹ ਹੈ ਕਿ ਇਸ ਨਾਲ ਕੋਡਿੰਗ ਕਰਨ ਵਿਚ ਲੱਗਣ ਵਾਲਾ ਸਮਾਂ ਘੱਟ ਜਾਂਦਾ ਹੈ ਅਤੇ ਪ੍ਰੋਗਰਾਮਰ ਦੁਆਰਾ ਯਾਦ ਰੱਖਣ ਵਾਲੀ ਜਾਣਕਾਰੀ ਵੀ ਘੱਟ ਜਾਂਦੀ ਹੈ।ਹਦਾਇਤਾਂ ਦੇ ਚਿੰਨ੍ਹਾਤਮਕ ਨਾਂਵਾਂ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਯਾਦ ਰੱਖਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਸ ਲਈ ਇਸ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚੋਂ ਗਲਤੀਆਂ ਲੱਭਣਾ ਅਤੇ ਉਹਨਾਂ ਨੂੰ ਠੀਕ ਕਰਨਾ ਆਸਾਨ ਹੁੰਦਾ ਹੈ।

ਇਹਨਾਂ ਗੁਣਾਂ ਦੇ ਬਾਵਜੂਦ, ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ ਕਰਨ ਲਈ ਅਜੇ ਵੀ ਹਾਰਡਵੇਅਰ ਭਾਗਾਂ ਦੀ ਅੰਦਰੂਨੀ ਲੈਵਲ ਦੀ ਤਕਨੀਕੀ ਜਾਣਕਾਰੀ ਰੱਖਣਾ ਬਹੁਤ ਜ਼ਰੂਰੀ ਹੈ। ਇਸ ਲਈ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ ਕਰਨ ਲਈ ਮਸ਼ੀਨ ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ ਸੰਬੰਧੀ ਜਾਣਕਾਰੀ ਹੋਣਾ ਜ਼ਰੂਰੀ ਹੈ। ਇਸ ਤਕਨੀਕੀ ਅੜਚਨ ਕਾਰਨ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਗਏ ਪ੍ਰੋਗਰਾਮ ਅਜੇ ਵੀ ਮਸ਼ੀਨ ਉੱਪਰ ਨਿਰਭਰ (ਮਸ਼ੀਨ-ਡਿਪੈਂਡੈਂਟ) ਹੁੰਦੇ ਹਨ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੋਇਆ ਕਿ ਇਹਨਾਂ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਭਿੰਨ੍ਹ ਬਣਤਰ ਵਾਲੇ ਕੰਪਿਊਟਰ ਸਿਸਟਮਾਂ ਉੱਪਰ ਨਹੀਂ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਓਪਰੇਸ਼ਨ ਕੋਡਸ (operation codes) ਲਈ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਚਿੰਨ੍ਹਾਤਮਕ ਨਾਵਾਂ ਨੂੰ ਨਮੋਨਿੱਕ ਕੋਡਸ (**Mnemonic Codes**) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ: ਜੋੜ, ਘਟਾਓ, ਗੁਣਾ ਅਤੇ ਭਾਗ ਕੰਮਾਂ ਲਈ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ADD, SUB, MUL ਅਤੇ DIV ਚਿੰਨ੍ਹਾਤਮਕ ਨਾਵਾਂ ਨੂੰ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

ਹੁਣ ਪ੍ਰਸ਼ਨ ਇਹ ਉਠੰਦਾ ਹੈ ਕਿ ਕੰਪਿਊਟਰ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੇ ਇਹਨਾਂ ਕੋਡਸ ਉੱਪਰ ਕੰਮ ਕਿਵੇਂ ਕਰਦਾ ਹੈ ਕਿਉਂਕਿ ਕੰਪਿਊਟਰ ਸਿਰਫ ਬਾਈਨਰੀ ਰੂਪ ਵਿਚ ਪ੍ਰਾਪਤ ਹਦਾਇਤਾਂ ਉੱਪਰ ਹੀ ਕੰਮ ਕਰ ਸਕਦਾ ਹੈ। ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿੱਚ ਲਿਖੀਆਂ ਹਦਾਇਤਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਉੱਪਰ ਚਲਾਉਣ ਲਈ, ਇਹਨਾਂ ਨੂੰ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਕੋਡ ਵਿਚ ਟਾਂਸਲੇਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਟ੍ਰਾਂਸਲੇਸ਼ਨ (ਅਨੁਵਾਦ) ਲਈ ਇਕ ਟ੍ਰਾਂਸਲੇਟਰ ਪ੍ਰੋਗਰਾਮ, ਜਿਸਦਾ ਨਾਂ ਅਸੈਂਬਲਰ ਹੈ, ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਅਸੈਂਬਲਰ ਇਕ ਭਾਸ਼ਾ ਟ੍ਰਾਂਸਲੇਟਰ ਪ੍ਰੋਗਰਾਮ ਹੈ ਜੋ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਟ੍ਰਾਂਸਲੇਟ ਕਰਦੇ ਹਨ। ਇਸ ਪਾਠ ਦੇ ਅਗਲੇ ਭਾਗਾਂ ਵਿਚ ਅਸੀਂ ਅਸੈਂਬਲਰ ਬਾਰੇ ਵਿਸਥਾਰ ਵਿਚ ਚਰਚਾ ਕਰਾਂਗੇ। ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵੀ ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਵਾਲੀਆਂ ਕਈ ਹਾਨੀਆਂ ਨਾਲ ਪ੍ਰਭਾਵਿਤ ਹੈ। ਪਰ ਇਸ ਭਾਸ਼ਾ ਦੇ ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਦੀ ਤੁਲਨਾ ਵਿਚ ਆਪਣੇ ਕਈ ਲਾਭ ਵੀ ਹਨ। ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੇ ਇਹ ਲਾਭ ਅਤੇ ਹਾਨੀਆਂ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ:

ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੇ ਲਾਭ (Advantages of Assembly Language) :

- ਇਸ ਭਾਸ਼ਾ ਨੂੰ ਸਿੱਖਣਾ ਅਤੇ ਇਸਦੇ ਕੋਡਜ਼ ਨੂੰ ਯਾਦ ਰੱਖਣਾ ਆਸਾਨ ਹੁੰਦਾ ਹੈ ਕਿਉਂਕਿ ਇਹ ਭਾਸ਼ਾ ਬਾਈਨਰੀ ਅੰਕਾਂ ਦੀ ਬਜਾਏ ਅੰਗਰੇਜ਼ੀ ਭਾਸ਼ਾ ਵਿਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਕੋਡਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ।
- ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਗਲਤੀਆਂ ਲੱਭਣਾ ਅਤੇ ਉਹਨਾਂ ਨੂੰ ਠੀਕ ਕਰਨਾ ਆਸਾਨ ਹੁੰਦਾ ਹੈ। ।
- ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮ ਵੀ ਕੰਮ ਕਰਨ ਵਿਚ ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮਾਂ ਜਿੰਨ੍ਹਾਂ ਹੀ ਨਿਪੁੰਨ ਹੁੰਦੇ ਹਨ

ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੀਆਂ ਹਾਨੀਆਂ (Disadvantages of Assembly Language):

- ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ ਲਈ ਹਾਰਡਵੇਅਰ ਦੇ ਹੇਠਲੇ ਪੱਧਰ ਦੇ ਅੰਦਰੂਨੀ ਵੇਰਵਿਆਂ ਦੀ ਜਾਣਕਾਰੀ ਹੋਣਾ ਜ਼ਰੂਰੀ ਹੈ। । ਇਸ ਲਈ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ ਕਰਨ ਲਈ ਹਾਰਡਵੇਅਰ ਨਾਲ ਸੰਬਧਿਤ ਤਕਨੀਕੀ ਕੁਸ਼ਲਤਾ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ।
- ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿੱਚ ਤਿਆਰ ਕੀਤੇ ਗਏ ਪ੍ਰੋਗਰਾਮ ਮਸ਼ੀਨ ਉੱਪਰ ਨਿਰਭਰ (ਮਸ਼ੀਨ ਡਿਪੈਂਡੈਂਟ) ਹੁੰਦੇ ਹਨ ਕਿਉਂਕਿ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੀਆਂ ਹਦਾਇਤਾਂ ਉਸ ਕੰਪਿਊਟਰ ਦੀ ਬਣਤਰ ਉੱਪਰ ਅਧਾਰਿਤ ਹੁੰਦੀਆਂ ਹਨ ਜਿਸ ਉੱਪਰ ਉਹਨਾਂ ਨੂੰ ਬਣਾਇਆ ਗਿਆ ਹੁੰਦਾ ਹੈ। ਇਹਨਾਂ ਹਦਾਇਤਾਂ ਨੂੰ ਭਿੰਨ੍ਹ ਬਣਤਰ ਵਾਲੇ ਕੰਪਿਊਟਰਾਂ ਉੱਪਰ ਨਹੀਂ ਚਲਾਇਆ ਜਾ ਸਕਦਾ।

ਅ. ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ (High Level Languages) :

ਜਿਵੇਂ ਕਿ ਉੱਪਰ ਦਿੱਤੀ ਚਰਚਾ ਦਰਸ਼ਾ ਰਹੀ ਹੈ ਕਿ ਲੋਅ-ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਪ੍ਰੋਗਰਾਮਰ ਦੁਆਰਾ ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰਨ ਲਈ ਹਾਰਡਵੇਅਰ ਭਾਗਾਂ ਦੀ ਲੋਅ-ਲੇਵਲ ਅੰਦਰੂਨੀ ਬਣਤਰ ਸੰਬੰਧੀ ਜਾਣਕਾਰੀ ਹੋਣਾ ਜਰੂਰੀ ਹੈ। ਇਸਦੇ ਉਲਟ, ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ ਲਈ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਕਿਸੇ ਹਾਰਡਵੇਅਰ ਭਾਗਾਂ ਦੀ ਲੋਅ ਲੇਵਲ ਅੰਦਰੂਨੀ ਬਣਤਰ ਸੰਬੰਧੀ ਜਾਣਕਾਰੀ ਹੋਣਾ ਲਾਜ਼ਮੀ ਨਹੀਂ ਹੈ। ਇਹੀ ਕਾਰਨ ਹੈ ਕਿ ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ।

ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਤਿਆਰ ਕਰਨ ਦਾ ਮੁੱਖ ਮਕਸਦ ਇਹ ਹੈ ਕਿ ਇਹ ਭਾਸ਼ਾਵਾਂ ਲੋਕਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਹਾਰਡਵੇਅਰ ਸੰਬੰਧੀ ਲੋਅ-ਲੇਵਲ ਅੰਦਰੂਨੀ ਜਾਣਕਾਰੀ ਪ੍ਰਾਪਤ ਕੀਤੇ ਬਿਨ੍ਹਾਂ ਪ੍ਰੋਗਰਾਮ ਜਾਂ ਸਾਫਟਵੇਅਰ ਬਨਾਉਣ ਦੀ ਸਹੂਲਤ ਪ੍ਰਦਾਨ ਕਰਦੀਆਂ ਹਨ। ਇਹ ਭਾਸ਼ਾਵਾਂ ਮਸ਼ੀਨ-ਇੰਡੀਪੈਨਡੈਂਟ (Machine Independent) ਹੁੰਦੀਆਂ ਹਨ। ਦੂਸਰੇ ਸ਼ਬਦਾ ਵਿੱਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਲਿਖੇ ਗਏ ਪ੍ਰੋਗਰਾਮ ਕਿਸੇ ਮਸ਼ੀਨ ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ ਉੱਪਰ ਨਿਰਭਰ ਨਹੀਂ ਕਰਦੇ ਅਤੇ ਇਹੀ ਕਾਰਨ ਹੈ ਕਿ ਇਹ ਪ੍ਰੋਗਰਾਮ ਵੱਖ-ਵੱਖ ਬਣਤਰਾਂ ਵਾਲੇ ਕੰਪਿਊਟਰਾਂ ਉਪਰ ਵੀ ਚਲਾਏ ਜਾ ਸਕਦੇ ਹਨ।

ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਅੰਗਰੇਜ਼ੀ ਭਾਸ਼ਾ ਵਰਗੀਆਂ ਹੁੰਦੀਆਂ ਹਨ। ਇਹ ਭਾਸ਼ਾਵਾਂ ਪ੍ਰੋਗਰਾਮਿੰਗ ਲਈ ਅਲਫਾਨੁਮੈਰਿਕ ਚਿੰਨ੍ਹਾਂ (alphanumeric symbols) ਦੀ ਵਰਤੋਂ ਕਰਦੀਆਂ ਹਨ। ਇਸ ਲਈ ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਸਿੱਖਣਾ ਅਤੇ ਇਹਨਾਂ ਵਿਚ

ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰਨਾ ਸੌਖਾ ਹੁੰਦਾ ਹੈ। ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੀ ਗਈ ਹਦਾਇਤ ਨੂੰ ਆਮ ਤੌਰ ਤੇ ਸਟੇਟਮੈਂਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਹਰ ਇਕ ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਹਦਾਇਤਾਂ ਲਿਖਣ ਦੇ ਆਪਣੇ ਬਣਤਰ ਅਤੇ ਗਰਾਮਰ ਦੇ ਨਿਯਮ ਹੁੰਦੇ ਹਨ। ਇਹਨਾਂ ਨਿਯਮਾਂ ਨੂੰ ਭਾਸ਼ਾ ਦੇ ਸਿੰਟੈਕਸ (Syntax) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। PYTHON, BASIC, COBOL, FORTRAN, PASCAL, C, C++, JAVA, C SHARP ਆਦਿ ਕੁੱਝ ਆਮ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ

ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੀ ਤਰ੍ਹਾਂ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਵੀ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੁਆਰਾ ਸਿੱਧੇ ਤੌਰ ਤੇ ਨਹੀਂ ਸਮਝਿਆ ਜਾ ਸਕਦਾ। ਭਾਸ਼ਾ ਟਾਂਸਲੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਲਿਖੇ ਕੋਡਜ਼ ਨੂੰ ਮਸ਼ੀਨ ਦੁਆਰਾ ਸਮਝਣ ਯੋਗ ਕੋਡਜ਼ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਦੇ ਕੋਡਜ਼ ਨੂੰ ਮਸ਼ੀਨ ਕੋਡਜ਼ ਵਿਚ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨ ਦੀਆਂ ਦੋ ਵਿਧੀਆਂ ਹਨ: ਪਹਿਲੀ ਵਿਧੀ ਕੰਪਾਈਲਰ (Compiler) ਦੀ ਵਰਤੋਂ ਨਾਲ ਦੂਜੀ ਇੰਟਰਪ੍ਰੈਟਰ (Interpreter) ਦੀ ਵਰਤੋਂ ਨਾਲ। ਇਹਨਾਂ ਟਾਂਸਲੇਟਰਾਂ ਦੀ ਵਿਸਥਾਰ ਵਿਚ ਵਿਆਖਿਆ ਇਸ ਪਾਠ ਦੇ ਆਉਣ ਵਾਲੇ ਭਾਗਾਂ ਵਿਚ ਕੀਤੀ ਗਈ ਹੈ। ਹਰ ਇਕ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾ ਦਾ ਆਪਣਾ ਅਲੱਗ ਕੰਪਾਈਲਰ ਹੁੰਦਾ ਹੈ। ਅਸੀਂ ਇਕ ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕਿਸੇ ਦੂਸਰੀ ਭਾਸ਼ਾ ਦੇ ਕੰਪਾਈਲਰ ਨਾਲ ਟਾਂਸਲੇਟ ਨਹੀਂ ਕਰ ਸਕਦੇ। ਉਦਾਹਰਣ ਲਈ: ਅਸੀਂ ਸੀ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ COBOL ਭਾਸ਼ਾ ਦੇ ਕੰਪਾਈਲਰ ਨਾਲ ਟ੍ਰਾਂਸਲੇਟ ਨਹੀਂ ਕਰ ਸਕਦੇ ਅਤੇ ਨਾਂ ਹੀ ਇਸਦੇ ਉਲਟ COBOL ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸੀ-ਭਾਸ਼ਾ ਦੇ ਕੰਪਾਈਲਰ ਨਾਲ ਟ੍ਰਾਂਸਲੇਟ ਕਰ ਸਕਦੇ ਹਾਂ।

ਜਿਵੇਂ ਕਿ ਉਪਰ ਦੱਸਿਆ ਗਿਆ ਹੈ, ਬਹੁਤ ਸਾਰੀਆਂ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਮੌਜੂਦ ਹਨ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਮਾਪਦੰਡਾਂ ਅਨੁਸਾਰ ਵੱਖ-ਵੱਖ ਸ਼੍ਰੇਣੀਆਂ ਵਿਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਹੇਠਾਂ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਕੁੱਝ ਮੁੱਖ ਸ਼੍ਰੇਣੀਆਂ ਦੀ ਚਰਚਾ ਕੀਤੀ ਗਈ ਹੈ:

(I) **ਪ੍ਰੋਸੀਜ਼ਰਲ ਜਾਂ ਪ੍ਰੋਸੀਜ਼ਰ ਓਰੀਐਂਟੇਡ ਭਾਸ਼ਾਵਾਂ (Procedural or Procedure Oriented Languages)** : ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਤੀਸਰੀ ਜੈਨਰੇਸ਼ਨ ਦੀਆਂ ਭਾਸ਼ਾਵਾਂ **(3GLs)** ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਪ੍ਰੋਸੀਜ਼ਰਲ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਲਈ ਉਸਨੂੰ ਛੋਟੇ ਪ੍ਰੋਸੀਜ਼ਰਾਂ ਜਾਂ ਸਬਰੁਟੀਨਾਂ (procedures or subroutines) ਵਿਚ ਵੰਡ ਕੇ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਹਰ ਇਕ ਪ੍ਰੋਸੀਜ਼ਰ ਵਿਚ ਕਿਸੇ ਖਾਸ ਕੰਮ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਹਦਾਇਤਾਂ ਦੀ ਇਕ ਲੜੀ ਲਿਖੀ ਜਾਂਦੀ ਹੈ। ਪ੍ਰੋਸੀਜ਼ਰ ਬਨਾਉਣ ਤੋਂ ਬਾਅਦ ਅਸੀਂ ਇਹਨਾਂ ਨੂੰ ਇੱਕ ਜਾਂ ਇੱਕ ਤੋਂ ਵੱਧ ਵਾਰ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਕਿਸੇ ਵੀ ਥਾਂ ਤੇ ਵਰਤ ਸਕਦੇ ਹਾਂ। ਇਹ ਭਾਸ਼ਾਵਾਂ ਹੱਲ ਕੀਤੀ ਜਾਣ ਵਾਲੀ ਸਮੱਸਿਆ ਦੇ ਲੋਜਿਕ ਨੂੰ ਦਰਸ਼ਾਉਣ ਲਈ ਬਣਾਈਆਂ ਗਈਆਂ ਹਨ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਪ੍ਰੋਗਰਾਮ ਦੀਆਂ ਹਦਾਇਤਾਂ ਦਾ ਕੰਮ ਬਹੁਤ ਮਹੱਵਪੂਰਨ ਹੁੰਦਾ ਹੈ।FORTAN, COBOL, Pascal, C ਆਦਿ ਕੁੱਝ ਮਸ਼ਹੂਰ ਪ੍ਰੋਸੀਜ਼ਰਲ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ।

(ii) **ਪ੍ਰੋਬਲਮ ਓਰੀਐਂਟੇਡ ਜਾਂ ਨਾਨ-ਪ੍ਰੋਸੀਜ਼ਰਲ ਭਾਸ਼ਾਵਾਂ (Problem-Oriented or Non-Procedural Languages):** ਪ੍ਰੋਬਲਮ ਓਰੀਐਂਟੇਡ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਨਾਨ-ਪ੍ਰੋਸੀਜ਼ਰਲ ਭਾਸ਼ਾਵਾਂ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀ **ਚੌਥੀ ਜੈਨਰੇਸ਼ਨ ਦੀਆਂ ਭਾਸ਼ਾਵਾਂ (4GLs)** ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਹਦਾਇਤਾਂ ਦੇਣ ਲਈ ਸਾਧਾਰਣ, ਅੰਗਰੇਜ਼ੀ ਨਾਲ ਮਿਲਦੇ ਜੁਲਦੇ ਨਿਯਮ ਵਰਤੇ ਜਾਂਦੇ ਹਨ ਅਤੇ ਇਹਨਾਂ ਨੂੰ ਡਾਟਾਬੇਸ ਅਸੈਸ (database access) ਕਰਨ ਲਈ ਵੀ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਕਿਸੇ ਕੰਮ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਇੱਕਲਾ-ਇੱਕਲਾ ਸਟੈਪ ਦੱਸਣ ਦੀ ਬਜਾਏ ਇਹ ਦੱਸਿਆ ਜਾਂਦਾ ਹੈ ਕਿ ਸਾਨੂੰ ਪ੍ਰੋਗਰਾਮ ਦੀ ਆਊਟਪੁੱਟ ਕੀ ਚਾਹੁੰਦੀ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੋਇਆ ਕਿ ਸਾਨੂੰ ਨਤੀਜਾ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਡਾਟਾ ਉਪਰ ਕੰਮ ਕਰਨ ਦੇ ਸਾਰੇ ਵੇਰਵੇ ਦੱਸਣ ਦੀ ਜਰੂਰਤ ਨਹੀਂ ਹੈ। ਇਹ ਭਾਸ਼ਾਵਾਂ ਤੀਸਰੀ ਜੈਨਰੇਸ਼ਨ ਦੀਆਂ ਭਾਸ਼ਾਵਾਂ ਤੋਂ ਇਕ ਕਦਮ ਅੱਗੇ ਹਨ। ਇਹ ਭਾਸ਼ਾਵਾਂ ਪ੍ਰੋਗਰਾਮ ਲਿਖਣ ਵੇਲੇ ਹਦਾਇਤਾਂ ਦੇਣ ਲਈ ਯੂਜ਼ਰ-ਫਰੈਂਡਲੀ ਟੂਲਜ਼ (user-friendly tools) ਮੁਹੱਈਆ ਕਰਵਾਉਂਦੀਆਂ ਹਨ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਯੂਜ਼ਰ ਐਪਲੀਕੇਸ਼ਨ ਜੈਨਰੇਟਰ (application generator) ਦੀ ਮਦਦ ਨਾਲ ਪ੍ਰੋਗਰਾਮ ਲਿਖਦੇ ਹਨ ਤਾਂ ਜੋ ਡਾਟਾ ਨੂੰ ਡਾਟਾਬੇਸ ਵਿੱਚ ਸਟੋਰ ਕਰਵਾਇਆ ਜਾ ਸਕੇ। ਪ੍ਰੋਗਰਾਮ ਯੂਜ਼ਰ ਨੂੰ ਡਾਟਾ ਭਰਨ ਲਈ ਕਹਿੰਦਾ ਹੈ ਅਤੇ ਡਾਟਾ ਲੈਣ ਤੋਂ ਬਾਅਦ ਇਹ ਡਾਟਾ ਦੀ ਵੈਦਤਾ (Validity) ਨੂੰ ਚੈਕ ਕਰਦਾ ਹੈ। SQL (Structure Query Language), Visual Basic, C# ਆਦਿ ਭਾਸ਼ਾਵਾਂ ਪ੍ਰੋਬਲਮ ਓਰੀਐਂਟੇਡ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਦਾ ਮਕਸਦ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਦੀ ਰਫ਼ਤਾਰ ਵਧਾਉਣਾ ਅਤੇ ਪ੍ਰੋਗਰਾਮ ਲਿਖਣ ਵੇਲੇ ਹੋਣ ਵਾਲੀਆਂ ਗਲਤੀਆਂ ਨੂੰ ਘਟਾਉਣਾ ਹੈ।।

- ਆਬਜੈਕਟ - ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ (Object-Oriented Programming Langauges): ਆਬਜੈਕਟ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸੰਕਲਪ 1960 ਦੇ ਅਖੀਰ ਵਿੱਚ ਪੇਸ਼ ਕੀਤਾ ਗਿਆ ਸੀ। ਪਰ ਹੁਣ ਇਹ ਸਾਫਟਵੇਅਰ ਬਨਾਉਣ ਦੀ ਸਭ ਤੋਂ ਵੱਧ ਮਸ਼ਹੂਰ ਵਿਧੀ ਬਣ ਚੁੱਕੀ ਹੈ। ਇਹਨਾਂ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਕਿਸੇ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਉਸਨੂੰ ਆਬਜੈਕਟਾਂ ਵਿਚ ਵੰਡਿਆ ਜਾਂਦਾ ਹੈ। ਆਬਜੈਕਟ ਓਰੀਐਂਟੇਡ ਭਾਸ਼ਾਵਾਂ ਆਬਜੈਕਟ (Object), ਕਲਾਸ (Class), ਏਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation), ਡਾਟਾ ਹਾਈਡਿੰਗ (Data Hiding), ਇਨਹੈਰੀਟੈਂਸ (Inheritence) ਅਤੇ ਪਾਲੀਮਾਰਫੀਜ਼ਮ (Polymorphisms) ਆਦਿ ਸੰਕਲਪਾਂ ਦਾ ਸਮਰਥਣ ਕਰਦੀਆਂ ਹਨ। ਅੱਜ ਕੱਲ C++ ਅਤੇ JAVA ਸਭ ਤੋਂ ਵੱਧ ਮਸ਼ਹੂਰ ਅਤੇ ਜ਼ਿਆਦਾ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਆਬਜੈਕਟ ਓਰੀਐਂਟੇਡ ਭਾਸ਼ਾਵਾਂ ਹਨ।
- ਲਾਜੀਕ ਓਰੀਐਂਟੇਡ ਭਾਸ਼ਾਵਾਂ (Logic Oriented Languages): ਇਹ ਭਾਸ਼ਾਵਾਂ ਵੱਖ ਵੱਖ ਕੰਪਿਊਟੇਸ਼ਨਲ ਸਮੱਸਿਆਵਾਂ ਦੇ ਹੱਲ ਲਈ ਡਿਜ਼ਾਈਨ ਵਿਧੀ ਵਜੋਂ ਲਾਜੀਕ ਪ੍ਰੋਗਰਾਮਿੰਗ ਮਾਡਲ ਦੇ ਤੌਰ ਤੇ ਵਰਤਦੀਆਂ ਜਾਂਦੀਆਂ ਹਨ। ਲਾਜੀਕ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਲਿਖਿਆ ਕੋਈ ਵੀ ਪ੍ਰੋਗਰਾਮ ਲਾਜ਼ੀਕਲ ਰੂਪ ਵਿੱਚ ਵਾਕਾਂ (Statement) ਦਾ ਸਮੂਹ ਹੁੰਦਾ ਹੈ। ਇਹ ਵਾਕ ਕਿਸੇ ਸਮੱਸਿਆ ਦੇ ਤੱਥਾਂ ਅਤੇ ਨਿਯਮਾਂ ਨੂੰ ਦਰਸਾਉਂਦੇ ਹਨ। ਮੁੱਖ ਲਾਜੀਕ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ PROLOG, Answer set programming (ASP) ਅਤੇ Datalog ਭਾਸ਼ਾਵਾਂ ਸ਼ਾਮਲ ਹਨ। ਇਨ੍ਹਾਂ ਸਾਰੀਆਂ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਨਿਯਮਾਂ ਨੂੰ ਕਲਾਜਿਸ (Clauses) ਦੇ ਰੂਪ ਵਿੱਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ। ਅਜਿਹੀਆਂ ਭਾਸ਼ਾਵਾਂ Artificial Intelligence ਅਤੇ Robotics ਦੇ ਖੇਤਰ ਵਿੱਚ ਬਹੁਤ ਫਾਇਦੇਮੰਦ ਹਨ।

ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਂਵਾਂ ਦੀਆਂ ਹਾਨੀਆਂ (Advantages of High-Level Languages)

ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਦੇ ਕੁੱਝ ਮਹੱਤਵਪੂਰਣ ਲਾਭ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹਨ:

- ਲੋਅ-ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਦੇ ਮੁਕਾਬਲੇ ਹਾਈ-ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਸਿੱਖਣਾ ਅਤੇ ਸਮਝਣਾ ਆਸਾਨ ਹੈ। ਇਸ ਦਾ ਕਾਰਨ ਇਹ ਹੈ ਕਿ ਇਨ੍ਹਾਂ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਅੰਗਰੇਜ਼ੀ ਦੇ ਵਾਕਾਂ ਵਰਗੇ ਹੁੰਦੇ ਹਨ।
- ਇੱਕ ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਗਲਤੀਆਂ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਲੱਭਿਆ ਅਤੇ ਠੀਕ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਜਿਆਦਾਤਰ ਗਲਤੀਆਂ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਈਲ ਕਰਦੇ ਸਮੇਂ ਲੱਭੀਆਂ ਅਤੇ ਠੀਕ ਕਰ ਦਿੱਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।
- ਇਹ ਭਾਸ਼ਾਵਾਂ ਬਹੁਤ ਸਾਰੇ ਬਿਲਟ-ਇਨ ਫੰਕਸ਼ਨ ਪ੍ਰਦਾਨ ਕਰਦੀਆਂ ਹਨ ਜਿਹਨਾਂ ਨੂੰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੌਰਾਨ ਖਾਸ ਕੰਮ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਬਿਲਟ-ਇਨ ਫੰਕਸ਼ਨਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਪ੍ਰੋਗਰਾਮਰ ਦਾ ਬਹੁਤ ਸਾਰਾ ਸਮਾਂ ਬਚਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।
- ਉੱਚ-ਪੱਧਰੀ ਭਾਸ਼ਾ ਵਿੱਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਮਸ਼ੀਨ-ਇੰਡੀਪੈਨਡੈਂਟ ਹੁੰਦੇ ਹਨ। ਇੱਕ ਕੰਪਿਊਟਰ ਉੱਪਰ ਲਿਖਿਆ ਪ੍ਰੋਗਰਾਮ ਹੋਰ ਭਿੰਨ ਬਣਤਰਾਂ ਵਾਲੇ ਕੰਪਿਊਟਰਾਂ ਉੱਤੇ ਥੋੜੀ-ਬਹੁਤੀ ਤਬਦੀਲੀ ਨਾਲ ਜਾਂ ਬਿਨ੍ਹਾਂ ਕਿਸੇ ਤਬਦੀਲੀ ਕੀਤੇ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਂਵਾਂ ਦੀਆਂ ਹਾਨੀਆਂ (Disadvantages of High-Level Languages)

ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਕੁੱਝ ਮਹੱਤਵਪੂਰਣ ਕਮੀਆਂ ਹੇਠਾਂ ਦਿੱਤੀਆਂ ਗਈਆਂ ਹਨ:

- ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਦੀ ਕੁਸ਼ਲਤਾ ਅਤੇ ਸਪੀਡ ਲੋਅ-ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮਾਂ ਦੀ ਤੁਲਨਾ ਵਿੱਚ ਘੱਟ ਹੁੰਦੀ ਹੈ।
- ਉੱਚ ਪੱਧਰੀ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਵਿੱਚ ਵਧੇਰੇ ਸਮਾਂ ਲੱਗਦਾ ਹੈ ਅਤੇ ਇਹਨਾਂ ਨੂੰ ਕੰਮ ਕਰਨ ਲਈ ਵਧੇਰੇ ਮੈਮੋਰੀ ਦੀ ਲੋੜ ਵੀ ਹੁੰਦੀ ਹੈ।
- ਲੋਅ-ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਦੇ ਮੁਕਾਬਲੇ ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਘੱਟ ਲਚਕਦਾਰ ਹੁੰਦੀਆਂ ਹਨ ਕਿਉਂਕਿ ਇਨਬਾਂ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਕੰਪਿਊਟਰ ਦੇ ਹਾਰਡਵੇਅਰ, ਜਿਵੇਂ ਕਿ: ਸੀ.ਪੀ.ਯੂ., ਮੈਮੋਰੀ ਅਤੇ ਰਜਿਸਟਰਾਂ ਆਦਿ ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਵਾਲੀਆਂ ਹਦਾਇਤਾਂ ਨਹੀਂ ਹੁੰਦੀਆਂ।

3.4 ਭਾਸ਼ਾ ਟ੍ਰਾਂਸਲੇਟਰਜ਼ (LANGUAGE Processor)

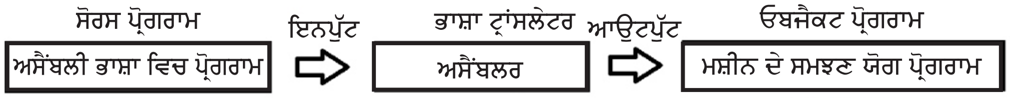
ਭਾਸ਼ਾ ਟ੍ਰਾਂਸਲੇਟਰਾਂ ਨੂੰ ਭਾਸ਼ਾ ਪ੍ਰੋਸੈਸਰ (Language Processor) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਸਿਸਟਮ ਪ੍ਰੋਗਰਾਮ ਹੁੰਦੇ ਹਨ ਜੋ ਹੋਰ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਤਿਆਰ ਕਰਨ ਵਿਚ ਮਦਦ ਕਰਦੇ ਹਨ। ਭਾਸ਼ਾ ਟ੍ਰਾਂਸਲੇਟਰਾਂ ਨੂੰ ਤਿਆਰ ਕਰਨ ਦਾ ਮਕਸਦ ਦੋ ਮੁੱਖ ਕੰਮਾਂ ਨੂੰ ਕਰਵਾਉਣ ਹੈ ਜੋ ਕਿ ਹੇਠਾਂ ਲਿਖੇ ਹਨ:

- ਇਹ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮਾਂ (Source Programs) ਨੂੰ ਮਸ਼ੀਨੀ ਆਬਜੈਕਟ ਕੋਡ (Machine's Object Code) ਵਿੱਚ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨ ਲਈ ਤਿਆਰ ਕੀਤੇ ਗਏ ਹਨ। ਸੋਰਸ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਜਾਂ ਉੱਚ ਪੱਧਰੀ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਕਿ ਆਬਜੈਕਟ ਕੋਡ ਇੱਕ ਅਜਿਹਾ ਕੋਡ ਹੁੰਦਾ ਹੈ ਜਿਸ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਬਿਨਾਂ ਕਿਸੇ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਦੇ ਸਮਝ ਸਕਦਾ ਹੈ॥
- ਇਹਨਾਂ ਟ੍ਰਾਂਸਲੇਟਰ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਸਿੰਟੈਕਸ ਦੀਆਂ ਗਲਤੀਆਂ (Syntax Errors) ਦਾ ਪਤਾ ਲਗਾਉਣ ਲਈ ਵੀ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ। ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਦਾ ਆਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਫਲਤਾਪੂਰਵਕ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਕੇਵਲ ਤਾਂ ਹੀ ਹੋਵੇਗਾ ਜੇਕਰ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੋਈ ਸਿੰਟੈਕਸ ਦੀ ਗਲਤੀ ਨਾਂ ਹੋਵੇ।

ਹਰੇਕ ਭਾਸ਼ਾ ਦਾ ਆਪਣਾ ਟ੍ਰਾਂਸਲੇਟਰ ਪ੍ਰੋਗਰਾਮ ਹੁੰਦਾ ਹੈ ਜੋ ਸਿਰਫ ਉਸ ਖਾਸ ਭਾਸ਼ਾ ਵਿੱਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਹੀ ਟ੍ਰਾਂਸਲੇਟ ਕਰ ਸਕਦਾ ਹੈ। ਅਸੈਂਬਲਰ ਇੱਕ ਟ੍ਰਾਂਸਲੇਟਰ ਪ੍ਰੋਗਰਾਮ ਹੈ ਜੋ ਸਿਰਫ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿੱਚ ਲਿਖੇ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਟਾਂਸਲੇਟ ਕਰ ਸਕਦਾ ਹੈ। ਇਸੇ ਤਰ੍ਹਾਂ ਹਰੇਕ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾ ਦਾ ਆਪਣਾ ਟ੍ਰਾਂਸਲੇਟਰ ਪ੍ਰੋਗਰਾਮ ਹੁੰਦਾ ਹੈ, ਜੋ ਕੰਪਾਈਲਰ ਅਤੇ ਇੰਟਰਪ੍ਰੈਟਰ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਕੁਝ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਕੰਪਾਈਲਰ ਦੀ ਵਰਤੋਂ ਕਰਦੀਆਂ ਹਨ (ਉਦਾਹਰਣ ਵਜੋਂ: C/C++ ਭਾਸ਼ਾ) ਜਦੋਂ ਕਿ ਕੁਝ ਹੋਰ ਇੰਟਰਪ੍ਰੈਟਰ ਦੀ ਵਰਤੋਂ ਕਰਦੀਆਂ ਹਨ (ਉਦਾਹਰਣ ਵਜੋਂ BASIC ਭਾਸ਼ਾ) ।ਪਰ ਕੁਝ ਭਾਸ਼ਾਵਾਂ ਅਜਿਹੀਆਂ ਵੀ ਹਨ ਜੋ ਵੱਖ-ਵੱਖ ਲੈਵਲ ਤੇ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਲਈ ਕੰਪਾਈਲਰ ਅਤੇ ਇੰਟਰਪ੍ਰੈਟਰ ਦੋਹਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੀਆਂ ਹਨ, ਉਦਾਹਰਣ ਵਜੋਂ JAVA ਇਕ ਅਜਿਹੀ ਭਾਸ਼ਾ ਹੈ ਜਿਸ ਵਿਚ ਕੰਪਾਈਲਰ ਅਤੇ ਇੰਟਰਪੇਟਰ ਦੋਹਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹਨਾਂ ਵੱਖ ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਟ੍ਰਾਂਸਲੇਟਰਾਂ ਦਾ ਵਰਨਣ ਹੇਠਾਂ ਦਿੱਤਾ ਗਿਆ ਹੈ:

3.4.1 ਅਸੈਂਬਲਰ (Assembler) :

ਇਹ ਇਕ ਭਾਸ਼ਾ ਟਾਂਸਲੇਟਰ ਹੈ ਜੋ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਵਿਚ ਤਬਦੀਲ ਕਰਦਾ ਹੈ। ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਗਏ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿੱਧੇ ਤੌਰ ਤੇ ਨਹੀਂ ਸਮਝ ਸਕਦਾ। ਇਸ ਲਈ ਇਸਨੂੰ ਚਲਾਉਣ ਲਈ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਫਾਰਮੈਟ ਵਿਚ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨਾ ਜਰੂਰੀ ਹੈ। ਅਸੈਂਬਜਰ ਟ੍ਰਾਂਸਲੇਟਰ ਹੀ ਹੈ ਜੋ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦੇ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣਯੋਗ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਤਬਦੀਲ ਕਰਦਾ ਹੈ। ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਦੀ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਤੋਂ ਬਾਅਦ ਬਣੇ ਕੋਡ ਨੂੰ ਆਬਜੈਕਟ ਕੋਡ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਅਤੇ ਇਹਨਾਂ ਆਬਜੈਕਟ ਕੋਡ ਦਾ ਸਮੂਹ ਆਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਅਖਵਾਉਂਦਾ ਹੈ।ਇਹ ਆਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਹੀ ਪ੍ਰੋਸੈਸਰ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਚਲਾਉਣ ਲਈ ਮੁਹੱਈਆ ਕਰਵਾਇਆ ਜਾਂਦਾ ਹੈ।



ਚਿੱਤਰ 3.4 ਅਸੈਂਬਲਰ ਦੀ ਕਾਰਜਪ੍ਰਣਾਲੀ

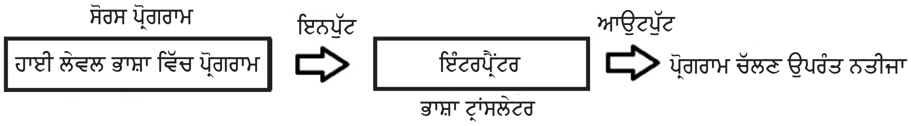
ਜਿਵੇਂ ਕਿ ਉਪਰ ਦਰਸ਼ਾਇਆ ਗਿਆ ਹੈ, ਅਸੈਂਬਲਰ ਨੂੰ ਇਨਪੁੱਟ ਦੇ ਤੌਰ ਤੇ ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਦਾ ਪ੍ਰੋਗਰਾਮ ਦਿਤਾ ਗਿਆ ਹੈ ਅਤੇ ਇਹ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣਯੋਗ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਆਉਟਪੁੱਟ ਦੇ ਤੌਰ ਤੇ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।

3.4.2 ਇੰਟਰਪ੍ਰੇਟਰ ਅਤੇ ਕੰਪਾਈਲਰ (Interpreter and Compiler) :

ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਲਈ ਭਾਸ਼ਾ ਟ੍ਰਾਂਸਲੇਟਰ ਕਰਨ ਦੀਆਂ ਦੋ ਤਕਨੀਕਾਂ ਹਨ: ਇੰਟਰਪ੍ਰੇਟਰ ਅਤੇ ਕੰਪਾਈਲਰ। ਇਹਨਾਂ ਦੋਵਾਂ ਦੀ ਵਰਤੋਂ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਲਿਖੇ ਗਏ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣਯੋਗ ਫਾਰਮੈਟ ਵਿਚ ਤਬਦੀਲ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ਪਹਿਲੀ ਤਕਨੀਕ, ਭਾਵ ਇੰਟਰਪੈਟਰ ਵਿਚ, ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਇਕ ਸਮੇਂ ਤੇ ਇਕ ਸਟੇਟਮੈਂਟ ਨੂੰ ਲਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ

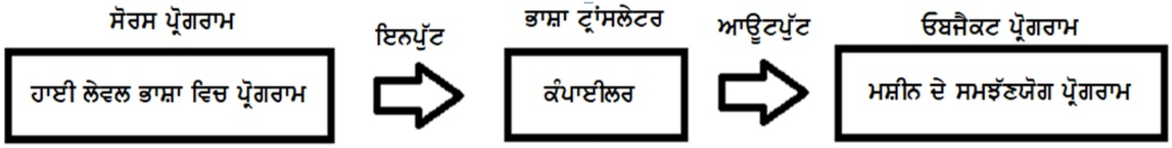
ਇਸਨੂੰ ਮਸ਼ੀਨੀ ਹਦਾਇਤਾਂ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਇਹਨਾਂ ਮਸ਼ੀਨੀ ਹਦਾਇਤਾਂ ਨੂੰ ਪ੍ਰੋਸੈਸਰ ਦੁਆਰਾ ਤੁਰੰਤ ਲਾਗੂ ਕਰ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਦਾ ਭਾਵ ਇਹ ਹੋਇਆ ਕਿ ਇਸ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਤਕਨੀਕ ਵਿਚ ਕੋਈ ਵੀ ਓਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਸੇਵ ਨਹੀਂ ਕੀਤਾ ਜਾਂਦਾ। ਜਦੋਂ ਵੀ ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਣਾ ਹੋਵੇਗਾ ਸਾਨੂੰ ਹਰ ਵਾਰ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨਾ ਪਿਆ ਕਰੇਗਾ।।



ਚਿੱਤਰ 3.5 ਇੰਟਰਪ੍ਰੇਟਰ ਦੀ ਕਾਰਜ ਪ੍ਰਣਾਲੀ

ਇੰਟਰਪ੍ਰੈਟਰ ਦੁਆਰਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਟ੍ਰਾਂਸਲੇਟ ਅਤੇ ਚਲਾਉਣ ਲਈ ਜਿਆਦਾ ਮੈਮਰੀ ਦੀ ਜਰੂਰਤ ਨਹੀਂ ਹੁੰਦੀ। ਇੰਟਰਪ੍ਰੈਟਰ ਦਾ ਮੁੱਖ ਨੁਕਸਾਨ ਇਹ ਹੈ ਕਿ ਉਹਨਾਂ ਨੂੰ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਉਪਰ ਪ੍ਰੋਗਰਾਮ ਚਲਾਉਣ ਲਈ ਜਿਆਦਾ ਸਮੇਂ ਦੀ ਜ਼ਰੂਰਤ ਪੈਂਦੀ ਹੈ ਕਿਉਂਕਿ ਜਦੋਂ ਵੀ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ ਇਸਦੇ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਦੀ ਹਰ ਸਟੇਟਮੈਂਟ ਨੂੰ ਹਰ ਵਾਰ ਟ੍ਰਾਂਸਲੇਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਦੂਸਰੀ ਵਿਧੀ, ਕੰਪਾਈਲਰ, ਵਿਚ ਹਾਈ ਲੈਵਲ ਪ੍ਰੋਗਰਾਮ ਦੀਆਂ ਸਾਰੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਇਕ ਸਮੇਂ ਵਿਚ ਹੀ ਇਕੱਠਾ ਪੜ੍ਹ ਕੇ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਰੂਪ ਵਿੱਚ ਇੱਕ ਵਾਰ ਵਿਚ ਹੀ ਟਾਂਸਲੇਟ ਕਰ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਮੈਮਰੀ ਵਿਚ ਓਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਦੇ ਰੂਪ ਵਿਚ ਸਟੋਰ ਕਰ ਦਿਤਾ ਜਾਂਦਾ ਹੈ। ਜਦੋਂ ਵੀ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ ਇਹ ਓਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਹੀ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਨੂੰ ਮੁਹਈਆ ਕਰਵਾਇਆ ਜਾਂਦਾ ਹੈ। ਇੰਟਰਪ੍ਰੈਟਰ ਦੇ ਮੁਕਾਬਲੇ ਵਿਚ ਕੰਪਾਈਲਰ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨ ਵਿਚ ਜਿਆਦਾ ਸਮਾਂ ਲੈਂਦੇ ਹਨ। ਪ੍ਰੰਤੂ ਕੰਪਾਈਲ ਕੀਤਾ ਗਿਆ ਓਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਇੰਟਰਪ੍ਰੈਟ ਕੀਤੇ ਗਏ ਪ੍ਰੋਗਰਾਮ ਦੀ ਤੁਲਨਾ ਵਿਚ ਬਹੁਤ ਤੇਜੀ ਨਾਲ ਚਲਦੇ ਹਨ।।



ਚਿੱਤਰ 3.6 ਕੰਪਾਇਲਰ ਦੀ ਕਾਰਜ ਪ੍ਰਣਾਲੀ

ਹਰ ਇਕ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾ ਦਾ ਆਪਣਾ ਕੰਪਾਈਲਰ ਹੁੰਦਾ ਹੈ। ਅਸੀਂ ਇਕ ਭਾਸ਼ਾ ਦੇ ਸੋਰਸ ਕੋਡ ਨੂੰ ਦੂਜੀ ਭਾਸ਼ਾ ਦੇ ਕੰਪਾਈਲਰ ਨਾਲ ਕੰਪਾਈਲ ਨਹੀਂ ਕਰ ਸਕਦੇ। ਉਦਾਹਰਣ ਲਈ: FORTRAN ਭਾਸ਼ਾ ਦੇ ਕੰਪਾਈਲਰ ਦੀ ਵਰਤੋਂ ਨਾਲ COBOL ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਸੋਰਸ ਕੋਡ ਨੂੰ ਕੰਪਾਈਲ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਅਤੇ ਨਾਂ ਹੀ ਇਸਦਾ ਉਲਟਾ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ।ਇੰਟਰਪ੍ਰੈਟਰ ਅਤੇ ਕੰਪਾਈਲਰ ਦੇ ਵਿਚਕਾਰ ਅੰਤਰ ਨੂੰ ਹੇਠਾਂ ਦਿੱਤੀ ਉਦਾਹਰਣ ਦੀ ਸਹਾਇਤਾ ਨਾਲ ਸਮਝਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਮੰਨ ਲਓ ਕਿ ਅਸੀਂ ਇੱਕ ਭਾਸ਼ਣ ਦਾ ਤਮਿਲ ਤੋਂ ਹਿੰਦੀ ਵਿੱਚ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ। ਅਸੀਂ ਇਸ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਨੂੰ ਕਰਨ ਲਈ ਦੋ ਤਰੀਕਿਆਂ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ: ਪਹਿਲੀ ਵਿਧੀ ਵਿਚ, ਟ੍ਰਾਂਸਲੇਟਰ ਇਕ ਵਾਕ ਸੁਣਦਾ ਹੈ ਅਤੇ ਤੁਰੰਤ ਇਸ ਦਾ ਹਿੰਦੀ ਵਿਚ ਟ੍ਰਾਂਸਲੇਟਰ ਕਰ ਦਿੰਦਾ ਹੈ। ਦੂਸਰੀ ਵਿਧੀ ਵਿਚ, ਟ੍ਰਾਂਸਲੇਟਰ ਤਮਿਲ ਵਿਚ ਸਾਰਾ ਪੈਰ੍ਹਾ ਸੁਣਦਾ ਹੈ ਅਤੇ ਫਿਰ ਹਿੰਦੀ ਦੇ ਬਰਾਬਰ ਦਾ ਪੈਰ੍ਹਾ ਟ੍ਰਾਂਸਲੇਟ ਕਰਕੇ ਦਿੰਦਾ ਹੈ। ਇੰਟਰਪੈਟਰ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਦੀ ਪਹਿਲੀ ਵਿਧੀ ਦੇ ਸਮਾਨ ਹੈ ਜਿਥੇ ਵਾਕ ਦਰ-ਵਾਕ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਕਿ ਕੰਪਾਈਲਰ ਦੂਜੀ ਵਿਧੀ ਦੇ ਸਮਾਨ ਹੈ ਜਿਥੇ ਪੂਰਾ ਪੈਰਾ ਇਕੋ ਕਦਮ ਵਿਚ ਟ੍ਰਾਂਸਲੇਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਕੁੱਝ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ, ਜਿਵੇਂ ਕਿ ਜਾਵਾ, ਵਿਚ ਕੰਪਾਇਲਰ (Compilar) ਅਤੇ ਇੰਟਰਪ੍ਰੈਟਰ (Interpretor) ਦੋਵਾਂ ਦੀ ਵਰਤੋਂ ਪ੍ਰੋਗਰਾਮ ਟਾਂਸਲੇਟ ਕਰਨ ਅਤੇ ਉਸਨੂੰ ਚਲਾਉਣ (Execution) ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਵਿਚ ਕੰਪਾਇਲਰ ਦੀ ਵਰਤੋਂ ਸੋਰਸ ਕੋਡ ਨੂੰ ਮਸ਼ੀਨ ਨਿਊਟ੍ਰਲ (Machine Neutral) ਕੋਡ, ਜੋ ਕਿ ਕਿਸੇ ਵੀ ਮਸ਼ੀਨ ਦੁਆਰਾ ਸਮਝਿਆ ਜਾ ਸਕਦਾ ਹੈ, ਵਿਚ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਕਿ ਇੰਟਰਪ੍ਰੈਟਰ ਦੀ ਵਰਤੋਂ ਕੰਪਾਇਲਡ ਮਸ਼ੀਨ ਨਿਊਟ੍ਰਲ ਕੋਡ ਨੂੰ ਅੰਡਰਲੇਇੰਗ (Underlying) ਮਸ਼ੀਨ ਅਨੁਸਾਰ ਟ੍ਰਾਂਸਲੇਟ ਕਰਕੇ ਉਸਨੂੰ ਚਲਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

3.5 ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰੋਸੈਸ (PROGRAMMING PROCESS)

ਅਸੀਂ ਜਾਣਦੇ ਹਾਂ ਕਿ ਕੰਪਿਊਟਰ ਨਾ ਤਾਂ ਆਪਣੇ ਆਪ ਸੋਚ ਸਕਦਾ ਹੈ ਅਤੇ ਨਾ ਹੀ ਕੋਈ ਫੈਸਲਾ ਲੈ ਸਕਦਾ ਹੈ। ਕਿਸੇ ਵੀ ਕੰਪਿਊਟਰ ਲਈ ਸੁਤੰਤਰ ਤੌਰ 'ਤੇ ਕਿਸੇ ਪ੍ਰੋਗਰਾਮ ਦਾ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨਾ ਅਤੇ ਫਿਰ ਇਸ ਦੇ ਹੱਲ ਦਾ ਤਰੀਕਾ ਖ਼ੁਦ ਲੱਭਣਾ ਸੰਭਵ ਨਹੀਂ ਹੈ। ਕੰਪਿਊਟਰ ਨੂੰ ਇਹ ਦੱਸਣ ਲਈ, ਕਿ ਇਸਨੇ ਕੀ ਕਰਨਾ ਹੈ, ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੀ ਜ਼ਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਦੀਆਂ ਹਦਾਇਤਾਂ ਕੰਪਿਊਟਰ ਨੂੰ ਇਹ ਦੱਸਦੀਆਂ ਹਨ ਕਿ ਦਿੱਤੀ ਗਈ ਸਮੱਸਿਆ ਨੂੰ ਕਿਵੇਂ ਹੱਲ ਕੀਤਾ ਜਾਵੇਗਾ। ਪਰ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣਾ ਸਾਧਾਰਨ ਅਤੇ ਸੌਖਾ ਕੰਮ ਨਹੀਂ ਹੈ। ਇੱਕ ਪ੍ਰੋਗਰਾਮਰ ਲਈ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸਫਲਤਾਪੂਰਵਕ ਬਨਾਉਣ ਲਈ ਇੱਕ ਖਾਸ ਪਕੀਰਿਆ ਵਿਚੋਂ ਲੰਘਣਾ ਪੈਂਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰਕਿਰਿਆ ਦੇ ਕਦਮਾਂ ਵਿੱਚ ਪ੍ਰੋਗਰਾਮਰ ਦੇ ਤਜ਼ਰਬੇ ਦੇ ਅਧਾਰ ਤੇ ਥੋੜ੍ਹੀ ਬਹੁਤ ਸੋਧ ਹੋ ਸਕਦੀ ਹੈ, ਪਰ ਵੱਡੇ ਪੱਧਰ ਤੇ, ਹੇਠਾਂ ਦਿੱਤੇ ਕਦਮਾਂ ਨੂੰ ਉਸੇ ਤਰਤੀਬ ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

1. ਹੱਲ ਕੀਤੀ ਜਾਣ ਵਾਲੀ ਸਮੱਸਿਆ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨਾ (Defining the problem to be solved)
2. ਸਮੱਸਿਆ ਦੇ ਹੱਲ ਲਈ ਯੋਜਨਾ ਤਿਆਰ ਕਰਨਾ (Plan the solution of the problem)
3. ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਵਿੱਚ ਹੱਲ ਲਈ ਕੋਡਿੰਗ ਕਰਨਾ (Coding the solution in the high-level language)
4. ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਈਲ ਕਰਨਾ (Compile the program)
5. ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਟੈਸਟ ਅਤੇ ਡਿਬੱਗ ਕਰਨਾ (Test and Debug the program)
6. ਪ੍ਰੋਗਰਾਮ ਦਾ ਦਸਤਾਵੇਜ਼ੀਕਰਨ ਕਰਨਾ (Documenting the program)

ਇਹਨਾਂ ਸਟੈਪਾਂ ਦੀ ਵਿਸਥਾਰ ਵਿਚ ਵਿਆਖਿਆ ਹੇਠਾਂ ਦਿਤੀ ਗਈ ਹੈ:

3.5.1 ਸਮੱਸਿਆ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨਾ (Defining the problem) :

ਇਹ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰਕਿਰਿਆ ਦਾ ਪਹਿਲਾ ਕਦਮ ਹੈ। ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਆਪਣਾ ਕੰਮ ਸ਼ੁਰੂ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ, ਸਮੱਸਿਆ ਦੇ ਵੇਰਵੇ ਜਾਣਨੇ ਜਰੂਰੀ ਹੁੰਦੇ ਹਨ। ਸਮੱਸਿਆ ਦੇ ਵੇਰਵੇ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਪ੍ਰਦਾਨ ਕੀਤੇ ਜਾਣੇ ਚਾਹੀਦੇ ਹਨ ਤਾਂ ਕਿ ਉਹ ਇਹਨਾਂ ਵੇਰਵਿਆਂ ਦਾ ਅਧਿਐਨ ਕਰਕੇ ਸਮੱਸਿਆ ਨੂੰ ਸਪੱਸ਼ਟ ਰੂਪ ਵਿਚ ਸਮਝ ਸਕੇ। ਸਮੱਸਿਆ ਦੇ ਵਿਸ਼ਲੇਸ਼ਣ ਕਰਨ ਨਾਲ ਸਮੱਸਿਆ ਦੇ ਹੱਲ ਲਈ ਲੋੜੀਂਦੀ ਜਾਣਕਾਰੀ ਅਤੇ ਨਤੀਜੇ ਪ੍ਰਾਪਤ ਹੁੰਦੇ ਹਨ। ਸਮੱਸਿਆ ਦੀ ਸਪੱਸ਼ਟ ਸਮਝ ਹੋਣ ਤੋਂ ਬਾਅਦ ਹੀ ਪ੍ਰੋਗਰਾਮਰ ਦਿੱਤੀ ਗਈ ਸਮੱਸਿਆ ਨੂੰ ਕਿਵੇਂ ਹੱਲ ਕੀਤਾ ਜਾਵੇਗਾ, ਬਾਰੇ ਸੋਚਣਾ ਸ਼ੁਰੂ ਕਰਦਾ ਹੈ।

3.5.2 ਹੱਲ ਦੀ ਯੋਜਨਾ ਬਨਾਉਣਾ (Planning the Solution) :

ਸਮੱਸਿਆ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਤੋਂ ਬਾਅਦ ਅਗਲਾ ਕਦਮ ਸਮੱਸਿਆ ਦੇ ਹੱਲ ਲਈ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਕਦਮਾਂ ਦੀ ਇੱਕ ਵਿਸਥਾਰਿਤ ਸੂਚੀ ਤਿਆਰ ਕਰਨਾ ਹੈ। ਆਓ ਇਕ ਉਦਾਹਰਣ ਦੀ ਮਦਦ ਨਾਲ ਇਸ ਤੱਥ ਨੂੰ ਸਮਝੀਏ ਕਿ ਸਮੱਸਿਆ ਦੇ ਹੱਲ ਲਈ ਯੋਜਨਾਬੰਦੀ ਕਿਉਂ ਜ਼ਰੂਰੀ ਹੈ। ਮੰਨ ਲਵੋ ਇੱਕ ਅਧਿਆਪਕ ਨੇ ਇੱਕ ਵਿਦਿਆਰਥੀ ਨੂੰ ਗਣਿਤ ਦੀ ਵਿਸ਼ੇਸ਼ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਕਿਹਾ, ਅਤੇ ਵਿਦਿਆਰਥੀ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਵਾਲੇ ਕਦਮਾਂ ਤੋਂ ਜਾਣੂ ਨਹੀਂ ਹੈ, ਤਾਂ ਉਹ ਇਸ ਨੂੰ ਹੱਲ ਨਹੀਂ ਕਰ ਸਕੇਗਾ। ਇਹੋ ਸਿਧਾਂਤ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਲਿਖਣ ਲਈ ਵੀ ਲਾਗੂ ਹੁੰਦਾ ਹੈ। ਇੱਕ ਪ੍ਰੋਗਰਾਮਰ ਕਿਸੇ ਵੀ ਪ੍ਰੋਗਰਾਮ ਲਈ ਹਦਾਇਤਾਂ ਉਦੋਂ ਤੱਕ ਨਹੀਂ ਲਿਖ ਸਕਦਾ ਜਦੋਂ ਤੱਕ ਕਿ ਉਹ ਇਹ ਨਹੀਂ ਸਮਝਦਾ ਕਿ ਸਮੱਸਿਆ ਨੂੰ ਕਿਵੇਂ ਹੱਲ ਕਰਨਾ ਹੈ। ।

ਜੇ ਕੋਈ ਪ੍ਰੋਗਰਾਮਰ ਸਮੱਸਿਆ ਦੇ ਹੱਲ ਲਈ ਸਟੈਪਾਂ ਨੂੰ ਜਾਣਦਾ ਤਾਂ ਹੈ ਪਰ ਇਸ ਨੂੰ ਹੱਲ ਕਰਨ ਵੇਲੇ ਉਹ ਕਦਮਾਂ ਦੀ ਗਲਤ ਤਰਤੀਬ ਲਾਗੂ ਕਰਦਾ ਹੈ ਜਾਂ ਉਹ ਇਨ੍ਹਾਂ ਵਿੱਚੋਂ ਕੁਝ ਕਦਮਾਂ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਭੁੱਲ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਉਸਨੂੰ ਗਲਤ ਆਉਟਪੁੱਟ ਮਿਲੇਗੀ। ਇੱਕ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਪ੍ਰੋਗਰਾਮ ਲਿਖਣ ਲਈ, ਇੱਕ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਸਾਰੀਆਂ ਹਦਾਇਤਾਂ ਨੂੰ ਸਹੀ ਤਰਤੀਬ ਵਿੱਚ ਲਿਖਣਾ ਜਰੂਰੀ ਹੁੰਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਹਦਾਇਤਾਂ ਦੀ ਲੜੀ ਗੁੰਝਲਦਾਰ ਹੋ ਸਕਦੀ ਹੈ। ਇਸ ਲਈ ਪ੍ਰੋਗਰਾਮ ਦੀਆਂ ਹਦਾਇਤਾਂ ਦੀ ਸਹੀ ਵਰਤੋਂ ਅਤੇ ਸਹੀ ਤਰਤੀਬ ਤੈਅ ਕਰਨ ਲਈ, ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਤੋਂ ਪਹਿਲਾਂ ਇਸਦੀ ਯੋਜਨਾ ਬਣਾਉਣਾ ਲਾਜ਼ਮੀ ਹੁੰਦੀ ਹੈ। ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਸਟੈਪਾਂ ਦੀ ਪਲੈਨਿੰਗ ਅਤੇ ਪਰਿਭਾਸ਼ਾ ਲਈ, ਪ੍ਰੋਗਰਾਮਰ ਆਮ ਤੌਰ ਤੇ ਐਲਗੋਰਿਥਮ (Algorithms) ਅਤੇ ਫਲੋ-ਚਾਰਟਸ (Flow-Charts) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ।

3.5.3 ਹੱਲ ਦੀ ਕੋਡਿੰਗ ਕਰਨਾ (Coding the Solution) :

ਫਲੋ-ਚਾਰਟ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਓਪਰੇਸ਼ਨਾਂ ਦੇ ਕੰਮ ਨੂੰ ਕਿਸੇ ਵੀ ਉੱਚ ਪੱਧਰੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ, ਜਿਵੇਂ ਕਿ C, C++, ਅਤੇ PASCAL ਆਦਿ ਦੀ ਵਰਤੋਂ ਨਾਲ ਪ੍ਰੋਗਰਾਮ ਹਦਾਇਤਾਂ ਵਿੱਚ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਕਿਸੇ ਵੀ ਕੰਪਿਊਟਰ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਨਾਲ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੀਆਂ ਹਦਾਇਤਾਂ ਲਿਖਣ ਨੂੰ ਕੋਡਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਪ੍ਰੋਗਰਾਮ ਫਾਈਲ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਜਾਂ ਸੋਰਸ ਕੋਡ ਵਜੋਂ ਜਾਣੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਕੋਡ ਇੱਕ ਡਿਸਕ ਫਾਈਲ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਲੋਜਿਕ ਜਾਂ ਸਮੱਸਿਆ ਦੇ ਹੱਲ ਲਈ ਸਟੈਪ ਲਿਖੇ ਜਾਂਦੇ ਹਨ।

3.5.4 ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਈਲ ਕਰਨਾ (Compile the program) :

ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਵਿੱਚ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਤੋਂ ਬਾਅਦ ਇਸ ਨੂੰ ਅਜਿਹੇ ਰੂਪ ਵਿੱਚ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨਾ ਪੈਂਦਾ ਹੈ। ਜਿਸਨੂੰ ਕੰਪਿਊਟਰ ਸਮਝ ਸਕੇ ਅਤੇ ਲਾਗੂ ਕਰ ਸਕੇ, ਕਿਉਂਕਿ ਅਸੀਂ ਜਾਣਦੇ ਹਾਂ ਕਿ ਇੱਕ ਕੰਪਿਊਟਰ ਸਿਰਫ ਬਾਈਨਰੀ ਫਾਰਮੈਟ ਵਿੱਚ ਹੀ ਹਦਾਇਤਾਂ ਨੂੰ ਸਮਝ ਸਕਦਾ ਹੈ। ਇਸ ਲਈ ਕੰਪਿਊਟਰ ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਲਿਖੇ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸਮਝਣ ਦੇ ਯੋਗ ਨਹੀਂ ਹੁੰਦਾ। ਇਸ ਲਈ ਭਾਸ਼ਾ ਟ੍ਰਾਂਸਲੇਟਰਾਂ (ਕੰਪਾਈਲਰ ਆਦਿ) ਦੀ ਮਦਦ ਨਾਲ ਸੋਰਸ ਪ੍ਰੋਗ੍ਰਾਮ ਨੂੰ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਰੂਪ ਵਿਚ ਟਾਂਸਲੇਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਕੋਡ ਨੂੰ ਆਬਜੈਕਟ ਕੋਡ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਜੋ ਭਾਸ਼ਾ ਸਲੇਟਰ ਦੁਆਰਾ ਤਿਆਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਸੋਰਸ ਕੋਡ ਦੇ ਆਬਜੈਕਟ ਕੋਡ ਵਿਚ ਇਸ ਸਲੇਸ਼ਨ ਨੂੰ ਕੰਪਾਈਲੇਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਕੰਪਾਈਲੇਸ਼ਨ ਦੌਰਾਨ ਕੰਪਾਈਲਰ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਿੰਟੈਕਸ-ਗਲਤੀਆਂ (Syntrap Error) ਵੀ ਸਕੈਨ ਕਰਦਾ ਹੈ। ਜੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਿੰਟੈਕਸ ਗਲਤੀਆਂ ਹੋਣ ਤਾਂ ਕੰਪਾਈਲਰ ਗਲਤੀਆਂ ਨਾਲ ਸੰਬੰਧਤ ਸੰਦੇਸ਼ ਦਰਸ਼ਾਉਂਦਾ ਹੈ। ਆਬਜੈਕਟ ਕੋਡ ਬਨਾਉਣ ਲਈ ਇਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਠੀਕ ਕਰਨਾ ਜਰੂਰੀ ਹੁੰਦਾ ਹੈ। ਆਬਜੈਕਟ ਕੋਡ ਨੂੰ ਡਿਸਕ ਫਾਈਲ ਵਿਚ ਸਟੋਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜਦੋਂ ਵੀ ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਣਾ ਚਾਹੁੰਦੇ ਹਾਂ, ਇਹ ਆਬਜੈਕਟ ਕੋਡ ਹੀ ਕੰਪਿਊਟਰ ਨੂੰ ਚਲਾਉਣ ਲਈ ਮੁਹਿਆ ਕਰਵਾਇਆ ਜਾਂਦਾ ਹੈ।

3.5.5 ਪ੍ਰੋਗਰਾਮ ਦੀ ਟੈਸਟਿੰਗ ਅਤੇ ਡੀਬੱਗਿੰਗ ਕਰਨਾ (Testing and Debugging the Program) :

ਟੈਸਟਿੰਗ ਅਤੇ ਡੀਬੱਗਿੰਗ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਦੇ ਮਹੱਤਵਪੂਰਣ ਸਟੈਪ ਹਨ। ਟੈਸਟਿੰਗ ਇੱਕ ਅਜਿਹੀ ਪ੍ਰਕਿਰਿਆ ਹੈ ਜੋ ਇਹ ਤੈਅ ਕਰਦੀ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮ ਉਦੇਸ਼ਿਤ ਕਾਰਜ (Program's objective) ਨੂੰ ਪੂਰਾ ਕਰਦਾ ਹੈ। ਟੈਸਟਿੰਗ ਕਰਨ ਵਿਚ ਕਾਫੀ ਸਮਾਂ ਲੱਗਦਾ ਹੈ। ਜਦੋਂ ਤੱਕ ਮਨੁੱਖ ਪ੍ਰੋਗਰਾਮ ਬਨਾਵੇਗਾ, ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਗਲਤੀਆਂ ਮੌਜੂਦ ਰਹਿਣਗੀਆਂ। ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਇਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਬੱਗ (Bugs) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਲੱਭਣਾ ਅਤੇ ਠੀਕ ਕਰਨ ਦੀ ਪ੍ਰਕਿਰੀਆ ਨੂੰ ਡੀਬੱਗਿੰਗ (Debugging) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

3.5.6 ਪ੍ਰੋਗਰਾਮ ਦਾ ਦਸਤਾਵੇਜ਼ੀਕਰਨ ਕਰਨਾ (Documenting the Program) :

ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਦਾ ਆਖਰੀ ਪੜਾਅ ਹੈ ਦਸਤਾਵੇਜ਼ੀਕਰਣ। ਸ਼ਬਦ ਦਸਤਾਵੇਜ਼ੀਕਰਣ ਦਾ ਅਰਥ ਹੈ ਪ੍ਰੋਗਰਾਮ ਬਾਰੇ ਮਹੱਤਵਪੂਰਣ ਜਾਣਕਾਰੀ ਨੂੰ ਨਿਰਧਾਰਿਤ (Specify) ਕਰਨਾ। ਦਸਤਾਵੇਜ਼ੀਕਰਣ ਦੂਜੇ ਪ੍ਰੋਗਰਾਮਰ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਦੇ ਲਾਜੀਕ ਅਤੇ ਉਦੇਸ਼ ਨੂੰ ਸਮਝਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀ ਹੈ। ਇਹ ਪ੍ਰੋਗਰਾਮ ਦੀ ਸੰਭਾਲ (Maintenance) ਵਿਚ ਵੀ ਮਦਦਗਾਰ ਸਾਬਤ ਹੁੰਦਾ ਹੈ।

3.6 ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਗਲਤੀਆਂ ਦੀਆਂ ਕਿਸਮਾਂ (TYPES OF ERRORS IN THE PROGRAM) :

ਐਰਰਜ਼/ ਗਲਤੀਆਂ (Errors) : ਉਹ ਨੁਕਸ (Fault) ਹੁੰਦੇ ਹਨ ਜੋ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਮੌਜੂਦ ਹੋ ਸਕਦੇ ਹਨ। ਇਹ ਗਲਤੀਆਂ ਪ੍ਰੋਗਰਾਮ ਦੇ ਵਿਵਹਾਰ ਨੂੰ ਅਸਾਧਾਰਨ (Abnormal Behaviour) ਬਣਾਉਂਦੀਆਂ ਹਨ। ਪ੍ਰੋਗਰਾਮਿੰਗ ਐਰਰਜ਼ ਨੂੰ ਬੱਗ ਜਾਂ ਨੁਕਸ (Bugs or Errors) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਬੱਗਸ (Bugs) ਨੂੰ ਹਟਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਡੀਬੱਗਿੰਗ (Debugging) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਆਮ ਤੌਰ 'ਤੇ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿੱਚ ਹੇਠਾਂ ਦਿਤੀਆਂ ਕਿਸਮਾਂ ਦੀਆਂ ਗਲਤੀਆਂ ਪਾਈਆਂ ਜਾਂਦੀਆਂ ਹਨ:

ਸਿੰਟੈਕਸ ਗਲਤੀਆਂ (Syntax Errors) : ਇਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਕੰਪਾਇਲੇਸ਼ਨ ਐਰਰਜ਼ (Compilation Errors) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਗਲਤੀਆਂ ਉਦੋਂ ਹੁੰਦੀਆਂ ਹਨ ਜਦੋਂ ਅਸੀਂ ਪ੍ਰੋਗ੍ਰਾਮਿੰਗ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰਨ ਵਾਲੇ ਨਿਯਮਾਂ ਜਾਂ ਸਿੰਟੈਕਸ ਦੀ ਵਰਤੋਂ ਨਹੀਂ ਕਰਦੇ। ਕੰਪਾਈਲਰਾਂ ਦੁਆਰਾ ਕੰਪਾਇਲੇਸ਼ਨ ਪ੍ਰਕਿਰਿਆ ਦੌਰਾਨ ਇਸ ਕਿਸਮ ਦੀਆਂ ਗਲਤੀਆਂ ਆਪਣੇ ਆਪ ਲੱਭੀਆਂ ਜਾਂਦੀਆਂ ਹਨ। ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸਫਲਤਾਪੂਰਵਕ ਉਸ ਸਮੇਂ ਤੱਕ ਕੰਪਾਈਲ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਜਦੋਂ ਤੱਕ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚਲੀਆਂ ਸਿੰਟੈਕਸ ਗਲਤੀਆਂ ਨੂੰ ਠੀਕ ਨਹੀਂ ਕਰ ਦਿੱਤਾ ਜਾਂਦਾ ਸੀ। ਭਾਸ਼ਾ ਵਿੱਚ ਸਿੰਟੈਕਸ ਗਲਤੀਆਂ ਦੀਆਂ ਉਦਾਹਰਣ ਹਨ: ਸੈਮੀਕਾਲਨ ਨਾ ਲਗਾਉਣਾ (Missing semicolon), ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਕਰਨਾ ਭੁੱਲ ਜਾਣਾ (variable not declared), ਅਨ-ਟਰਮੀਨੇਟਡ ਸਟ੍ਰਿੰਗ (Unterminated String), ਕੰਪਾਊਂਡ ਸਟੇਟਮੈਂਟ ਲਾਪਤਾ ਹੋਣਾ (Missing Compound Statement) ਆਦਿ।

ਲਾਜੀਕਲ ਗਲਤੀਆਂ (Logical Errors) : ਇਹ ਗਲਤੀਆਂ ਉਦੋਂ ਹੁੰਦੀਆਂ ਹਨ ਜਦੋਂ ਪ੍ਰੋਗਰਾਮ ਦੇ ਲਾਜੀਕ ਵਿੱਚ ਗਲਤੀਆਂ ਹੁੰਦੀਆਂ ਹਨ। ਜੇਕਰ ਸਾਡੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਲਾਜੀਕਲ-ਗਲਤੀਆਂ ਹੋਣ, ਤਾਂ ਇਹ ਸਫਲਤਾਪੂਰਵਕ ਕੰਪਾਇਲ ਤਾਂ ਹੋ ਜਾਵੇਗਾ ਪਰ ਇਹ ਗਲਤ ਆਉਟਪੁੱਟ ਪੈਦਾ ਕਰ ਸਕਦਾ ਹੈ। ਕੰਪਾਈਲਰ ਦੁਆਰਾ ਅਜਿਹੀਆਂ ਗਲਤੀਆਂ ਲੱਭੀਆਂ ਨਹੀਂ ਜਾ ਸਕਦੀਆਂ। ਇਹ ਗਲਤੀਆਂ ਜਾਂ ਤਾਂ ਪ੍ਰੋਗਰਾਮਰ ਦੁਆਰਾ ਖੁੱਦ ਲੱਭੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ ਜਾਂ ਕੁਝ ਡੀਬੱਗਿੰਗ ਟੂਲਜ਼ (debugging tools) ਅਜਿਹੀਆਂ ਗਲਤੀਆਂ ਨੂੰ ਲੱਭਣ ਲਈ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਪ੍ਰੋਗਰਾਮ ਆਉਟਪੁੱਟ ਦੀ ਜਾਂਚ ਕਰਕੇ ਕੋਈ ਵੀ ਗਲਤ ਲਾਜੀਕ ਲੱਭ ਸਕਦਾ ਹੈ।

ਰਨਟਾਈਮ ਗਲਤੀਆਂ (Runtime Errors) : ਇੱਕ ਰਨਟਾਈਮ ਗਲਤੀ ਇੱਕ ਅਜਿਹੀ ਗਲਤੀ ਹੁੰਦੀ ਹੈ ਜੋ ਪ੍ਰੋਗਰਾਮ ਦੇ ਸਫਲਤਾਪੂਰਵਕ ਕੰਪਾਈਲ ਹੋਣ ਤੋਂ ਬਾਅਦ ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਗਜ਼ੀਕਿਉਸ਼ਨ-ਸਮੇਂ ਵਾਪਰਦੀ ਹੈ। ਕਈ ਵਾਰ ਜਦੋਂ ਕੋਈ ਪ੍ਰੋਗਰਾਮ ਚੱਲ ਰਿਹਾ ਹੁੰਦਾ ਹੈ ਤਾਂ ਉਹ ਆਪਣਾ ਕੰਮ ਕਰਨ ਯੋਗ ਨਹੀਂ ਹੁੰਦਾ। ਇਹ ਰਨ-ਟਾਈਮ ਗਲਤੀ ਦੇ ਕਾਰਨ ਹੋ ਸਕਦਾ ਹੈ। ਇਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਲੱਭਣਾ ਬਹੁਤ ਮੁਸ਼ਕਲ ਹੁੰਦਾ ਹੈ, ਕਿਉਂਕਿ ਇਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਡਿਟੈਕਟ (detect) ਨਹੀਂ ਕਰ ਸਕਦਾ। ਰਨ-ਟਾਈਮ ਗਲਤੀਆਂ ਦੀਆਂ ਆਮ ਉਦਾਹਰਨਾਂ ਇਸ ਪ੍ਰਕਾਰ ਹਨ :

- ਇਨਪੁੱਟ ਆਉਟਪੁੱਟ ਸੰਬੰਧੀ ਗਲਤੀ (IO error)
- ਕਿਸੇ ਮੁੱਲ ਨੂੰ ਜ਼ੀਰੋ ਨਾਲ ਭਾਗ ਕਰਨ ਸਮੇਂ ਗਲਤੀ (Division by zero errors)
- ਕਿਸੇ ਮੁੱਲ ਦਾ ਰੇਂਜ਼ ਤੋਂ ਬਾਹਰ ਚਲੇ ਜਾਣਾ (Out of range errors)

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਹਦਾਇਤਾਂ ਦਾ ਸਮੂਹ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਕਿ ਇੱਕ ਸਾਫਟਵੇਅਰ ਪ੍ਰੋਗਰਾਮਾਂ ਦਾ ਸਮੂਹ।
2. ਪ੍ਰੋਗਰਾਮਰ ਉਹ ਵਿਅਕਤੀ ਹੁੰਦਾ ਹੈ ਜੋ ਪ੍ਰੋਗਰਾਮ ਲਿਖਦਾ ਹੈ।
3. ਪ੍ਰੋਗਰਾਮ ਲਿਖਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
4. ਮਸ਼ੀਨੀ ਭਾਸ਼ਾ ਕੰਪਿਊਟਰ ਦੁਆਰਾ ਸਿੱਧੇ ਤੌਰ ਤੇ ਸਮਝੀ ਜਾਂਦੀ ਹੈ ਜੋ ਕਿ ਬਾਈਨਰੀ ਅੰਕਾਂ 0 ਅਤੇ 1 ਤੋਂ ਮਿਲਕੇ ਬਣਦੀ ਹੈ।
5. ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਹਦਾਇਤਾਂ ਲਿਖਣ ਲਈ ਨਿਊਮੈਰਿਕ ਕੋਡਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ
6. ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਹਦਾਇਤਾਂ ਲਿਖਣ ਲਈ ਅੈਲਫਾਨੁਮੈਰਿਕ ਕੋਡਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਦੀ ਹੈ।
7. ਅਸੈਬਲਰ ਇੱਕ ਭਾਸ਼ਾ ਟਾਂਸਲੇਟਰ ਹੈ ਜੋ ਅਸੈਬਲੀ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਟਾਂਸਲੇਟ ਕਰਦੀ ਹੈ ਜਿਸਨੂੰ ਆਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
8. ਕੰਪਾਈਲਰ ਇੱਕ ਭਾਸ਼ਾ ਟਾਂਸਲੇਟਰ ਹੈ ਜੋ ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਟਾਂਸਲੇਟ ਕਰਦੀ ਹੈ ਜਿਸਨੂੰ ਆਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
9. ਕਿਸੇ ਵੀ ਕੰਪਿਊਟਰ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਨਾਲ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੀਆਂ ਹਦਾਇਤਾਂ ਲਿਖਣ ਨੂੰ ਕੋਡਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
10. ਐਰਰਜ਼/ਗਲਤੀਆਂ (Errors) ਉਹ ਨੁਕਸ (Fault) ਹੁੰਦੇ ਹਨ ਜੋ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਮੌਜੂਦ ਹੋ ਸਕਦੇ ਹਨ।
11. ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਗਲਤੀਆਂ ਨੂੰ ਲੱਭਣਾ ਅਤੇ ਠੀਕ ਕਰਨ ਦੀ ਪ੍ਰਕਿਰੀਆ ਨੂੰ ਡੀਬੱਗਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
12. ਸਿੰਟੈਕਸ ਗਲਤੀਆਂ ਨੂੰ ਕੰਪਾਈਲੇਸ਼ਨ ਗਲਤੀਆਂ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
13. ਲਾਜ਼ੀਕਲ ਅਤੇ ਰਨਟਾਈਮ ਗਲਤੀਆਂ ਨੂੰ ਕੰਪਾਈਲਰ ਦੁਆਰਾ ਡਿਟੈਕਟ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ।

ਅਭਿਆਸ

1. ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ

1. ਹਦਾਇਤਾਂ ਦੇ ਸਮੂਹ ਨੂੰ ....ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

   (ੳ) ਗਰੁੱਪ (Group) (ਅ) ਸਾਫਟਵੇਅਰ (Soft ware)

   (ੲ) ਪ੍ਰੋਗਰਾਮ (Program) (ਸ) ਇਹਨਾਂ ਵਿਚੋਂ ਕੋਈ ਨਹੀਂ

2. ਕਿਹੜੀ ਭਾਸ਼ਾ ਬਿਨ੍ਹਾਂ ਕਿਸੇ ਟ੍ਰਾਂਸਲੇਸ਼ਨ ਦੇ ਕੰਪਿਊਟਰ ਦੁਆਰਾ ਸਿੱਧੀ ਸਮਝੀ ਜਾਂਦੀ ਹੈ।

   ੳ. ਪ੍ਰੋਸੀਜ਼ਰ ਓਰੀਐਂਟਡ ਭਾਸ਼ਾ (Procedural Oriented Language)

   ਅ. ਮਸ਼ੀਨ ਭਾਸ਼ਾ (Machine Language)

   ੲ. ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ (Assembly Language)

   ਸ. ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ (High level Language)

3. ਨਮੋਨਿੱਕ (Mnemonic) ਕੋਡ ਅਤੇ ਚਿੰਨਾਤਮਕ (Symbolic) ਐਡਰੈਸ ਕਿਹੜੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਵਿਚ ਵਰਤੇ ਜਾਂਦੇ ਹਨ ?

   ੳ. ਆਬਜੈਕਟ ਓਰੀਐਂਟੇਡ ਭਾਸ਼ਾ (Object Oriented Language)

   ਅ. ਨਾਨ-ਪ੍ਰੋਸੀਜ਼ਰਲ ਭਾਸ਼ਾ (Non-procedural Language)

ੲ. ਅਸੈਂਬਲੀ ਭਾਸ਼ਾ (Assembly Language)
ਸ. ਮਸ਼ੀਨ ਭਾਸ਼ਾ (Machine Language)

4. ਕਿਹੜਾ ਟ੍ਰਾਂਸਲੇਟਰ ਹਾਈ ਲੇਵਲ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਸੋਰਸ ਕੋਡ ਨੂੰ ਫਾਂਸਲੇਟ ਕਰਦੇ ਸਮੇਂ ਆਬਜੈਕਟ ਕੋਡ ਵਿਚ ਸਟੋਰ ਨਹੀਂ ਕਰਦਾ ?

ੳ. ਇੰਟਰਪ੍ਰੇਟਰ (Interpretor) ਅ. ਕੰਪਾਈਲਰ (Compilor)
ੲ. ਅਸੈਂਬਲਰ (Assembler) ਸ. ਇਹਨਾਂ ਵਿੱਚੋਂ ਕੋਈ ਨਹੀਂ

5. ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਗਲਤੀਆਂ ਨੂੰ ਲੱਭਣਾ ਅਤੇ ਉਹਨਾਂ ਨੂੰ ਠੀਕ ਕਰਨ ਦੀ ਪ੍ਰਕਿਰਿਆ ਨੂੰ...........ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਕੰਪਾਈਲੇਸ਼ਨ (Compilation) ਅ. ਕੋਡਿੰਗ (Coding)
ੲ. ਡੀਬੱਗਿੰਗ (Debugging) ਸ. ਦਸਤਾਵੇਜ਼ੀਕਰਣ (Documentations)

2. ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ।

1. ਇਕ ਵਿਅਕਤੀ ਜੋ ਪ੍ਰੋਗਰਾਮ ਲਿਖਦਾ ਹੈ ਉਸਨੂੰ ... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
2. ਹਾਰਡਵੇਅਰ ਸੰਬੰਧੀ ਲੋਅ ਲੇਵਲ ਅੰਦਰੂਨੀ ਜਾਣਕਾਰੀ, ...ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਲਈ ਲੋੜੀਂਦੀ ਹੈ।
3. ਹਾਈ ਲੈਵਲ ਭਾਸ਼ਾ ਵਿਚ ਲਿਖੇ ਸੋਰਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਆਬਜੈਕਟ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਤਬਦੀਲ ਕਰਨ ਦੀ ਪ੍ਰਕਿਰਿਆ ਨੂੰ.......... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
4. ਉਹ ਗਲਤੀਆਂ ਜੋ ਕੰਪਾਈਲਰ ਦੁਆਰਾ ਨਹੀਂ ਲੱਭੀਆਂ ਜਾ ਸਕਦੀਆਂ ਉਹਨਾਂ ਨੂੰ .... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

3 : ਹੇਠ ਲਿਖਿਆਂ ਦੇ ਪੂਰੇ ਰੂਪ ਲਖੋ ।

1. Opcode
2. Operand
3. 4GL
4. OOP

4. ਛੋਟੇ ਉਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ

1. ਪ੍ਰੋਗਰਾਮਿੰਗ ਕੀ ਹੈ ?
2. ਲੋਅ ਲੇਵਲ ਭਾਸ਼ਾਵਾਂ ਕੀ ਹੁੰਦੀਆਂ ਹਨ ?
3. ਤੁਸੀਂ ਓਬਜੈਕਟ ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਬਾਰੇ ਕੀ ਜਾਣਦੇ ਹੋ ?
4. ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰੋਸੈਸ ਵਿਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਸਟੈਪਾਂ ਦੇ ਨਾਂ ਲਿਖੋ।
5. ਸਿੰਟੈਕਸ-ਗਲਤੀਆਂ (Syntax Errors) ਕੀ ਹੁੰਦੀਆਂ ਹਨ ?
6. ਅਸੈਂਬਲਰ ਕੀ ਹੁੰਦਾ ਹੈ ?
7. ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਵਿਚ ਹਦਾਇਤ ਦਾ ਫਾਰਮੇਟ ਲਿਖੋ।
8. ਹਾਈ ਲੈਵਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਟ੍ਰਾਂਸਲੇਟ ਕਰਨ ਲਈ ਕਿਹੜੇ ਟਾਂਸਲੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ?

5. ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ

1. ਹਾਈ-ਲੇਵਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਕੀ ਹੁੰਦੀਆਂ ਹਨ ? ਵਿਆਖਿਆ ਕਰੋ।
2. ਭਾਸ਼ਾ ਟਾਂਸਲੇਟਰ ਕੀ ਹੁੰਦੇ ਹਨ ? ਕੰਪਾਈਲਰ ਅਤੇ ਇੰਟਰਪ੍ਰੈਟਰ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
3. ਐਰਰਜ਼/ਗਲਤੀਆਂ ਕੀ ਹੁੰਦੀਆਂ ਹਨ ? ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਮਿਲਣ ਵਾਲੀਆਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀਆਂ ਗਲਤੀਆਂ ਦਾ ਵਰਨਣ ਕਰੋ।
4. ਮਸ਼ੀਨ ਭਾਸ਼ਾ ਕੀ ਹੁੰਦੀ ਹੈ ? ਇਸ ਦੇ ਲਾਭ ਅਤੇ ਕਮੀਆਂ ਦਾ ਵਰਨਣ ਕਰੋ।
5. ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰੋਸੈਸ ਵਿੱਚ ਸ਼ਾਮਿਲ ਵੱਖ-ਵੱਖ ਪੜਾਵਾਂ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।

ਜਾਵਾ ਨਾਲ OOP ਦੀ ਜਾਣ-ਪਛਾਣ
(Introduction to OOP with Java)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

4.1 ਓਬਜੈਕਟ ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਨਾਲ ਜਾਣ-ਪਛਾਣ

4.2 ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਸਿਧਾਂਤ - ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction), ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation), ਇਨਹੈਰੀਟੈਂਸ (Inheritance), ਅਤੇ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism)

4.3 ਜਾਵਾਂ ਨਾਲ ਜਾਣ ਪਛਾਣ

4.4 ਜਾਵਾ ਦਾ ਇਤਿਹਾਸ

4.5 ਜਾਵਾ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ

4.6 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਮੁੱਢਲੀ ਬਣਤਰ

4.7 ਇਕ ਸਾਧਾਰਣ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣਾ ਅਤੇ ਚਲਾਉਣਾ

4.8 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਮੁੱਢਲੇ ਤੱਤ: ਕਰੈਕਟਰ ਸੈੱਟ, ਟੋਕਨਜ਼, ਕਮੈਂਟਸ

4.1 ਆਬਜੈਕਟ ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਨਾਲ ਜਾਣ-ਪਛਾਣ (INTRODUCTION TO OBJECT ORIENTED PROGRAMMING)

ਪ੍ਰੋਸੀਜ਼ਰਲ ਪ੍ਰੋਗਰਾਮਿੰਗ ਡਾਟਾ ਉਪਰ ਵੱਖ-ਵੱਖ ਕੰਮਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਬਣਾਏ ਜਾਣ ਵਾਲੇ ਪ੍ਰੋਸੀਜ਼ਰਾਂ (procedures) ਜਾਂ ਮੈਥਡਜ਼ (methods) ਨਾਲ ਸੰਬੰਧਤ ਹੈ, ਜਦੋਂ ਕਿ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਆਬਜੈਕਟ (object) ਬਣਾ ਕੇ ਕੰਮ ਕਰਨ ਨਾਲ ਸੰਬੰਧਤ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ "ਆਬਜੈਕਟ **(object)**" ਦੀ ਧਾਰਨਾ 'ਤੇ ਕੰਮ ਕਰਦੀਆਂ ਹਨ। ਆਬਜੈਕਟਸ ਅਸਲ-ਸੰਸਾਰ ਦੀਆਂ ਵਸਤੂਆਂ (entities) ਹੁੰਦੀਆਂ ਹਨ ਜਿਵੇਂ ਕਿ: ਵਿਦਿਆਰਥੀ, ਵਿਅਕਤੀ, ਪੈਨ, ਟੇਬਲ, ਟੈਲੀਵਿਜ਼ਨ, ਕੰਪਿਊਟਰ ਆਦਿ।

ਸੰਸਾਰ ਵਿੱਚ ਕਿਸੇ ਵੀ ਚੀਜ਼ ਨੂੰ ਇੱਕ ਆਬਜੈਕਟ ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ, ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ ਨੂੰ ਉਸਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (properties) ਅਤੇ ਵਿਵਹਾਰ (behaviour) ਦੇ ਰੂਪ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ: ਜੇਕਰ ਆਬਜੈਕਟ ਇੱਕ ਵਿਅਕਤੀ ਹੈ ਤਾਂ ਉਸਦਾ ਨਾਮ, ਉਮਰ, ਕੱਦ, ਭਾਰ, ਆਦਿ ਉਸ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹੋਣਗੀਆਂ ਅਤੇ ਤੁਰਨਾ, ਬੋਲਣਾ, ਸੌਣਾ ਆਦਿ ਉਸਦਾ ਵਿਵਹਾਰ (ਕਿਰਿਆਵਾਂ) ਹੋ ਸਕਦੇ ਹਨ। ਹਰੇਕ ਆਬਜੈਕਟ/ਵਸਤੂ ਦੀਆਂ ਆਪਣੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਇਹ ਦੱਸਦੀਆਂ ਹਨ ਕਿ ਉਹ ਕੀ ਚੀਜ਼ ਹੈ ਜਾਂ ਉਹ ਕੀ ਕਰਦੀ ਹੈ।

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿਧੀ ਵਿੱਚ, ਕਲਾਸਾਂ ਅਤੇ ਆਬਜੈਕਟਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਵਿਧੀ ਕੁਝ ਸੰਕਲਪਾਂ (concepts) ਜਿਵੇਂ ਕਿ: ਆਬਜੈਕਟ (object), ਕਲਾਸ (class), ਇਨਹੈਰੀਟੈਂਸ (Inheritance), ਪੋਲੀਮੋਰਫਿਜ਼ਮ (polymorphism), ਐਬਸਟਰੈਕਸ਼ਨ (absraction) ਅਤੇ ਐਨਕੈਪਸੂਲੇਸ਼ਨ (encapsulation) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਸਾਫਟਵੇਅਰ ਨੂੰ ਤਿਆਰ ਕਰਨ ਅਤੇ ਉਸਦੇ ਰੱਖ ਰਖਾਅ ਦੇ ਪ੍ਰੋਸੈਸ ਨੂੰ ਆਸਾਨ ਬਣਾਉਂਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (paradigm) ਜਿੱਥੇ ਹਰ ਚੀਜ਼ ਨੂੰ ਇੱਕ ਆਬਜੈਕਟ ਦੇ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਟਰੂਲੀ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ (Truly Object-Oriented Programming Language) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

Simula ਨੂੰ ਪਹਿਲੀ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਕਿ Smalltalk ਨੂੰ ਪਹਿਲੀ ਟਰੂਲੀ

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। Java, C++, C#, Phython, R, PHP, Visual Basic. NET, JavaScript, Ruby, Perl, Object Pascal, Dart, Swift, Scala, Kotlin, Common Lisp, MATLAB ਅਤੇ Smalltalk ਆਦਿ ਮਹੱਤਵਪੂਰਨ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ।

4.2 ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਸਿਧਾਂਤ (PRINCIPLES OF OBJECT-ORIENTED PROGRAMMING)

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸਭ ਤੋਂ ਵੱਧ ਵਰਤਿਆ ਜਾਣ ਵਾਲਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ ਹੈ। ਇਹ ਕਲਾਸਾਂ ਅਤੇ ਆਬਜੈਕਟਸ ਦੇ ਰੂਪ ਵਿੱਚ ਮੋਡਿਊਲਸ (modules) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਵੱਡੇ ਪੱਧਰ (large scale) ਦੇ ਐਪਲੀਕੇਸ਼ਨ ਸਾਫਟਵੇਅਰਾਂ ਨੂੰ ਵਿਕਸਤ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ। ਇਹਨਾਂ ਮੋਡੀਊਲਸ ਦੀ ਮਦਦ ਨਾਲ ਕਈ ਡਿਵੈਲਪਰ ਆਪਸ ਵਿਚ ਮਿਲ ਕੇ ਕੰਮ ਕਰਦੇ ਹੋਏ ਪੂਰਾ ਸਿਸਟਮ ਆਸਾਨੀ ਨਾਲ ਬਣਾ ਸਕਦੇ ਹਨ। ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇੱਕ ਐਪਲੀਕੇਸ਼ਨ ਦਾ ਐਂਟੀਟੀਜ਼ ਜਾਂ ਆਬਜੈਕਟਸ ਦੇ ਰੂਪ ਵਿੱਚ ਵਿਸ਼ਲੇਸ਼ਣ ਅਤੇ ਡਿਜ਼ਾਈਨ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਸਦਾ ਅਰਥ ਇਹ ਹੈ ਕਿ ਐਪਲੀਕੇਸ਼ਨ ਵਿਚ ਐਂਟੀਟੀਜ਼ ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ ਲਾਗੂ ਕਰਨਾ ਹੁੰਦਾ ਹੈ ਜਿਵੇਂ ਕਿ ਉਹ ਅਸਲ ਜੀਵਨ ਵਿੱਚ ਵੇਖੀਆਂ ਜਾਂਦੀਆਂ ਹਨ। ਇਸ ਲਾਗੂਕਰਨ ਵਿਚ ਹਰੇਕ ਐਂਟੀਟੀ ਨੂੰ ਕਿਰਿਆਵਾਂ (actions) ਅਤੇ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (attributes) ਨਾਲ ਜੋੜ ਕੇ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

OOP ਵਿੱਚ ਕੋਡ ਅਤੇ ਡਾਟਾ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਇਕਾਈ- ਆਬਜੈਕਟ (Object) ਵਿਚ ਇੱਕਠਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜਦੋਂ ਅਸੀਂ ਕਿਸੇ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਆਬਜੈਕਟ-ਅਧਾਰਿਤ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਉਸ ਵਿਚ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਫੰਕਸ਼ਨਾਂ (functions) ਜਾਂ ਪ੍ਰੋਸੀਜ਼ਰਾਂ (procedures) ਵਿੱਚ ਵੰਡਣ ਬਾਰੇ ਨਹੀਂ ਸੋਚਦੇ, ਸਗੋਂ ਸਮੱਸਿਆ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਉਸਨੂੰ ਆਬਜੈਕਟਸ (objects) ਵਿੱਚ ਵੰਡ ਕੇ ਹੱਲ ਕਰਨ ਬਾਰੇ ਸੋਚਦੇ ਹਾਂ। ਹਰ ਉਹ ਕੰਮ ਜੋ ਅਸੀਂ ਹੱਲ ਕਰਨ ਬਾਰੇ ਸੋਚਦੇ ਹਾਂ ਉਸਨੂੰ ਕਲਾਸਾਂ (classes) ਅਤੇ ਆਬਜੈਕਟਸ (Objects) ਦੇ ਰੂਪ ਵਿੱਚ ਹੱਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।OOP ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਅਸੀਂ ਫੰਕਸ਼ਨੈਲਿਟੀ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਵਾਲੀਆਂ ਕਲਾਸਾਂ (classes) ਬਣਾਉਂਦੇ ਹਾਂ ਅਤੇ ਫਿਰ ਉਸ ਕਲਾਸ ਦਾ ਇੱਕ ਆਬਜੈਕਟ ਬਣਾ ਕੇ ਉਸ ਕਲਾਸ ਦੇ ਫੰਕਸ਼ਨ ਨੂੰ ਕਾਲ ਕਰ ਸਕਦੇ ਹਾਂ।

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (paradigm) ਦੇ ਚਾਰ ਮੁੱਖ ਸਿਧਾਂਤ ਹੁੰਦੇ ਹਨ:

1. ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction)
2. ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation)
3. ਇਹੈਰੀਟੈਂਸ (Inheritance)
4. ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphim)

ਇਹਨਾਂ ਚਾਰਾਂ ਸਿਧਾਂਤਾਂ ਨੂੰ ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੇ ਚਾਰ ਥੰਮ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਇੱਕ ਸਫਲ ਪ੍ਰੋਗਰਾਮਰ ਬਣਨ ਲਈ ਇਹਨਾਂ ਨੂੰ ਸਮਝਣਾ ਬਹੁਤ ਜ਼ਰੂਰੀ ਹੈ। ਪਰ ਇਹਨਾਂ ਸੰਕਲਪਾਂ ਦੀ ਚਰਚਾ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ, ਇਹ ਜਾਣਨਾ ਜ਼ਰੂਰੀ ਹੈ ਕਿ ਕਲਾਸਾਂ (classes) ਅਤੇ ਆਬਜੈਕਟਸ (objects) ਕੀ ਹੁੰਦੇ ਹਨ ? ਆਉ OOP ਦੀ ਬਿਹਤਰ ਸਮਝ ਲਈ ਇਹਨਾਂ ਧਾਰਨਾਵਾਂ ਸੰਬੰਧੀ ਚਰਚਾ ਸ਼ੁਰੂ ਕਰਦੇ ਹਾਂ:

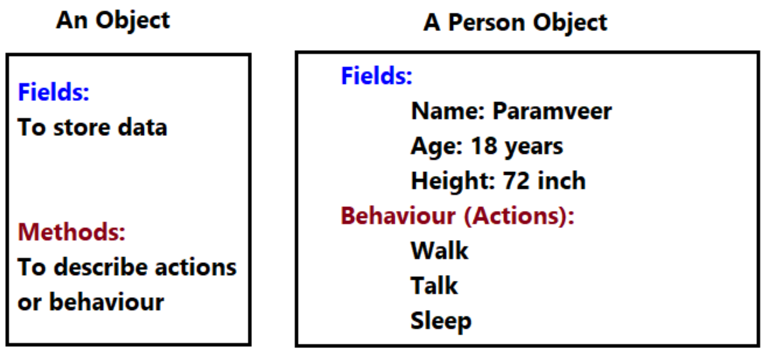
4.2.1 ਆਬਜੈਕਟਸ (Objects)

ਕੋਈ ਵੀ ਅਸਲ ਸੰਸਾਰ ਦੀ ਐਂਟੀਟੀ (entity) ਜਿਸਦੀ ਇਕ ਸਥਿਤੀ/ਸਟੇਟ (state) ਹੁੰਦੀ ਹੈ ਅਤੇ ਜੋ ਕੋਈ ਵਿਵਹਾਰ/ਕਿਰਿਆ (behaviour) ਕਰਦੀ ਹੈ, ਨੂੰ ਇੱਕ ਆਬਜੈਕਟ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: ਇੱਕ ਕੁਰਸੀ (chair), ਪੈੱਨ (pen), ਮੇਜ਼ (table), ਕੀਬੋਰਡ (keyboard), ਬਾਈਕ (bike), ਆਦਿ। ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ:

- ਅਸੀਂ ਆਬਜੈਕਟ ਦੀ ਸਥਿਤੀ/ਸਟੇਟ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ ਉਸ ਆਬਜੈਕਟ ਦੇ ਡਾਟਾ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਕਿਸੇ ਆਬਜੈਕਟ ਦਾ ਇਹ ਡਾਟਾ ਜਾਂ ਸਟੇਟ ਨੂੰ ਫੀਲਡਸ/ਵੇਰੀਏਬਲਜ਼ **(fields/variables)** ਦੇ ਰੂਪ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਫੀਲਡਸ **(fields)** ਨੂੰ ਐਟਰੀਬਿਊਟਸ (attributes) ਜਾਂ ਪ੍ਰਾਪਰਟੀਜ਼ (*properties*) ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
- ਆਬਜੈਕਟ ਦੇ ਵਿਵਹਾਰਕਿਰਿਆ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ, ਅਸੀਂ ਕੋਡ (Code) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਇਹ ਕੋਡ **(Code)** ਮੈਥਡਜ਼ ਜਾਂ ਫੰਕਸ਼ਨਜ਼ **(methods or functions)** ਦੇ ਰੂਪ ਵਿੱਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੈਥਡਜ਼ (methods) ਆਬਜੈਕਟ ਦੇ ਡਾਟਾ ਫੀਲਡਾਂ ਨੂੰ ਅਸੈੱਸ (access) ਅਤੇ ਉਹਨਾਂ ਵਿਚ ਸ਼ੋਧ (modify) ਕਰ ਸਕਦੇ ਹਨ।

ਹਰੇਕ ਆਬਜੈਕਟ ਦੀਆਂ ਆਪਣੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (properties) ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਇਹ ਦੱਸਦੀਆਂ ਹਨ ਕਿ ਇਹ ਕੀ ਹੈ ਜਾਂ ਇਹ ਕੀ ਕਰਦਾ ਹੈ ? ਰਨਟਾਈਮ 'ਤੇ ਆਬਜੈਕਟਸ ਬਣਾਉਣ ਲਈ ਟੈਂਪਲੇਟਸ (templates) ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਇਹਨਾਂ ਟੈਂਪਲੇਟਸ ਨੂੰ

ਕਲਾਸਾਂ (classes) ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇੱਕ ਆਬਜੈਕਟ ਮੈਮੋਰੀ ਵਿੱਚ ਕੁਝ ਥਾਂ ਲੈਂਦਾ ਹੈ। ਹੇਠਾਂ ਦਿੱਤਾ ਚਿੱਤਰ ਐਂਟੀਟੀ (entity) ਦੀ ਅਸਲ ਜੀਵਨ (real-life) ਆਧਾਰਿਤ ਉਦਾਹਰਨ ਸਹਿਤ ਇੱਕ ਆਬਜੈਕਟ ਦੀ ਧਾਰਨਾ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ:

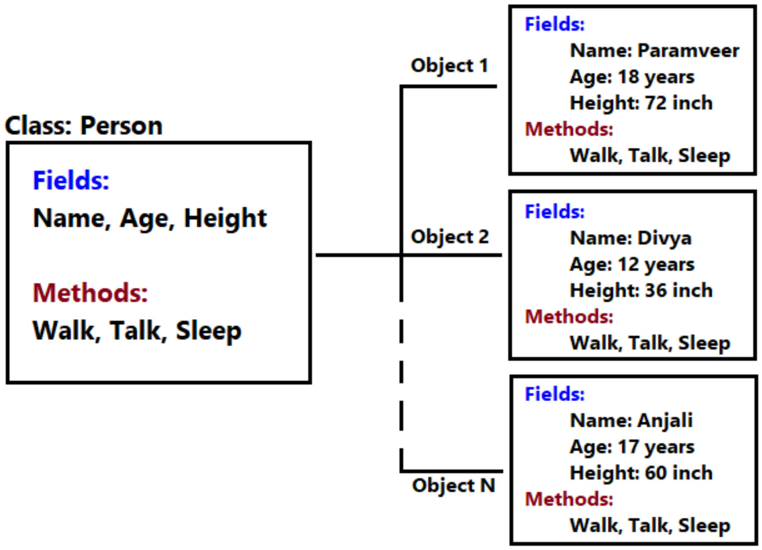


ਚਿੱਤਰ : 4.1 ਆਬਜੈਕਟ ਅਤੇ ਇਸਦੀਆਂ ਉਦਾਹਰਣਾਂ (Person ਆਬਜੈਕਟ)

4.2.2 ਕਲਾਸ (Class): ਇੱਕ ਕਲਾਸ ਉਹਨਾਂ ਆਬਜੈਕਟਸ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦੀ ਹੈ, ਜਿਹਨਾਂ ਵਿੱਚ ਸਮਾਨ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (same properties) ਅਤੇ ਸਾਂਝਾ ਵਿਵਹਾਰ (common behaviour) ਹੁੰਦਾ ਹੈ। ਕਲਾਸ ਨੂੰ ਇੱਕ ਬਲੂਪ੍ਰਿੰਟ (blueprint) ਵੀ ਮੰਨਿਆ ਜਾ ਸਕਦਾ ਹੈ ਜਿਸ ਤੋਂ ਅਸੀਂ ਇੱਕ ਆਬਜੈਕਟ (objects) ਬਣਾ ਸਕਦੇ ਹਾਂ। ਜਦੋਂ ਇਕ ਕਲਾਸ ਤੋਂ ਆਬਜੈਕਟਸ ਨੂੰ ਬਣਾਈਆਂ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਉਹ ਕਲਾਸ ਤੋਂ ਸਾਰੇ ਵੇਰੀਏਬਲਜ਼ ਅਤੇ ਮੈਥਡਜ਼ (methods) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰ ਲੈਂਦੇ ਹਨ।

ਇਕ ਕਲਾਸ ਉਸਦੇ ਆਬਜੈਕਟਸ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦੀ ਹੈ ਹਾਲਾਂਕਿ ਇਕ ਆਬਜੈਕਟ ਬਣਾਏ ਜਾਣ ਤੋਂ ਬਾਅਦ ਹੀ ਉਸਦੇ ਮੁੱਲ ਨਿਰਧਾਰਤ ਕੀਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਹਰੇਕ ਆਬਜੈਕਟ ਨੂੰ ਉਸਦੀ ਕਲਾਸ ਦਾ ਇੱਕ ਉਦਾਹਰਣ (instance) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। Person ਕਲਾਸ ਵਿੱਚ ਇਕ ਵਿਸ਼ੇਸ਼ ਵਿਅਕਤੀ ਜਿਵੇਂ ਕਿ: ਪਰਮਵੀਰ, ਦਿਵਿਆ, ਅੰਜਲੀ ਆਦਿ Person ਕਲਾਸ ਦੇ ਆਬਜੈਕਟਸ ਹੋਣਗੇ।

ਇੱਕ ਕਲਾਸ ਕੋਈ ਮੈਮੋਰੀ ਸਪੇਸ ਨਹੀਂ ਵਰਤਦੀ। ਇਹ ਡਾਟਾ ਦੀ ਸਿਰਫ ਇੱਕ ਲੋਜੀਕਲ ਪ੍ਰਤੀਨਿਧਤਾ (logical representation) ਹੈ। ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇੱਕ ਕਲਾਸ ਆਬਜੈਕਟ ਲਈ ਇੱਕ ਟੈਂਪਲੇਟ (template) ਹੁੰਦੀ ਹੈ ਅਤੇ ਇੱਕ ਆਬਜੈਕਟ ਕਲਾਸ ਦੀ ਇੱਕ ਉਦਾਹਰਣ (instance) ਹੁੰਦੀ ਹੈ। ਹੇਠਾਂ ਦਿੱਤਾ ਚਿੱਤਰ ਕਲਾਸ ਅਤੇ ਇਸਦੇ ਆਬਜੈਕਟਸ ਦੀ ਧਾਰਨਾ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ:



ਚਿੱਤਰ : 4.2 ਕਲਾਸ ਅਤੇ ਆਬਜੈਕਟ ਦੀ ਧਾਰਨਾ

4.2.3 ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) : ਇਹ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੀਆਂ ਸਭ ਤੋਂ ਮਹੱਤਵਪੂਰਨ ਧਾਰਨਾਵਾਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ। ਇਹ ਸਿਧਾਂਤ ਇਹ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮਿੰਗ ਐਂਟੀਟੀ ਵਿੱਚ ਇੱਕ ਅਸਲ-ਸੰਸਾਰ ਐਂਟੀਟੀ (real world entity) ਨੂੰ ਕਿਵੇਂ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਇੱਕ ਅਜਿਹੀ ਪ੍ਰਕਿਰਿਆ ਹੈ ਜਿਸ ਵਿੱਚ ਅਸੀਂ ਯੂਜ਼ਰ ਨੂੰ ਆਬਜੈਕਟ ਨਾਲ ਸੰਬੰਧਿਤ ਵੇਰਵੇ ਦਿਖਾਉਣ ਲਈ ਡਾਟਾ ਦੇ ਇੱਕ ਵੱਡੇ ਸੰਗ੍ਰਹ ਵਿਚੋਂ ਲੋੜੀਂਦਾ ਡਾਟਾ ਚੁਣਦੇ ਹਾਂ। ਇਹ ਪ੍ਰੋਗਰਾਮਿੰਗ ਜਟਿਲਤਾਵਾਂ (complexity) ਅਤੇ ਕੋਸ਼ਿਸ਼ਾਂ (efforts) ਨੂੰ ਘਟਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ।

ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਐਬਸਟਰੈਕਸ਼ਨ ਨੂੰ ਆਬਜੈਕਟ ਦੇ ਅੰਦਰੂਨੀ ਲਾਗੂਕਰਨ ਨੂੰ ਲੁਕਾਉਣ (hidding internal implementation) ਅਤੇ ਯੂਜ਼ਰਜ਼ ਨੂੰ ਸਿਰਫ਼ ਲੋੜੀਂਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਪ੍ਰਦਾਨ ਕਰਨ ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਆਓ ਇੱਕ ਉਦਾਹਰਣ ਦੀ ਮਦਦ ਨਾਲ ਇਸਨੂੰ ਸਮਝਦੇ ਹਾਂ:

ਐਬਸਟਰੈਕਸ਼ਨ ਦੀ ਅਸਲ ਜ਼ਿੰਦਗੀ ਦੀ ਉਦਾਹਰਨ: ਮੰਨ ਲਵੋ ਕਿ ਅਸੀਂ ਇੱਕ ਬਾਈਕ (bike) ਚਲਾ ਰਹੇ ਹਾਂ। ਇੱਥੇ ਅਸੀਂ ਸਿਰਫ ਬਾਈਕ ਚਲਾਉਣ ਬਾਰੇ ਸੋਚਾਂਗੇ: ਜਿਵੇਂ ਕਿ ਬਾਈਕ ਨੂੰ ਸਟਾਰਟ ਸਟਾਪ ਕਰਨਾ, ਤੇਜ਼ ਕਰਨਾ/ਬੇਕ ਲਗਾਉਣਾ, ਆਦਿ। ਬਾਈਕ ਚਲਾਉਣ ਸਮੇਂ ਅਸੀਂ ਇਹ ਨਹੀਂ ਸੋਚਦੇ ਕਿ ਅਸਲ ਵਿਚ ਇਹ ਸਟਾਰਟ/ਸਟਾਪ ਕਿਵੇਂ ਹੋ ਰਹੀ ਹੈ ਜਾਂ ਤੇਜ਼ ਚਲਾਉਣ/ਬ੍ਰੇਕ ਲਗਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਅੰਦਰੂਨੀ ਤੌਰ 'ਤੇ ਕਿਵੇਂ ਕੰਮ ਕਰ ਰਹੀ ਹੈ। ਸਾਨੂੰ ਉਹਨਾਂ ਵੇਰਵਿਆਂ ਵਿੱਚ ਕੋਈ ਦਿਲਚਸਪੀ ਨਹੀਂ ਹੁੰਦੀ, ਅਸੀਂ ਸਿਰਫ ਬਾਈਕ ਦੀ ਵਰਤੋਂ ਕਰਨ ਬਾਰੇ ਹੀ ਸੋਚਦੇ ਹਾਂ।

4.2.4 ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation) : ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਡਾਟਾ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਸਮੇਟਣ (wrapping) ਦੀ ਇੱਕ ਪ੍ਰਕਿਰਿਆ ਹੈ। ਇਸ ਸਿੰਗਲ ਯੂਨਿਟ ਨੂੰ ਕਲਾਸ (class) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਇਸਨੂੰ ਇੱਕ ਅਜਿਹੇ ਸੁਰੱਖਿਆ ਰੈਪਰ (protective wrapper) ਵਜੋਂ ਮੰਨ ਸਕਦੇ ਹਾਂ ਜੋ ਰੈਪਰ ਦੇ ਅੰਦਰ ਪਰਿਭਾਸ਼ਿਤ ਕੋਡ ਦੀ ਰੈਂਡਮ ਅਸੈਸ ਨੂੰ ਬਾਹਰੋਂ ਰੋਕਦਾ ਹੈ। ਇਹ ਇੱਕ ਕੈਪਸੂਲ ਵਾਂਗ ਹੁੰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਕਈ ਤਰ੍ਹਾਂ ਦੀਆਂ ਦਵਾਈਆਂ ਦਾ ਮਿਸ਼ਰਣ ਹੁੰਦਾ ਹੈ।

ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਡਾਟਾ ਦੀ ਸੁਰੱਖਿਆ ਨੂੰ ਵਧਾਉਂਦੀ ਹੈ, ਕਿਉਂਕਿ ਇਸਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇੱਕ ਸਿੰਗਲ ਟਾਸਕ ਨਾਲ ਸਬੰਧਤ ਹਰ ਚੀਜ਼ ਨੂੰ ਇਕੱਠਾ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਲੋੜ ਅਨੁਸਾਰ ਡਾਟਾ ਦਾ ਅਸੈਸ ਪ੍ਰਦਾਨ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਅਤੇ ਇਸ ਕੰਮ ਲਈ ਡਾਟਾ ਛੁਪਾਉਣ (Data Hiding) ਦੀ ਧਾਰਨਾ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦਾ ਹੈ। ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ ਬਿਆਨ ਕੀਤਾ ਜਾ ਸਕਦਾ

Encapsulation = Data Hiding + Abstraction

ਇਸ ਤਰ੍ਹਾਂ ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਸਾਡੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਛੁਪਾ ਕੇ (selective hiding of properties and methods) ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਲਪੇਟ (wrapping) ਕੇ ਰੱਖਦੀ ਹੈ, ਜਿਸਨੂੰ ਕਲਾਸ (class) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਅਸੈੱਸ ਮੋਡੀਫਾਇਰ **private** ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਡਾਟਾ ਨੂੰ ਛੁਪਾ ਕੇ ਰੱਖਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਪ੍ਰਾਈਵੇਟ (Private) ਡਾਟਾ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਕਲਾਸ ਤੋਂ ਬਾਹਰ ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ ਦੁਆਰਾ ਐਕਸੈਸ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਆਓ ਇੱਕ ਉਦਾਹਰਣ ਦੀ ਮਦਦ ਨਾਲ ਸਮਝੀਏ:

ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਦੀ ਅਸਲ ਜ਼ਿੰਦਗੀ ਦੀ ਉਦਾਹਰਨ : ਬਾਜ਼ਾਰ ਵਿੱਚ ਵੱਖ-ਵੱਖ ਸਿਹਤ ਸਮੱਸਿਆਵਾਂ ਦੇ ਇਲਾਜ ਲਈ ਕੈਪਸੂਲਜ਼ (Capsules) ਉਪਲਬਧ ਹਨ। ਕੈਪਸੂਲ ਵਿੱਚ ਇੱਕ ਪੂਰੀ ਦਵਾਈ ਬਣਾਉਣ ਲਈ ਵੱਖ-ਵੱਖ ਰਚਨਾਵਾਂ ਦਾ ਸਮੂਹ ਇੱਕਠਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਕਿਸੇ ਖਾਸ ਸਿਹਤ ਸਮੱਸਿਆ ਨੂੰ ਠੀਕ ਕਰਦਾ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਕੈਪਸੂਲ ਵਿੱਚ ਵੱਖ-ਵੱਖ ਰਚਨਾਵਾਂ ਦੇ ਸਮੂਹ ਨੂੰ ਇੱਕਠਾ ਕਰਕੇ ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਦੇ ਤੌਰ ਤੇ ਰੱਖਣਾ ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਦਾ ਹੀ ਇੱਕ ਰੂਪ ਹੈ।

ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ:

1. ਇਹ ਡਾਟਾ ਲੁਕਾਉਣ ਦੀ ਧਾਰਨਾ (concept of Data Hiding) ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਕਲਾਸ ਦੇ ਮੈਂਬਰਾਂ ਨੂੰ ਵਾਧੂ ਸੁਰੱਖਿਆ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।
2. ਇਹ ਡਾਟਾ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਇੱਕਠਾ ਕਰਕੇ ਰੱਖਦਾ ਹੈ।
3. ਇਹ ਡਾਟਾ ਨੂੰ ਵਾਧੂ ਸੁਰੱਖਿਆ ਮੁਹਈਆ ਕਰਵਾਉਂਦਾ ਹੈ ਜੋ ਅਣਅਧਿਕਾਰਤ (unauthorized) ਲੋਕਾਂ ਦੁਆਰਾ ਡਾਟਾ ਨੂੰ ਅਸੈਸ ਕਰਨ 'ਤੇ ਪਾਬੰਦੀ (Restrictions) ਲਗਾਉਂਦਾ ਹੈ।

4.2.5 ਇਨਹੈਰੀਟੈਂਸ (Inheritance): ਇਹੈਰੀਟੈਂਸ ਇੱਕ ਅਜਿਹੀ ਵਿਧੀ ਹੈ ਜਿਸ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਆਪਣੇ ਪੇਰੈਂਟ (parent) ਆਬਜੈਕਟ ਦੀਆਂ ਸਾਰੀਆਂ ਅਵਸਥਾਵਾਂ ਅਤੇ ਵਿਵਹਾਰਾਂ (states and behaviours) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਇੱਕ ਬੱਚੇ ਨੂੰ ਉਸਦੇ ਮਾਤਾ-ਪਿਤਾ ਦੇ ਗੁਣ ਵਿਰਾਸਤ (inherits) ਵਿੱਚ ਮਿਲਦੇ ਹਨ।

ਆਬਜੈਕਟ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ ਇਨਹੈਰੀਟੈਂਸ ਇਕ ਅਜਿਹੀ ਪ੍ਰਕਿਰਿਆ ਹੈ ਜਿਸ ਵਿਚ ਮੌਜੂਦਾ ਕਲਾਸਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ

ਨਵੀਆਂ ਕਲਾਸਾਂ ਬਣਾਈਆਂ (creating new classes using the existing classes) ਜਾਂਦੀਆਂ ਹਨ। ਇਸ ਪ੍ਰਕਿਰਿਆ ਦੀ ਵਰਤੋਂ ਨਾਲ ਨਵੀਂ ਕਲਾਸ ਮੌਜੂਦਾ ਕਲਾਸ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਨੂੰ ਗ੍ਰਹਿਣ ਕਰਦੀ ਹੈ। ਇਹੈਰੀਟੈਂਸ ਨਾਲ ਅਸੀਂ ਮੌਜੂਦਾ ਕਲਾਸ ਦੇ ਫੀਲਡਜ਼ ਅਤੇ ਮੈਥਡਜ਼ (fields and methods) ਦੀ ਮੁੜ ਵਰਤੋਂ (reuse) ਕਰ ਸਕਦੇ ਹਾਂ। ਇਸ ਤਰ੍ਹਾਂ ਇਹੈਰੀਟੈਂਸ ਮੁੜ-ਵਰਤੋਂਯੋਗਤਾ (Reusability) ਦੀ ਸਹੂਲਤ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ ਅਤੇ ਇਹ OOP ਦੀ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਧਾਰਨਾ ਮੰਨੀ ਜਾਂਦੀ ਹੈ।

ਮੰਨ ਲਓ ਇੱਕ ਪੇਰੈਂਟ ਕਲਾਸ (parent class) ਉਪਲਬਧ ਹੈ ਜਿਸ ਵਿਚ ਕੁੱਝ ਮੈਥਡਜ਼ (methods) ਉਪਲਬਧ ਹਨ ਅਤੇ ਹੁਣ ਅਸੀਂ ਇੱਕ ਨਵੀਂ ਕਲਾਸ ਬਣਾਵਾਂਗੇ, ਜਿਸਨੂੰ ਚਾਈਲਡ ਕਲਾਸ (child class) ਕਹਾਂਗੇ, ਜਿਸ ਵਿਚ ਇਸ ਕਲਾਸ ਦੇ ਆਪਣੇ ਮੈਥਡਜ਼ (methods) ਹੋਣਗੇ। ਹੁਣ ਜਦੋਂ ਇੱਕ ਚਾਈਲਡ ਕਲਾਸ ਨੂੰ ਪੇਰੈਂਟ ਕਲਾਸ ਤੋਂ ਇਨਹੈਰਿਟ (inherit) ਕੀਤਾ ਜਾਵੇਗਾ, ਤਾਂ ਪੇਰੈਂਟ ਕਲਾਸ ਨਾਲ ਸਬੰਧਤ ਨਾਨ ਪ੍ਰਾਇਵੇਟ ਮੈਥਡਜ਼ ਚਾਈਲਡ ਕਲਾਸ ਵਿੱਚ ਚਾਈਲਡ ਕਲਾਸ ਦੇ ਆਪਣੇ ਮੈਥਡਜ਼ ਨਾਲ ਉਪਲਬਧ ਹੋਣਗੇ। ਮੌਜੂਦਾ ਕਲਾਸਾਂ (existing classes) ਤੋਂ ਨਵੀਆਂ ਵਧੀਆਂ ਕਲਾਸਾਂ (new enhanced classes) ਬਣਾਉਣ ਦੀ ਅਜਿਹੀ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਇਨਹੈਰੀਟੈਂਸ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ਇਨਹੈਰੀਟੈਂਸ ਲਈ ਦੋ ਸ਼ਬਦਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ - ਬੇਸ ਕਲਾਸ (Base Class) ਅਤੇ ਡਰਾਈਵਡ ਕਲਾਸ (Derived Class)।

- **ਬੇਸ ਕਲਾਸ (Base Class)** - ਇਸਨੂੰ ਪੇਰੈਂਟ ਕਲਾਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੁੱਖ ਕਲਾਸ ਹੁੰਦੀ ਹੈ।ਜਿਸ ਵਿਚ ਬੁਨਿਆਦੀ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ (basic properties and methods) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੁੰਦਾ ਹੈ।
- **ਡਰਾਈਵਡ ਕਲਾਸ (Derived Class)** - ਇਹ ਬੇਸ ਕਲਾਸ ਦੀ ਐਕਸਟੈਨਸ਼ਨ ਹੁੰਦੀ ਹੈ ਅਤੇ ਜਿਸਨੂੰ ਚਾਈਲਡ ਕਲਾਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਵਿੱਚ ਆਪਣੀਆਂ ਖੁੱਦ ਦੀਆਂ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਦੇ ਨਾਲ ਉਹ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਵੀ ਹੁੰਦੇ ਹਨ ਜੋ ਬੇਸ ਕਲਾਸ ਵਿੱਚ ਮੌਜੂਦ ਹੁੰਦੇ ਹਨ।

ਇਨਹੇਰੀਟੈਂਸ ਦੀ ਅਸਲ ਜ਼ਿੰਦਗੀ ਦੀ ਉਦਾਹਰਨ: ਅਸੀਂ ਸਭ ਨੇ ਮੋਬਾਈਲ ਫ਼ੋਨਾਂ ਲਈ ਅੱਪਡੇਟ ਦੇਖੇ ਹਨ। ਸ਼ੁਰੂ ਵਿੱਚ ਮੋਬਾਈਲ ਫ਼ੋਨ ਸਿਰਫ਼ ਗੱਲ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਸਨ ਅਤੇ ਫਿਰ ਮੀਡੀਆ ਅਸੈਸ, ਇੰਟਰਨੈਟ, ਕੈਮਰਾ ਆਦਿ ਲਈ ਵਰਤੇ ਜਾਣ ਲੱਗੇ। ਇਹ ਉਹ ਵਿਕਾਸ ਹੈ ਜਿਸ ਵਿਚ ਪੂਰੀ ਕਾਰਜਕੁਸ਼ਲਤਾ ਦਾ ਮੁੜ ਨਿਰਮਾਣ ਕੀਤੇ ਬਿਨਾਂ, ਮੌਜੂਦਾ ਤਕਨੀਕ ਵਿਚ ਕੁੱਝ ਵਾਧੂ ਵਿਸ਼ੇਸ਼ਤਾ ਜੋੜ ਕੇ ਨਵਾਂ ਉਤਪਾਦ ਬਣਾ ਕੇ ਪੇਸ਼ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਲਈ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਹੀ ਇੱਕ ਉਦਾਹਰਣ ਹੈ।

4.2.6 ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism): ਪੌਲੀਮੋਰਫਿਜ਼ਮ ਕਿਸੇ ਆਬਜੈਕਟ ਦੀ ਕਈ ਰੂਪ ਧਾਰਨ ਕਰਨ ਦੀ ਯੋਗਤਾ ਹੁੰਦੀ ਹੈ। ਇਹ ਆਬਜੈਕਟ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਸਿਧਾਂਤ ਦਾ ਸਭ ਤੋਂ ਜ਼ਰੂਰੀ ਸੰਕਲਪ ਹੈ। ਪੌਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਸ਼ਬਦ ਦੋ ਸ਼ਬਦਾਂ ਤੋਂ ਬਣਿਆ ਹੈ: **Poly + morph** (ਪੌਲੀ + ਮੋਰਫ)। ਇੱਥੇ 'poly' ਪੌਲੀ ਦਾ ਅਰਥ ਹੈ 'ਬਹੁਤ ਸਾਰੇ' ਅਤੇ (ਮੋਰਫ) ਦਾ ਅਰਥ ਹੈ 'ਰੂਪ'। ਇਸ ਤਰ੍ਹਾਂ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਦਾ ਅਰਥ ਹੁੰਦਾ ਹੈ ਕਈ ਰੂਪ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਇੱਕ ਵਸਤੂ ਦੇ ਕਈ ਰੂਪਾਂ ਨੂੰ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਆਬਜੈਕਟ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ ਜਾਂ ਮੈਥਡ ਨਾਲ ਇੱਕ ਤੋਂ ਵੱਧ ਨਾਮ ਜੁੜੇ ਹੋ ਸਕਦੇ ਹਨ। ਇਹ ਹੋਰ ਕੁੱਝ ਨਹੀਂ ਸਗੋਂ **ਪੋਲੀਮੋਰਫਿਜ਼ਮ** ਹੀ ਹੁੰਦਾ ਹੈ। ਆਉ ਅਸੀਂ ਇੱਕ ਅਸਲ-ਜੀਵਨ ਦੀ ਉਦਾਹਰਨ ਨਾਲ ਪੋਲੀਮੋਫਿਜ਼ਮ ਦੀ ਧਾਰਨਾ ਨੂੰ ਸਮਝੀਏ:

ਪੌਲੀਮੋਰਫਿਜ਼ਮ ਦੀਆਂ ਅਸਲ ਜ਼ਿੰਦਗੀ ਦੀਆਂ ਉਦਾਹਰਣਾਂ:

- ਇੱਕ ਅਧਿਆਪਕ ਦਾ ਵਿਦਿਆਰਥੀਆਂ ਨਾਲ ਵਿਵਹਾਰ।
- ਇੱਕ ਅਧਿਆਪਕ ਦਾ ਆਪਣੇ ਸੀਨੀਅਰਾਂ ਨਾਲ ਵਿਵਹਾਰ।

ਇੱਥੇ ਅਧਿਆਪਕ ਇੱਕ ਆਬਜੈਕਟ ਹੈ ਪਰ ਉਸਦਾ ਵਿਵਹਾਰ ਵੱਖ-ਵੱਖ ਸਥਿਤੀਆਂ ਵਿੱਚ ਵੱਖਰਾ-ਵੱਖਰਾ ਹੋਵੇਗਾ, ਭਾਵ ਉਸਦਾ ਵਿਵਹਾਰ ਵਿਦਿਆਰਥੀਆਂ ਨਾਲ ਵੱਖਰਾ ਹੋਵੇਗਾ ਅਤੇ ਆਪਣੇ ਸੀਨੀਅਰਾਂ ਨਾਲ ਵੱਖਰਾ ਹੋਵੇਗਾ, ਜੋ ਕੇ ਅਧਿਆਪਕ ਦੇ ਇਕ ਤੋਂ ਵੱਧ ਰੂਪਾਂ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ।

ਪੌਲੀਮੋਰਫਿਜ਼ਮ ਦੀਆਂ ਕਿਸਮਾਂ (Types of Polymorphism):

ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਦੋ ਤਰ੍ਹਾਂ ਦੇ ਹੁੰਦੇ ਹਨ:

1. ਕੰਪਾਈਲ-ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Compile-Time Polymorphism): ਇਸਨੂੰ ਸਟੈਟਿਕ (static) ਪੌਲੀਮੋਰਫਿਜ਼ਮ ਜਾਂ ਅਰਲੀ ਬਾਈਡਿੰਗ (early binding) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਕੰਪਾਈਲਰ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਈਲ ਕਰਦੇ ਸਮੇਂ ਮੈਥਡ ਸਿਗਨੇਚਰ (method signature) ਦੀ ਜਾਂਚ ਕਰਦਾ ਹੈ ਅਤੇ ਇਹ ਨਿਰਧਾਰਤ ਕਰਦਾ ਹੈ ਕਿ ਦਿੱਤੇ ਮੈਥਡ ਨੂੰ ਕਾਲ (given method call) ਕਰਨ ਲਈ ਕਿਸ ਮੈਥਡ ਨੂੰ ਕਾਲ ਕੀਤਾ ਜਾਵੇ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਸ ਕਿਸਮ ਦੇ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਵਿੱਚ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਇਲ ਕਰਨ ਸਮੇਂ ਇੱਕ ਆਬਜੈਕਟ ਆਪਣੀ ਕਾਰਜਸ਼ੀਲਤਾ ਨਾਲ ਬੱਝੀਆ (object is bound with its functionality) ਹੁੰਦਾ ਹੈ।

ਕੰਪਾਈਲ ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ **(Method Overloading)** ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਾਸਿਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading) : ਜਦੋਂ ਇੱਕ ਤੋਂ ਵੱਧ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕੋ ਨਾਮ ਨਾਲ ਘੋਸ਼ਿਤ (declared) ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਪਰ ਉਹਨਾਂ ਨੂੰ ਪੈਰਾਮੀਟਰਾਂ (Parameters) ਦੀ ਇੱਕ ਵੱਖਰੀ ਸੰਖਿਆ ਜਾਂ ਪੈਰਾਮੀਟਰ ਦੇ ਵੱਖਰੇ ਡਾਟਾ-ਟਾਈਪ ਦੇ ਨਾਲ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਉਸਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

2. ਰਨ-ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Run-Time Polymorphism) : ਇਸਨੂੰ ਡਾਇਨਾਮਿਕ (dynamic) ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਜਾਂ ਲੇਟ ਬਾਈਡਿੰਗ (Late Binding) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਰਨਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਵਿੱਚ, ਰਨਟਾਈਮ ਦੌਰਾਨ ਮੈਥਡ ਨੂੰ ਕਾਲ ਕਰਨ ਲਈ ਜਾਣਕਾਰੀ ਇਕੱਠੀ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਸ ਕਿਸਮ ਦੇ ਪੋਲੀਮੋਫਿਜ਼ਮ ਵਿੱਚ ਰਨ ਟਾਈਮ ਸਮੇਂ ਇੱਕ ਆਬਜੈਕਟ ਨੂੰ ਉਸਦੀ ਕਾਰਜਸ਼ੀਲਤਾ ਨਾਲ ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ।ਰਨ-ਟਾਈਮ ਪੋਲੀਮੋਫਿਜ਼ਮ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ (Method Overriding) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ (Method Overloading) : ਜਦੋਂ ਇੱਕ ਤੋਂ ਵੱਧ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕੋ ਨਾਮ ਅਤੇ ਇੱਕੋ ਹਸਤਾਖਰ (same signature) ਨਾਲ ਵੱਖਰੀਆਂ ਕਲਾਸਾਂ ਵਿੱਚ ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਸ ਕੀਰਿਆ ਨੂੰ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਡਰਾਈਵਡ ਕਲਾਸ ਵਿਚ ਇਕੋ ਨਾਮ ਨਾਲ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਮੈਥਡ ਬੇਸ ਕਲਾਸ ਵਿਚ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤੇ ਗਏ ਮੈਥਡ ਨੂੰ ਓਵਰਰਾਈਡ ਕਰੇਗਾ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੋਇਆ ਕਿ ਬੇਸ ਕਲਾਸ ਦੇ ਮੈਥਡ ਨੂੰ ਡਰਾਈਵਡ ਕਲਾਸ ਦੇ ਮੈਥਡ ਨਾਲ ਸ਼ੈਡੋਡ (shadowed) ਕੀਤਾ ਗਿਆ ਹੈ ਅਤੇ ਜਦੋਂ ਇਸ ਓਵਰਰੀਡਨ ਮੈਥਡ (Overidden Method) ਨੂੰ ਕਾਲ ਕੀਤਾ ਜਾਵੇਗਾ ਤਾਂ ਉਸਨੂੰ ਓਵਰਰਾਈਡ ਮੈਥਡ ਦੇ ਅਧਾਰ ਤੇ ਕਾਲ ਕੀਤਾ ਜਾਵੇਗਾ।

4.3 ਜਾਵਾ ਨਾਲ ਜਾਣ ਪਛਾਣ (INTRODUCTION TO JAVA)

JAVA ਨੂੰ ਸਾਲ 1995 ਵਿੱਚ ਸਨ ਮਾਈਕੋਸਿਸਟਮਜ਼ **(Sun Microsystems)** ਕੰਪਨੀ ਵਿੱਚ ਜੇਮਸ ਗੋਸਲਿੰਗ (James Gosling) ਦੁਆਰਾ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਸੀ। ਬਾਅਦ ਵਿੱਚ ਇਸਦੀ ਮਲਕੀਅਤ ਓਰੇਕਲ (Oracle) ਕਾਰਪੋਰੇਸ਼ਨ ਦੁਆਰਾ ਹਾਸਿਲ ਕਰ ਲਈ ਗਈ ਸੀ। ਜਾਵਾ ਇੱਕ ਸਧਾਰਨ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਹੈ। ਜਾਵਾ ਇੱਕ ਆਮ ਉਦੇਸ਼ (general-purpose), ਕਲਾਸ-ਅਧਾਰਿਤ (class-based), ਆਬਜੈਕਟ-ਅਧਾਰਿਤ (object-oriented) ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਹੈ, ਜੋ ਕਿ ਵਿੰਡੋਜ਼, ਮੈਕ (Mac) ਅਤੇ ਲਾਇਨੇਕਸ (Linux) ਵਰਗੇ ਵੱਖ ਵੱਖ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮਾਂ 'ਤੇ ਕੰਮ ਕਰਦੀ ਹੈ। JAVA ਵਿਚ ਪ੍ਰੋਗਰਾਮਿੰਗ, ਕੰਪਾਇਲੇਸ਼ਨ, ਅਤੇ ਡੀਬੱਗਿੰਗ ਕਰਨਾ ਆਸਾਨ ਹੁੰਦਾ ਹੈ। ਇਹ ਮੁੜ ਵਰਤੋਂ ਯੋਗ ਕੋਡ (reusable code) ਅਤੇ ਮਾਡਿਊਲਰ (modular) ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦੀ ਹੈ।JAVA ਵਿਚ ਇਕ ਵਾਰ ਪ੍ਰੋਗਰਾਮ ਬਨਾਉਣ ਤੋਂ ਬਾਅਦ ਕਿਸੇ ਵੀ ਤਰ੍ਹਾਂ ਦੀ ਮਸ਼ੀਨ ਉਪਰ ਚੱਲਣ ਯੋਗ (Write Once Run Anywhere) ਹੁੰਦਾ ਹੈ। ਸਾਧਾਰਣ ਸ਼ਬਦਾਂ ਵਿਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਕੰਪਾਇਲ ਕੀਤਾ ਗਿਆ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਨੂੰ ਸਪੋਰਟ ਕਰਨ ਵਾਲੇ ਸਾਰੇ ਪਲੇਟਫਾਰਮਾਂ 'ਤੇ ਚੱਲ ਸਕਦਾ ਹੈ। ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਈਲ ਹੋਣ ਤੋਂ ਬਾਅਦ ਬਾਈਟ ਕੋਡ ਵਿਚ ਤਬਦੀਲ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿਸੇ ਵੀ ਜਾਵਾ ਵਰਚੁਅਲ ਮਸ਼ੀਨ (Java Virtual Machine) 'ਤੇ ਚੱਲ ਸਕਦਾ ਹੈ। ਜਾਵਾ ਦਾ ਸਿੰਟੈਕਸ C/C++ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੇ ਸਮਾਨ ਹੁੰਦਾ ਹੈ।

ਅਸੀਂ ਜਾਵਾ ਦੀ ਵਰਤੋਂ ਹੇਠਾਂ ਦਿਤੀਆਂ ਕਿਸਮਾਂ ਦੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਡਿਵੈਲਪ ਕਰਨ ਲਈ ਕਰ ਸਕਦੇ ਹਾਂ:

- ❖ ਡੈਸਕਟਾਪ ਐਪਲੀਕੇਸ਼ਨਾਂ (Desktop Applications)
- ❖ ਵੈੱਬ ਐਪਲੀਕੇਸ਼ਨਾਂ (Web Applications)
- ❖ ਮੋਬਾਈਲ ਐਪਲੀਕੇਸ਼ਨਾਂ (Andriod Mobile Application)
- ❖ ਵੈੱਬ ਅਤੇ ਐਪਲੀਕੇਸ਼ਨ ਸਰਵਰ (Web and Applications Server)
- ❖ ਬਿੱਗ ਡਾਟਾ ਪ੍ਰੋਸੈਸਿੰਗ ਐਪਲੀਕੇਸ਼ਨਾਂ (Big data Proceesing Application)
- ❖ ਏਮਬੈਡਡ (Embedded) ਸਿਸਟਮ ਅਤੇ ਹੋਰ ਵੀ ਬਹੁਤ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀਆਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਜਾਵਾ ਵਿਚ ਤਿਆਰ ਕੀਤੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ।

4.3.1 ਜਾਵਾ ਵਾਤਾਵਰਨ (Java Environment)

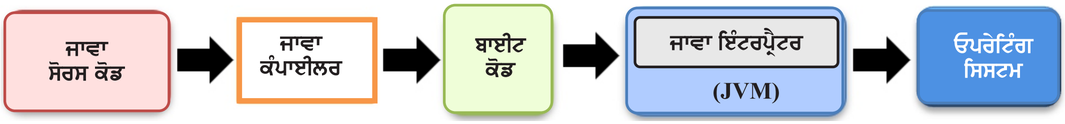
C ਅਤੇ C++ ਸਮੇਤ ਕਈ ਹੋਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੇ ਵਿਪਰੀਤ, ਜਦੋਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਹ ਕਿਸੇ ਵਿਸ਼ੇਸ਼ ਪਲੇਟਫਾਰਮ-ਮਸ਼ੀਨ (platform specific machine) ਵਿੱਚ ਕੰਪਾਇਲ ਨਹੀਂ ਹੁੰਦਾ; ਇਸ ਦੀ ਬਜਾਏ ਇਸ ਨੂੰ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਬਾਈਟ-ਕੋਡ (platform independent byte-code) ਵਿੱਚ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਬਾਈਟ-ਕੋਡ ਨੂੰ ਫਿਰ ਜਾਵਾ ਵਰਚੁਅਲ ਮਸ਼ੀਨ (JVM) ਦੁਆਰਾ ਇੰਟਰਪ੍ਰੇਟ (ਐਗਜ਼ੀਕਿਊਟ) ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਬਾਈਟ-ਕੋਡ ਕਿਸੇ ਵੀ ਪਲੇਟਫਾਰਮ 'ਤੇ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ਭਾਵੇਂ ਉਹ ਵਿੰਡੋਜ਼ (Windows), ਲੀਨਕਸ (Linux) ਜਾਂ ਮੈਕ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ (macOS)

ਹੋਵੇ।ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜੇਕਰ ਅਸੀਂ ਵਿੰਡੋਜ਼ ਉੱਤੇ ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਇਲ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਇਸਨੂੰ ਬਿਨ੍ਹਾਂ ਕੋਈ ਬਦਲਾਵ ਕੀਤੇ ਲੀਨਕਸ ਉੱਤੇ ਵੀ ਚਲਾ ਸਕਦੇ ਹਾਂ ਅਤੇ ਠੀਕ ਇਸੇ ਤਰ੍ਹਾਂ ਲੀਕਨਸ ਪਲੇਟਫਾਰਮ ਉਪਰ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਈਲ ਕਰਕੇ ਉਸ ਕੋਡ ਨੂੰ ਵਿੰਡੋਜ਼ ਪਲੇਟਫਾਰਮ ਉਪਰ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਹਰੇਕ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦਾ ਇੱਕ ਵੱਖਰਾ JVM ਹੁੰਦਾ ਹੈ, ਪਰ ਬਾਈਟ-ਕੋਡ ਦੇ ਲਾਗੂ (exection) ਹੋਣ ਤੋਂ ਬਾਅਦ ਸਾਰੇ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦੁਆਰਾ ਪੈਦਾ ਕੀਤੀ ਗਈ ਆਉਟਪੁੱਟ ਇੱਕੋ ਜਿਹੀ ਹੁੰਦੀ ਹੈ। ਇਸ ਲਈ ਜਾਵਾ ਨੂੰ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਭਾਸ਼ਾ (platform indepent language) ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ।

ਲੀਨਕਸ ਅਤੇ ਵਿੰਡੋਜ਼ ਪਲੇਟਫਾਰਮਾਂ ਵਿਚ ਜਾਵਾ ਦੀਆਂ ਵਾਤਾਵਰਣ ਸੈਟਿੰਗਾਂ ਨੂੰ ਹੇਠਾਂ ਦਿਖਾਇਆ ਗਿਆ ਹੈ।JVM, JRE ਅਤੇ JDK ਇਹ ਸਾਰੇ JAVA ਦੇ ਜ਼ਰੂਰੀ ਭਾਗ ਹਨ ਅਤੇ ਇਹ ਸਾਰੇ ਪਲੇਟਫਾਰਮ-ਨਿਰਭਰ (platform-dependent) ਹਨ ਕਿਉਂਕਿ ਹਰੇਕ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦੀ ਸੰਰਚਨਾ (configuration) ਵੱਖਰੀ ਹੁੰਦੀ ਹੈ। ਪਰ ਜਾਵਾ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ (platform indepent) ਹੈ। ਇਸ ਪਾਠ ਵਿਚ ਅੱਗੇ ਵਧਣ ਤੋਂ ਪਹਿਲਾਂ ਸਾਨੂੰ ਜਾਵਾ ਵਾਤਾਵਰਣ ਨਾਲ ਸੰਬੰਧਤ ਕੁੱਝ ਗੱਲਾਂ ਸਪੱਸ਼ਟ ਹੋਣੀਆਂ ਚਾਹੀਦੀਆਂ ਹਨ ਜੋ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਦੀ ਵਰਤੋਂ ਨਾਲ ਬਿਹਤਰ ਸਮਝੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ:

ਚਿੱਤਰ : 4.3 ਜਾਵਾ ਦੇ ਵਾਤਾਵਰਣ ਦੀ ਸਟਰਕਚਰ (JDJ, JRE ਅਤੇ JVM)

- **JDK (ਜਾਵਾ ਡਿਵੈਲਪਮੈਂਟ ਕਿੱਟ) :** ਇਹ JAVA ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਲਈ ਇੱਕ ਪ੍ਰਮੁੱਖ ਪਲੇਟਫਾਰਮ ਕੰਪੋਨੈਂਟ ਹੈ। ਇਸਦਾ ਮੁੱਖ ਭਾਗ ਜਾਵਾ ਕੰਪਾਈਲਰ ਹੈ। ਇਸ ਵਿੱਚ JRE + ਡਿਵੈਲਪਮੈਂਟ ਟੂਲਜ਼ ਸ਼ਾਮਲ ਹਨ। ਡਿਵੈਲਪਮੈਂਟ ਟੂਲਜ਼ ਵਿੱਚ JAVA ਕੰਪਾਈਲਰ (JAVAC), JAVADOC, Debugger ਆਦਿ ਸ਼ਾਮਲ ਹਨ। JDK ਕੰਪੋਨੈਂਟ ਸਾਫਟਵੇਅਰ ਡਿਵੈਲਪਰਾਂ ਲਈ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ।
- **JRE (ਜਾਵਾ ਰਨਟਾਈਮ ਐਨਵਾਇਰਮੈਂਟ) :** JRE ਵਿੱਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੀਆਂ ਜਾਵਾ ਲਾਇਬ੍ਰੇਰੀਆਂ ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ।JRE ਨੂੰ ਸਿਰਫ਼ JAVA ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਇੱਕ ਸਟੈਂਡਅਲੋਨ ਕੰਪੋਨੈਂਟ (standalone component) ਵਜੋਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ, ਪਰ ਇਹ JDK ਦਾ ਵੀ ਹਿੱਸਾ ਹੁੰਦੀ ਹੈ। JDK ਨੂੰ ਕੰਮ ਕਰਨ ਲਈ JRE ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ ਕਿਉਂਕਿ JAVA ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣਾ ਪ੍ਰੋਗਰਾਮ ਡਿਵੇਲਪਮੈਂਟ ਦਾ ਹੀ ਹਿੱਸਾ ਹੁੰਦਾ ਹੈ। ਇਹ ਕੰਪੋਨੈਂਟ ਅੰਤਮ ਉਪਭੋਗਤਾਵਾਂ (End-Users) ਲਈ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ।
- **JVM (ਜਾਵਾ ਵਰਚੁਅਲ ਮਸ਼ੀਨ):** JVM ਇੱਕ ਐਬਸਟਰੈਕਟ (abstract) ਮਸ਼ੀਨ ਹੁੰਦੀ ਹੈ। ਇਹ ਇੱਕ ਸਪੈਸੀਫੀਕੇਸ਼ਨ (specification) ਹੈ ਜੋ ਇਕ ਅਜਿਹਾ ਰਨਟਾਈਮ ਵਾਤਾਵਰਣ (runtime enviroment) ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਜਾਵਾ ਬਾਈਟਕੋਡ ਨੂੰ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।JVM ਬਹੁਤ ਸਾਰੇ ਵੱਖ-ਵੱਖ ਹਾਰਡਵੇਅਰ ਅਤੇ ਸਾਫਟਵੇਅਰ ਪਲੇਟਫਾਰਮਾਂ ਲਈ ਉਪਲਬਧ ਹਨ। ਅਸੀਂ ਇਹ ਵੀ ਕਹਿ ਸਕਦੇ ਹਾਂ JVM ਕੰਪੋਨੈਂਟ JAVA ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਇੱਕ ਰਨ-ਟਾਈਮ ਇੰਜਣ ਵਜੋਂ ਕੰਮ ਕਰਦਾ ਹੈ।JVM ਉਹ ਕੰਪੋਨੈਂਟ ਹੁੰਦਾ ਹੈ ਜੋ ਅਸਲ ਵਿੱਚ ਜਾਵਾ ਕੋਡ ਵਿੱਚ ਮੌਜੂਦ ਮੇਨ ਮੈਥਡ (main method) ਨੂੰ ਕਾਲ ਕਰਦਾ ਹੈ।JVM ਕੰਪੋਨੈਂਟ JRE (ਜਾਵਾ ਰਨਟਾਈਮ ਵਾਤਾਵਰਣ) ਦਾ ਹੀ ਇੱਕ ਹਿੱਸਾ ਹੁੰਦਾ ਹੈ। ਜਾਵਾ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ WORA (Write Once Run Anywhere) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮਰ ਇੱਕ ਸਿਸਟਮ ਤੇ ਜਾਵਾ ਕੋਡ ਡਿਵੇਲਪ ਕਰਕੇ ਉਸ ਵਿਚ ਬਿਨਾਂ ਕੋਈ ਬਦਲਾਵ ਕਿਤੇ ਕਿਸੇ ਹੋਰ ਜਾਵਾ-ਸਮਰਥਿਤ (JVM-enabled) ਸਿਸਟਮ ਤੇ ਚੱਲਾ ਸਕਦਾ ਹੈ। ਇਹ ਸਭ JVM ਦੇ ਕਾਰਨ ਹੀ ਸੰਭਵ ਹੋਇਆ ਹੈ।



ਚਿੱਤਰ : 4.4 JVM ਫੰਕਸ਼ਨ

4.4 ਜਾਵਾ ਦਾ ਇਤਿਹਾਸ (HISTORY OF JAVA) :

ਜਾਵਾ ਨੂੰ ਜੂਨ 1991 ਵਿੱਚ ਜੇਮਜ਼ ਗੋਸਲਿੰਗ ਅਤੇ ਉਸਦੀ ਟੀਮ (ਪੈਟਰਿਕ ਨੌਂਟਨ, ਕ੍ਰਿਸ ਵਾਰਥ, ਮਾਈਕ ਸ਼ੈਰੀਡਨ, ਅਤੇ ਐਡ ਫਰੈਂਕ) ਦੁਆਰਾ "ਓਕ (Oak)" ਨਾਮਕ ਇੱਕ ਪ੍ਰੋਜੈਕਟ ਵਜੋਂ ਸ਼ੁਰੂ ਕੀਤਾ ਗਿਆ ਸੀ। ਇਹ ਪ੍ਰੋਜੈਕਟ ਇੱਕ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਭਾਸ਼ਾ (platfrom-independent language) ਵਿਕਸਿਤ ਕਰਨ ਅਤੇ ਖਪਤਕਾਰ ਇਲੈਕਟ੍ਰਾਨਿਕ ਯੰਤਰਾਂ (Consumer Electronic Devices) ਲਈ ਏਮਬੈਡਡ (embedded) ਸਾਫਟਵੇਅਰ ਬਣਾਉਣ ਦੇ ਵਿਚਾਰ ਨਾਲ ਸ਼ੁਰੂ ਕੀਤਾ ਗਿਆ ਸੀ। ਗੋਸਲਿੰਗ ਦਾ ਟੀਚਾ ਇੱਕ ਵਰਚੁਅਲ ਮਸ਼ੀਨ ਅਤੇ ਇੱਕ ਅਜਿਹੀ ਭਾਸ਼ਾ ਨੂੰ ਲਾਗੂ ਕਰਨਾ ਸੀ ਜਿਸ ਵਿੱਚ ਜਾਣੀ-ਪਛਾਣੀ ਭਾਸ਼ਾ C ਵਰਗੀ ਨੋਟੇਸ਼ਨ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਵੇ ਪਰ ਇਹ C/C ++ ਨਾਲੋਂ ਵੱਧ ਇਕਸਾਰ ਅਤੇ ਸਰਲ ਹੋਵੇ। ਇਸ ਨੂੰ ਵਿਕਸਤ ਕਰਨ ਵਿੱਚ 18 ਮਹੀਨੇ ਲੱਗੇ ਅਤੇ ਇਸਦਾ ਸ਼ੁਰੂਆਤੀ ਨਾਮ ਓਕ (Oak) ਸੀ। ਪਰੰਤੂ ਕਾਪੀਰਾਈਟ ਮੁੱਦਿਆਂ ਕਾਰਨ 1995 ਵਿੱਚ ਓਕ (Oak) ਦਾ ਨਾਮ ਬਦਲ ਕੇ ਜਾਵਾ (Java) ਰੱਖਿਆ ਗਿਆ ਸੀ। ਜਾਵਾ ਦੀ ਪਹਿਲੀ ਜਨਤਕ ਸਥਾਪਨਾ (public implemention) JAVA 1.0 ਨਾਲ 1995 ਵਿੱਚ ਕੀਤੀ ਗਈ। ਜਾਵਾ ਕਾਫ਼ੀ ਸੁਰੱਖਿਅਤ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ।

ਜਾਵਾ ਇੰਡੋਨੇਸ਼ੀਆ ਵਿੱਚ ਇੱਕ ਟਾਪੂ ਦਾ ਨਾਮ ਹੈ ਜਿੱਥੇ ਪਹਿਲੀ ਕੌਫੀ (ਜਿਸਦਾ ਨਾਮ ਜਾਵਾ ਕੌਫੀ ਸੀ) ਪੈਦਾ ਕੀਤੀ ਗਈ ਸੀ; ਜੇਮਸ ਗੋਸਲਿੰਗ ਨੇ ਨਵੀਂ ਭਾਸ਼ਾ ਦਾ ਜਾਵਾ ਨਾਮ ਆਪਣੇ ਦਫਤਰ ਦੇ ਨੇੜੇ ਕੌਫੀ ਪੀਂਦੇ ਹੋਏ ਚੁਣਿਆ ਸੀ। ਜੇਮਸ ਗੋਸਲਿੰਗ ਨੂੰ ਜਾਵਾ ਦਾ ਪਿਤਾਮਾ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ॥

4.5 ਜਾਵਾ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (FEATURES OF JAVA) :

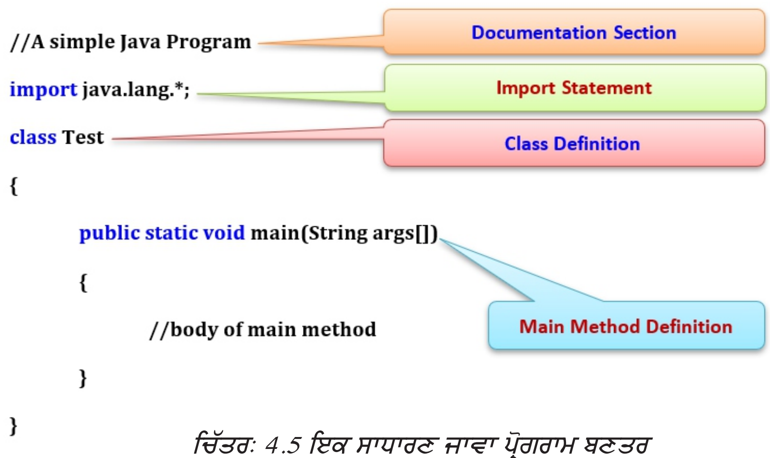
ਜਾਵਾ ਦੁਨੀਆ ਦੀ ਸਭ ਤੋਂ ਪ੍ਰਸਿੱਧ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚੋਂ ਇੱਕ ਹੈ। ਇਹ ਸਿੱਖਣ ਲਈ ਆਸਾਨ ਅਤੇ ਵਰਤਣ ਲਈ ਇਕ ਸਧਾਰਨ ਭਾਸ਼ਾ ਹੈ। ਇਸ ਵਿੱਚ ਬਹੁਤ ਸਾਰੀਆਂ ਅਜਿਹੀਆਂ ਸ਼ਕਤੀਸ਼ਾਲੀ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹਨ ਜੋ ਇਸਨੂੰ ਇੱਕ ਸ਼ਕਤੀਸ਼ਾਲੀ ਅਤੇ ਇੱਕ ਬਹੁਤ ਮਸ਼ਹੂਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਬਣਾਉਂਦੀਆਂ ਹਨ। ਹੇਠਾਂ ਜਾਵਾਂ ਦੀਆਂ ਕੁੱਝ ਮਹੱਤਵਪੂਰਨ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦਾ ਵਰਨਣ ਕੀਤਾ ਗਿਆ ਹੈ:

- **ਕੰਪਾਈਲਡ ਅਤੇ ਇੰਟਰਪ੍ਰੈਟਿਡ (Compiled and Interpretered) :** ਜ਼ਿਆਦਾਤਰ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਇਸ ਤਰ੍ਹਾਂ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਹੈ ਕਿ ਜਾਂ ਤਾਂ ਉਹ ਕੰਪਾਇਲਡ ਭਾਸ਼ਾਵਾਂ ਹਨ ਜਾਂ ਉਹ ਇੰਟਰਪ੍ਰੈਟਿਡ ਭਾਸ਼ਾਵਾਂ ਹਨ। ਜਾਵਾ ਕੋਲ ਕੰਪਾਈਲਡ ਅਤੇ ਇੰਟਰਪੇਟਿਡ ਦੋਵੇਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹਨ। ਜਾਵਾ ਕੰਪਾਈਲਰ ਸੋਰਸ ਕੋਡ ਨੂੰ ਬਾਈਟਕੋਡ ਵਿਚ ਕੰਪਾਇਲ ਕਰਦਾ ਹੈ ਅਤੇ JVM ਇਸ ਬਾਈਟਕੋਡ ਨੂੰ ਮਸ਼ੀਨ OS ਉਪਰ ਨਿਰਭਰ ਐਗਜ਼ੀਕਿਊਟੇਬਲ ਕੋਡ ਵਿਚ ਤਬਦੀਲ ਕਰਕੇ ਕਦਮ ਦਰ ਕਦਮ ਚਲਾਉਂਦਾ ਹੈ।
- **ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ (Platform Independent) :** ਜਾਵਾ ਭਾਸ਼ਾ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜਾਵਾ ਦਾ ਪ੍ਰੋਗਰਾਮ ਇੱਕ ਕਿਸਮ ਦੀ ਮਸ਼ੀਨ ਤੋਂ ਦੂਜੀ ਕਿਸਮ ਦੀ ਮਸ਼ੀਨ ਉਪਰ ਆਸਾਨੀ ਨਾਲ ਪ੍ਰਯੋਗ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਸਦਾ ਕਾਰਨ ਇਹ ਹੈ ਕਿ ਕੰਪਾਈਲਰ ਦੁਆਰਾ ਸੋਰਸ ਕੋਡ ਨੂੰ ਬਾਈਟਕੋਡ ਵਿੱਚ ਬਦਲਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਫਿਰ JVM ਦੁਆਰਾ ਇਸ ਬਾਈਟਕੋਡ ਨੂੰ ਮਸ਼ੀਨ ਦੇ ਸਮਝਣ ਯੋਗ ਕੋਡ ਵਿਚ ਤਬਦੀਲ ਕਰਕੇ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਬਾਈਟਕੋਡ ਕਿਸੇ ਵੀ ਪਲੇਟਫਾਰਮ 'ਤੇ ਚੱਲ ਸਕਦਾ ਹੈ ਭਾਵੇਂ ਉਹ ਵਿੰਡੋਜ਼, ਲੀਨਕਸ, ਜਾਂ ਮੈਕ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਹੋਵੇ, ਜਿਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜੇਕਰ ਅਸੀਂ ਵਿੰਡੋਜ਼ ਪਲੇਟਫਾਰਮ ਉਪਰ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਇਲ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਇਸਨੂੰ ਲੀਨਕਸ 'ਤੇ ਵੀ ਚਲਾ ਸਕਦੇ ਹਾਂ।
- **ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ (Object-Oriented) :** ਜਾਵਾ ਸ਼ੁੱਧ ਰੂਪ ਵਿਚ ਇਕ OOP ਭਾਸ਼ਾ ਹੈ। ਜਾਵਾ ਭਾਸ਼ਾ ਦਾ ਸਾਰਾ ਕੋਡ ਕਲਾਸਾਂ ਅਤੇ ਆਬਜੈਕਟਾਂ ਵਿੱਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ OOP ਦੇ ਸਾਰੇ ਚਾਰ ਮੁੱਖ ਸੰਕਲਪਾਂ ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction), ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation), ਇਨਹੇਰੀਟੈਂਸ (Inheritance), ਅਤੇ ਪੋਲੀਮੋਫਿਜ਼ਮ (Polymorphism) ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ।
- **ਮਜਬੂਤ (Robust) :** ਜਾਵਾ ਇਕ ਮਜਬੂਤ ਭਾਸ਼ਾ ਹੈ ਜਿਸਤੋਂ ਭਾਵ ਹੈ ਇਕ ਭਰੋਸੇਯੋਗ ਭਾਸ਼ਾ। ਇਸ ਭਾਸ਼ਾ ਨੂੰ ਇਸ ਤਰੀਕੇ ਨਾਲ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਹੈ ਕਿ ਇਹ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਜਲਦੀ ਤੋਂ ਜਲਦੀ ਗਲਤੀਆਂ ਦੀ ਜਾਂਚ ਕਰ ਸਕੇ। ਜਾਵਾ ਕੰਪਾਈਲਰ ਉਹਨਾਂ ਗਲਤੀਆਂ ਨੂੰ ਵੀ ਲੱਭਣ ਦੇ ਯੋਗ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿਸੇ ਹੋਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਦੁਆਰਾ ਲੱਭਣਾ ਆਸਾਨ ਨਹੀਂ ਹੁੰਦਾ। ਗਾਰਬੇਜ਼ ਕੁਲੈਕਸ਼ਨ (garbage collection), ਐਕਸੇਪਸ਼ਨ ਹੈਂਡਲਿੰਗ (Exception Handling), ਅਤੇ ਮੈਮੋਰੀ ਐਲੋਕੇਸ਼ਨ (memory allocation) ਆਦਿ ਜਾਵਾ ਦੀਆਂ ਅਜਿਹੀਆਂ ਮੁੱਖ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹਨ ਜੋ ਇਸਨੂੰ ਮਜ਼ਬੂਤ ਬਣਾਉਂਦੀਆਂ ਹਨ।

- **ਸੁਰੱਖਿਅਤ (Secure) :** ਜਾਵਾ ਕਈ ਕਾਰਨਾਂ ਕਰਕੇ ਸੁਰੱਖਿਅਤ ਹੈ॥ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਇੱਕ ਵਰਚੁਅਲ ਮਸ਼ੀਨ ਦੇ ਅੰਦਰ ਚੱਲਦੇ ਹਨ ਜਿਸਨੂੰ ਸੈਂਡਬਾਕਸ (Sandbox) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। JAVA, C ਭਾਸ਼ਾ ਵਿਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਪੁਆਇੰਟਰਜ਼ ਨੂੰ ਸਪੋਰਟ ਨਹੀਂ ਕਰਦਾ। ਬਾਈਟਕੋਡ ਵੈਰੀਫਾਇਰ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਉਹਨਾਂ ਗੈਰ ਕਾਨੂੰਨੀ ਕੋਡਜ਼ (illegal codes) ਦੀ ਜਾਂਚ ਕਰਦਾ ਹੈ। ਜੋ ਆਬਜੈਕਟ ਦੇ ਅਧਿਕਾਰ ਦੀ ਉਲੰਘਣਾ (violate access right) ਕਰ ਸਕਦੇ ਹਨ। JAVA ਵਿਚ ਕਈ ਅਜਿਹੀਆਂ ਕਲਾਸ ਲਾਇਬ੍ਰੇਰੀਆਂ ਮੌਜੂਦ ਹਨ ਜੋ ਸੁਰੱਖਿਅਤ API ਪ੍ਰਦਾਨ ਕਰਦੀਆਂ ਹਨ। ਇਹਨਾਂ API ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਰੇਕ ਅੰਦਰੂਨੀ ਸੰਚਾਰ ਦੀ ਪ੍ਰਮਾਣਿਕਤਾ ਦੀ ਜਾਂਚ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।
- **ਡਿਸਟੀਬਿਊਟਿਡ (Distribulted) :** ਅਸੀਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਡਿਸਟ੍ਰੀਬਿਊਟਿਡ ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾ ਸਕਦੇ ਹਾਂ। ।ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਅਜਿਹੇ ਇੱਕ ਜਾਂ ਵਧੇਰੇ ਸਿਸਟਮਾਂ 'ਤੇ ਆਸਾਨੀ ਨਾਲ ਵੰਡਿਆ (distribute) ਜਾ ਸਕਦਾ ਹੈ। ਜੋ ਇੰਟਰਨੈਟ ਕਨੈਕਸ਼ਨ ਦੁਆਰਾ ਇੱਕ ਦੂਜੇ ਨਾਲ ਜੁੜੇ ਹੁੰਦੇ ਹਨ। ਇਸ ਤਰ੍ਹਾਂ ਜਾਵਾ ਨੂੰ ਵਿਸ਼ੇਸ਼ ਤੌਰ 'ਤੇ ਉਹਨਾਂ ਇੰਟਰਨੈਟ-ਯੂਜ਼ਰਜ਼ ਲਈ ਤਿਆਰ ਕੀਤਾ ਗਿਆ ਸੀ।ਜੋ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਰਿਮੋਟ ਕੰਪਿਊਟਰਾਂ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਨ। ।
- **ਸਧਾਰਨ, ਛੋਟੀ ਅਤੇ ਜਾਣੂ (Simple, Small and Familiar) :** ਜਾਵਾ ਇੱਕ ਸਧਾਰਨ ਭਾਸ਼ਾ ਹੈ ਕਿਉਂਕਿ ਇਸ ਵਿੱਚ C ਅਤੇ C++ ਵਰਗੀਆਂ ਹੋਰ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਬਹੁਤ ਸਾਰੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਸ਼ਾਮਲ ਹਨ। ਜਾਵਾ ਵਿਚ ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਦੀਆਂ ਗੁੰਝਲਦਾਰ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਜਿਵੇਂ ਕਿ: ਪੁਆਇੰਟਰਜ਼ (Pointers), ਸਟੋਰੇਜ਼ ਕਲਾਸਾਂ (Storge Classes) ਅਤੇ goto ਸਟੇਟਮੈਂਟ ਆਦਿ ਨੂੰ ਹਟਾਇਆ ਗਿਆ ਹੈ।ਇਹ ਮਲਟੀਪਲ ਇਨਹੇਰੀਟੈਂਸ (Inheritance), ਓਪਰੇਟਰ ਓਵਰਲੋਡਿੰਗ (Overloding), ਸਪੱਸ਼ਟ ਮੈਮੋਰੀ ਐਲੋਕੇਸ਼ਨ (Explicit Memory Allocation) ਆਦਿ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਨੂੰ ਸਪੋਰਟ ਨਹੀਂ ਕਰਦਾ ਜੋ C++ ਦੁਆਰਾ ਸਪੋਰਟ ਕੀਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।
- **ਮਲਟੀਥਰੈਡਡ ਅਤੇ ਇੰਟਰਐਕਟਿਵ (Multithreaded and Interactive) :** ਜਾਵਾ ਮਲਟੀਥ੍ਰੈਡਿੰਗ ਨੂੰ ਸਪੋਰਟ ਕਰਦਾ ਹੈ॥ ਇਹ ਇੱਕ ਅਜਿਹੀ ਜਾਵਾ ਵਿਸ਼ੇਸ਼ਤਾ ਹੈ ਜੋ CPU ਦੀ ਵੱਧ ਤੋਂ ਵੱਧ ਉਪਯੋਗਤਾ (Maximum Utilization) ਲਈ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੇ ਦੋ ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਭਾਗਾਂ ਨੂੰ ਇੱਕੋ ਸਮੇਂ ਚਲਾਉਣ ਦੀ ਆਗਿਆ ਦਿੰਦੀ ਹੈ।
- **ਪੋਰਟੇਬਲ (Portable) :** ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਜਾਣਦੇ ਹਾਂ ਕਿ ਇਕ ਮਸ਼ੀਨ ਉਪਰ ਲਿਖਿਆ ਜਾਵਾ ਕੋਡ ਦੂਜੀ ਮਸ਼ੀਨ 'ਤੇ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਜਿਸਨੂੰ Write Once Run Anywhere-WORA ਵਿਸ਼ੇਸ਼ਤਾ ਨਾਲ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ॥ ਜਾਵਾ ਦੀ ਪਲੇਟਫਾਰਮ-ਸੁਤੰਤਰ (Patfrom-Independent) ਵਿਸ਼ੇਸ਼ਤਾ ਜਿਸ ਵਿੱਚ ਇਸਦੇ ਬਾਈਟਕੋਡ (Bytecode) ਨੂੰ ਐਗਜ਼ੀਕਿਊਸ਼ਨ (exection) ਲਈ ਕਿਸੇ ਵੀ ਪਲੇਟਫਾਰਮ 'ਤੇ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ, ਜਾਵਾ ਨੂੰ ਪੋਰਟੇਬਲ ਬਣਾਉਂਦੀ ਹੈ॥
- **ਡਾਇਨਾਮਿਕ ਅਤੇ ਐਕਸਟੈਂਸੀਬਲ ਕੋਡ (Dynamic and Extensible Code) :** ਜਾਵਾ ਪੂਰੀ ਤਰ੍ਹਾਂ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਿਡ ਹੋਣ ਕਾਰਨ ਸਾਨੂੰ ਨਵੀਆਂ ਕਲਾਸਾਂ, ਮੌਜੂਦਾ ਕਲਾਸਾਂ ਵਿੱਚ ਨਵੇਂ ਮੈਥਡਜ਼ (Methods) ਅਤੇ ਸਬ-ਕਲਾਸਾਂ (sub-classes) ਦੀ ਵਰਤੋਂ ਨਾਲ ਨਵੀਆਂ ਕਲਾਸਾਂ ਬਣਾਉਣ ਦੀ ਲਚਕਤਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। JAVA ਹੋਰ ਭਾਸ਼ਾਵਾਂ ਜਿਵੇਂ ਕਿ C, C++ ਵਿੱਚ ਲਿਖੇ ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਵੀ ਸਪੋਰਟ ਕਰਦਾ ਹੈ ਜਿਨ੍ਹਾਂ ਨੂੰ ਨੇਟਿਵ ਮੈਥਡਜ਼ (Native Methods) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

4.6 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਮੁੱਢਲੀ ਬਣਤਰ (BASIC STRUCTURE OFJAVA PROGRAM)

ਹਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਵਿੱਚ ਕੁੱਝ ਪਹਿਲਾਂ ਤੋਂ ਪ੍ਰਭਾਸ਼ਿਤ ਕੋਡ ਬਣਤਰ ਅਤੇ ਸਿੰਟੈਕਸ ਵਜੋਂ ਜਾਣੇ ਜਾਂਦੇ ਨਿਯਮਾਂ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦਾ ਹੈ। ਜੇਕਰ ਕੋਡ ਲਿਖਣ ਵੇਲੇ ਇਹਨਾਂ ਨਿਯਮਾਂ ਦੀ ਉਲੰਘਣਾ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਤਾਂ ਇਹ ਇੱਕ ਐਰਰ (Error) ਦਰਸ਼ਾਵੇਗਾ। ਹਰੇਕ ਭਾਸ਼ਾ ਦੇ ਸਿੰਟੈਕਸ ਦੀ ਪਾਲਣਾ ਕਰਨਾ ਬਹੁਤ ਮਹੱਤਵਪੂਰਨ ਹੁੰਦਾ ਹੈ। ਇਸ ਤੋਂ ਪਹਿਲਾਂ ਕਿ ਅਸੀਂ ਇਹ ਸਿੱਖਣਾ ਸ਼ੁਰੂ ਕਰੀਏ ਕਿ ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਕਿਵੇਂ ਲਿਖਣਾ ਹੈ, ਆਓ ਪਹਿਲਾਂ ਇਹ ਸਮਝੀਏ ਕਿ ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦਾ ਬੁਨਿਆਦੀ ਢਾਂਚਾ ਕੀ ਹੁੰਦਾ ਹੈ।



ਚਿੱਤਰ: 4.5 ਇਕ ਸਾਧਾਰਣ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਤਰ

ਇਹ ਚਿੱਤਰ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਬਹੁਤ ਹੀ ਬੁਨਿਆਦੀ ਬਣਤਰ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ।ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਹੋਰ ਤੱਤ (elements) ਵੀ ਹੋ ਸਕਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ ਪੈਕੇਜ ਸਟੇਟਮੈਂਟ (package statement), ਇੰਟਰਫੇਸ ਸੈਕਸ਼ਨ (Inerface section) ਆਦਿ॥ ਅਸੀਂ ਇਹਨਾਂ ਭਾਗਾਂ ਬਾਰੇ ਚਰਚਾ ਨਹੀਂ ਕਰਾਂਗੇ ਕਿਉਂਕਿ ਇਹ ਟੋਪਿਕਸ ਇਸ ਕਿਤਾਬ ਦੇ ਦਾਇਰੇ (scope) ਤੋਂ ਬਾਹਰ ਹਨ॥ ਆਓ ਹੁਣ ਅਸੀਂ ਉਪਰੋਕਤ ਪ੍ਰੋਗਰਾਮ-ਸਟ੍ਰਕਚਰ ਵਿੱਚ ਦਰਸਾਏ ਪ੍ਰੋਗਰਾਮ ਤੱਤਾਂ ਦੇ ਵਰਣਨ 'ਤੇ ਇੱਕ ਨਜ਼ਰ ਮਾਰੀਏ:

❖ **ਡਾਕੂਮੈਂਟੇਸ਼ਨ ਸੈਕਸ਼ਨ (Documentation Section) :** ਇਹ ਪ੍ਰੋਗਰਾਮ ਦੀ ਪੜ੍ਹਨਯੋਗਤਾ (readability) ਨੂੰ ਬਿਹਤਰ ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।। ਇਸ ਵਿੱਚ JAVA ਦੇ ਕਮੈਂਟਸ ਸ਼ਾਮਿਲ ਹੁੰਦੇ ਹਨ। ਜੋ ਕੋਡ ਦੀ ਸਮੀਖਿਆ (reviewing) ਜਾਂ ਡੀਬੱਗਿੰਗ (debugging) ਕਰਨ ਵੇਲੇ ਪ੍ਰੋਗਰਾਮਰ ਦੇ ਕੰਮ ਨੂੰ ਆਸਾਨ ਬਣਾਉਂਦੇ ਹਨ॥ ਇਹ ਸਟੇਟਮੈਂਟਸ ਆਪਸ਼ਨਲ ਹੁੰਦੀਆਂ ਹਨ। :

❖ **ਇੰਪੋਰਟ ਸਟੇਟਮੈਂਟ (Impornt Statement) :** ਜਾਵਾ ਵਿੱਚ ਬਹੁਤ ਸਾਰੀਆਂ ਪੂਰਵ-ਪ੍ਰਭਾਸ਼ਿਤ (predefined) ਕਲਾਸਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਪੈਕੇਜਾਂ (packages) ਦੇ ਰੂਪ ਵਿਚ ਸੰਗਠਿਤ (organized) ਹੁੰਦੀਆਂ ਹਨ। ਜੇਕਰ ਅਸੀਂ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਦੂਜੇ ਪੈਕੇਜਾਂ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕਲਾਸਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ, ਤਾਂ ਸਾਨੂੰ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ import ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਕਰਨੀ ਪਵੇਗੀ।

❖ **ਕਲਾਸ ਪਰਿਭਾਸ਼ਾ (Class Defination) :** ਕਲਾਸਾਂ ਕਿਸੇ ਵੀ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦਾ ਜ਼ਰੂਰੀ ਹਿੱਸਾ ਹੁੰਦੀਆਂ ਹਨ। ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕਈ ਕਲਾਸ ਪਰਿਭਾਸ਼ਾਵਾਂ ਹੋ ਸਕਦੀਆਂ ਹਨ। JAVA ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਣ ਲਈ ਮੇਨ main ਮੈਥਡ ਵਾਲੀ ਇੱਕ ਕਲਾਸ ਬਨਾਉਣੀ ਜਰੂਰੀ ਹੁੰਦੀ ਹੈ॥ਜਿਸ ਕਲਾਸ ਵਿੱਚ ਮੇਨ ਮੈਥਡ ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ ਉਸਨੂੰ ਪਬਲਿਕ (public) ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ।

  ❖ **class Test -** ਜਾਵਾ ਵਿਚ ਇਹ ਸਟੇਟਮੈਂਟ ਇੱਕ ਕਲਾਸ ਬਣਾਵੇਗੀ ਜਿਸਦਾ ਨਾਮ Test ਹੋਵੇਗਾ॥ ਜਾਵਾ ਦੇ ਸਟੈਂਡਰਡ ਕਨਵੈਨਸ਼ਨ ਅਨੁਸਾਰ ਕਲਾਸ ਦੇ ਨਾਮ ਲਈ ਟਾਈਟਲਕੇਸ (Title Case) ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ॥ ਇਸ ਲਈ ਜਾਵਾ ਵਿਚ ਕਲਾਸ ਦਾ ਨਾਮ ਤੈਅ ਕਰਦੇ ਹੋਏ ਇਹ ਯਕੀਨੀ ਬਣਾਓ ਕਿ ਕਲਾਸ ਦਾ ਨਾਮ ਟਾਈਟਲਕੇਸ (ਨਾਮ ਵਿੱਚ ਖਾਲੀ ਥਾਂ ਨਹੀਂ ਹੋਣੀ ਚਾਹੀਦੀ) ਵਿੱਚ ਹੋਵੇ।

  ❖ **ਬਰੇਸਿਸ (Braces{}) (ਓਪਨਿੰਗ ਅਤੇ ਕਲੋਜ਼ਿੰਗ ਦੋਵੇਂ ਤਰ੍ਹਾਂ ਦੇ) -** ਕਰਲੀ ਬਰੇਸਿਸ ({ }) ਦੀ ਵਰਤੋਂ ਕਈ ਸਟੇਟਮੈਂਟਸ ਨੂੰ ਇੱਕ ਗਰੁੱਪ ਵਿਚ ਇੱਕਠਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ॥ਕਰਲੀ ਬਰੇਸਿਸ ਦੀ ਵਰਤੋਂ ਇਹ ਦਰਸ਼ਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਦਿੱਤੀਆਂ ਗਈਆਂ ਸਟੇਟਮੈਂਟਸ ਕਲਾਸ ਜਾਂ ਮੈਥਡ ਨਾਲ ਸਬੰਧਤ ਹਨ॥

  ❖ **main() ਮੈਥਡ ਪਰਿਭਾਸ਼ਾ (main () Method Defintion) :** ਜਾਵਾ ਵਿੱਚ ਮੇਨ ਮੈਥਡ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਂਟਰੀ ਪੁਆਇੰਟ ਵਜੋਂ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ॥ ਜਦੋਂ ਯੂਜ਼ਰ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਂਦਾ ਹੈ ਤਾਂ ਮੇਨ ਮੈਥਡ ਨੂੰ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਦੁਆਰਾ ਕਾਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ: ਉਦਾਹਰਣ ਲਈ:

```
public static void main(String args[])
```

  ❖ **public –** ਜਦੋਂ ਮੇਨ (main( )) ਮੈਥਡ ਨੂੰ ਪਬਲਿਕ (public) ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਸਦਾ ਮਤਲਬ ਹੁੰਦਾ ਹੈ ਕਿ ਇਸਨੂੰ ਘੋਸ਼ਿਤ ਕਲਾਸ (declared class) ਤੋਂ ਬਾਹਰ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

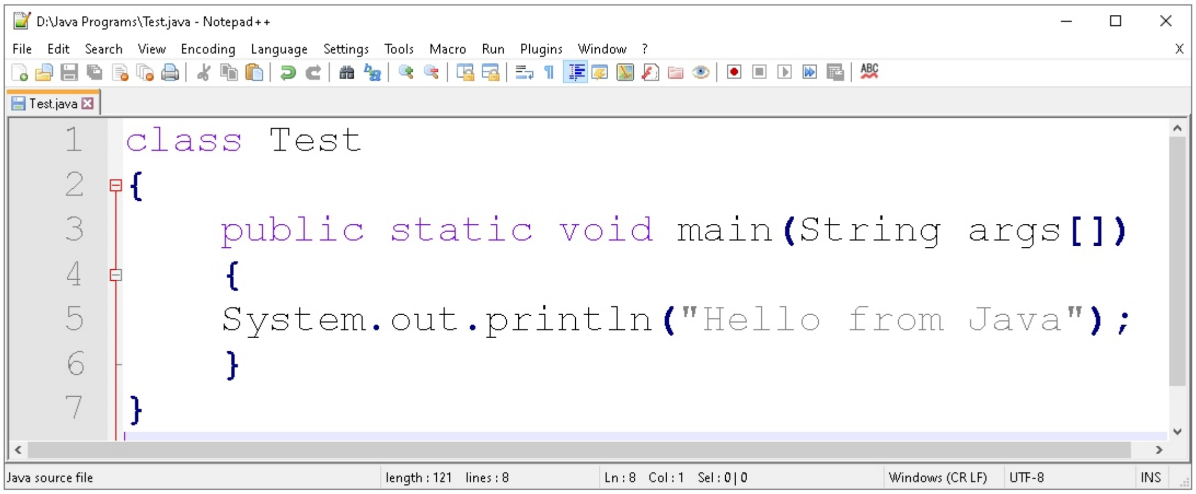
- ❖ **static** – ਸਟੈਟਿਕ ਸ਼ਬਦ ਦਾ ਮਤਲਬ ਹੈ ਕਿ ਅਸੀਂ ਉਸ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਏ ਬਿਨਾਂ ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ ਜਿਸ ਵਿੱਚ ਮੈਥਡ ਘੋਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੈ॥ਜਾਵਾ ਵਿਚ ਅਸੀਂ ਮੇਨ ਮੈਥਡ ਵਾਲੀ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਏ ਬਿਨਾਂ ਮੇਨ ਮੈਥਡ ਨੂੰ ਕਾਲ ਕਰਦੇ ਹਾਂ ਜਿਸ ਕਾਰਨ ਮੇਨ ਮੈਥਡ ਨਾਲ Static ਕੀਅਵਰਡ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
- ❖ **void** – ਕਿਸੇ ਵੀ ਮੈਥਡ ਨਾਲ void ਸ਼ਬਦ ਇਹ ਦਰਸਾਉਂਦਾ ਹੈ ਕਿ ਉਹ ਮੈਥਡ ਕੋਈ ਵੀ ਮੁੱਲ ਯੂਜ਼ਰ ਨੂੰ ਵਾਪਿਸ ਨਹੀਂ ਕਰੇਗਾ। ਮੇਨ ਮੈਥਡ ਨੂੰ ਇਸੇ ਕਰਕੇ void ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਉਹ ਕੋਈ ਵੀ ਮੁੱਲ ਯੂਜ਼ਰ ਨੂੰ ਵਾਪਸ ਨਹੀਂ (does not return any value) ਕਰਦਾ।
- ❖ **main** - ਇਹ ਇਕ ਮੈਥਡ ਹੈ, ਜੋ ਕਿ ਕਿਸੇ ਵੀ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦਾ ਜ਼ਰੂਰੀ ਹਿੱਸਾ ਹੁੰਦਾ ਹੈ।
- ❖ **String args[ ]** - ਇਹ ਇੱਕ ਐਰੇ (array) ਹੈ ਜਿੱਥੇ ਹਰੇਕ ਤੱਤ (element) ਇੱਕ ਸਟਿੰਗ ਹੁੰਦਾ ਹੈ, ਜਿਸਨੂੰ args ਨਾਮ ਦਿਤਾ ਗਿਆ ਹੈ। ।ਜੇਕਰ ਅਸੀਂ ਜਾਵਾ ਕੋਡ ਨੂੰ ਕੰਸੋਲ (console) ਰਾਹੀਂ ਚਲਾਉਂਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਇਸ ਐਰੇ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇਨਪੁਟ ਪੈਰਾਮੀਟਰਜ਼ ਪਾਸ ਕਰ ਸਕਦੇ ਹਾਂ।main() ਮੈਥਡ ਇਸਨੂੰ ਇੱਕ ਇਨਪੁੱਟ ਵਜੋਂ ਲੈਂਦਾ ਹੈ।

4.7 ਇੱਕ ਸਧਾਰਨ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਣਾ ਅਤੇ ਲਾਗੂ ਕਰਨਾ (CREATING AND EXECUTING A SIMPLE JAVA PROGRAM)

ਆਪਣਾ ਪਹਿਲਾ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਣ ਲਈ ਤੁਹਾਨੂੰ ਹੇਠਾਂ ਦਿੱਤੇ ਦੋ ਸਾਫਟਵੇਅਰਾਂ ਦੀ ਜਰੂਰਤ ਪਵੇਗੀ:

1. **ਜਾਵਾ ਡਿਵੇਲਪਮੈਂਟ ਕਿੱਟ (Java Development Kit (JDK)) :** ਇਹ JAVA ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਲਈ ਇੱਕ ਪ੍ਰਮੁੱਖ ਪਲੇਟਫਾਰਮ ਕੰਪੋਨੈਂਟ ਹੈ। ।ਇਸਦਾ ਮੁੱਖ ਭਾਗ ਜਾਵਾ ਕੰਪਾਈਲਰ ਹੁੰਦਾ ਹੈ। ਇਸ JRE + ਡਿਵੈਲਪਮੈਂਟ ਟੂਲਜ਼ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ॥
2. **ਟੈਕਸਟ ਐਡੀਟਰ (Text Editor) :** ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਦਾ ਸੋਰਸ ਕੋਡ ਲਿਖਣ ਲਈ ਕਿਸੇ ਵੀ ਟੈਕਸਟ ਐਡੀਟਰ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ, ਜਿਵੇਂ ਕਿ Notepad, Notepad++ ਆਦਿ। ਨਿਮਨਲਿਖਤ ਪ੍ਰੋਗਰਾਮ ਦੀ ਉਦਾਹਰਨ ਵਿੱਚ ਅਸੀਂ Notepad ++ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹਾਂ। ।ਅਸੀਂ ਕਿਸੇ ਹੋਰ ਵੱਖਰੇ ਟੈਕਸਟ ਐਡੀਟਰ ਦੀ ਵਰਤੋਂ ਵੀ ਕਰ ਸਕਦੇ ਹਾਂ। Notepad ਵਿੰਡੋਜ਼ ਓਪਰੇਟਿੰਗ ਸਿਸਟਮ ਵਿੱਚ ਪਹਿਲਾਂ ਤੋਂ ਮੌਜੂਦ ਸਧਾਰਨ ਟੈਕਸਟ-ਐਡੀਟਰ ਹੁੰਦਾ ਹੈ, ਜਾਂ ਆਨਲਾਈਨ ਜਾਵਾ ਕੰਪਾਈਲਰ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਾਉਣ ਅਤੇ ਚਲਾਉਣ ਲਈ ਹੇਠਾਂ ਦਿੱਤੇ ਸਟੈਪ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ:

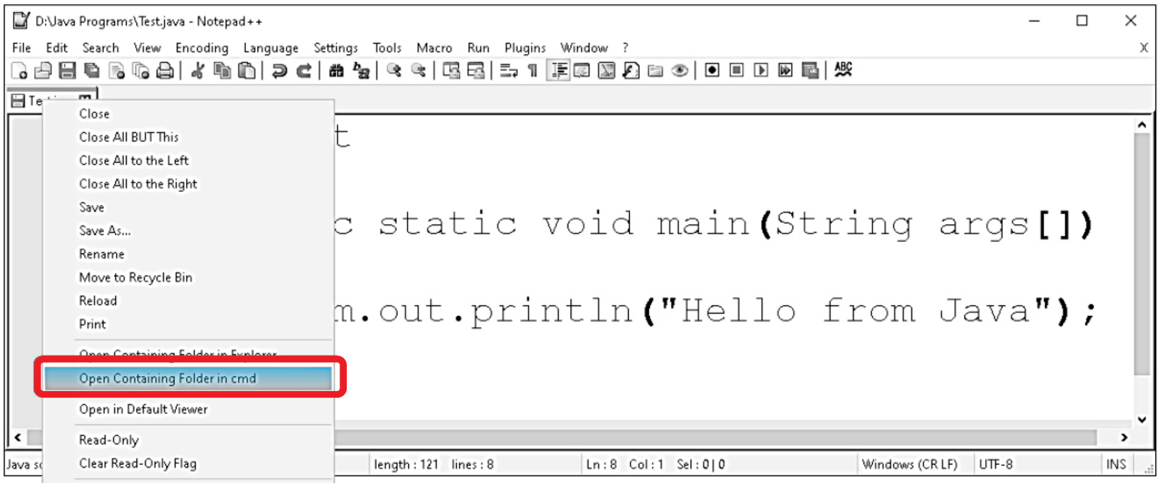
ਸਟੈਪ 1: Notepad++ ਓਪਨ ਕਰੋ (Start button, Notepad++) ਅਤੇ ਉਸ ਵਿਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਟਾਈਪ ਕਰੋ, ਫਿਰ ਉਸਨੂੰ **Java** ਐਕਸਟੈਂਸ਼ਨ (extension) ਨਾਲ ਸੇਵ (Ctrl+S) ਕਰੋ, ਉਦਾਹਰਣ ਲਈ:



ਚਿੱਤਰ 4.6 JAVA ਪ੍ਰੋਗਰਾਮ ਦੀ Notepad++ ਵਿੱਚ ਉਦਾਹਰਣ

ਸਟੈਪ 2: ਮੌਜੂਦਾ ਫਾਈਲ ਦੀ ਲੋਕੇਸ਼ਨ ਨੂੰ ਕਮਾਂਡ ਪ੍ਰਾਮਪਟ ਉਪਰ ਖੋਲਣ ਲਈ Notepad++ ਦੇ ਮੌਜੂਦਾ ਫਾਈਲ ਟੈਬ ਉੱਪਰ Right

ਕਲਿੱਕ ਕਰੋ ਅਤੇ ਫਿਰ ਓਪਨ ਹੋਏ ਪ੍ਰਾਪਰਟੀਜ਼ ਮੀਨੂੰ ਵਿਚੋਂ **"Open Containing Folder in CMD** ਆਪਸ਼ਨ ਉਪਰ ਕਲਿੱਕ ਕਰੋ, ਜਿਵੇਂ ਕਿ ਅੱਗੇ ਚਿੱਤਰ ਵਿਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:



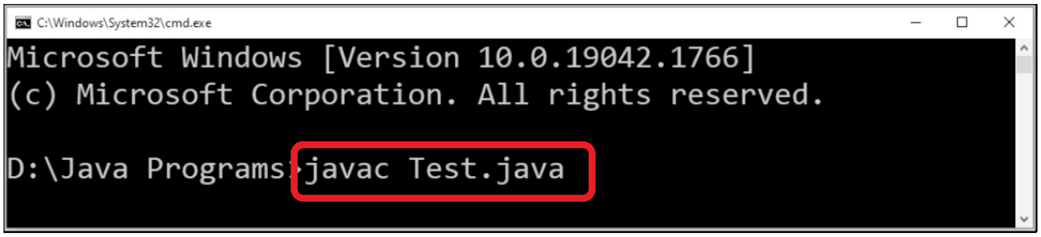
ਚਿੱਤਰ: 4.7 Notepad+ + ਵਿਚ ਮੌਜੂਦਾ ਫਾਈਲ ਲਈ ਕਮਾਂਡ ਪ੍ਰੋਮਪਟ ਖੋਲਣਾ

ਚਿੱਤਰ: 4.8 ਮੌਜੂਦਾ ਫਾਈਲ ਦੀ ਲੋਕੇਸ਼ਨ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਕਮਾਂਡ ਪ੍ਰਾਮਪਟ

ਸਟੈੱਪ 3: ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਈਲ ਕਰਨ ਲਈ ਹੇਠਾਂ ਦਿਤੇ ਕਮਾਂਡ ਸਿੰਟੈਕਸ ਦੀ ਵਰਤੋਂ ਕਰੋ:

javac filename.java

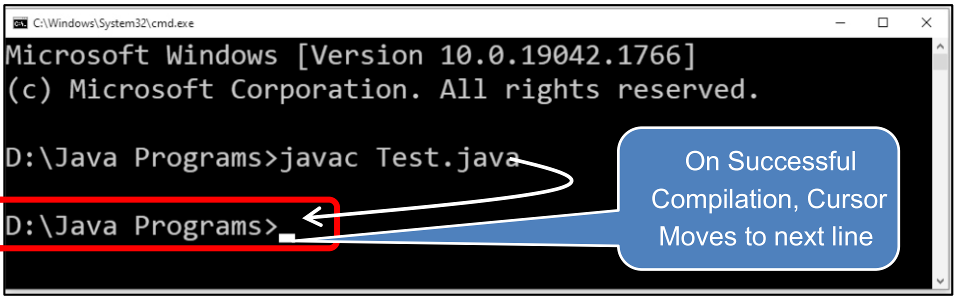
ਹੇਠਾਂ ਦਰਸ਼ਾਏ ਅਨੁਸਾਰ Test.java ਫਾਈਲ ਨੂੰ ਕੰਪਾਈਲ ਕਰੋ:

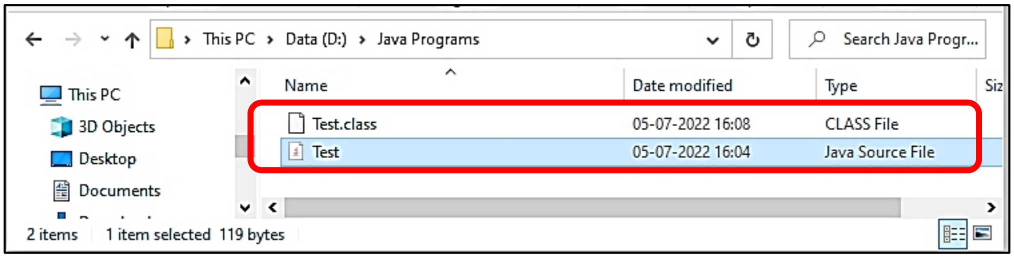


ਚਿੱਤਰ 4.9: ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ Test.java ਨੂੰ ਕੰਪਾਈਲ ਕਰਨਾ

ਕੰਸੋਲ (Console) ਸਕੀਨ ਉਪਰ JAVA Test.java ਟਾਈਪ ਕਰਨ ਤੋਂ ਬਾਅਦ ਐਂਟਰ ਕੀਅ ਦਬਾਓ॥ ਇਹ ਕਮਾਂਡ Test.java ਫਾਈਲ ਵਿੱਚ ਸਟੋਰ ਸਾਡੇ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੇ ਸੋਰਸ ਕੋਡ ਨੂੰ ਕੰਪਾਇਲ ਕਰੇਗਾ। ਜੇਕਰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੋਈ ਗਲਤੀ ਨਹੀਂ ਹੁੰਦੀ, ਤਾਂ ਇਹ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ ਦਰਸਾਏ ਅਨੁਸਾਰ ਕਮਾਂਡ ਪ੍ਰੋਮਪਟ ਨੂੰ ਅਗਲੀ ਲਾਈਨ ਵਿੱਚ ਦਿਖਾਏਗਾ।

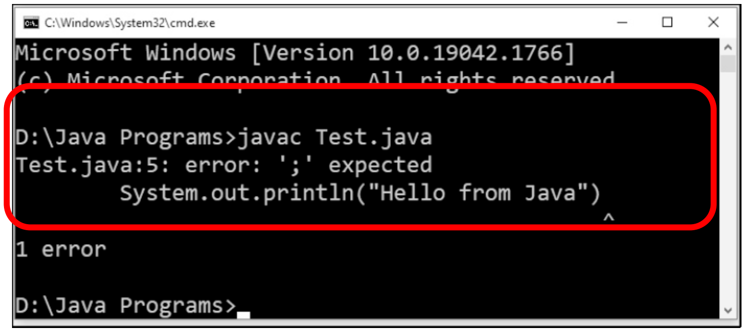
ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਈਲ ਹੋਣ ਤੋਂ ਬਾਅਦ ਇਹ ਉਸੇ ਫੋਲਡਰ ਵਿੱਚ ਇੱਕ .class ਫਾਈਲ ਤਿਆਰ ਕਰ ਦੇਵੇਗਾ। ਜਿਸ ਵਿੱਚ ਸਾਡੇ ਪ੍ਰੋਗਰਾਮ ਦਾ ਬਾਈਟਕੋਡ ਸਟੋਰ ਹੋ ਜਾਵੇਗਾ॥ ਇਸ ਬਾਈਟਕੋਡ ਨੂੰ ਹੀ ਜਾਵਾ ਇੰਟਰਪ੍ਰੇਟਰ ਦੁਆਰਾ JVM ਵਿੱਚ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ।





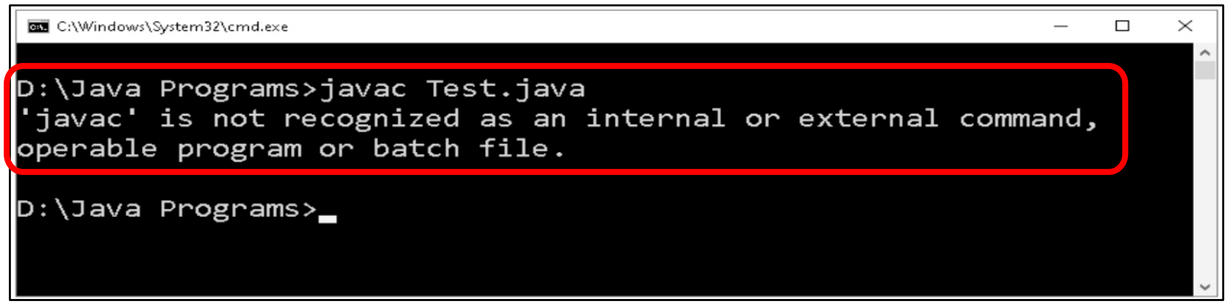
ਚਿੱਤਰ : 4.10 ਫਾਈਲ ਦੇ ਕੰਪਾਈਲੇਸ਼ਨ ਤੋਂ ਬਾਅਦ Test.class ਤਿਆਰ ਹੋ ਜਾਵੇਗੀ।

ਪਰ ਜੇਕਰ ਸਾਡੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੁੱਝ ਸਿੰਟੈਕਸ ਐਰਰਜ਼ (syntax errors) ਹੋਣ ਤਾਂ ਇਹ ਸੋਰਸ ਕੋਡ ਫਾਈਲ ਵਿਚ ਮੌਜੂਦ ਐਰਰਜ਼ ਨਾਲ ਸੰਬੰਧਤ ਮੈਸੇਜ, ਸਮੇਤ ਲਾਈਨ ਨੰਬਰ ਦਿਖਾਏਗਾ; ਜਿਵੇਂ ਕਿ ਹੇਠਾਂ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:



ਚਿੱਤਰ 4.11 ਕੰਪਾਇਲੇਸ਼ਨ ਐਰਰਜ਼ (Compilition Erros) (ਜੇ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਮੌਜੂਦ ਹੋਣ)

ਜੇਕਰ ਸਾਨੂੰ ਕੰਸੋਲ ਸਕੀਨ ਉਪਰ ਹੇਠਾਂ ਦਿੱਤਾ ਐਰਰ ਮੈਸੇਜ (Error Message) ਦਿਖਾਈ ਦੇ ਰਿਹਾ ਹੈ, ਤਾਂ ਸਾਨੂੰ ਆਪਣੇ ਸਿਸਟਮ ਦੀਆਂ ਪਾਥ ਸੈਟਿੰਗਾਂ (Path Settings) ਨੂੰ ਐਡਿਟ ਕਰਨਾ ਪਵੇਗਾ।ਤਾਂ ਜੋ ਅਸੀਂ ਆਪਣੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸਫਲਤਾਪੂਰਵਕ ਕੰਪਾਇਲ ਕਰ ਸਕੀਏ।



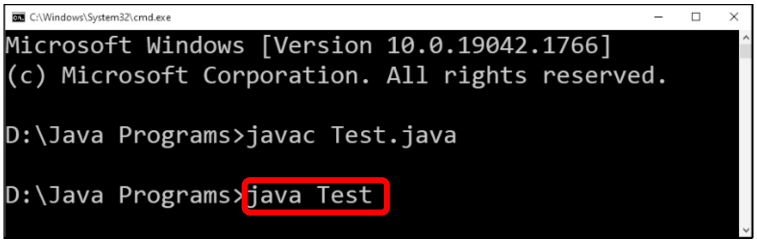
ਚਿੱਤਰ : 4.12 JAVA ਇੰਸਟਾਲੇਸ਼ਨ ਨਾਲ ਸੰਬਧਤ ਪਾਥ ਸੈਟਿੰਗ ਐਰਰ

ਸਟੈਂਪ 4: JAVA ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਣ ਲਈ ਹੇਠ ਦਿੱਤੇ ਕਮਾਂਡ ਟੈਕਸ ਦੀ ਵਰਤੋਂ ਕਰੋ:

Syntax:

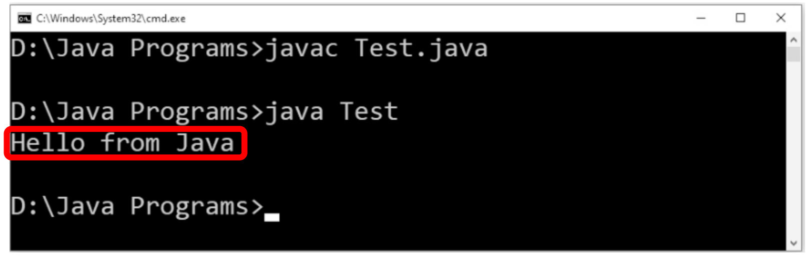
java classname (ਨੋਟ: ਕਲਾਸ ਦਾ ਨਾਮ ਲਿਖਦੇ ਸਮੇਂ ਕੇਸਿੰਗ (CASEING) ਦਾ ਧਿਆਨ ਰੱਖੋ)

ਅੱਗੇ ਦਿੱਤਾ ਚਿੱਤਰ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ Test.Java ਦੀ ਸਫਲਤਾਪੂਰਵਕ ਕੰਪਾਇਲੇਸ਼ਨ ਤੋਂ ਬਾਅਦ ਉਸ ਨੂੰ ਚਲਾਉਣ (execute) ਸੰਬੰਧੀ ਕਮਾਂਡ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ:



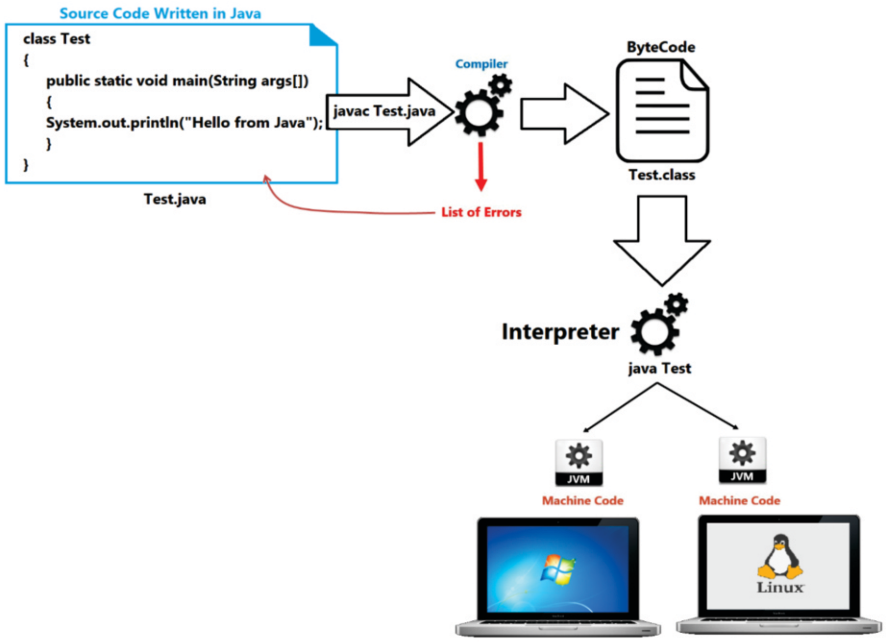
ਚਿੱਤਰ 4.13: ਸਫਲਤਾਪੂਰਵਕ ਕੰਪਾਇਲੇਸ਼ਨ ਤੋਂ ਬਾਅਦ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਚਲਾਉਣਾ

ਕੰਸੋਲ ਸਕ੍ਰੀਨ ਉਪਰ java Test ਟਾਈਪ ਕਰਨ ਤੋਂ ਬਾਅਦ ਕੀਬੋਰਡ ਤੋਂ ਐਂਟਰ ਕੀਅ ਦਬਾਓ ਅਤੇ ਇਹ ਹੇਠਾਂ ਦਰਸਾਏ ਅਨੁਸਾਰ ਪ੍ਰੋਗਰਾਮ ਦੀ ਆਉਟਪੁੱਟ ਨੂੰ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰੇਗਾ:



ਚਿੱਤਰ: 4.14: Test.java ਪ੍ਰੋਗਰਾਮ ਦੀ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਤੋਂ ਬਾਅਦ ਆਉਟਪੁੱਟ

ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਇਲ ਕਰਨ ਅਤੇ ਉਸਨੂੰ ਚਲਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:

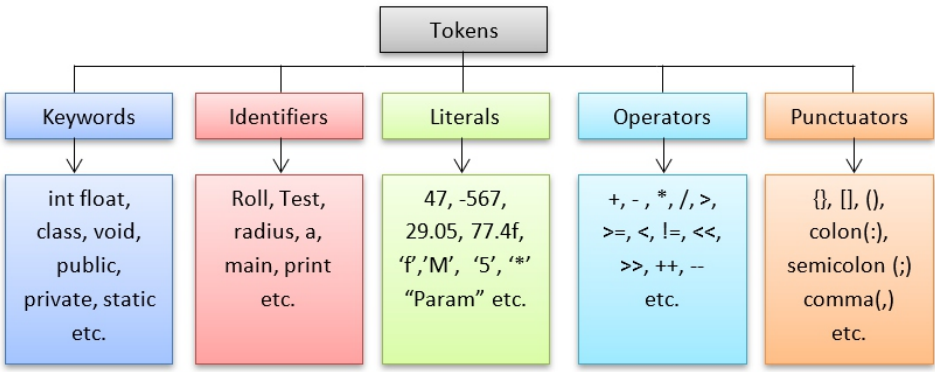


ਚਿੱਤਰ: 4.15 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ ਅਤੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਪ੍ਰੋਸੈਸ

4.8 ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਲਈ ਬੁਨਿਆਦੀ ਤੱਤ (Basic Elements for Java Programming) : ਮਨੁੱਖਾਂ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਲਈ ਅਸੀਂ ਕੁਦਰਤੀ ਭਾਸ਼ਾਵਾਂ, ਜਿਵੇਂ ਕਿ ਅੰਗਰੇਜ਼ੀ, ਹਿੰਦੀ ਅਤੇ ਪੰਜਾਬੀ ਆਦਿ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਪਰ ਕੰਪਿਊਟਰਾਂ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਲਈ ਅਸੀਂ ਸਿਰਫ ਉਹੀ ਭਾਸ਼ਾਵਾਂ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ ਜਿਹੜੀਆਂ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੁਆਰਾ ਸਿੱਧੇ ਜਾਂ ਅਸਿੱਧੇ ਢੰਗ ਨਾਲ ਸਮਝੀਆਂ ਜਾ ਸਕਦੀਆਂ ਹਨ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਹੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਇੱਕ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਹੈ। ਜਿਵੇਂ ਕਿ ਸਾਨੂੰ ਕਿਸੇ ਵੀ ਕੁਦਰਤੀ ਭਾਸ਼ਾ ਦੀ ਵਰਤੋਂ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ ਇਸਨੂੰ ਸਿੱਖਣਾ ਪੈਂਦਾ ਹੈ, ਠੀਕ ਉਸੇ ਤਰ੍ਹਾਂ ਸਾਨੂੰ ਕੰਪਿਊਟਰਾਂ ਨਾਲ ਸੰਚਾਰ **(communication)** ਕਰਨ ਲਈ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਵੀ ਸਿੱਖਣੀਆਂ ਪੈਂਦੀਆਂ ਹਨ। ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਸਿੱਖਣਾ ਵੀ ਕਿਸੇ ਵੀ ਕੁਦਰਤੀ ਭਾਸ਼ਾ, ਜਿਵੇਂ ਹਿੰਦੀ, ਪੰਜਾਬੀ ਆਦਿ, ਨੂੰ ਸਿੱਖਣ ਵਾਂਗ ਹੀ ਹੁੰਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਕਿਵੇਂ ਲਿਖਣਾ ਹੈ ਇਹ ਸਿੱਧਾ ਸਿੱਖਣ ਦੀ ਬਜਾਏ ਸਾਨੂੰ ਪਹਿਲਾਂ ਇਹ ਪਤਾ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਕਿ ਜਾਵਾ ਵਿੱਚ ਕਿਹੜੇ ਅੱਖਰ, ਨੰਬਰ ਅਤੇ ਚਿੰਨ੍ਹ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਫਿਰ ਇਨ੍ਹਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਵੱਖ ਵੱਖ ਟੋਕਨਾਂ ਨੂੰ ਕਿਵੇਂ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਅੰਤ ਵਿੱਚ ਇਹਨਾਂ ਟੋਕਨਾ ਨੂੰ ਜੋੜ ਕੇ ਹਦਾਇਤਾਂ ਨੂੰ ਕਿਵੇਂ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਹਦਾਇਤਾਂ ਦਾ ਇੱਕ ਸਮੂਹ ਆਖਿਰ ਵਿੱਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰਨ ਲਈ ਜੋੜਿਆ ਜਾਵੇਗਾ। ਆਉ ਹੁਣ ਜਾਵਾ ਦੇ ਕਰੈਕਟਰ ਸੈਂਟ ਨੂੰ ਸਮਝਦੇ ਹੋਏ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਨੂੰ ਸਿੱਖਣ ਦਾ ਪ੍ਰੋਸੈਸ ਸ਼ੁਰੂ ਕਰੀਏ।

4.8.1 ਕਰੈਕਟਰ ਸੈੱਟ (Character Set) : ਇਹ ਕਿਸੇ ਵੀ ਭਾਸ਼ਾ, ਚਾਹੇ ਉਹ ਕੁਦਰਤੀ ਹੋਵੇ ਜਾਂ ਕੰਪਿਊਟਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ, ਨੂੰ ਸਿੱਖਣ ਦਾ ਪਹਿਲਾ ਸਟੈਪ ਹੈ। ਅਸੀਂ ਕੋਈ ਵੀ ਭਾਸ਼ਾ ਤਾਂ ਹੀ ਸਿੱਖ ਸਕਦੇ ਹਾਂ ਜੇ ਅਸੀਂ ਉਸ ਭਾਸ਼ਾ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਅੱਖਰਾਂ ਅਤੇ ਚਿੰਨ੍ਹਾਂ (characters and symbols) ਦੀ ਜਾਣਕਾਰੀ ਰੱਖਦੇ ਹਾਂ। ਇਸਲਈ ਜਾਵਾ ਭਾਸ਼ਾ ਸਿੱਖਣ ਤੋਂ ਪਹਿਲਾਂ ਸਾਨੂੰ ਇਸ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਕਰੈਕਟਰਾਂ ਅਤੇ ਚਿੰਨ੍ਹਾਂ ਤੋਂ ਜਾਣੂ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ॥C ਅਤੇ C++ ਵਰਗੀਆਂ ਹੋਰ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾਵਾਂ ਦੇ ਉਲਟ, ਜਾਵਾ ਵਿਚ ਅੱਖਰਾਂ (ਚਹੳਰੳਚਟੲਰਸ) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ ਯੂਨੀਕੋਡ (Unicode) ਸਟੈਂਡਰਡ ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ। ਯੂਨੀਕੋਡ ਇੱਕ 16-ਬਿੱਟ ਕਰੈਕਟਰ ਕੋਡ ਸੈਂਟ ਹੈ। ਜੋ ਸਾਰੀਆਂ ਮਨੁੱਖੀ ਭਾਸ਼ਾਵਾਂ ਜਿਵੇਂ ਕਿ: ਅੰਗਰੇਜ਼ੀ, ਹਿੰਦੀ, ਫ੍ਰੈਂਚ, ਜਰਮਨ, ਚੀਨੀ, ਜਾਪਾਨੀ ਆਦਿ ਵਿੱਚ ਉਪਲਬਧ ਸਾਰੇ ਅੱਖਰਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ। ਯੂਨੀਕੋਡ ਕਰੈਕਟਰ ਸੈਂਟ ਜਾਵਾ ਨੂੰ ਇੱਕ ਵਿਸ਼ਵਵਿਆਪੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਬਣਾਉਂਦਾ ਹੈ।।

4.8.2 ਟੋਕਨਜ਼ (Tokens) : ਟੋਕਨ ਅੰਗਰੇਜ਼ੀ ਭਾਸ਼ਾ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਸ਼ਬਦਾਂ ਅਤੇ ਵਿਸ਼ੇਸ਼ ਚਿੰਨ੍ਹਾਂ Punchuation Marks ਵਾਂਗ ਹੁੰਦੇ ਹਨ। ਕਿਸੇ ਵੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ ਵਿੱਚ, ਜਿਵੇਂ ਕਿ ਜਾਵਾ ਭਾਸ਼ਾ ਵਿੱਚ, ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਟੋਕਨਾਂ ਦਾ ਬਣਿਆ ਹੁੰਦਾ ਹੈ। ਟੋਕਨ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਭ ਤੋਂ ਛੋਟੀਆਂ ਵਿਅਕਤੀਗਤ ਇਕਾਈਆਂ Small Indivldual Units ਹੁੰਦੀਆਂ ਹਨ। ਜਦੋਂ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਕੰਪਾਈਲਰ ਸੋਰਸ ਕੋਡ ਨੂੰ ਸਕੈਨ ਕਰਦਾ ਹੈ ਅਤੇ ਸਿੰਟੈਕਸ ਐਰਰਜ਼ ਨੂੰ ਲੱਭਣ ਲਈ ਸੋਰਸ ਕੋਡ ਨੂੰ ਟੋਕਨਾਂ ਵਿੱਚ ਪਾਰਸ (Parse) ਕਰਦਾ ਹੈ। ਜਾਵਾ ਟੋਕਨਾਂ ਨੂੰ ਮੋਟੇ ਤੌਰ 'ਤੇ ਕੀਵਰਡਸ, ਆਈਡੈਂਟੀਫਾਇਰਜ਼, ਲਿਟਰਲਜ਼ ਕਾਂਸਟੈਂਟਸ), C ਅਤੇ ਪੰਕਚੁਏਟਰਜ਼ (ਵਿਸ਼ੇਸ਼ ਚਿੰਨ੍ਹ) ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਗਿਆ ਹੈ:



ਚਿੱਤਰ: 4.16 ਜਾਵਾ ਵਿਚ ਟੋਕਨਜ਼ ਦੀਆਂ ਕਿਸਮਾਂ

4.8.2.1 ਕੀਵਰਡਜ਼ (Keywords) : ਕੀਵਰਡਜ਼ ਨੂੰ ਰਿਜ਼ਰਵ ਵਰਡਜ਼ (Reserve Words) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਸ਼ਬਦ ਜਾਵਾ ਕੰਪਾਈਲਰ ਵਿੱਚ ਪਹਿਲਾਂ ਤੋਂ ਪਰਿਭਾਸ਼ਿਤ ਹੁੰਦੇ ਹਨ। ਇਨ੍ਹਾਂ ਸ਼ਬਦਾਂ ਦੇ ਅਰਥ ਪਹਿਲਾਂ ਤੋਂ ਪਰਿਭਾਸ਼ਿਤ ਹੁੰਦੇ ਹਨ।ਕੀਵਰਡਜ਼ ਨੂੰ ਉਹਨਾਂ ਵਿਸ਼ੇਸ਼ ਉਦੇਸ਼ਾਂ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਜਿਨ੍ਹਾਂ ਲਈ ਉਹਨਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੈ ਅਤੇ ਇਹੀ ਕਾਰਨ ਹੈ ਕਿ ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਯੂਜ਼ਰ ਦੁਆਰਾ ਪਰਿਭਾਸ਼ਿਤ ਨਾਮ (ਆਈਡੈਂਟੀਫਾਇਰਜ਼) ਵਜੋਂ ਨਹੀਂ ਕੀਤੀ ਜਾ ਸਕਦੀ। ਅਸੀਂ ਕੀਵਰਡਜ਼ ਦੇ ਅਰਥ ਨਹੀਂ ਬਦਲ ਸਕਦੇ ॥ਇਹ ਕੀਵਰਡਜ਼ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਜਿੱਥੇ ਵੀ ਲੋੜੀਂਦੇ ਹਨ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿਚ ਸਾਰੇ ਕੀਵਰਡ ਸਿਰਫ ਛੋਟੇ ਅੱਖਰਾਂ Lower Case ਵਿੱਚ ਲਿਖੇ ਜਾਣੇ ਚਾਹੀਦੇ ਹਨ। ਕਿਉਂਕਿ ਜਾਵਾ ਕੇਸ ਸੰਵੇਦਨਸ਼ੀਲ (Case-Sensitive) ਭਾਸ਼ਾ ਹੈ, ਇਸ ਲਈ ਜੇਕਰ ਅਸੀਂ ਇਹਨਾਂ ਕੀਵਰਡਜ਼ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵੱਡੇ-ਕੇਸ (upper-case) ਵਿੱਚ ਲਿਖਦੇ ਹਾਂ, ਤਾਂ ਇਹ ਕੰਪਾਇਲੇਸ਼ਨ ਐਰਰਜ਼ ਨੂੰ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰੇਗਾ। ਕੇਸ ਸੰਵੇਦਨਸ਼ੀਲ ਭਾਸ਼ਾ ਇੱਕ ਅਜਿਹੀ ਭਾਸ਼ਾ ਹੁੰਦੀ ਹੈ।

ਜੋ ਛੋਟੇ ਅੱਖਰਾਂ ਅਤੇ ਵੱਡੇ ਅੱਖਰਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਤੱਤਾਂ ਵਜੋਂ ਮੰਨਦੀ ਹੈ ॥ਜਾਵਾ ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਵੱਖ-ਵੱਖ ਕੀਵਰਡਜ਼ ਹੇਠਾਂ ਸੂਚੀਬੱਧ ਕੀਤੇ ਗਏ ਹਨ:

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
enum	extends	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	package	private	protected	public
return	short	static	strictfp	super
switch	synchronized	this	throw	throws
transient	try	void	volatile	while

ਟੇਬਲ 4.1 ਜਾਵਾ ਕੀਵਰਡਜ਼ ਦੀ ਸੂਚੀ

4.8.2.2 ਆਈਡੈਂਟੀਫਾਇਰਜ਼ (Identifiers) : ਆਈਡੈਂਟੀਫਾਇਰ ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਲੀਮੈਂਟ (ਤੱਤ) ਜਿਵੇਂ ਕਿ: ਵੇਰੀਏਬਲ, ਕਾਂਸਟੈਂਟਸ, ਐਰੇ, ਮੈਥਡਜ਼, ਕਲਾਸਾਂ, ਇੰਟਰਫੇਸ, ਆਦਿ ਨੂੰ ਦਿੱਤੇ ਗਏ ਨਾਮ ਹੁੰਦੇ ਹਨ। ਹਰੇਕ ਪ੍ਰੋਗਰਾਮ ਐਲੀਮੈਂਟ ਨੂੰ ਦੂਜੇ ਐਲੀਮੈਂਟਸ ਤੋਂ ਵੱਖਰਾ ਕਰਨ ਲਈ ਨਾਮ ਦਿੱਤਾ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ। ਐਲੀਮੈਂਟਸ ਨੂੰ ਦਿੱਤਾ ਗਿਆ ਨਾਮ ਸਾਰਥਕ (meaningful) ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਤਾਂ ਜੋ ਪ੍ਰੋਗਰਾਮ ਐਲੀਮੈਂਟਸ ਨੂੰ ਆਸਾਨੀ ਨਾਲ ਸਮਝਿਆ ਜਾ ਸਕੇ। ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਲੀਮੈਂਟਸ ਨੂੰ ਨਾਮ ਦੇਣ ਤੋਂ ਬਾਅਦ ਉਹਨਾਂ ਨੂੰ ਉਹਨਾਂ ਦੇ ਨਾਮ ਦੁਆਰਾ ਪਛਾਣਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਐਲੀਮੈਂਟਸ ਦੇ ਨਾਮ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ, ਕੁਝ ਨਾਮਕਰਨ ਨਿਯਮਾਂ (naming rules) ਦੀ ਪਾਲਣਾ ਕੀਤੀ ਜਾਣੀ ਚਾਹੀਦੀ ਹੈ। ਇਹ ਨਾਮਕਰਨ ਨਿਯਮ ਹੇਠਾਂ ਦਿੱਤੇ ਗਏ ਹਨ:

a) ਆਈਡੈਂਟੀਫਾਇਰ ਵਿੱਚ ਵੱਡੇ ਅਤੇ ਛੋਟੇ ਅੱਖਰ (Uppercase and Lowercase Letters), ਅੰਕ (Digits), ਡਾਲਰ ਚਿੰਨ੍ਹ ($) ਅਤੇ ਇੱਕ ਅੰਡਰਸਕੋਰ (_) ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ।
b) ਆਈਡੈਂਟੀਫਾਇਰ ਅੱਖਰ (letter), ਡਾਲਰ ਦੇ ਚਿੰਨ੍ਹ ਜਾਂ ਇੱਕ ਅੰਡਰਸਕੋਰ ਨਾਲ ਸ਼ੁਰੂ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ, ਭਾਵ ਇਹ ਅੰਕ ਨਾਲ ਸ਼ੁਰੂ ਨਹੀਂ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।
c) ਆਈਡੈਂਟੀਫਾਇਰ ਅੱਖਰ-ਸੰਵੇਦਨਸ਼ੀਲ (Case-Sensitive) ਹੁੰਦੇ ਹਨ, ਭਾਵ ਵੱਡੇ ਅੱਖਰਾਂ (Uppercase) ਵਿੱਚ ਲਿਖਿਆ ਹੋਇਆ ਆਈਡੈਂਟੀਫਾਇਰ ਛੋਟੇ ਅੱਖਰਾਂ (Lowercase) ਵਿੱਚ ਲਿਖੇ ਗਏ ਆਈਡੈਂਟੀਫਾਇਰ ਤੋਂ ਵੱਖਰਾ (different) ਹੁੰਦਾ ਹੈ।
d) ਇੱਕ ਕੀਵਰਡ ਨੂੰ ਆਈਡੈਂਟੀਫਾਇਰ ਵਜੋਂ ਨਹੀਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਕਿਉਂਕਿ ਇਹ ਇੱਕ ਰਾਖਵਾਂ ਸ਼ਬਦ (reserved word) ਹੁੰਦਾ ਹੈ ਅਤੇ ਇਸਦਾ ਇਕ ਵਿਸ਼ੇਸ਼ ਅਰਥ ਹੁੰਦਾ ਹੈ।
e) ਪ੍ਰੋਗਰਾਮ ਜਾਂ ਸਕੋਪ ਦੇ ਇੱਕ ਸੈਕਸ਼ਨ ਅੰਦਰ, ਹਰੇਕ ਯੂਜ਼ਰ-ਪਰਿਭਾਸ਼ਿਤ ਆਈਟਮ (User Defined Item) ਦਾ ਇੱਕ ਵਿਲੱਖਣ (Unique) ਆਈਡੈਂਟੀਫਾਇਰ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।
f) ਆਈਡੈਂਟੀਫਾਇਰ ਕਿਸੇ ਵੀ ਲੰਬਾਈ ਦੇ ਹੋ ਸਕਦੇ ਹਨ।
g) ਇੱਕ ਆਈਡੈਂਟੀਫਾਇਰ ਵਿੱਚ ਵਾਈਟ ਸਪੇਸ (white space) ਅਤੇ ਹੋਰ ਅੱਖਰ ਨਹੀਂ ਹੋਣੇ ਚਾਹੀਦੇ ,ਜਿਵੇਂ ਕਿ- *, ; ਆਦਿ। ਹੇਠਾਂ ਸਹੀ **(Legal)** ਅਤੇ ਗਲਤ **(IIlegal)** ਨਾਮ ਵਾਲੇ ਆਈਡੈਂਟੀਫਾਇਰਜ਼ ਦੀਆਂ ਕੁਝ ਉਦਾਹਰਣਾਂ ਦਿੱਤੀਆਂ ਗਈਆਂ ਹਨ:

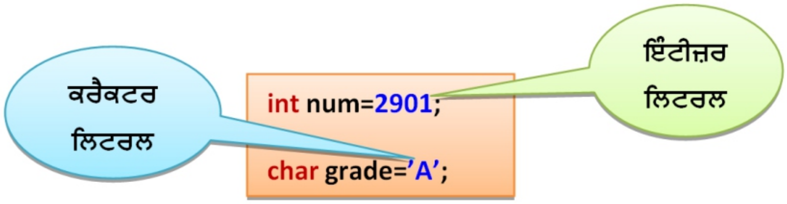
❖ **ਸਹੀ ਨਾਮ ਵਾਲੇ ਆਈਡੈਂਟੀਫਾਇਰਜ਼ (Legal identifiers) ਦੀਆਂ ਉਦਾਹਰਣਾਂ :**
MinNumber, total, vk49, hello_world, $amount, _under_value

❖ ਗਲਤ ਨਾਮ ਵਾਲੇ ਆਈਡੈਂਟੀਫਾਇਰਜ਼ (illegal identifiers) ਦੀਆਂ ਉਦਾਹਰਣਾਂ :
4avk, amount, Roll No, Hous

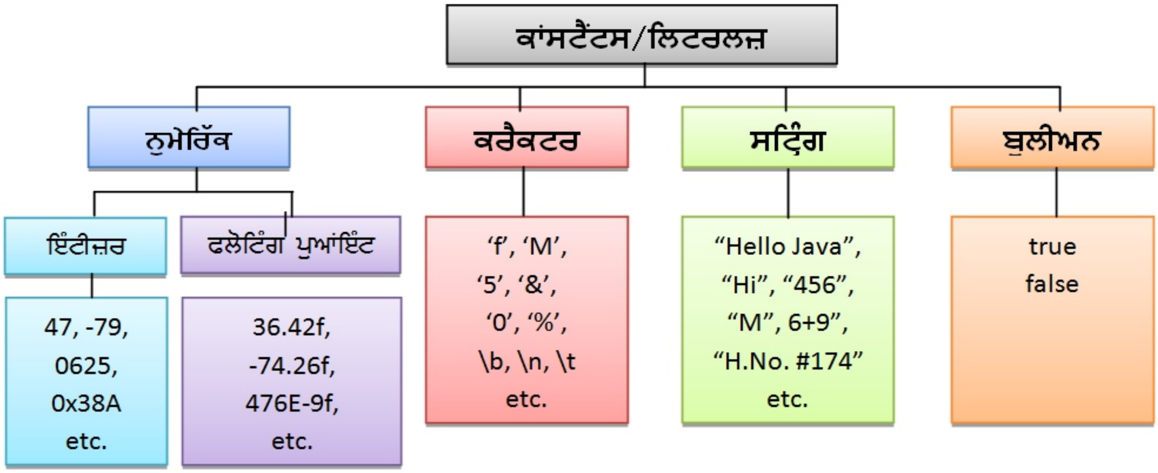
ਜਾਵਾ ਦੇ ਸਟੈਂਡਰਡ ਨੇਮਿੰਗ ਕਨਵੈਨਸ਼ਨਜ਼ ਅਨੁਸਾਰ, ਵੇਰੀਏਬਲ ਲਈ ਸਾਰੇ ਲੈਟਰਜ਼ (letters) ਛੋਟੇ-ਅੱਖਰਾਂ (Lowercase) ਵਿੱਚ ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ। ਵੇਰੀਏਬਲਾਂ ਦੇ ਨਾਵਾਂ ਲਈ ਅੰਡਰਸਕੋਰ ਦੀ ਸਿਫ਼ਾਰਸ਼ ਨਹੀਂ ਕੀਤੀ ਜਾਂਦੀ। ਕਾਂਸਟੈਂਟਸ (static, final, attributes ਅਤੇ enums) ਲਈ ਸਾਰੇ ਲੈਟਰਜ਼ (letters) ਵੱਡੇ ਅੱਖਰਾਂ (Uppercase) ਵਿੱਚ ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ।

4.8.2.3 ਲਿਟਰਲਜ਼ (Literals):

ਲਿਟਰਲਜ਼ ਨੂੰ ਸਥਿਰ ਮੁੱਲ (Constant Values) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਨਿਸ਼ਚਿਤ (Fixed) ਮੁੱਲ ਹੁੰਦੇ ਹਨ ਜੋ ਆਮ ਤੌਰ ’ਤੇ ਪ੍ਰੋਗਰਾਮ ਦੇ ਤੱਤਾਂ ਲਈ ਨਿਰਧਾਰਤ ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ: 2901, 47.29f, “Param”, ‘A’ ਆਦਿ ਸਾਰੇ ਸਥਿਰ ਮੁੱਲ ਹਨ।



ਮੁੱਲ ਦੀ ਕਿਸਮ ਦੇ ਆਧਾਰ ’ਤੇ C ਲਿਟਰਲਕਾਂਸਟੈਂਟਸ ਨੂੰ ਮੋਟੇ ਤੌਰ ’ਤੇ ਚਾਰ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ: ਨੁਮੇਰਿੱਕ (Numeric) ਕਾਂਸਟੈਂਟਸ, ਕਰੈਕਟਰ (Character) ਕਾਂਸਟੈਂਟਸ, ਸਟਿੰਗ (String) ਕਾਂਸਟੈਂਟਸ ਅਤੇ ਬੁਲੀਅਨ (Boolean) ਕਾਂਸਟੈਂਟਸ :



ਚਿੱਤਰ: 4.17 ਜਾਵਾ ਵਿਚ ਲਿਟਰਲ ਕਾਂਸਟੈਂਟਸ ਦੀਆਂ ਕਿਸਮਾਂ

ੳ. ਨੁਮੇਰਿੱਕ ਲਿਟਰਲਜ਼ (Numeric Literals) :

ਨੁਮੇਰਿਕ ਲਿਟਰਲਜ਼ ਉਹ ਸੰਖਿਆਤਮਕ ਮੁੱਲ ਹੁੰਦੇ ਹਨ ਜੋ ਸੰਖਿਆਤਮਕ ਗਣਨਾਵਾਂ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਇਹਨਾਂ ਦਾ ਮੁੱਲ ਅੰਕਾਂ ਦੀ ਲੜੀ (squence of digits) ਦੇ ਰੂਪ ਵਿਚ(ਦਸ਼ਮਲਵ ਬਿੰਦੂ ਦੇ ਨਾਲ ਜਾਂ ਦਸ਼ਮਲਵ ਬਿੰਦੂ ਤੋਂ ਬਿਨਾਂ) ਹੁੰਦਾ ਹੈ ਜੋ ਸਕਾਰਾਤਮਕ (Positive) ਜਾਂ ਨਕਾਰਾਤਮਕ (Negative) ਹੋ ਸਕਦਾ ਹੈ। ਮੂਲ ਰੂਪ ਵਿੱਚ (By default), ਨੁਮੇਰਿੱਕ ਲਿਟਰਲਜ਼ ਨੂੰ ਅੱਗੇ ਦੋ ਕਿਸਮਾਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ:

i. **ਇੰਟੀਜ਼ਰ ਲਿਟਰਲਜ਼ (Integer Literals) :** ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਵਿਚ ਅੰਸ਼ਿਕ/ਦਸ਼ਮਲਵ (fractional/decimal) ਭਾਗ ਨਹੀਂ ਹੁੰਦਾ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਵਿੱਚ ਸਕਾਰਾਤਮਕ (+) ਜਾਂ ਨਕਾਰਾਤਮਕ (−) ਚਿੰਨ ਦੇ ਨਾਲ 0 ਤੋਂ 9 ਤੱਕ ਦੇ ਅੰਕ ਹੁੰਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ: 56, +26, −96 ਆਦਿ ਇੰਟੀਜ਼ਰ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ। ਕਿਸੇ ਮੁੱਲ ਨੂੰ ਲਾਂਗ ਇੰਟੀਜ਼ਰ (long integer) ਵਜੋਂ ਦਰਸਾਉਣ ਲਈ ਅਸੀਂ ਉਸ ਮੁੱਲ ਦੇ ਨਾਲ ਪਿਛੇਤਰ (suffix) ਵਜੋਂ L ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ, ਉਦਾਹਰਣ ਵੱਜੋਂ 3456L ਲਿਟਰਲ, ਇਹ ਲਾਂਗ ਇੰਟੀਜ਼ਰ ਨੂੰ ਦਰਸਾਵੇਗਾ। ਇੰਟੀਜ਼ਰ ਲਿਟਰਲਜ਼ ਨੂੰ ਤਿੰਨ ਵੱਖ-ਵੱਖ ਨੰਬਰ ਸਿਸਟਮਾਂ ਦੁਆਰਾ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ:

- ❖ **ਡੇਸੀਮਲ (Decimal (base 10)) :** ਕਿਸੇ ਸੰਖਿਆ ਨੂੰ ਡੇਸੀਮਲ (ਦਸ਼ਮਲਵ) ਫਾਰਮੈਟ ਵਿਚ ਦਰਸਾਉਣ ਲਈ ਉਸਦੇ ਸਭ ਤੋਂ ਖੱਬੇ (left most) ਅੰਕ ਨੂੰ ਗੈਰ-ਜ਼ੀਰੋ (Non-Zero) ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ: 56, 5689 ਆਦਿ।
- ❖ **ਓਕਟਲ (Octal (base 8)) :** ਅਸੀਂ 0 ਤੋਂ 7 ਅੰਕਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਬਣੇ ਨੰਬਰਾਂ ਤੋਂ ਪਹਿਲਾਂ ਜ਼ੀਰੋ ਲਗਾ ਕੇ ਉਸ ਨੰਬਰ ਨੂੰ ਓਕਟਲ ਫਾਰਮੈਟ ਵਿਚ ਦਰਸਾ ਸਕਦੇ ਹਾਂ। ਉਦਾਹਰਣ ਲਈ: 0625, 034 ਆਦਿ।
- ❖ **ਹੇਕਸਾਡੇਸੀਮਲ (Hexadecimal (base 16)) :** ਕਿਸੇ ਸੰਖਿਆ ਨੂੰ ਹੇਕਸਾਡੇਸੀਮਲ ਫਾਰਮੈਟ ਵਿਚ ਦਰਸ਼ਾਉਣ ਲਈ ਉਸ ਸੰਖਿਆ ਤੋਂ ਪਹਿਲਾਂ Ox ਜਾਂ OX ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਉਦਾਹਰਣ ਲਈ: 0×38A, 0×C4 ਆਦਿ।

ii. **ਫਲੋਟਿੰਗ ਪੁਆਂਇੰਟ ਲਿਟਰਲਜ਼ (Floating Point Literals) :** ਇਹਨਾਂ ਨੂੰ ਰੀਅਲ (ਅਸਲ) ਸੰਖਿਆਵਾਂ (Real Numbers) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਫੈਕਸ਼ਨਲ/ਦਸ਼ਮਲਵ (fractional/decimal) ਭਾਗ ਵਾਲੀਆਂ ਸੰਖਿਆਵਾਂ ਹੁੰਦੀਆਂ ਹਨ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਵਿੱਚ ਸਕਾਰਾਤਮਕ (+) ਜਾਂ ਨਕਾਰਾਤਮਕ (−) ਚਿੰਨ੍ਹ ਦੇ ਨਾਲ 0 ਤੋਂ 9 ਤੱਕ ਦੇ ਅੰਕ ਹੁੰਦੇ ਹਨ। ਇਸ ਵਿੱਚ ਇੱਕ ਦਸ਼ਮਲਵ ਬਿੰਦੂ (.) ਵੀ ਹੁੰਦਾ ਹੈ ਜੋ ਰੀਅਲ ਸੰਖਿਆ ਦੇ ਇੰਟੀਜ਼ਰ (Integral) ਅਤੇ ਫ੍ਰੈਕਸ਼ਨ (Fractional) ਭਾਗਾਂ ਨੂੰ ਵੱਖ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਇੰਟੀਜ਼ਰ (Integral) ਭਾਗ ਨਾਲ ਕੋਈ ਚਿੰਨ੍ਹ ਨਹੀਂ ਹੈ, ਤਾਂ ਇਸਨੂੰ ਸਕਾਰਾਤਮਕ (Postive) ਰੀਅਲ ਲਿਟਲ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: 3.14, +256.5896, 96.14, 36.00 ਆਦਿ ਰੀਅਲ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ। ਜਾਵਾ ਵਿੱਚ ਦੋ ਤਰ੍ਹਾਂ ਦੇ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਨੰਬਰ ਹੁੰਦੇ ਹਨ: ਫਲੋਟ (float) ਅਤੇ ਡਬਲ (double) ਅਸੀਂ ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਲਿਟਰਲਜ਼ ਲਈ ਇੱਕ ਪਿਛੇਤਰ ਜਿਵੇਂ ਕਿ D, d, F, f ਵੀ ਲਗਾ ਸਕਦੇ ਹਾਂ। ਪਿਛੇਤਰ ਦ ਜਾਂ ਧ ਦੀ ਵਰਤੋਂ ਡਬਲ-ਪੀਸੀਜ਼ਨ (Double-Precision) ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਲਿਟਰਲ ਮੰਨਿਆਂ ਜਾਂਦਾ ਹੈ।ਉਦਾਹਰਣ ਲਈ:

65563.554	Double-precision floating-point literal (by default)
34578.343D	Double-precision floating-point literal
45344.01f	Floating-point literal

ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਲਿਟਰਜ਼ ਨੂੰ ਤਰੀਕਿਆਂ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਸਟੈਂਡਰਡ ਨੋਟੇਸ਼ਨ ਅਤੇ ਵਿਗਿਆਨਕ ਨੋਟਸ਼ਨ 1.

- ❖ **ਸਟੈਂਡਰਡ ਨੋਟੇਸ਼ਨ (Standard Notation) :** ਸਟੈਂਡਰਡ ਨੋਟੇਸ਼ਨ ਵਿੱਚ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਨੰਬਰ ਇੱਕ ਪੂਰਨ ਅੰਕ (integer) ਅਤੇ ਇੱਕ ਫਰੈਕਸ਼ਨ (fractional) ਹਿੱਸੇ ਤੋਂ ਬਣਿਆ ਹੁੰਦਾ ਹੈ ਅਤੇ ਇਹਨਾਂ ਦੋਵੇਂ ਹਿੱਸਿਆਂ ਵਿਚਕਾਰ ਇੱਕ ਦਸ਼ਮਲਵ ਬਿੰਦੂ (decimal point) ਲੱਗਿਆ ਹੁੰਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: 130.35, 41.0, 0.56, 0.67, 0.67, 7.71 ਆਦਿ।
- ❖ **ਵਿਗਿਆਨਕ ਨੋਟੇਸ਼ਨ (Scientific Notation) :** ਵਿਗਿਆਨਕ ਨੋਟੇਸ਼ਨ ਵਿੱਚ ਇੱਕ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਲਿਟਰਲ ਵਿੱਚ ਇੱਕ ਮੈਟੀਸਾ (mantissa) ਅਤੇ ਇੱਕ ਐਕਸਪੋਨੈਂਟ (exponent) ਭਾਗ ਹੁੰਦਾ ਹੈ। ਮੈਨਟੀਸਾ ਭਾਗ ਸਟੈਂਡਰਡ ਨੋਟੇਸ਼ਨ ਵਿੱਚ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਨੰਬਰ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਅਤੇ ਐਕਸਪੋਨੈਂਟ ਭਾਗ 10 ਦੀ ਉਸ ਪਾਵਰ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ ਜਿਸ ਨਾਲ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਨੰਬਰ ਨੂੰ ਗੁਣਾ ਕੀਤਾ ਜਾਣਾ ਹੈ। ਮੈਨਟੀਸਾ ਭਾਗ ਅਤੇ ਐਕਸਪੋਨੈਂਟ ਭਾਗ ਨੂੰ ਅੱਖਰ **E** (ਅਪਰਕੇਸ ਜਾਂ ਲੋਅਰਕੇਸ) ਦੁਆਰਾ ਵੱਖ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: ਇੱਕ ਨੰਬਰ 231.54 ਨੂੰ 2.3154**e**2 (2.3154E102) ਨਾਲ ਐਕਸਪੋਨੈਂਟ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਇੱਥੇ 2.3154 ਮੈਨਟੀਸਾ ਭਾਗ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਅਤੇ ਅੱਖਰ ਦਾ **e** ਤੋਂ ਬਾਅਦ ਵਾਲਾ ਹਿੱਸਾ 2 ਐਕਸਪੋਨੈਂਟ ਭਾਗ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਜਿਸਦਾ ਅਧਾਰ ਮੁੱਲ 10 ਹੁੰਦਾ ਹੈ। ਐਕਸਪੋਨੈਂਟ ਰੂਪ

ਬਹੁਤ ਵੱਡੀਆਂ ਸੰਖਿਆਵਾਂ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਉਪਯੋਗੀ ਹੁੰਦਾ ਹੈ ਕਿਉਂਕਿ ਜਦੋਂ ਕਿਸੇ ਵੱਡੀ ਸੰਖਿਆ ਨੂੰ ਇਸ ਰੂਪ ਵਿਚ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਉਸ ਸੰਖਿਆ ਨਾਲ ਲਿਖੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਜ਼ੀਰੋਆਂ ਦੀ ਵੱਡੀ ਗਿਣਤੀ ਨੂੰ ਲਿਖਣ ਤੋਂ ਬਚਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: 6000000 ਨੂੰ 6.0e6 ਵਜੋਂ ਲਿਖਿਆ ਜਾ ਸਕਦਾ ਹੈ।

ਅ. ਕਰੈਕਟਰ ਲਿਟਰਲਜ਼ (Character Literals) : ਇਹ ਸਿੰਗਲ ਕਰੈਕਟਰ ਮੁੱਲ ਹੁੰਦੇ ਹਨ ਜੋ ਆਮ ਤੌਰ 'ਤੇ ਗਣਨਾਵਾਂ ਵਿੱਚ ਸ਼ਾਮਲ ਨਹੀਂ ਹੁੰਦੇ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਨੂੰ ਸਿੰਗਲ ਕੋਮਿਆ (single quotes) ਵਿੱਚ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਵਿੱਚ ਇੱਕ ਤੋਂ ਵੱਧ ਪ੍ਰਿੰਟੇਬਲ ਕਰੈਕਟਰਾਂ ਨੂੰ ਰੱਖਣ ਦੀ ਆਗਿਆ ਨਹੀਂ ਹੁੰਦੀ।ਨਾਨ-ਪ੍ਰਿੰਟੇਬਲ ਕਰੈਕਟਰ ਵੀ ਇਸ ਸ਼੍ਰੇਣੀ ਵਿੱਚ ਆਉਂਦੇ ਹਨ ਹਾਲਾਂਕਿ ਇਨ੍ਹਾਂ ਕਰੈਕਟਰਾਂ ਵਿੱਚ ਦੋ ਚਿੰਨ ਵਰਤੇ ਜਾਂਦੇ ਹਨ, ਉਦਾਹਰਣ ਵਜੋਂ ਨਵਾਂ ਲਾਈਨ ਅੱਖਰ (\n - ਬੈਕਸਲੈਸ਼ (/) ਅਤੇ ਇੱਕ ਅੱਖਰ n), ਪਰ ਫਿਰ ਵੀ ਉਹਨਾਂ ਨੂੰ ਇੱਕ ਕਰੈਕਟਰ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਇੱਕਹਿਰੇ ਕਰੈਕਟਰ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਕੁੱਝ ਉਦਾਹਰਣ ਇਸ ਪ੍ਰਕਾਰ ਹਨ: 'A', 'g', '‘', +, $, \n, \t, ਆਦਿ। ਮੁੱਲ 'ab', '45' ਆਦਿ ਇੱਕਹਿਰੇ ਕਰੈਕਟਰ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਸਹੀ ਉਦਾਹਰਣਾਂ ਨਹੀਂ ਹਨ ਕਿਉਂਕਿ ਇਹਨਾਂ ਵਿੱਚ ਇੱਕ ਤੋਂ ਵੱਧ ਅੱਖਰਾਂ ਨੂੰ ਸਿੰਗਲ ਕਾਮਿਆਂ ਵਿੱਚ ਰੱਖਿਆ ਗਿਆ ਹੈ ਜਿਸ ਦੀ ਇੱਕਹਿਰੇ ਕਰੈਕਟਰ ਲਿਟਰਲਜ਼ ਵਿੱਚ ਆਗਿਆ ਨਹੀਂ ਹੁੰਦੀ।

ੲ. ਸਟਿੰਗ ਲਿਟਰਲਜ਼ (String Literals) : ਇਹ ਲਿਟਰਲਜ਼ ਕਿਸੇ ਵੀ ਗਿਣਤੀ ਵਿਚ ਕਰੈਕਟਰਾਂ ਦੇ ਸਮੂਹ(combination of any number of characters) ਨੂੰ ਦੋਹਰੇ ਕਾਮਿਆਂ (double quotes) ਵਿੱਚ ਰੱਖ ਕੇ ਲਿਖੇ ਜਾ ਸਕਦੇ ਹਨ। ਇਹ ਲਿਟਰਲਜ਼ ਅੰਗਰੇਜ਼ੀ ਦੇ ਅੱਖਰਾਂ, ਅੰਕਾਂ, ਖਾਸ ਚਿੰਨ੍ਹਾਂ ਅਤੇ ਖਾਲੀ ਥਾਵਾਂ ਆਦਿ ਦੇ ਸਮੂਹ ਨਾਲ ਬਣੇ ਹੋ ਸਕਦੇ ਹਨ। ਇਹਨਾਂ ਲਿਟਰਲਜ਼ ਦੀਆਂ ਕੁੱਝ ਉਦਾਹਰਣਾਂ ਇਸ ਪ੍ਰਕਾਰ ਹਨ: "Paramveer", "A", "House#196", "1829" ਆਦਿ।

ਸ. ਬੂਲੀਅਨ ਲਿਟਰਲਜ਼ (Boolen Literals) : ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ true ਅਤੇ false ਮੁੱਲਾਂ ਨੂੰ ਵੀ ਲਿਟਲ ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ। ਜਦੋਂ ਅਸੀਂ ਕਿਸੇ ਬੂਲੀਅਨ ਟਾਈਪ ਦੇ ਵੇਰੀਏਬਲ ਲਈ ਕੋਈ ਮੁੱਲ ਸੈੱਟ ਕਰਨਾ ਹੋਵੇ ਤਾਂ ਅਸੀਂ ਸਿਰਫ ਇਹਨਾਂ ਦੋ ਮੁੱਲਾਂ ਦੀ ਹੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ। C ਭਾਸ਼ਾ ਤੋਂ ਉਲਟ ਅਸੀਂ ਜਾਵਾ ਵਿਚ ਇਹ ਨਹੀਂ ਮੰਨ ਸਕਦੇ ਕਿ 1 ਦਾ ਮੁੱਲ true ਦੇ ਬਰਾਬਰ ਹੈ ਅਤੇ 0 false ਦੇ ਬਰਾਬਰ ਹੈ। ਬੂਲੀਅਨ ਮੁੱਲ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਸਾਨੂੰ ture ਅਤੇ false ਮੁੱਲਾਂ ਦੀ ਹੀ ਵਰਤੋਂ ਕਰਨੀ ਪਵੇਗੀ। ਉਦਾਹਰਣ ਲਈ:

```
Boolean flag = true;
```

4.8.2.4 ਆਪਰੇਟਰਜ਼ (Operators) :

ਆਪਰੇਟਰ ਉਹ ਚਿੰਨ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿ ਕੁੱਝ ਗਣਿਤਕ ਜਾਂ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ ॥ ਉਦਾਹਰਣ ਲਈ: +,-,#, %, ++, -- ਆਦਿ। ਆਪਰੇਟਰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਮੁੱਲਾਂ ਵੇਰੀਏਬਲਾਂ ਉੱਪਰ ਕੰਮ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਇਹਨਾਂ ਮੁੱਲਾਂ/ਵੇਰੀਏਬਲਾਂ ਨੂੰ **ਓਪਰੈਂਡਜ਼** (operands) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਸੀ ਭਾਸ਼ਾ ਵਿੱਚ ਬਹੁਤ ਸਾਰੇ ਆਪਰੇਟਰਜ਼ ਪਹਿਲਾਂ ਤੋਂ ਹੀ ਬਣੇ ਹੋਏ ਹਨ। ਇਹਨਾਂ ਸਾਰੇ ਓਪਰੇਟਰਾਂ ਨੂੰ ਤਿੰਨ ਵਿਆਪਕ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ: ਯੂਨਰੀ (unary), ਬਾਈਨਰੀ (binary) ਅਤੇ ਟਰਨਰੀ (Ternary). ਜਾਵਾ ਵਿਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਓਪਰੇਟਰਾਂ ਦੀ ਵਿਸਤਾਰ ਵਿਚ ਵਿਆਖਿਆ ਅਗਲੇ ਪਾਠ ਵਿਚ ਕੀਤੀ ਗਈ ਹੈ।

4.8.2.5 ਪੰਕਚੁਏਟਰਜ਼ (Punctuators):

ਇਹਨਾਂ ਚਿੰਨ੍ਹਾਂ ਨੂੰ ਵਿਸ਼ੇਸ਼ ਚਿੰਨ੍ਹ ਵਜੋਂ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਹਰੇਕ ਚਿੰਨ੍ਹਾਂ ਦੀ ਆਪਣੀ ਵੱਖਰੀ ਵਿਸ਼ੇਸ਼ਤਾ ਹੁੰਦੀ ਹੈ ॥ਹਰ ਇੱਕ ਚਿੰਨ੍ਹਾਂ ਦੀ ਵਰਤੋਂ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੁੱਝ ਖਾਸ ਕੰਮ ਦਰਸਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਉਦਾਹਰਣ ਦੇ ਤੌਰ ਤੇ: ਸੈਮੀਕਾਲਨ; ਚਿੰਨ੍ਹ ਦੀ ਵਰਤੋਂ ਸਟੇਟਮੈਂਟ ਨੂੰ ਖਤਮ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਕਾਮੇ (,) ਨੂੰ ਸੈਪਰੇਟਰ ਵਜੋਂ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ, ਫੰਕਸ਼ਨਾਂ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਗੋਲ ਬਰੈਕਟ (()), ਐਰੇ ਲਈ ਚਕੋਰ ਬਰੈਕਟ ([]) ਸਟੇਟਮੈਂਟਸ ਨੂੰ ਗਰੁੱਪ ਕਰਨ ਲਈ ਘੁੰਡੀਦਾਰ ਬਰੈਕਟ ({}) ਵਰਤੇ ਜਾਂਦੇ ਹਨ ॥

4.8.3 ਕਮੈਂਟਸ (Comments) :

ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਸਿਰਫ਼ ਡਾਕੂਮੈਂਟੇਸ਼ਨ (documentation) ਦੇ ਮੰਤਵ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ॥ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਪ੍ਰੋਗਰਾਮ ਦੇ ਅੰਦਰ ਕੋਡ ਦਾ ਵਰਣਨ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜੇਕਰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਤਾਂ ਕੋਡ ਨੂੰ ਸਮਝਣਾ ਆਸਾਨ ਹੋ ਜਾਂਦਾ ਹੈ। ਕੰਪਾਈਲਰ ਕੰਪਾਈਲੇਸ਼ਨ ਦੌਰਾਨ ਕਮੈਂਟਸ ਨੂੰ ਨਜ਼ਰਅੰਦਾਜ਼ (ignores) ਕਰਦਾ ਹੈ। ਕਮੈਂਟਸ ਲਈ ਜਾਵਾ ਵਿੱਚ ਹੇਠ ਲਿਖੇ ਸਟਾਈਲਜ਼ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ:

- **ਬਲਾਕ ਸਟਾਈਲ ਕਮੈਂਟਸ (Block style comments)** ਦੀ ਵਰਤੋਂ ਕਈ ਲਾਈਨਾਂ ਵਾਲੇ ਕਮੈਂਟਸ (Multiple Line comments) ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਕਮੈਂਟਸ /* ਨਾਲ ਸ਼ੁਰੂ ਹੁੰਦੇ ਹਨ ਅਤੇ */ ਨਾਲ ਖਤਮ ਹੁੰਦੇ ਹਨ।
- **ਲਾਈਨ ਸਟਾਈਲ ਕਮੈਂਟਸ (Line style comments)** ਦੀ ਵਰਤੋਂ ਇਕ ਸਿੰਗਲ ਲਾਈਨ ਕਮੈਂਟ (single line comment) ਬਨਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ।ਇਹ ਕਮੈਂਟ // ਚਿੰਨ੍ਹ ਨਾਲ ਸ਼ੁਰੂ ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਆਬਜੈਕਟਸ ਅਸਲ ਸੰਸਾਰ ਦੀਆਂ ਵਸਤੂਆਂ (entities) ਹੁੰਦੀਆਂ ਹਨ ਜਿਵੇਂ ਕਿ: ਵਿਦਿਆਰਥੀ, ਵਿਅਕਤੀ, ਪੈਨ, ਟੇਬਲ, ਟੈਲੀਵਿਜ਼ਨ, ਕੰਪਿਊਟਰ ਆਦਿ।
2. ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (paradigm) ਜਿੱਥੇ ਹਰ ਚੀਜ਼ ਨੂੰ ਇੱਕ ਆਬਜੈਕਟ ਦੇ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਟਰੂਲੀ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਭਾਸ਼ਾ *(Truly Object-Oriented Progamming Languge)* ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
3. OOP ਵਿੱਚ ਕੋਡ ਅਤੇ ਡਾਟਾ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਇਕਾਈ - ਆਬਜੈਕਟ (object) ਵਿਚ ਇੱਕਠਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
4. ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (paradigm) ਦੇ ਚਾਰ ਮੁੱਖ ਸਿਧਾਂਤ ਹੁੰਦੇ ਹਨ: ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction), ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation), ਇਹੈਰੀਟੈਂਸ (Interitance) ਅਤੇ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism)
5. ਇੱਕ ਕਲਾਸ ਉਹਨਾਂ ਆਬਜੈਕਟਸ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦੀ ਹੈ, ਜਿਹਨਾਂ ਵਿੱਚ ਇਕ ਸਮਾਨ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (same properties) ਅਤੇ ਸਾਂਝਾ ਵਿਵਹਾਰ (common behaviour) ਹੁੰਦਾ ਹੈ॥
6. ਐਬਸਟਰੈਕਸ਼ਨ ਸਿਧਾਂਤ ਇਹ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ ਕਿ ਪ੍ਰੋਗਰਾਮਿੰਗ ਐਂਟੀਟੀ ਵਿੱਚ ਇੱਕ ਅਸਲ ਸੰਸਾਰ ਐਂਟੀਟੀ (real world entity) ਨੂੰ ਕਿਵੇਂ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।
7. ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਡਾਟਾ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਸਮੇਟਣ (wrapping) ਦੀ ਇੱਕ ਪ੍ਰਕਿਰਿਆ ਹੈ। ਇਸ ਸਿੰਗਲ ਯੂਨਿਟ ਨੂੰ ਕਲਾਸ (class) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
8. ਐਨਕੈਪਸੂਲੇਸ਼ਨ ਸਾਡੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਛੁਪਾ ਕੇ (selective hiding of properties and methods) ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਲਪੇਟ (wrapping) ਕੇ ਰੱਖਦੀ ਹੈ, ਜਿਸਨੂੰ ਕਲਾਸ (class) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
9. ਇਨਹੈਰੀਟੈਂਸ ਇੱਕ ਅਜਿਹੀ ਵਿਧੀ ਹੈ ਜਿਸ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਆਪਣੇ ਪੇਰੈਂਟ (parent) ਆਬਜੈਕਟ ਦੀਆਂ ਸਾਰੀਆਂ ਅਵਸਥਾਵਾਂ ਅਤੇ ਵਿਵਹਾਰਾਂ (states and behaviours) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ।
10. ਇੱਕ ਆਬਜੈਕਟ ਦੇ ਕਈ ਰੂਪਾਂ ਨੂੰ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
11. ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ ਦੋ ਕਿਸਮ ਦੇ ਪੋਲੀਮੋਫਿਜ਼ਮ ਹਨ: ਕੰਪਾਈਲ-ਟਾਈਮ ਪੋਲੀਮੋਫਿਜ਼ਮ ਅਤੇ ਰਨ-ਟਾਈਮ ਪੋਲੀਮੋਫਿਜ਼ਮ।
12. ਕੰਪਾਈਲ ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (method overriding) ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਾਸਿਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
13. ਰਨ-ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (method overriding) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰਾਪਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
14. Java ਨੂੰ ਸਾਲ 1995 ਵਿੱਚ ਸਨ ਮਾਇਕ੍ਰੋਸਿਸਟਮਜ਼ (Sun Microsystems) ਕੰਪਨੀ ਵਿੱਚ ਜੇਮਸ ਗੋਸਲਿੰਗ (James Gosling) ਦੁਆਰਾ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਸੀ।
15. ਜਦੋਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਕਿਸੇ ਵਿਸ਼ੇਸ਼ ਪਲੇਟਫਾਰਮ ਮਸ਼ੀਨ (platform specific machine) ਵਿੱਚ ਕੰਪਾਇਲ ਨਹੀਂ ਹੁੰਦਾ। ਇਸ ਦੀ ਬਜਾਏ ਇਸ ਨੂੰ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਬਾਈਟ-ਕੋਡ (platform independent bytecode) ਵਿੱਚ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

16. JDK JAVA ਐਪਲੀਕੇਸ਼ਨਾਂ ਬਣਾਉਣ ਲਈ ਇੱਕ ਪ੍ਰਮੁੱਖ ਪਲੇਟਫਾਰਮ ਕੰਪੋਨੈਂਟ ਹੈ।

17. JRE ਵਿੱਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੀਆਂ ਜਾਵਾ ਲਾਇਬ੍ਰੇਰੀਆਂ ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ।

18. JVM ਇੱਕ ਐਬਸਟਰੈਕਟ (abstract) ਮਸ਼ੀਨ ਹੁੰਦੀ ਹੈ ॥ਇਹ ਇੱਕ ਸਪੈਸੀਫੀਕੇਸ਼ਨ (specification) ਹੈ।ਜੋ ਇਕ ਅਜਿਹਾ ਰਨਟਾਈਮ ਵਾਤਾਵਰਣ (runtime environment) ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ।ਜਿਸ ਵਿੱਚ ਜਾਵਾ ਬਾਈਟਕੋਡ ਨੂੰ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

19. ਜਾਵਾ ਐਪਲੀਕੇਸ਼ਨਾਂ ਨੂੰ WORA (Write Once Run Anywhere) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

20. ਜਾਵਾ ਭਾਸ਼ਾ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ ਹੈ।ਇਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਜਾਵਾ ਦਾ ਪ੍ਰੋਗਰਾਮ ਇੱਕ ਕਿਸਮ ਦੀ ਮਸ਼ੀਨ ਤੋਂ ਦੂਜੀ ਕਿਸਮ ਦੀ ਮਸ਼ੀਨ ਉਪਰ ਆਸਾਨੀ ਨਾਲ ਪ੍ਰਯੋਗ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

21. ਟੋਕਨ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਭ ਤੋਂ ਛੋਟੀਆਂ ਵਿਅਕਤੀਗਤ ਇਕਾਈਆਂ (smallest individual units) ਹੁੰਦੀਆਂ ਹਨ।

22. ਕੀਵਰਡਜ਼ ਨੂੰ ਰਿਜ਼ਰਵ ਵਰਡਜ਼ (Reserve Words) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਸ਼ਬਦ ਜਾਵਾ ਕੰਪਾਈਲਰ ਵਿੱਚ ਪਹਿਲਾਂ ਤੋਂ ਪਰਿਭਾਸ਼ਿਤ ਹੁੰਦੇ ਹਨ।

23. ਆਈਡੈਂਟੀਫਾਇਰ ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਲੀਮੈਂਟ ਤੱਤ ਜਿਵੇਂ ਕਿ: ਵੇਰੀਏਬਲ, ਕਾਂਸਟੈਂਟਸ, ਐਰੇ, ਮੈਥਡਜ਼, ਕਲਾਸਾਂ, ਇੰਟਰਫੇਸ, ਆਦਿ ਨੂੰ ਦਿੱਤੇ ਗਏ ਨਾਮ ਹੁੰਦੇ ਹਨ।

24. ਲਿਟਰਲਜ਼ ਨੂੰ ਸਥਿਰ ਮੁੱਲ (Constant Values) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।ਇਹ ਨਿਸ਼ਚਿਤ (Fixed) ਮੁੱਲ ਹੁੰਦੇ ਹਨ ਜੋ ਆਮ ਤੌਰ ਤੇ ਪ੍ਰੋਗਰਾਮ ਦੇ ਤੱਤਾਂ ਲਈ ਨਿਰਧਾਰਤ ਕੀਤੇ ਜਾਂਦੇ ਹਨ।

25. ਆਪਰੇਟਰ ਉਹ ਚਿੰਨ੍ਹ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿ ਕੁੱਝ ਗਣਿਤਕ ਜਾਂ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

26. ਕਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਪ੍ਰੋਗਰਾਮ ਦੇ ਅੰਦਰ ਕੋਡ ਦਾ ਵਰਣਨ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਕੰਪਾਈਲਰ ਕੰਪਾਈਲੇਸ਼ਨ ਦੌਰਾਨ ਕਮੈਂਟਸ ਨੂੰ ਨਜ਼ਰਅੰਦਾਜ਼ (ignores) ਕਰਦਾ ਹੈ।

ਅਭਿਆਸ

ਪ੍ਰਸ਼ਨ 1 ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ:

i. OOP ਵਿੱਚ ਕੋਡ ਅਤੇ ਡਾਟਾ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਇਕਾਈ................ ਵਿਚ ਇੱਕਠਾ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਪ੍ਰੋਗਰਾਮ (Program) ਅ. ਕਲਾਸ (class)

ੲ. ਆਬਜੈਕਟ (object) ਸ. ਯੂਨਿਟ (Unit)

ii. ....ਸਿਧਾਂਤ ਇਹ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ ਕਿ ਇਕ ਪ੍ਰੋਗਰਾਮਿੰਗ ਐਂਟੀਟੀ ਵਿੱਚ ਇੱਕ ਅਸਲ ਸੰਸਾਰ ਐਂਟੀਟੀ (real world entity) ਨੂੰ ਕਿਵੇਂ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

ੳ. ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) ਅ. ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation)

ੲ. ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਸ. ਇਹੈਰੀਟੈਂਸ (Inheritance)

iii. ....ਇੱਕ ਅਜਿਹੀ ਵਿਧੀ ਹੈ ਜਿਸ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਆਪਣੇ ਪੇਰੈਂਟ (parent) ਆਬਜੈਕਟ ਦੀਆਂ ਸਾਰੀਆਂ ਅਵਸਥਾਵਾਂ ਅਤੇ ਵਿਵਹਾਰਾਂ (states and behaviours) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ।

ੳ. ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) ਅ. ਐਨਕੈਪਸੂਲੇਸ਼ਨ (Encapsulation)

ੲ. ਪੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਸ. ਇਨਹੈਰੀਟੈਂਸ (Inheritance)

iv. .... ਪੋਲੀਮੋਰਫਿਜ਼ਮ ਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading) ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਾਸਿਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਰਨ-ਟਾਈਮ (Run-Time) ਅ. ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਟਾਈਮ (Execution Time)

ੲ. ਕੰਪਾਈਲ ਟਾਈਮ (Compile-Time) ਸ. ਲੋਡਿੰਗ ਟਾਈਮ (Loading independent)

v. ਜਦੋਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਸ ਨੂੰ ਪਲੇਟਫਾਰਮ ਸੁਤੰਤਰ (platfrom independent) .... ਵਿੱਚ ਕੰਪਾਇਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਸੋਰਸ ਕੋਡ (Source Code) ਅ. ਆਬਜੈਕਟ ਕੋਡ (Object Code)

ੲ. ਬਿੱਟ ਕੋਡ (Bit Code) ਸ. ਬਾਈਟ-ਕੋਡ (Byte Code)

vi. .... ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਭ ਤੋਂ ਛੋਟੀਆਂ ਵਿਅਕਤੀਗਤ ਇਕਾਈਆਂ (smallest individual units) ਹੁੰਦੀਆਂ ਹਨ।

ੳ. ਆਬਜੈਕਟਸ (Objects) ਅ. ਟੋਕਨਜ਼ (Tokens)

ੲ. ਲਿਟਰਲਜ਼ (Literals) ਸ. ਕਲਾਸਾਂ (Classes)

vii. ....ਉਹ ਚਿੰਨ੍ਹ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿ ਕੁੱਝ ਗਣਿਤਕ ਜਾਂ ਲਾਜ਼ੀਕਲ ਓਪਰੇਸ਼ਨ ਕਰਨ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

ੳ. ਅੰਕ (Digits) ਅ. ਓਪਰੈਂਡਜ਼ (Operands)

ੲ. ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (Expressions) ਸ. ਆਪਰੇਟਰਜ਼ (Operators)

2. ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ:

i. .... ਸਾਡੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਪ੍ਰਾਪਰਟੀਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਛੁਪਾ ਕੇ (selective hiding of properties and methods) ਇੱਕ ਸਿੰਗਲ ਯੂਨਿਟ ਵਿੱਚ ਲਪੇਟ (wrapping) ਕੇ ਰੱਖਦੀ ਹੈ, ਜਿਸਨੂੰ ਕਲਾਸ (class) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ii. ਇੱਕ ਆਬਜੈਕਟ ਦੇ ਕਈ ਰੂਪਾਂ ਨੂੰ .... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

iii. JAVA ਨੂੰ ਸਾਲ 1995 ਵਿੱਚ ਸਨ ਮਾਇਕ੍ਰੋਸਿਸਟਮਜ਼ (Sun Microsystems) ਕੰਪਨੀ ਵਿੱਚ .... ਦੁਆਰਾ ਵਿਕਸਤ ਕੀਤਾ ਗਿਆ ਸੀ।

iv. .... ਵਿੱਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਚਲਾਉਣ ਲਈ ਲੋੜੀਂਦੀਆਂ ਜਾਵਾ ਲਾਇਬ੍ਰੇਰੀਆਂ ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ।

v. .... ਇੱਕ ਐਬਸਟਰੈਕਟ (abstract) ਮਸ਼ੀਨ ਹੁੰਦੀ ਹੈ। ਇਹ ਇੱਕ ਸਪੈਸੀਫੀਕੇਸ਼ਨ (specification) ਹੈ ਜੋ ਇਕ ਅਜਿਹਾ ਰਨਟਾਈਮ ਵਾਤਾਵਰਣ (runtime environment) ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਜਾਵਾ ਬਾਈਟਕੋਡ ਨੂੰ ਚਲਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

vi. ....ਪ੍ਰੋਗਰਾਮ ਦੇ ਐਲੀਮੈਂਟ (ਤੱਤ) ਜਿਵੇਂ ਕਿ: ਵੇਰੀਏਬਲ, ਕਾਂਸਟੈਂਟਸ, ਐਰੇ, ਮੈਥਡਜ਼, ਕਲਾਸਾਂ, ਇੰਟਰਫੇਸ, ਆਦਿ ਨੂੰ ਦਿੱਤੇ ਗਏ ਨਾਮ ਹੁੰਦੇ ਹਨ।

vii. ਕੰਪਾਈਲਰ ਕੰਪਾਈਲੇਸ਼ਨ ਦੌਰਾਨ...................... ਨੂੰ ਨਜ਼ਰਅੰਦਾਜ਼ (ingores) ਕਰਦਾ ਹੈ।

3. ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਆਬਜੈਕਟ ਓਰੀਐਂਟਿਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (Object-Oriented Programming Paradigm) ਦੇ ਚਾਰ ਮੁੱਖ ਸਿਧਾਂਤ ਕਿਹੜੇ ਹਨ ?

ii. ਆਬਜੈਕਟ (Object) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।

iii. ਇਨਹੈਰੀਟੈਂਸ (Inheritance) ਕੀ ਹੈ ?

iv. ਇਨਕੈਪਸੁਲੇਸ਼ਨ (Encapsultion) ਕੀ ਹੈ ?

v. ਤੁਸੀਂ ਜਾਵਾ ਬਾਰੇ ਕੀ ਜਾਣਦੇ ਹੋ ?

vi. JVM ਕੀ ਹੈ ?

vii. ਕੀਵਰਡਜ਼ (Keywords) ਕੀ ਹੁੰਦੇ ਹਨ ?

viii. ਕਮੈਂਟਸ (Comments) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।

ix. ਆਪਰੇਟਰਜ਼ (Operators) ਕੀ ਹਨ ?

x. ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਨੂੰ ਪਲੇਟਫਾਰਮ-ਸੁਤੰਤਰ (Platfrom Independence) ਵਜੋਂ ਕਿਉਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ ?

4 ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਪ੍ਰੋਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism) ਕੀ ਹੈ ? ਪੋਲੀਮੈਰਫਿਜ਼ਮ ਦੀਆਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਨਾਂ ਲਿਖੋ।

ii. ਜਾਵਾ ਦੀਆਂ ਕੋਈ ਪੰਜ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦਾ ਵਰਨਣ ਕਰੋ।

iii. ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੀ ਮੁੱਢਲੀ ਬਣਤਰ ਲਿਖੋ।।

iv. ਟੋਕਨਜ਼ (Tokens) ਕੀ ਹੁੰਦੇ ਹਨ ? ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਟੋਕਨਜ਼ ਸੰਬੰਧੀ ਜਾਣਕਾਰੀ ਦਿਓ।

v. ਆਈਡੈਂਟੀਫਾਇਰ (Identifier) ਕੀ ਹੁੰਦੇ ਹਨ ? ਆਈਡੈਂਟੀਫਾਇਰਜ਼ ਦੇ ਨਾਮਕਰਨ ਸੰਬੰਧੀ ਨਿਯਮਾਂ (naming rules) ਦਾ ਵਰਨਣ ਕਰੋ।।

vi. ਲਿਟਰਲਜ਼ (Literals) ਕੀ ਹੁੰਦੇ ਹਨ ? ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਲਿਟਰਲਜ਼ ਦਾ ਵਰਨਣ ਕਰੋ।

5 ਜਾਵਾ ਵਿਚ ਡਾਟਾ ਟਾਈਪਸ ਅਤੇ ਆਪਰੇਟਰਜ਼

(Data Types and Operators in Java)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

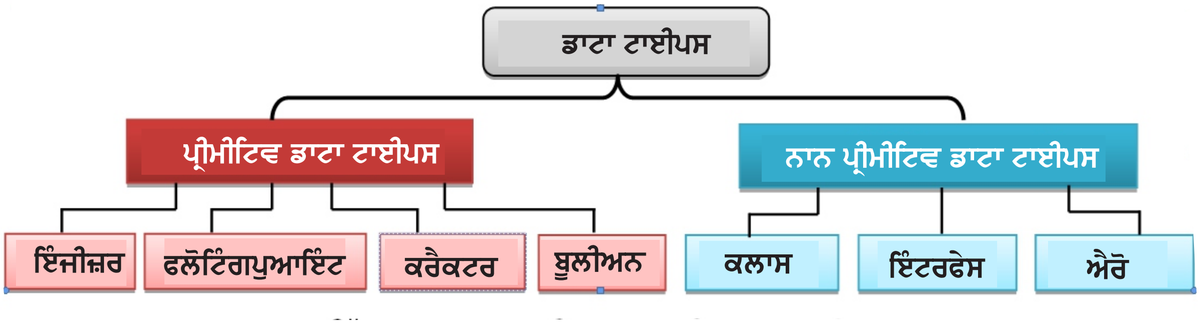
5.1 ਡਾਟਾ ਟਾਈਪਸ (Data Types)
5.2 ਵੇਰੀਏਬਲਜ਼ (Variables)
5.3 ਓਪਰੇਟਰਜ਼ (Operators)
5.4 ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (Expressions)
5.5 ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators)
5.6 ਟਾਈਪ ਕਨਵਰਜ਼ਨ (Type Conversion)

5.1 ਡਾਟਾ ਟਾਈਪਸ (DATA TYPES)

ਡਾਟਾ ਟਾਈਪਸ ਇਹ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦੀਆਂ ਹਨ ਕਿ ਪ੍ਰੋਗਰਾਮ ਦੇ ਤੱਤਾਂ ਵਿੱਚ ਕਿਸ ਕਿਸਮ ਦਾ ਡਾਟਾ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ, ਜਿਵੇਂ ਕਿ ਵੇਰੀਏਬਲ, ਕਾਂਸਟੈਂਟ, ਐਰੇ ਆਦਿ। ਡਾਟਾ ਟਾਈਪਸ ਇੱਕ ਖਾਸ ਕਿਸਮ, ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ ਅਤੇ ਡਾਟਾ ਉਪਰ ਕੀਤੇ ਜਾ ਸਕਣ ਵਾਲੇ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਨਿਰਧਾਰਤ ਕਰਦੀਆਂ ਹਨ। ਜਾਵਾ ਇੱਕ ਸਟ੍ਰਾਂਗਲੀ ਟਾਈਪਡ ਭਾਸ਼ਾ (strongly typed language) ਹੈ ਇਸ ਲਈ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਡਿਕਲੇਰੇਸ਼ਨ (declaration) ਸਮੇਂ ਸਾਰੇ ਵੇਰੀਏਬਲਾਂ ਦੀਆਂ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਡਿਕਲੇਅਰ (ਘੋਸ਼ਿਤ) ਕਰਨ ਲਾਜ਼ਮੀ ਹੁੰਦਾ ਹੈ।

ਜਾਵਾ ਕਈ ਕਿਸਮਾਂ ਦੀਆਂ ਡਾਟਾ ਟਾਈਪਸ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਅਤੇ ਹਰੇਕ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਕੰਪਿਊਟਰ ਦੀ ਮੈਮੋਰੀ ਵਿੱਚ ਵੱਖਰੇ ਢੰਗ ਨਾਲ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਦੁਆਰਾ ਪ੍ਰਦਾਨ ਕੀਤੀਆਂ ਗਈਆਂ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਮੁੱਖ ਤੌਰ 'ਤੇ ਦੋ ਕਿਸਮਾਂ ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ:

- ❖ ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Primitive Data Types)
- ❖ ਨਾਨ- ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Non-Primitive Data Types)



ਚਿੱਤਰ 5.1: ਜਾਵਾ ਵਿੱਚ ਡਾਟਾ ਟਾਈਪਸ ਦਾ ਵਰਗੀਕਰਨ

5.1.1 ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Primitive Data Types) :

ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਬਿਲਟ-ਇਨ (built-in) ਡਾਟਾ ਟਾਈਪ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਚਾਰ ਕਿਸਮਾਂ ਦੀਆਂ ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਹਨ :

I. ਇੰਟੀਜ਼ਰ ***(Integer):*** byte, short, int, ਅਤੇ long

II. ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ***(Floating Point:*** float ਅਤੇ double)

III. ਕਰੈਕਟਰ ***(Character):*** char

IV. ਬੁਲੀਅਨ ***(Boolean):*** boolean

ਇਹਨਾਂ ਡਾਟਾ ਟਾਈਪਸ ਦਾ ਵਿਸਥਾਰ ਵਿੱਚ ਵਰਣਨ ਇਸ ਪ੍ਰਕਾਰ ਹੈ:

I. ਇੰਟੀਜ਼ਰ ਟਾਈਪ (Integer Type): ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਪੂਰਨ ਅੰਕ ਮੁੱਲਾਂ (ਬਿਨ੍ਹਾਂ ਫ੍ਰੈਕਸ਼ਨਲ (fractional) ਭਾਗ ਵਾਲੇ ਅੰਕ) ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਿਵੇਂ ਕਿ:49, 174, +2901, 54896 ਆਦਿ।JAVA ਚਾਰ ਕਿਸਮਾਂ ਦੀਆਂ ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਸਪੋਰਟ ਕਰਦਾ ਹੈ: byte, short int. ਅਤੇ long ਟਾਈਪਸ। ਇਹਨਾਂ ਸਾਰੀਆਂ ਡਾਟਾ ਟਾਈਪਸ ਨਾਲ ਸਕਾਰਾਤਮਕ ਜਾਂ ਨਕਾਰਾਤਮਕ ਚਿੰਨ੍ਹ (positive or negative sign) ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਅਨਸਾਈਂਡ ਇੰਟੀਜ਼ਰ (unsigned integer) ਦੀ ਕੋਈ ਧਾਰਨਾ ਨਹੀਂ ਹੈ। ਇਹਨਾਂ ਡਾਟਾ ਟਾਈਪਸ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ 0 ਹੁੰਦਾ ਹੈ। ਇਹਨਾਂ ਟਾਈਪਸ ਦੀ ਸਟੋਰੇਜ ਸਮਰੱਥਾ ਵੱਖ-ਵੱਖ ਹੁੰਦੀ ਹੈ। ਹੇਠਾਂ ਦਿੱਤਾ ਟੇਬਲ ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪਸ ਲਈ ਲੋੜੀਂਦੀਆਂ ਮੈਮੋਰੀ ਜ਼ਰੂਰਤਾਂ (memory requirements) ਅਤੇ ਉਹਨਾਂ ਵਿਚ ਸਟੋਰੇ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ਼ ਨੂੰ ਦਰਸਾ ਰਿਹਾ ਹੈ:

ਡਾਟਾ ਟਾਈਪ	ਸਟੋਰੇਜ਼ ਸਾਈਜ਼	ਡਾਟਾ ਦੀ ਰੇਂਜ਼
byte	1 byte	-2^7 to $+2^7 - 1$ OR (-127 to +128)
short	2 bytes	-2^{15} to $+2^{15} - 1$ OR (-32, 768 to +32, 767)
int	4 bytes	-2^{31} to $+2^{31} - 1$ OR (-2, 147, 483, 648 to (+2, 147, 483, 647)
long	8 bytes	-2^{63} to $+2^{63} - 1$ OR (-9, 223, 372, 036, 854, 775, 808 to +9, 223, 372, 036, 854, 775, 808)

ਟੇਬਲ 5.1 ਇੰਟੀਜਰ ਡਾਟਾ ਟਾਇਪ ਦਾ ਸੰਖੇਪ ਵਿੱਚ ਜਾਣਕਾਰੀ

ਅਸੀਂ ਆਪਣੀ ਜਰੂਰਤ ਅਨੁਸਾਰ ਪੂਰਨਅੰਕ ਮੁੱਲ (Integer value) ਦੀ ਕਿਸਮ ਅਤੇ ਰੇਂਜ ਦੇ ਅਧਾਰ ਤੇ ਵੇਰੀਏਬਲ ਲਈ ਢੁਕਵੀਂ ਡਾਟਾ ਟਾਈਪ ਦੀ ਚੋਣ ਕਰ ਸਕਦੇ ਹਾਂ ॥

II. ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਟਾਈਪ (Floating-Point Type) : ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਡਾਟਾਟਾਈਪ ਦੀ ਵਰਤੋਂ ਰੀਅਲ ਨੰਬਰਾਂ ਜਿਵੇਂ ਕਿ: 3.14, 74,5884569, +29.05, -524836.458 ਆਦਿ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।(Java) ਦੋ ਤਰ੍ਹਾਂ ਦੇ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਸਪੋਰਟ ਕਰਦੀ ਹੈ: float ਅਤੇ double ਟਾਈਪਸ। ਹੇਠਾਂ ਦਿੱਤਾ ਟੇਬਲ ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਟਾਈਪਸ ਲਈ ਲੋੜੀਂਦੀਆਂ ਮੈਮੋਰੀ ਜ਼ਰੂਰਤਾਂ (Memory Requirement) ਅਤੇ ਉਹਨਾਂ ਵਿਚ ਸਟੋਰੇ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ਼ ਨੂੰ ਦਰਸਾ ਰਿਹਾ ਹੈ:

ਡਾਟਾ ਟਾਈਪ	ਸਟੋਰੇਜ਼ ਸਾਈਜ਼	ਡਾਟਾ ਦੀ ਰੇਂਜ਼
float	4 bytes	3. 4E-38 to 3. 4E+38
double	8 bytes	1.7E-308 to 1. 7E+308

ਟੇਬਲ 5.2 ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਡਾਟਾ ਦੀ ਸੰਖੇਪ ਵਿੱਚ ਜਾਣਕਾਰੀ

float ਡਾਟਾ ਟਾਈਪ ਇੱਕ ਸਿੰਗਲ-ਪ੍ਰੀਸੀਜ਼ਨ (single-precision) ਨੰਬਰ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ ਜਦੋਂ ਕਿ double ਟਾਈਪ ਡਬਲ-ਪ੍ਰੀਸੀਜ਼ਨ (double-precision) ਨੰਬਰ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ। ਸਿੰਗਲ ਪ੍ਰੀਸੀਜ਼ਨ ਨੰਬਰ ਡਬਲ ਪ੍ਰੀਸੀਜ਼ਨ ਨਾਲੋਂ ਘੱਟ ਥਾਂ ਘੇਰਦਾ ਹੈ। ਜਦੋਂ ਸਟੋਰ ਕੀਤਾ ਜਾਣ ਵਾਲਾ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਮੁੱਲ ਜ਼ਿਆਦਾ ਵੱਡਾ ਹੋਵੇ ਤਾਂ ਉਸ ਨੂੰ ਸਿੰਗਲ- ਪ੍ਰੀਸੀਜ਼ਨ ਟਾਈਪ ਵਿਚ ਸਟੋਰ ਕਰਵਾਉਣ ਨਾਲ ਉਸ ਮੁੱਲ ਦੀ ਸ਼ੁੱਧਤਾ ਘੱਟ ਜਾਂਦੀ ਹੈ। ਇਸ ਲਈ ਜਦੋਂ ਸਾਨੂੰ ਵੱਡੇ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਮੁੱਲਾਂ ਨੂੰ ਸਟੋਰ ਕਰਨ ਦੀ ਲੋੜ ਪਵੇ ਤਾਂ ਡਬਲ ਟਾਈਪ ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਸਭ ਤੋਂ ਵਧੀਆ ਆਪਸ਼ਨ ਰਹੇਗਾ। ਉਦਾਹਰਨ ਲਈ: ਜੇਕਰ ਅਸੀਂ ਵਿਦਿਆਰਥੀਆਂ ਦੁਆਰਾ ਪ੍ਰਾਪਤ ਅੰਕਾਂ ਦੇ ਮੁੱਲ ਨੂੰ ਸਟੋਰ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ, ਤਾਂ ਫਲੋਟ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਇਸੇ ਤਰ੍ਹਾਂ, ਜਦੋਂ ਅਸੀਂ ਗਣਿਤਿਕ ਫੰਕਸ਼ਨਾਂ ਜਿਵੇਂ ਕਿ sin(), cos(), sqrt (),/ ਆਦਿ ਨਾਲ ਕੰਮ ਕਰ ਰਹੇ ਹੁੰਦੇ ਹਾਂ, ਤਾਂ ਸਾਨੂੰ ਡਬਲ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਕਰਨੀ ਚਾਹੀਦੀ ਹੈ। ਫਲੋਟ ਡਾਟਾ ਟਾਈਪ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ 0.0f ਹੁੰਦਾ ਹੈ ਅਤੇ ਡਬਲ ਡਾਟਾ ਟਾਈਪ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ 0.0d ਹੁੰਦਾ ਹੈ।

III. ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ (Character Data Type) : ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਸਿੰਗਲ ਕੋਟ (') ਵਿੱਚ ਬੰਦ ਇੱਕ ਸਿੰਗਲ ਯੂਨੀਕੋਡ ਅੱਖਰ (single Unicode character enclosed in single quotes) ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਦਰਸਾਉਣ ਲਈ char ਕੀਵਰਡ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ 2 ਬਾਈਟ (16 ਬਿੱਟ) ਮੈਮੋਰੀ ਸਪੇਸ ਲੈਂਦਾ ਹੈ। char ਡਾਟਾ ਟਾਈਪ ਦੀ ਰੇਂਜ 0 ਤੋਂ 65535 ਹੁੰਦੀ ਹੈ। char ਡਾਟਾ ਟਾਈਪ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ '/u0000' ਹੁੰਦਾ ਹੈ।

ਡਾਟਾ ਟਾਈਪ	ਸਟੋਰੇਜ਼ ਸਾਈਜ਼	ਡਾਟਾ ਦੀ ਰੇਂਜ਼
char	2 bytes	0 to 65535

ਟੇਬਲ 5.3 ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਸੰਖੇਪ ਵਿੱਚ ਵਿਆਖਿਆ

IV. ਬੁਲੀਅਨ ਡਾਟਾ ਟਾਈਪ Boolean Data Type : ਬੁਲੀਅਨ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਵੇਰੀਏਬਲਾਂ ਵਿਚ ਬੁਲੀਅਨ ਮੁੱਲਾਂ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਡਾਟਾ ਟਾਈਪ ਸਿਰਫ਼ ਦੋ ਮੁੱਲਾਂ ਨੂੰ ਸਵੀਕਾਰ ਕਰਦਾ ਹੈ: true ਜਾਂ false ਮੁੱਲ। ਕੀਵਰਡ (boolean) ਦੀ ਵਰਤੋਂ ਬੁਲੀਅਨ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ 1-ਬਿੱਟ ਜਾਣਕਾਰੀ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਪਰ ਇਸਦਾ ਮੈਮਰੀ ਆਕਾਰ ਸਹੀ ਰੂਪ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ। ਬੁਲੀਅਨ ਡਾਟਾ ਟਾਈਪ ਦਾ ਡਿਫਾਲਟ ਮੁੱਲ false ਹੁੰਦਾ ਹੈ। ।

5.1.2 ਨਾਨ-ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Non-Primitive Data Types) : ਨਾਨ-ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਯੂਜ਼ਰ ਦੁਆਰਾ ਪਰਿਭਾਸ਼ਿਤ User defined ਡਾਟਾ ਟਾਈਪਸ ਜਾਂ ਰੈਫਰੈਂਸ Raference ਡਾਟਾ ਟਾਈਪਸ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਡਾਟਾ ਟਾਈਪਸ ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਤੋਂ ਹੀ ਤਿਆਰ ਕੀਤੀਆਂ ਗਈਆਂ ਹੁੰਦੀਆਂ ਹਨ। ਕਲਾਸਾਂ, ਇੰਟਰਫੇਸ ਅਤੇ ਐਰੇ ਜਾਵਾ ਵਿੱਚ ਨਾਨ-ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਦੀਆਂ ਉਦਾਹਰਣਾਂ ਹਨ।

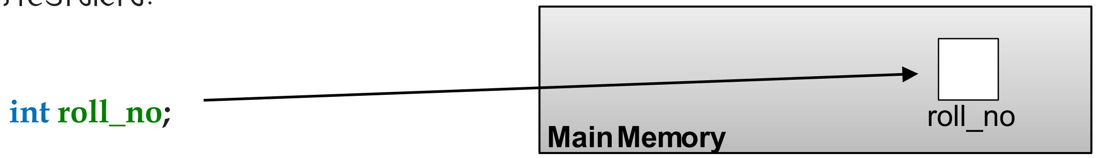
5.2 ਵੇਰੀਏਬਲਜ਼ (VARIABLES)

ਵੇਰੀਏਬਲ ਇੱਕ ਆਈਡੈਂਟੀਫਾਇਰ ਹੁੰਦਾ ਹੈ ਜੋ ਇੱਕ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ (memory location) ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ਇਹ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ ਡਾਟਾ/ਮੁੱਲ ਸਟੋਰ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਲੋਕੇਸ਼ਨਾਂ ਵਿੱਚ ਸਟੋਰ ਕੀਤੇ ਡਾਟਾ/ਮੁੱਲ ਨੂੰ ਵੇਰੀਏਬਲ ਦੇ ਨਾਮ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਐਕਸੈੱਸ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਦੀ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਸਮੇਂ (execution time) ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਹਰੇਕ ਵੇਰੀਏਬਲ ਦੀ ਇੱਕ ਡਾਟਾ ਟਾਈਪ ਹੁੰਦੀ ਹੈ, ਜੋ ਵੇਰੀਏਬਲ ਦੇ ਮੈਮਰੀ ਆਕਾਰ ਅਤੇ ਉਸ ਵਿਚ ਸਟੋਰ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਮੁੱਲ ਦੀ ਕਿਸਮ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ। ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵੇਰੀਏਬਲਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਲਈ ਉਹਨਾਂ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਡਿਕਲੇਅਰ/ਘੋਸ਼ਿਤ (declare) ਕਰਨਾ ਲਾਜ਼ਮੀ ਹੁੰਦਾ ਹੈ।

5.2.1 ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ (Variable Declaration) : ਜਾਵਾ ਇੱਕ ਸਟ੍ਰਾਂਗਲੀ ਟਾਈਪਡ ਭਾਸ਼ਾ (Stgrongly Typed Language) ਹੈ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੁੰਦਾ ਹੈ ਕਿ ਸਾਨੂੰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵੇਰੀਏਬਲਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ ਉਹਨਾਂ ਨੂੰ ਡਿਕਲੇਅਰ/ਘੋਸ਼ਿਤ ਕਰਨਾ ਲਾਜ਼ਮੀ ਹੁੰਦਾ ਹੈ। ਜੇਕਰ ਅਸੀਂ ਵੇਰੀਏਬਲਾਂ ਨੂੰ ਡਿਕਲੇਅਰ ਕੀਤੇ ਬਿਨਾਂ ਵਰਤਦੇ ਹਾਂ, ਤਾਂ ਕੰਪਾਈਲਰ ਸਿੰਟੈਕਸ ਐਰਰ (Syntax Error) ਦਿਖਾਵੇਗਾ। ਇੱਕ ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਵਿੱਚ ਦੋ ਭਾਗ ਹੁੰਦੇ ਹਨ: ਡਾਟਾ ਟਾਈਪ ਅਤੇ ਵੇਰੀਏਬਲ ਦਾ ਨਾਮ (ਆਈਡੈਂਟੀਫਾਇਰ)। ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵੇਰੀਏਬਲ ਡਿਕਲੇਅਰ ਕਰਨ ਲਈ ਹੇਠਾਂ ਦਿੱਤੇ ਸਿੰਟੈਕਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ:

data_type variable_name;

ਇੱਥੇ data-name ਕੰਪਾਈਲਰ ਨੂੰ ਇਹ ਦੱਸਦਾ ਹੈ ਕਿ ਵੇਰੀਏਬਲ ਵਿੱਚ ਕਿਸ ਕਿਸਮ ਦਾ ਮੁੱਲ ਸਟੋਰ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ, ਅਤੇ (variable-name) ਇੱਕ ਵੈਧ ਆਈਡੈਂਟੀਫਾਇਰ (valid indentifier) ਹੈ ਜੋ ਕੰਪਾਈਲਰ ਨੂੰ ਵੇਰੀਏਬਲ ਦੇ ਨਾਮ ਬਾਰੇ ਦੱਸਦਾ ਹੈ ਅਤੇ ਜਿਸਦੀ ਵਰਤੋਂ ਵੇਰੀਏਬਲ ਵਿੱਚ ਸਟੋਰ ਕੀਤੇ ਮੁੱਲ ਨੂੰ ਵਰਤਣ ਸਮੇ ਕੀਤੀ ਜਾਵੇਗੀ। ਹੇਠਾਂ ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਦੀ ਇੱਕ ਉਦਾਹਰਨ ਦਿੱਤੀ ਗਈ ਹੈ:



ਚਿੱਤਰ : 5.2 ਵੇਰੀਏਬਲ ਦੀ ਮੈਮਰੀ ਐਲੋਕੇਸ਼ਨ

ਇਸ ਉਦਾਹਰਣ ਵਿਚ int ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਜੋ ਆਈਡੈਂਟੀਫਾਇਰ roll_no ਨੂੰ 4 ਬਾਈਟ ਮੈਮੋਰੀ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਇਹ ਆਈਡੈਂਟੀਫਾਇਰ ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਦੀ ਵਰਤੋਂ ਸਮੇਂ ਉਸ ਵਿੱਚ ਸਟੋਰ ਮੁੱਲ ਨੂੰ ਐਕਸੈਸ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਵੇਗਾ। ਇਸ ਉਦਾਹਰਣ ਅਨੁਸਾਰ ਵੇਰੀਏਬਲ roll_no ਸਿਰਫ਼ ਪੂਰਨ ਅੰਕ ਮੁੱਲ ਹੀ ਸਟੋਰ ਕਰ ਸਕਦਾ ਹੈ। ਅਸੀਂ ਕਾਮੇ ਸੇਪਰੇਟਰ (comma separator) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇੱਕੋ ਡਾਟਾ ਟਾਈਪ ਵਾਲੇ ਕਈ ਵੇਰੀਏਬਲ ਵੀ ਡਿਕਲੇਅਰ ਕਰ ਸਕਦੇ ਹਾਂ। ਉਦਾਹਰਣ ਲਈ:

```
int a, b, c;
```

5.2.2 ਵੇਰੀਏਬਲ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ੇਸ਼ਨ (Variable Initialization): ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਵਿਚ ਉਸ ਨੂੰ ਸਿਰਫ ਮੈਮੋਰੀ ਐਲੋਕੇਟ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਉਸ ਵਿਚ ਕੋਈ ਡਾਟਾ ਸਟੋਰ ਨਹੀਂ ਕੀਤਾ ਜਾਂਦਾ। ਅਸੀਂ ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਸਮੇਂ ਉਸ ਨੂੰ ਇੱਕ ਮੁੱਲ ਵੀ ਅਸਾਈਨ (assign) ਕਰ ਸਕਦੇ ਹਾਂ, ਜਿਸਨੂੰ ਵੇਰੀਏਬਲ ਇਨੀਸ਼ੀਅਲਾਈਜੇਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ:

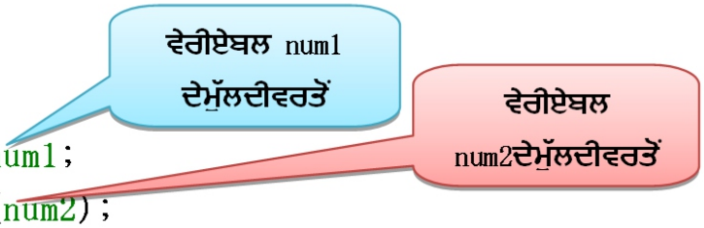
```
int roll_no=5;
```

ਚਿੱਤਰ : 5.3 ਵੇਰੀਏਬਲ ਵਿੱਚ ਮੁੱਲ ਸਟੋਰ ਕਰਨਾ

ਇਸ ਅਸਾਈਨ ਕੀਤੇ ਮੁੱਲ ਨੂੰ ਬਾਅਦ ਵਿੱਚ ਕਿਸੇ ਵੀ ਸਮੇਂ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ ਕਿਉਂਕਿ ਇੱਕ ਵੇਰੀਏਬਲ ਸਾਨੂੰ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੌਰਾਨ (during execution) ਕਿਸੇ ਵੀ ਸਮੇਂ ਇਸਦੇ ਮੁੱਲ ਨੂੰ ਬਦਲਣ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦਾ ਹੈ।

5.2.3 ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਦੀ ਵਰਤੋਂ ਕਰਨਾ (Accessing Value of a Variable) : ਵੇਰੀਏਬਲ ਵਿੱਚ ਸਟੋਰ ਮੁੱਲ ਨੂੰ ਐਕਸੈਸ ਕਰਨ/ਵਰਤਣ ਲਈ ਸਾਨੂੰ ਵੇਰੀਏਬਲ ਦਾ ਨਾਮ ਵਰਤਣਾ ਪੈਂਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ

```
int num1=5;
int num2 = num1 + num1;
System.out.println(num2);
```



ਉਪਰੋਕਤ ਉਦਾਹਰਣ ਵਿੱਚ ਅਸੀਂ ਦੋ ਵੇਰੀਏਬਲ num1 ਅਤੇ num2 ਨੂੰ ਡਿਕਲੇਅਰ ਅਤੇ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ ਕੀਤਾ ਹੈ। num1 ਵੇਰੀਏਬਲ ਦਾ ਮੁੱਲ num 2 ਦੇ ਮੁੱਲ ਦੀ ਗਣਨਾ (int num2 = num1 + num1) ਕਰਨ ਲਈ ਵਰਤਿਆ ਗਿਆ ਹੈ।।num2 ਦਾ ਮੁੱਲ ਦਿਖਾਉਣ ਲਈ ਅਸੀਂ ਜਾਵਾ ਦੇ println () ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹਾਂ।

5.3 ਆਪਰੇਟਰਜ਼ (OPERATORS)

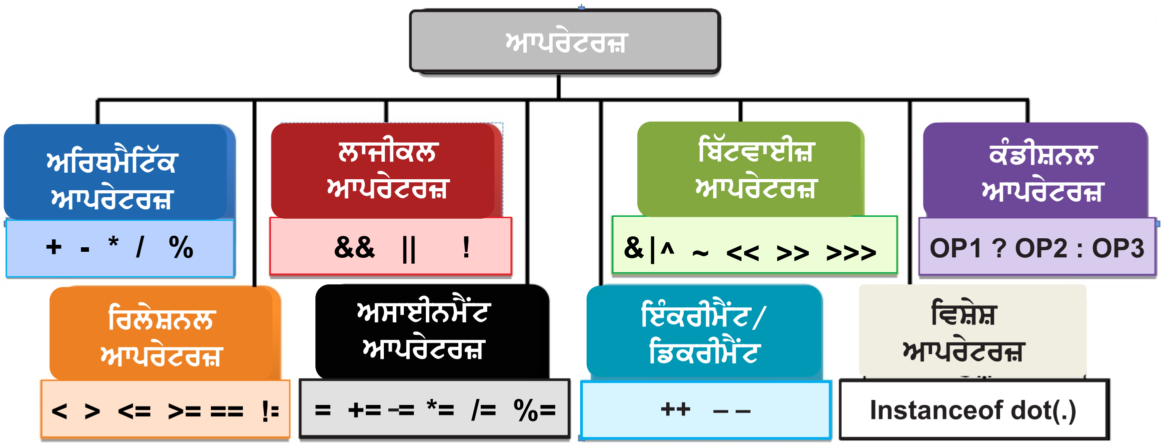
ਆਪਰੇਟਰਜ਼ ਉਹ ਚਿੰਨ੍ਹ ਹੁੰਦੇ ਹਨ ਜਿਨ੍ਹਾਂ ਦੀ ਵਰਤੋਂ ਡਾਟਾ ਉਪਰ ਕਿਸੇ ਖਾਸ ਆਪਰੇਸ਼ਨ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਉਦਾਹਰਣ ਲਈ: + ਚਿੰਨ੍ਹ ਦੀ ਵਰਤੋਂ ਜੋੜ ਕਰਨ ਲਈ, * ਚਿੰਨ੍ਹ ਦੀ ਵਰਤੋਂ ਗੁਣਾ ਕਰਨ ਲਈ, >= ਦੀ ਵਰਤੋਂ ਤੁਲਨਾ ਕਰਨ ਲਈ। ਇਹਨਾਂ ਉਦਾਹਰਣਾਂ ਵਿਚ +, *, >= ਆਪਰੇਟਰਜ਼ ਹਨ ਜੋ ਕਿ ਵੱਖ-ਵੱਖ ਆਪਰੇਸ਼ਨਾਂ (ਕੰਮਾਂ) ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਸਾਰੇ ਆਪਰੇਟਰਜ਼ ਆਪਣਾ ਕੰਮ ਕਰਨ ਤੋਂ ਬਾਅਦ ਇਕ ਮੁੱਲ ਵਾਪਿਸ ਕਰਦੇ ਹਨ।

ਕਿਸੇ ਵੀ ਤਰ੍ਹਾਂ ਦੇ ਕੰਮ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਸਾਨੂੰ ਆਪਰੈਂਡਜ਼ ਦੀ ਜ਼ਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਆਪਰੈਂਡਜ਼ ਉਹ ਡਾਟਾ ਆਈਟਮਾਂ ਹੁੰਦੀਆਂ ਹਨ

ਜਿਨ੍ਹਾਂ ਉਪਰ ਆਪਰੇਟਰਜ਼ ਆਪਣਾ ਕੰਮ ਕਰ ਸਕਦੇ ਹੁੰਦੇ ਹਨ। ਇਹ ਆਪਰੈਂਡ ਵੇਰੀਏਬਲਜ਼ ਜਾਂ ਕਾਂਸਟੈਂਟ ਮੁੱਲ ਕੁੱਝ ਵੀ ਹੋ ਸਕਦੇ ਹਨ। ਉਦਾਹਰਣ ਲਈ:

a + 5 * 10

ਇਸ ਉਦਾਹਰਣ ਵਿਚ + ਅਤੇ * ਆਪਰੇਟਰਜ਼ ਹਨ ਜੋ ਆਪਣਾ ਕੰਮ ਵੇਰੀਏਬਲ 'a' ਕਾਂਸਟੈਂਟ ਮੁੱਲ 5 ਅਤੇ 10 ਉਪਰ ਕਰ ਰਹੇ ਹਨ। ਇਸ ਵਿਚ 'a', 5 ਅਤੇ 10 ਨੂੰ ਆਪਰੈਂਡ ਕਿਹਾ ਜਾਵੇਗਾ। ਓਪਰੇਟਰਾਂ ਅਤੇ ਓਪਰੇਡਾਂ ਦੇ ਇੱਕ ਵੈਲੀਡ ਸਮੂਹ (valid combination) ਨੂੰ ਐਸਪ੍ਰੈਸ਼ਨ/ਸਮੀਕਰਨ (Expression) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਆਪਰੇਟਰਾਂ ਦੁਆਰਾ ਕੀਤੇ ਜਾਂਦੇ ਫੰਕਸ਼ਨਾਂ ਦੇ ਆਧਾਰ 'ਤੇ ਜਾਵਾ ਆਪਰੇਟਰਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ: ਅਰਿਥਮੈਟਿੱਕ (Arithmetic) ਆਪਰੇਟਰਜ਼, ਰਿਲੇਸ਼ਨਲ (Relational) ਆਪਰੇਟਰਜ਼, ਲਾਜ਼ੀਕਲ (Logical) ਆਪਰੇਟਰਜ਼, ਅਸਾਈਨਮੈਂਟ (Assignment) ਆਪਰੇਟਰਜ਼, ਬਿਟਵਾਈਜ਼ (Bitwise) ਆਪਰੇਟਰਜ਼, ਇਨਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ (Increment and Decrement) ਆਪਰੇਟਰਜ਼, ਕੰਡੀਸ਼ਨਲ (Conditional) ਆਪਰੇਟਰਜ਼ ਅਤੇ ਸਪੈਸ਼ਲ (Special) ਆਪਰੇਟਰਜ਼।



ਚਿੱਤਰ 5.4 ਜਾਵਾ ਵਿੱਚ ਆਪਰੇਟਰਜ਼ ਦੀਆਂ ਕਿਸਮਾਂ

5.3.1 ਅਰਿਥਮੈਟਿੱਕ ਆਪਰੇਟਰਜ਼ (Arithmetic Operators) :

ਅਰਿਥਮੈਟਿੱਕ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਅਰਥਮੈਟਿਕ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕਿਤੀ ਜਾਂਦੀ ਹੈ, ਜਿਵੇਂ ਕਿ: ਜੋੜ, ਘਟਾਓ, ਗੁਣਾ ਭਾਗ ਆਦਿ। ਜਾਵਾ ਭਾਸ਼ਾ ਵਿਚ 5 ਅਰਥਮੈਟਿਕ ਆਪਰੇਟਰ ਹਨ। ਇਹ ਸਾਰੇ ਆਪਰੇਟਰਜ਼ ਬਾਈਨਰੀ ਕਿਸਮ ਦੇ ਆਪਰੇਟਰ ਹਨ ਕਿਉਂਕਿ ਇਹਨਾਂ ਸਾਰੇ ਆਪਰੇਟਰਾਂ ਨੂੰ ਆਪਣਾ ਕੰਮ ਕਰਨ ਲਈ ਦੋ ਆਪਰੈਂਡਜ਼ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਹੇਠਾਂ ਦਿਤਾ ਟੇਬਲ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੇ ਕੰਮ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ:

ਨਾਮ	ਆਪਰੇਟਰ	ਵਿਆਖਿਆ	ਆਪਰੇਟਰ ਦੀ ਉਦਾਹਰਣ	
			ਇੰਟੀਜ਼ਰ ਮੁੱਲ	ਰੀਅਲ ਮੁੱਲ
ਐਡ (Add)	+	ਜੋੜ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।	2+4→6	2.0+4.0 →6.0
ਸਬਟ੍ਰੈਕਟ (Subtract)	-	ਘਟਾਓ ਕਰਨ ਲਈ ਜਾਂ ਯੂਨਰੀ ਮਾਈਨਟਸ ਵਜੋਂ ਵਰਤਿਆ ਜਾਂਦਾ ਹਹੈ।	6-2→4	6.0-4.0 → 2.0
ਮਲਟੀਪਲਾਈ (Multiply)	*	ਗੁਣਾ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।	7*2→ 14	7.0*2.0 → 14.0
ਡਿਵਾਈਡ (Divide)	/	ਭਾਗ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।	5/2→ 2	5.0/2.0 →2.5
ਮਾਡੂਲਸ (Modulus)	%	ਭਾਗ ਕਰਨ ਤੋਂ ਬਾਅਦ ਬਾਕੀ ਬਚੇ ਮੁੱਲ (ਸ਼ੇਸ਼ਫਲ) ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।	7%4→3	5.0%2.0 → 1.0

ਟੇਬਲ 5.4 ਅਰਿੱਥਮੈਟਿਕ ਆਪ੍ਰੇਟਰਜ਼

ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿਚ ਅਰਿਥਮੈਟਿੱਕ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ:

ਪ੍ਰੋਗਰਾਮ 5.1: ਜਾਵਾ ਵਿਚ ਅਰਿਥਮੈਟਿੱਕ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ

Test1.java

```
class Test1
{
    public static void main(String args[])
    {
        float num1=5.0f, num2=2.0f, result;
        result=num1+num2;
        System.out.println("Result of Addition is "+result);
        result=num1-num2;
        System.out.println("Result of Subtraction is "+result);
        result=num1*num2;
        System.out.println("Result of Multiplication is "+result);
        result=num1/num2;
        System.out.println("Result of Division is "+result);
        result=num1%num2;
        System.out.println("Result of Modulus is "+result);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 5.1 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Exeuction) ਅਤੇ ਆਊਟਪੁੱਟ:

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac Test1.java

D:\Java Programs>java Test1
Result of Addition is 7.0
Result of Subtraction is 3.0
Result of Multiplication is 10.0
Result of Division is 2.5
Result of Modulus is 1.0

D:\Java Programs>
```

ਪ੍ਰੋਗਰਾਮ 5.1 ਵਿੱਚ ਅਸੀਂ ਫਲੋਟਿੰਗ ਟਾਈਪ ਦੇ ਤਿੰਨ ਵੇਰੀਏਬਲ ਡਿਕਲੇਅਰ ਕੀਤੇ ਹਨ: num1, num2 ਅਤੇ result ਵੇਰੀਏਬਲਜ਼। ਵੇਰੀਏਬਲ num1 ਅਤੇ num2 ਨੂੰ ਕ੍ਰਮਵਾਰ ਫਲੋਟ ਮੁੱਲ 5.0f ਅਤੇ 2.0f ਨਾਲ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ ਕੀਤਾ ਹੈ। ਫਿਰ ਸਾਰੇ ਅਰਥਮੈਟਿੱਕ ਆਪਰੇਟਰਾਂ ਦੀ ਗਣਨਾ ਦਾ ਨਤੀਜਾ result ਵੇਰੀਏਬਲ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਹੈ ਜਿਸ ਨੂੰ ਜਾਵਾ ਦੇ System.out.println( ) ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਪ੍ਰਦਰਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੈ। ਅਸੀਂ ਇਹਨਾਂ ਆਪਰੇਟਰਾਂ ਨੂੰ ਇੰਟੀਜ਼ਰ ਅਤੇ ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਆਪਰੇਂਡਾਂ 'ਤੇ ਲਾਗੂ ਕਰ ਸਕਦੇ ਹਾਂ।

5.3.2 ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ (Relational Operators)

ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਤੁਲਨਾਤਮਕ (comparison) ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਆਪਰੈਂਡਜ਼ ਵਿਚਕਾਰ ਰਿਲੇਸ਼ਨਸ਼ਿਪ ਟੈਸਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਦੂਸਰੇ ਸ਼ਬਦਾਂ ਵਿਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਮੁੱਲਾਂ ਦੀ ਤੁਲਨਾ (compare) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਮੁੱਲਾਂ ਦੀ ਤੁਲਨਾ ਕਰਨ ਤੋਂ ਬਾਅਦ ਇਹ ਆਪਰੇਟਰ ਸਾਨੂੰ true ਜਾਂ false ਮੁੱਲ ਵਾਪਿਸ ਕਰਦੇ ਹਨ। ਜਾਵਾ ਭਾਸ਼ਾ ਵਿਚ ਇਹ ਸਾਰੇ ਆਪਰੇਟਰਜ਼ ਬਾਈਨਰੀ ਕਿਸਮ ਦੇ ਆਪਰੇਟਰਜ਼ ਹਨ। ਭਾਵ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਕੰਮ ਕਰਨ ਲਈ ਦੋ ਆਪਰੈਂਡਜ਼ ਦੀ ਜ਼ਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਜਾਵਾ ਵਿਚ ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ ਦੀ ਗਿਣਤੀ 6 ਹੈ ਜੋ ਕਿ ਹੇਠਾਂ ਟੇਬਲ ਵਿਚ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਦਿੱਤੇ ਗਏ ਹਨ:

ਨਾਂ	ਆਪਰੇਟਰ	ਵਿਆਖਿਆ	ਉਦਾਹਰਣ	ਨਤੀਜਾ
ਬਰਾਬਰ ਨਹੀਂ (Equals to)	==	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਦੋ ਮੁੱਲ ਬਰਾਬਰ ਹਨ	4==5 5==5	False True
ਬਰਾਬਰ ਨਹੀਂ ਹੈ (Not Equals to)	!=	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਦੋ ਮੁੱਲ ਬਰਾਬਰ ਨਹੀਂ ਹੈ	4! =5 4! =4	True False
ਵੱਡਾ ਹੈ (Greater than)	>	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਪਹਿਲਾ ਮੁੱਲ ਦੂਜੇ ਤੋਂ ਛੋਟਾ ਹੈ	4>5 5>4	False True
ਛੋਟਾ ਹੈ (Less than)	<	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਪਹਿਲਾ ਮੁੱਲ ਦੂਜੇ ਤੋਂ ਵੱਡਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ।	4<5 5<4	True False
ਵੱਡਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ (Greater than or equal to)	>=	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਪਹਿਲੇ ਮੁੱਲ ਦੂਜੇ ਤੋਂ ਵੱਡਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ।	5>=5 6>=8 10>=5	True False True
ਛੋਟਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ (Less than or equal to)	<=	ਇਹਨਾਂ ਦੀ ਵਰਤੋਂ ਇਹ ਪਤਾ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕੀ ਪਹਿਲਾ ਮੁੱਲ ਦੂਜੇ ਤੋਂ ਛੋਟਾ ਹੈ ਜਾਂ ਬਰਾਬਰ ਹੈ।	4<=5 4<=2 4<=4	True False True

ਟੇਬਲ 5.5 ਜਾਵਾ ਵਿੱਚ ਰਿਲੇਸ਼ਨਸ

ਪ੍ਰੋਗਰਾਮ 5.2 : ਜਾਵਾ ਵਿੱਚ ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ

Test2.java

```
class Test2
{
    public static void main(String args[])
    {
        int num1=5, num2=2;
        boolean result;
        System.out.println("Value of num1 is "+num1);
        System.out.println("Value of num2 is "+num2);
        result=num1==num2;
        System.out.println("Result of num1==num2 is "+result);
        result=num1!=num2;
        System.out.println("Result of num1!=num2 is "+result);
        result=num1>num2;
        System.out.println("Result of num1>num2 is "+result);
        result=num1<num2;
        System.out.println("Result of num1<num2 is "+result);
        result=num1>=num2;
        System.out.println("Result of num1>=num2 is "+result);
        result=num1<=num2;
        System.out.println("Result of num1<=num2 is "+result);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 5.2 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਉਸ਼ਨ (Execution) ਅਤੇ ਆਉਟਪੁੱਟ: ←on) ਅਤੇ ਆਉਟਪੁੱਟ:

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac Test2.java

D:\Java Programs>java Test2
Value of num1 is 5
Value of num2 is 2
Result of num1==num2 is false
Result of num1!=num2 is true
Result of num1>num2 is true
Result of num1<num2 is false
Result of num1>=num2 is true
Result of num1<=num2 is false

D:\Java Programs>_
```

ਆਮ ਤੌਰ ’ਤੇ ਰਿਲੇਸ਼ਨਲ ਓਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਉਹਨਾਂ ਟੈਸਟ ਐਕਸਪੈਸ਼ਨਜ਼ ਨੂੰ ਬਨਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜਿਹਨਾਂ ਦੀ ਵਰਤੋਂ ਬ੍ਰਾਂਚਿੰਗ, ਲੂਪਿੰਗ ਅਤੇ ਜੰਪਿੰਗ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟ ਵਿੱਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਨ੍ਹਾਂ ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਿਆਖਿਆ ਇਸ ਪੁਸਤਕ ਦੇ ਅਗਲੇ ਪਾਠ ਵਿਚ ਕੀਤੀ ਗਈ ਹੈ ॥

5.3.3 ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ (Logical Operators): ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਬੂਲੀਅਨ (Boolean) ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਮਿਸ਼ਰਿਤ (compound) ਰਿਲੇਸ਼ਨਲ ਐਕਸਪੈਸ਼ਨਜ਼ ਬਨਾਉਣ ਲਈ ਕਿਤੀ ਜਾਂਦੀ ਹੈ। ਦੂਸਰੇ ਸ਼ਬਦਾਂ ਵਿਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਉਸ ਸਮੇਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਅਸੀਂ ਇਕ ਸਮੇਂ ਵਿਚ ਇਕ ਤੋਂ ਵੱਧ ਟੈਸਟ ਕੰਡੀਸ਼ਨਾਂ ਦੀ ਜਾਂਚ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹੋਈਏ। ਸੀ ਭਾਸ਼ਾ ਵਿਚ 3 ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਮੌਜੂਦ ਹਨ: ‘ਲਾਜੀਕਲ AND, “ਲਾਜੀਕਲ OR” ਅਤੇ ‘ਲਾਜੀਕਲ NOT’, ਇਹਨਾਂ ਵਿਚ “ਲਾਜੀਕਲ AND” ਅਤੇ ‘ਲਾਜੀਕਲ OR’ ਬਾਈਨਰੀ ਆਪਰੇਟਰਜ਼ ਹਨ ਜਦੋਂ ਕਿ ‘ਲਾਜੀਕਲ NOT’ ਯੁਨਰੀ ਆਪਰੇਟਰ ਹੈ। ‘ਲਾਜੀਕਲ AND’ ਅਤੇ ਲਾਜੀਕਲ OR ਆਪਰੇਟਰਜ਼ ਨੂੰ ਕੰਮ ਕਰਨ ਲਈ ਦੋ ਆਪਰੈਂਡਜ਼ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ ਜਦੋਂ ਕਿ NOT ਆਪ੍ਰੇਟਰ ਨੂੰ ਇਕ ਆਪਰੇਂਡ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਸਾਰੇ ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ true ਜਾਂ false ਨਤੀਜਾ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ। ਲਾਜੀਕਲ AND ਆਪ੍ਰੇਟਰ true ਨਤੀਜਾ ਕੇਵਲ ਉਸ ਸਮੇਂ ਦੇਵੇਗਾ ਜਦੋਂ ਇਸਦੇ ਦੋਵੇਂ ਆਪਰੈਂਡਜ਼ true ਹੋਣ, ਇਸੇ ਤਰ੍ਹਾਂ ਲਾਜੀਕਲ OR ਆਪਰੇਟਰ ਦਾ true ਨਤੀਜਾ ਕੇਵਲ ਉਸ ਸਮੇਂ ਆਵੇਗਾ ਜਦੋਂ ਇਸਦਾ ਜਾਂ ਤਾਂ ਕੋਈ ਵੀ ਇਕ ਆਰੇਂਡ true ਹੋਵੇ ਜਾਂ ਦੋਵੇਂ ਆਪਰੈਂਡਜ਼ true ਹੋਣ। ਲਾਜੀਕਲ NOT ਆਪਰੇਟਰ ਕੇਵਲ ਉਸ ਸਮੇਂ true ਨਤੀਜਾ ਦੇਵੇਗਾ ਜੇਕਰ ਇਸਦਾ ਐਕਸਪ੍ਰੈਸ਼ਨ false ਹੋਵੇ। ਹੇਠਾਂ ਦਿੱਤੇ ਟੇਬਲ ਵਿਚ ਜਾਵਾ ਵਿਚ ਮੌਜੂਦ ਸਾਰੇ ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਢੁਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਦਰਸ਼ਾਇਆ ਗਿਆ ਹੈ:

ਨਾਂ	ਆਪਰੇਟਰ	ਵਿਆਖਿਆ	ਉਦਾਹਰਣ	ਵਿਆਖਿਆ ਅਤੇ ਨਤੀਜਾ
AND	&&	ਇਹ ਕੇਵਲ ਉਸ ਸਮੇਂ true ਨਤੀਜਾ ਦੇਵੇਗਾਜ ਦੇ ਦੋਵੇਂ ਅਕਸਪ੍ਰੈਸ਼ਨ true ਹੋਣ ਨਤੀਜਾ ਨਤੀਜਾ false ਆਵੇਗਾ।	3>5 && 4>5 3>5 && 4<5 3<5 && 4>5 3<5 && 4<5	False && False → False False && True → False True && False → False True && True → True
OR	||	ਇਹ ਕੇਵਲ ਉਸ ਸਮੇਂ true ਨਤੀਜਾ ਦੇਵੇਗਾ ਜਦੋਂ ਦੋਵੇਂ ਅਕਸਪ੍ਰੈਸ਼ਨਾਂ ਵਿੱਚੋਂ ਘੱਟ ਤੋਂ ਘੱਟ ਇਕ ਅਕਸਪ੍ਰੈਸ਼ਨ true ਹੋਵੇ ਨਹੀਂ ਤਾਂ ਨਤੀਜਾ false ਆਵੇਗਾ।	3>5 || 4>5 3>5 || 4<5 3<5 || 4>5 3<5 || 4<5	False || False → False False || True → True True || False → True True || True → True
NOT	!	ਇਹ ਕੇਵਲ ਉਸ ਸਮੇਂ true ਨਤੀਜਾ ਦੇਵੇਗਾ ਜਦੋਂ ਇਸਦਾ ਅਕਸਪ੍ਰੈਸ਼ਨ false ਹੋਵੇਗਾ ਨਹੀਂ ਤਾਂ ਨਤੀਜਾ false ਆਵੇਗਾ।	!(3<5) !(3>5)	!(True) → False !(False) → True

ਟੇਬਲ 5.4 ਜਾਵਾ ਵਿੱਚ ਲਾਜੀਕਲ ਆਪ੍ਰੇਟਰਜ਼

ਪ੍ਰੋਗਰਾਮ 5.3 ਜਾਵਾ ਵਿਚ ਲਾਜੀਕਲ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ

Test3.java

```
class Test3
{
    public static void main(String args[])
    {
        int num1=15, num2=10, num3=5;
        boolean result;
        System.out.println("Value of num1 is "+num1);
        System.out.println("Value of num2 is "+num2);
        System.out.println("Value of num3 is "+num3);
        result=num1>num2 && num2<=num3;
        System.out.println("Result of AND (num1>num2 && num2<=num3) is "+result);
        result=num1>num2 || num2<=num3;
        System.out.println("Result of OR (num1>num2 || num2<=num3) is "+result);
        result=!(num1==num2);
        System.out.println("Result of NOT (!(num1==num2)) is "+result);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 5 .3 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਉਟਪੁੱਟ

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac Test3.java

D:\Java Programs>java Test3
Value of num1 is 15
Value of num2 is 10
Value of num3 is 5
Result of AND (num1>num2 && num2<=num3) is false
Result of OR (num1>num2 || num2<=num3) is true
Result of NOT (!(num1==num2)) is true

D:\Java Programs>
```

5.3.4 ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰਜ਼ (Assignment operators):

ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਵੇਰੀਏਬਲ ਵਿਚ ਮੁੱਲ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦਾ ਚਿੰਨ੍ਹ = ਹੁੰਦਾ ਹੈ। ਹੇਠਾਂ ਦਿਤੀਆਂ ਉਦਾਹਰਣਾਂ ਜਾਵਾ ਭਾਸ਼ਾ ਵਿਚ ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦੀਆਂ ਹਨ :

```
int a = -2;          // assigns –ve value (-2) to the variable.
int b = 5;           // assigns value (5) to the variable.
int c = a + b;       // assigns the result of expression to the variable.
int a = a + 10;      // self-assignment of a variable.
```

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੇ ਖੱਬੇ ਪਾਸੇ ਹਮੇਸ਼ਾਂ ਇੱਕ ਵੈਧ ਆਈਡੈਂਟੀਫਾਇਰ (valid Indentifier) ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਜੋ ਇੱਕ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੋਵੇ। ਅਸੀਂ ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੇ ਖੱਬੇ ਪਾਸੇ ਐਕਸਪ੍ਰੈਸ਼ਨ/ਸਮੀਕਰਨ ਨਹੀਂ ਲਿੱਖ ਸਕਦੇ। ਉਦਾਹਰਨ ਲਈ ਨਿਮਨਲਿਖਤ ਅਸਾਈਨਮੈਂਟ ਸਟੇਟਮੈਂਟ ਗਲਤ ਹੋਵੇਗੀ:

```
a +b=c;              //Invalid assignment statement
```

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੀ ਮਦਦ ਨਾਲ ਕਈ ਵੇਰੀਏਬਲਾਂ ਨੂੰ ਇੱਕ ਸਾਂਝਾ ਮੁੱਲ ਵੀ ਨਿਰਧਾਰਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ:

```
a = b = c = 5;       //value 5 will be assigned to three variables a, b and c
```

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਕੰਪਾਊਂਡ ਜਾਂ ਸ਼ਾਰਟਹੈਂਡ ਅਸਾਈਨਮੈਂਟ (Compound or Shorthand Assignment) ਆਪਰੇਟਰਜ਼ ਵਜੋਂ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਸ਼ਾਰਟਹੈਂਡ ਆਪਰੇਟਰਜ਼ ਸੈਲਫ-ਅਸਾਈਨਮੈਂਟ ਸਟੇਟਮੈਂਟਜ਼ ਵਿਚ ਲਾਭਦਾਇਕ ਹੁੰਦੇ ਹਨ। ਹੇਠਾਂ ਦਿੱਤਾ ਟੇਬਲ ਜਾਵਾ ਭਾਸ਼ਾ ਵਿਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਸ਼ਾਰਟਹੈਂਡ ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ:

ਮੰਨ ਲਵੋ int a=5;

ਸ਼ਾਰਟਹੈਂਡ ਆਪਰੇਟਰ	ਉਦਾਹਰਣ	ਵਿਆਖਿਆ	ਨਤੀਜਾ
+=	a+ =2	a = a + 2	a=7
- =	a- =2	a = a – 2	a=3
*=	a* =2	a = a * 2	a=10
/=	a / =2	a = a / 2	a=2
%=	a%=2	a = a % 2	a=1

ਟੇਬਲ 5.7 ਅਰਥਮੈਟਿਕ ਆਪ੍ਰੇਟਰ ਲਈ ਸ਼ਾਰਟਹੈਂਡ ਅਸਾਇਨਮੈਂਟ

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ "=" ਅਤੇ Equals to ਆਪਰੇਟਰ "==" ਦੋਵੇਂ ਵੱਖ-ਵੱਖ ਆਪਰੇਟਰਜ਼ ਹਨ। ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ (=) ਦੀ ਵਰਤੋਂ ਕਿਸੇ variable ਵਿਚ ਮੁੱਲ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਕਿ ਇਕੁਐਲਟੀ (= =) (equality ਭਾਵ equals to) ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਇਹ ਤੈਅ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਦਿੱਤੇ ਗਏ ਆਪਰੈਂਡਜ਼ ਦਾ ਮੁੱਲ ਬਰਾਬਰ ਹੈ ਜਾਂ ਨਹੀਂ। ਇਹ ਦੋਵੇਂ ਆਪਰੇਟਰਜ਼ ਇਕ ਦੂਜੇ ਦੀ ਜਗ੍ਹਾਂ ਨਹੀਂ ਵਰਤੇ ਜਾ ਸਕਦੇ।

5.3.5 ਬਿੱਟਵਾਈਜ਼ ਆਪਰੇਟਰਜ਼ (Bitwise Operators):

ਬਿੱਟਵਾਈਜ਼ ਓਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਬਿੱਟ ਲੇਵਲ ਆਪਰੇਸ਼ਨ ਕਰਵਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹਨਾਂ ਨੂੰ ਕਿਸੇ ਵੀ ਇੰਟੀਜ਼ਰ ਟਾਈਪ (char, short, int, ਆਦਿ) ਮੁੱਲਾਂ ਨਾਲ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਹੇਠਾਂ ਦਿੱਤਾ ਟੇਬਲ ਬਿੱਟਵਾਈਜ਼ ਆਪਰੇਟਰਾਂ ਸੰਬੰਧੀ ਸੰਖੇਪ ਜਾਣਕਾਰੀ ਦਰਸਾ ਰਿਹਾ ਹੈ:

ਮੰਨ ਲਵੋ **int A=60** ਅਤੇ **int B=13** ਤਾਂ :

Operator	Description	Example
& (Bitwise AND)	ਇਹ ਆਪਰੇਅਰ ਇੱਕ ਬਾਈਨਰੀ ਓਪਰੇਟਰ ਹੈ,ਇਹ ਇਨਪੁੱਟ ਮੁੱਲਾਂ ਉੱਪਰ ਬਿੱਟ ਬਾਈ ਬਿੱਟ A©4 ਆਪਰੇਸ਼ਨ ਲਾਗੂ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਦੋਵੇਂ ਬਿੱਟ 1 ਹਨ ਤਾਂ ਇਹ 1 ਦਿੰਦਾ ਹੈ, ਨਹੀ ਤਾਂ ਇਹ 0 ਵਾਪਿਸ ਕਰਨਾ।	(A & B) ਦਾ ਨਤੀਜਾ 12 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 000 1100 ਹੋਵੇਗਾ।
I (Bitwise OR)	ਇਹ ਆਪਰੇਟਰ ਇੱਕ ਬਾਈਨਰੀ ਓਪਰੇਟਰ ਹੈ,ਇਹ ਇਨਪੁੱਟ ਮੁੱਲਾਂ ਉੱਪਰ ਬਿੱਟ ਬਾਈ ਬਿੱਟ OR ਆਪਰੇਸ਼ਨ ਲਾਗੂ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਦੋਵੇਂ ਬਿੱਟ 1 ਹਨ ਤਾਂ ਇਹ 1 ਦਿੰਦਾ ਹੈ, ਨਹੀ ਤਾਂ ਇਹ 0 ਵਾਪਿਸ ਕਰਨਾ।	(A) (B) ਦਾ ਨਤੀਜਾ 61 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 0011 1101 ਹੋਵੇਗਾ।
^ (Bitwise XOR)	ਇਹ ਆਪਰੇਟਰ ਇੱਕ ਬਾਈਨਰੀ ਓਪਰੇਟਰ ਹੈ,ਇਹ ਇਨਪੁੱਟ ਮੁੱਲਾਂ ਉੱਪਰ ਬਿੱਟ ਬਾਈ ਬਿੱਟ XOR ਆਪਰੇਸ਼ਨ ਲਾਗੂ ਕਰਦਾ ਹੈ ਜੇਕਰ ਸੰਬੰਧਿਤ ਬਿੱਟ ਵੱਖਰੇ ਹਨ ਤਾਂ ਇਹ 1 ਦਿੰਦਾ ਹੈ ਨਹੀਂ ਤਾਂ ਇਹ 0 ਵਾਪਿਸ ਕਰੇਗਾ।	(A) (B) ਦਾ ਨਤੀਜਾ 61 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 0011 1101 ਹੋਵੇਗਾ।
~ (Bitwise Compliment)	ਇਹ 1's ਕੰਪਲੀਮੈਂਟ ਯੂਨਰੀ ਆਪਰੇਟਰ ਹੈ ਜੋ ਬਿੱਟਸ ਨੂੰ ਫਲਿਪ ਕਰਦਾ ਹੈ ਭਾਵ 0 ਨੂੰ 1 ਅਤੇ 1 ਨੂੰ ਵਿੱਚ ਤਬਦੀਲ ਕਰਦਾ ਹੈ।	(A^ B) ਦਾ ਨਤੀਜਾ 49 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 0011 0001 ਹੋਵੇਗਾ।
<< (Right Shift)	ਇਹ ਲੈਫਟ ਸ਼ਿਫਟ ਆਪਰੇਟਰ ਹੈ। ਇਸ ਵਿੱਚ ਖੱਬੇ ਪਾਸੇ ਮੌਜੂਦ ਅਪਹੈਂਡ ਦੀਆਂ ਬਿੱਟਸ ਨੂੰ ਸੱਜੇ ਪਾਸੇ ਆਪਹੈਂਡ ਦੁਆਰਾ ਨਿਰਧਾਰਤ ਬਿੱਟਾਂ ਦੀ ਸੰਖਿਆ ਅਨੁਸਾਰ ਖੱਬੇ ਪਾਸੇ ਸ਼ਿਫਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।	(A) ਦਾ ਨਤੀਜਾ 61 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 2's ਕੰਪਲੀਮੈਂਟ ਰੂਪ ਵਿੱਚ 1100 0011 ਹੋਵੇਗਾ (ਕਿਉਂਕਿ ਨਤੀਜਾ signed binary number ਹੈ।)
>> (Right Shift)	ਇਹ ਰਾਈਟ ਸ਼ਿਫਟ ਆਪਰੇਟਰ ਹੈ। ਇਸ ਵਿੱਚ ਖੱਬੇ ਪਾਸੇ ਮੌਜੂਦ ਆਪਹੈਂਡ ਦੀਆਂ ਬਿੱਟਸ ਨੂੰ ਸੱਜੇ ਪਾਸੇ ਆਪਹੈਂਡ ਦੁਆਰਾ ਨਿਰਧਾਰਤ ਬਿੱਟਾਂ ਦੀ ਸੰਖਿਆ ਅਨੁਸਾਰ ਸੱਜੇ ਪਾਸੇ ਸ਼ਿਫਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।	A >> 2 ਦਾ ਨਤੀਜਾ 15 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 0000 1111 ਹੋਵੇਗਾ।
>>> (Zero Fill Right Shift)	ਇਹ ਰਾਈਟ ਸ਼ਿਫਟ ਜ਼ੀਰੋ ਫਿਲ ਓਪਰੇਟਰ ਹੈ। ਇਸ ਵਿੱਚ ਖੱਬੇ ਪਾਸੇ ਮੌਜੂਦ ਆਪਹੈਂਡ ਬਿੱਟਸ ਨੂੰ ਸੱਜੇ ਆਪਹੈਂਡ ਦੁਆਰਾ ਨਿਰਧਾਰਤ ਬਿਟਾਂ ਦੀ ਸੰਖਿਆ ਅਨੁਸਾਰ ਸੱਜੇ ਪਾਸੇ ਸ਼ਿਫਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਸ਼ਿਫਟ ਕੀਤੇ ਮੁੱਲ ਜ਼ੀਰੋ ਨਾਲ ਭਰੇ ਜਾਂਦੇ ਹਨ।	A >>> 2 ਦਾ ਨਤੀਜਾ 15 ਹੋਵੇਗਾ ਜਿਸ ਦਾ ਬਾਈਨਰੀ ਮੁੱਲ 00001111 ਹੋਵੇਗਾ।

ਟੇਬਲ 5.8 ਬਿਟਵਾਇਜ਼ ਆਪ੍ਰੇਟਰਜ਼

5.3.6 ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰਜ਼ (Increment and Decrement Operators):

ਇਹ ਯੂਨਰੀ ਆਪਰੇਟਰਜ਼ ਹਨ। ਚਿੰਨ ++ ਦੀ ਵਰਤੋਂ ਇੰਕਰੀਮੈਂਟ ਆਪਰੇਟਰ ਲਈ ਅਤੇ ਚਿੰਨ -- ਦੀ ਵਰਤੋਂ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰ ਵੱਜੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇੰਕਰੀਮੈਂਟ ਆਪਰੇਟਰ (++) ਆਪਣੇ ਆਪਰੈਂਡ ਦੇ ਮੁੱਲ ਵਿਚ ਇਕ ਨੰਬਰ ਦਾ ਵਾਧਾ ਕਰਦਾ ਹੈ ਜਦੋਂ ਕਿ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰ (- -) ਆਪਣੇ ਆਪਰੈਂਡ ਦੇ ਮੁੱਲ ਵਿਚੋਂ ਇਕ ਨੰਬਰ ਘਟਾਅ ਦਿੰਦਾ ਹੈ। ਇਹਨਾਂ ਆਪਰੇਟਰਾਂ ਨਾਲ ਵਰਤਿਆ ਜਾਣ ਵਾਲਾ ਆਪਰੈਂਡ ਇਕ ਵੇਰੀਏਬਲ ਹੀ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ। ਇਹਨਾਂ ਨੂੰ ਕਿਸੇ ਸਥਿਰ ਮੁੱਲ ਉਪਰ ਸਿੱਧੇ ਲਾਗੂ ਨਹੀਂ ਕਿਹਾ ਜਾ ਸਕਦਾ॥

ਉਦਾਹਰਣ ਲਈ:

int x=10;   (x ਇਕ ਇੰਟੀਜ਼ਰ ਵੇਰੀਏਬਲ ਹੋ ਜਿਸਦਾ ਮੁੱਲ ਹੈ 10)

ਹੇਠਾਂ ਦਿਤੀ ਐਕਸਪੈਸ਼ਨ x ਦੇ ਮੁੱਲ ਨੂੰ ਵਧਾ ਕੇ 11 ਕਰ ਦੇਵੇਗੀ।

++x;　　　　(ਜੋ ਕਿ x= x+1 ਦੇ ਬਰਾਬਰ ਹੈ)

ਇਸੇ ਤਰ੍ਹਾਂ, ਹੇਠਾਂ ਦਿੱਤੀ ਐਕਸਪ੍ਰੈਸ਼ਨ x ਦੇ ਅਸਲੀ ਮੁੱਲ (10) ਨੂੰ ਘਟਾਅ ਕੇ 9 ਕਰ ਦੇਵੇਗੀ।

--x,　　　　(ਜੋ ਕਿ x=x-1 ਦੇ ਬਰਾਬਰ ਹੈ)

ਜੇਕਰ ਅਸੀਂ 10++ ਜਾਂ - -10 ਦੀ ਵਰਤੋਂ ਕਰਾਂਗੇ ਤਾਂ ਇਹ ਗਲਤ ਸਟੇਟਮੈਂਟਸ ਮੰਨੀਆਂ ਜਾਣਗੀਆਂ ਜਿਵੇਂ ਕਿ ਪਹਿਲਾਂ ਹੀ ਦੱਸਿਆ ਜਾ ਚੁੱਕਿਆ ਹੈ ਕਿ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਕਿਸੇ ਸਥਿਰ ਮੁੱਲ ਉਪਰ ਸਿੱਧੇ ਲਾਗੂ ਨਹੀਂ ਕਿਤਾ ਜਾ ਸਕਦਾ।

ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਦੋ ਵੱਖ-ਵੱਖ ਤਰੀਕਿਆਂ ਨਾਲ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਤਰੀਕੇ ਇਸ ਗੱਲ ਉਪਰ ਨਿਰਭਰ ਕਰਦੇ ਹਨ ਕਿ ਆਪਰੇਟਰ ਨੂੰ ਆਪਰੈਂਡ ਤੋਂ ਬਾਅਦ ਲਿਖਿਆ ਗਿਆ ਹੈ ਜਾਂ ਪਹਿਲਾਂ। ਇਹ ਹੇਠ ਲਿਖੇ ਦੋ ਕਿਸਮਾਂ ਦੇ ਹੋ ਸਕਦੇ ਹਨ:

- ਪ੍ਰੀ-ਫਿਕਸ ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ (Prefix increment and decrement)
- ਪੋਸਟ-ਫਿਕਸ ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ (Postfix increment and decrement)

ਜੇਕਰ ਆਪਰੇਟਰ ਨੂੰ ਆਪਰੈਂਡ ਤੋਂ ਪਹਿਲਾਂ ਲਗਾਇਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਆਪਰੈਂਡ ਨੂੰ ਵਰਤਣ ਤੋਂ ਪਹਿਲਾਂ ਉਸਦਾ ਮੁੱਲ ਬਦਲਿਆ ਜਾਵੇਗਾ। ਇਸ ਨੂੰ ਪ੍ਰੀ-ਇੰਕਰੀਮੈਂਟ/ਡਿਕਰੀਮੈਂਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਪਰੰਤ, ਜੇਕਰ ਆਪਰੇਟਰ ਆਪਰੈਂਡ ਤੋਂ ਬਾਅਦ ਲਗਾਇਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਆਪਰੈਂਡ ਦਾ ਮੁੱਲ ਉਸਦੀ ਵਰਤੋਂ ਹੋਣ ਤੋਂ ਬਾਅਦ ਬਦਲਿਆ ਜਾਵੇਗਾ। ਇਸ ਨੂੰ ਪੋਸਟ ਇੰਕਰੀਮੈਂਟ/ਡਿਕਰੀਮੈਂਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।।

ਉਦਾਹਰਨ ਲਈ :

ਜੇਕਰ x ਦਾ ਮੁੱਲ ਸ਼ੁਰੂਆਤ ਵਿਚ 10 ਹੈ ਤਾ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਇਸਦਾ ਮੁੱਲ ਦੋ ਤਰੀਕਿਆਂ ਨਾਲ ਵਧਾਇਆ ਜਾ ਸਕਦਾ ਹੈ:

y = ++x;　　　　(ਪ੍ਰੀ-ਇੰਕਰੀਮੈਂਟ)

ਇਸ ਉਦਾਹਰਣ ਵਿਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ x ਦੇ ਮੁੱਲ ਵਿਚ ਵਾਧਾ ਕਰਕੇ 11 ਹੋ ਜਾਵੇਗਾ, ਉਸ ਤੋਂ ਬਾਅਦ x ਦਾ ਵਧਿਆ ਹੋਇਆ ਮੁੱਲ ਵੇਰੀਏਬਲ y ਵਿਚ ਸਟੋਰ ਕਰ ਦਿਤਾ ਜਾਵੇਗਾ, ਭਾਵ y ਦਾ ਮੁੱਲ ਵੀ 11 ਹੋ ਜਾਵੇਗਾ, (i.e. x =11 ਅਤੇ y =11)

y = x++;　　　　(ਪੋਸਟ-ਇੰਕਰੀਮੈਂਟ)

ਇਸ ਉਦਾਹਰਣ ਵਿਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ x ਦਾ ਮੁੱਲ ਵੇਰੀਏਬਲ y ਵਿਚ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ (ਭਾਵ y ਦਾ ਮੁੱਲ 10 ਹੋ ਜਾਵੇਗਾ) ਇਸ ਤੋਂ ਬਾਅਦ x ਦੇ ਮੁੱਲ ਵਿਚ ਵਾਧਾ ਹੋਣ ਤੋਂ ਬਾਅਦ 11 ਹੋ ਜਾਵੇਗਾ, (i.e. y = 10 ਅਤੇ x = 11)

ਇਸੇ ਤਰ੍ਹਾਂ ਡਿਕਰੀਮੈਂਟ ਆਪਰੇਟਰ ਨੂੰ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ

y = --x;　　　　(ਪ੍ਰੀ-ਇੰਕਰੀਮੈਂਟ)

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ x ਦੇ ਮੁੱਲ ਵਿੱਚੋਂ 1 ਘਟਾਇਆ ਜਾਵੇਗਾ।ਅਤੇ x ਦਾ ਮੁੱਲ 9 ਹੋ ਜਾਵੇਗਾ ਅਤੇ ਬਆਦ ਵਿੱਚ ਇਹ x ਦਾ 9 ਮੁੱਲ y ਮੁੱਲ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ ਅਤੇ y ਦਾ ਮੁੱਲ ਵੀ 9 ਹੋ ਜਾਵੇਗਾ (i.e. *x* = 9 and y=9)

y = x--;　　　　(ਪੋਸਟ-ਇੰਕਰੀਮੈਂਟ)

ਇਸ ਉਦਾਹਰਣ ਵਿੱਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ x ਦਾ ਮੁੱਲ ਵੇਰੀਏਬਲ y ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ ਇਸ ਤੋਂ ਬਾਅਦ x ਦੇ ਮੁੱਲ ਵਿੱਚੋਂ ਇੱਕ ਨੰਬਰ ਘਟਾਇਆ ਜਾਵੇਗਾ ਅਤੇ x ਦਾ ਮੁੱਲ 9 ਹੋ ਜਾਵੇਗਾ, (i.e. x = 9 ਅਤੇ y = 9)

5.3.7 ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ Conditional Operator (? :)

ਇਹ ਇੱਕ ਟਰਨਰੀ ਆਪਰੇਟਰ (ternary) ਹੈ। ਜਾਵਾ ਵਿੱਚ ਕੇਵਲ ਇੱਕ ਹੀ ਟਰਨਰੀ ਓਪਰੇਟਰ ਹੈ। ਇਸ ਆਪਰੇਟਰ ਨੂੰ ਆਪਣਾ ਕੰਮ ਕਰਨ ਲਈ ਤਿੰਨ ਆਪਰੈਂਡਾਂ ਦੀ ਜਰੂਰਤ ਪੈਂਦੀ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਟਰਨਰੀ ਓਪਰੇਟਰ ਨੂੰ ਦਰਸਾਉਣ ਲਈ "? :" ਚਿੰਨ੍ਹ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਸ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਹੇਠਾਂ ਦਿੱਤਾ ਗਿਆ ਹੈ:

exp1?exp2:exp3;

ਇੱਥੇ exp1 ਇੱਕ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨ ਹੈ ਜੋ true ਜਾਂ false ਨਤੀਜਾ ਪੈਦਾ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਐਕਸਪ੍ਰੈਸ਼ਨ exp1 ਦਾ ਨਤੀਜਾ true ਹੋਵੇਗਾ ਤਾਂ ਐਕਸਪ੍ਰੈਸ਼ਨ exp2 ਆਪਣਾ ਕੰਮ ਕਰੇਗੀ ਨਹੀਂ ਤਾਂ ਐਕਸਪ੍ਰੈਸ਼ਨ exp3 ਆਪਣਾ ਕੰਮ ਕਰੇਗੀ। ਉਦਾਹਰਣ ਲਈ:

a=5;
b=10;
c =**a > b ? a : b**;

ਇਸ ਉਦਾਹਰਣ ਵਿਚ ਐਕਸਪ੍ਰੈਸ਼ਨ (Operand1/exp1) a>b ਦਾ ਨਤੀਜਾ false ਹੈ, ਇਸ ਲਈ : ਤੋਂ ਬਾਅਦ ਵਾਲੀ ਐਕਸਪ੍ਰੈਸ਼ਨ (Operand3/exp3) ਵੇਰੀਏਬਲ b ਦਾ ਮੁੱਲ ਵੇਰੀਏਬਲ c ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾਵੇਗਾ। ਐਕਸਪ੍ਰੈਸ਼ਨ (Oprand2/exp2) ਵੇਰੀਏਬਲ a ਕੋਈ ਕੰਮ ਨਹੀਂ ਕਰੇਗਾ ਕਿਉਂਕਿ ਇਹ ਆਪਣਾ ਕੰਮ ਤਾਂ ਹੀ ਕਰੇਗਾ ਜੇਕਰ exp1 ਦਾ ਨਤੀਜਾ (True) ਹੋਵੇਗਾ।

ਪ੍ਰੋਗਰਾਮ 5 .4 ਜਾਵਾ ਵਿਚ ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਸੰਬਧੀ ਪ੍ਰੋਗਰਾਮ ਦੀ ਉਦਾਹਰਣ

Test4.java

```
class Test4
{
    public static void main(String args[])
    {
        int num1=15, num2=10;
        int result;
        System.out.println("Value of num1 is "+num1);
        System.out.println("Value of num2 is "+num2);
        result = num1>num2 ? num1 : num2;
        System.out.println("Largest Number is "+result);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 5 .4 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਊਟਪੁੱਟ:

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac Test4.java

D:\Java Programs>javac Test4.java

D:\Java Programs>java Test4
Value of num1 is 15
Value of num2 is 10
Largest Number is 15

D:\Java Programs>
```

5.3.8 ਸਪੈਸ਼ਲ ਆਪਰੇਟਰਜ਼ (Special Operators) :

JAVA ਕੁੱਝ ਵਿਸ਼ੇਸ਼ ਆਪਰੇਟਰਜ਼ ਵੀ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ, ਜਿਵੇਂ ਕਿ: instance of ਆਪਰੇਟਰ ਅਤੇ ਮੈਂਬਰ ਸਿਲੈਕਸ਼ਨ (Member Selection) ਆਪਰੇਟਰ:

❖ **instranceof ਆਪਰੇਟਰ:** ਇਸ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਇਹ ਜਾਂਚ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਦਿੱਤਾ ਗਿਆ ਆਬਜੈਕਟ ਕਿਸੇ ਵਿਸ਼ੇਸ਼ ਕਲਾਸ ਨਾਲ ਸਬੰਧਤ ਹੈ ਜਾਂ ਨਹੀਂ। ਇਹ ਇਸ ਗੱਲ 'ਤੇ ਨਿਰਭਰ ਕਰਦਾ ਹੈ ਕਿ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਖੱਬੇ ਪਾਸੇ ਵਾਲਾ ਆਬਜੈਕਟ ਸੱਜੇ ਪਾਸੇ ਵਾਲੀ ਕਲਾਸ ਦਾ ਇੰਸਟਾਂਸ (Instance) ਹੈ ਜਾਂ ਨਹੀਂ। ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੀ ਜਾਂਚ ਤੋਂ ਬਾਅਦ ਇਹ true ਜਾਂ false ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਉਦਾਹਰਣ ਲਈ:

mango **instanceof** fruit

ਇਹ ਸਟੇਟਮੈਂਟ true ਵਾਪਸ ਕਰੇਗਾ ਜੇਕਰ mango ਆਬਜੈਕਟ fruit ਕਲਾਸ ਨਾਲ ਸਬੰਧਤ ਆਬਜੈਕਟ ਹੋਵੇਗਾ, ਨਹੀਂ ਤਾਂ ਇਹ false ਵਾਪਸ ਕਰੇਗਾ।।

- **ਮੈਂਬਰ ਸਿਲੈਕਸ਼ਨ ਆਪਰੇਟਰ (Member Selection Operator):** ਇਸਨੂੰ ਡਾਟ (.) ਆਪਰੇਟਰ ਵਜੋਂ ਵੀ ਜਾਨਿਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਕਿਸੇ ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਦੁਆਰਾ ਉਸ ਕਲਾਸ ਦੇ ਵੇਰੀਏਬਲਾਂ ਅਤੇ ਮੈਥਡਾਂ ਦੀ ਵਰਤੋਂ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

```
Object.variable;        //accessing variable of a class through object
Object.method();        //accessing method of a class through object
```

5.4 ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (EXPRESSIONS)

ਐਕਸਪ੍ਰੈਸ਼ਨ ਗਣਿਤ ਵਿਚ ਇਕ ਫਾਰਮੂਲੇ ਦੀ ਤਰ੍ਹਾਂ ਹੁੰਦੀ ਹੈ। ਇਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਆਪਰੇਟਰਜ਼ ਅਤੇ ਓਪਰੈਂਡਜ਼ ਦਾ ਕੋਈ ਵੀ ਯੋਗ ਸੁਮੇਲ (valid combination) ਹੋ ਸਕਦਾ ਹੈ। ਇਕ ਯੋਗ ਸੁਮੇਲ ਅਜਿਹਾ ਸੁਮੇਲ ਹੁੰਦਾ ਹੈ ਜੋ JAVA ਭਾਸ਼ਾ ਦੇ ਸਿੰਟੈਕਸ ਨਿਯਮਾਂ ਦੀ ਪੁਸ਼ਟੀ ਕਰਦਾ ਹੈ। ਇਕ ਯੋਗ ਐਕਸਪ੍ਰੈਸ਼ਨ ਨੂੰ well formed-expression ਵਜੋਂ ਵੀ ਜਾਨਿਆ ਜਾਂਦਾ ਹੈ। ਮੁਲਾਂਕਣ ਤੋਂ ਬਾਅਦ ਐਕਸਪੈਸ਼ਨ ਹਮੇਸ਼ਾ ਇਕੋ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਇਸ ਨਤੀਜੇ ਨੂੰ JAVA ਭਾਸ਼ਾ ਦੇ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਐਕਸਪੈਸ਼ਨ ਇਕ ਸਿੰਗਲ ਮੁੱਲ ਜਿੰਨੀ ਸਾਧਾਰਣ ਵੀ ਹੋ ਸਕਦੀ ਹੈ ਅਤੇ ਇਕ ਵੱਡੀ ਕੈਲਕੁਲੇਸ਼ਨ ਜਿੰਨੀ ਗੁੰਝਲਦਾਰ ਵੀ। ਉਦਾਹਰਣ ਲਈ:

x=2.9;

ਇਹ ਇਕ ਸਧਾਰਣ ਐਕਸਪ੍ਰੈਸ਼ਨ ਹੈ ਜਿਸ ਨੂੰ ਅਪਰੈਂਡਜ਼ x ਅਤੇ 2.9 ਨਾਲ ਵਰਤਿਆ ਗਿਆ ਹੈ।

x = 2.9 * y + 3.6 > z - (3.4 / z);

ਇਹ ਕੁੱਝ ਗੁੰਝਲਦਾਰ ਐਕਸਪ੍ਰੈਸ਼ਨ ਹੈ ਜੋ ਕਿ ਕਈ ਆਪਰੇਟਰਾ ਅਤੇ ਆਪਰੈਂਡਜ਼ ਤੋਂ ਮਿਲ ਕੇ ਬਣੀ ਹੋਈ ਹੈ। ਇਸ ਵਿਚ =, *, +, >, –/ ਆਪਰੇਟਰਜ਼ ਹਨ ਅਤੇ x, 2.9, y, 3.6, 34, ਆਪਰੈਂਡਜ਼ ਹਨ। ਹੇਠਾਂ ਦਿਤੀ ਉਦਾਹਰਣ ਵਿਚ ਆਪਰੇਟਰਜ਼ ਅਤੇ ਅਪਰੈਂਡਜ਼ ਦਾ ਸੁਮੇਲ ਇਕ ਯੋਗ ਐਕਸਪੈਸ਼ਨ ਨਹੀਂ ਬਣਾ ਰਹੇ:

x + y = z;

ਇਸ ਉਦਾਹਰਣ ਵਿਚ ਭਾਵੇਂ ਅਸੀਂ ਯੋਗ ਆਪਰੇਟਰਜ਼ ਅਤੇ ਆਪਰੈਂਡਜ਼ ਦੀ ਵਰਤੋਂ ਕਰ ਰਹੇ ਹਾਂ ਪੰਤੂ ਫਿਰ ਵੀ ਉਹਨਾਂ ਦਾ ਸੁਮੇਲ ਇਕ ਯੋਗ ਐਕਸਪ੍ਰੈਸ਼ਨ ਨਹੀਂ ਬਣਾ ਰਿਹਾ ਕਿਉਂਕਿ ਇਹਨਾਂ ਆਪਰੇਟਰਜ਼ ਅਤੇ ਆਪਰੈਂਡਜ਼ ਨੂੰ JAVA ਭਾਸ਼ਾ ਦੇ ਸਿੰਟੈਕਸ ਨਿਯਮਾਂ ਅਨੁਸਾਰ ਨਹੀਂ ਵਰਤਿਆ ਗਿਆ। ਇਸ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ z ਦਾ ਮੁੱਲ ਸਟੋਰ ਕਰਵਾਉਣ ਲਈ = ਆਪਰੇਟਰ ਦੇ ਖੱਬੇ ਪਾਸੇ ਇਕ ਯੋਗ ਮੈਮਰੀ ਲੋਕੇਸ਼ਨ (ਆਈਡੈਂਟੀਫਾਇਰ ਦਾ ਹੋਣਾ ਜਰੂਰੀ ਹੈ, ਜਦੋਂ ਕਿ ਉਪਰੋਕਤ ਉਦਾਹਰਣ ਵਿਚ ਖੱਬੇ ਪਾਸੇ x + y ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਗਈ ਹੈ ਜੋ ਕਿ ਇਕ ਯੋਗ ਆਈਡੈਂਟੀਫਾਇਰ ਨਹੀਂ ਹੋ ਸਕਦਾ ਕਿਉਂਕਿ ਇਕ ਯੋਗ ਆਈਡੈਂਟੀਫਾਇਰ ਵਿਚ ਅੰਡਰਸਕੋਰ ਤੋਂ ਇਲਾਵਾ ਹੋਰ ਕੋਈ ਵੀ ਵਿਸ਼ੇਸ਼ ਚਿੰਨ੍ਹ ਨਹੀਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ (ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਪਿਛਲੇ ਪਾਠ ਵਿਚ ਪੜਿਆ ਹੈ)

ਜਾਵਾ ਭਾਸ਼ਾ ਦੀਆਂ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਨੂੰ ਹੇਠ ਲਿਖੀਆਂ ਦੋ ਸ਼੍ਰੇਣੀਆਂ ਵਿਚ ਵੰਡਿਆ ਜਾ ਸਦਕਾ ਹੈ:

5.4.1 ਨੁਮੈਰੀਕਲ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (Numerical Expressions):

ਇਹਨਾਂ ਐਸਪੈਸ਼ਨਜ਼ ਦੀ ਵਰਤੋਂ ਲਾਜੀਕਲ ਅਤੇ ਕੰਡੀਸ਼ਨਲ ਕੰਮਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਹਮੇਸ਼ਾ ਦੋ ਸੰਭਵ ਮੁੱਲਾਂ ਵਿਚੋਂ ਇਕ ਮੁੱਲ ਵਾਪਿਸ ਕਰਦੀਆਂ ਹਨ:

4 + 3

3.2 - 7.8

ਉਪਰੋਕਤ ਨੁਮੈਰੀਕਲ ਐਕਸਪੈਸ਼ਨਜ਼ ਮੁਲਾਂਕਣ ਤੋਂ ਬਾਅਦ ਨੁਮੈਰੀਕਲ ਮੁੱਲ 7 ਅਤੇ –4.6 ਪ੍ਰਦਾਨ ਕਰਣਗੀਆਂ।

5.4.2 ਲਾਜੀਕਲ ਜਾਂ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ (Logical or Conditional Expressions):

ਇਹਨਾਂ ਐਸਪੈਸ਼ਨਜ਼ ਦੀ ਵਰਤੋਂ ਲਾਜੀਕਲ ਅਤੇ ਕੰਡੀਸ਼ਨਲ ਕੰਮਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਹਮੇਸ਼ਾ ਦੋ ਸੰਭਵ ਮੁੱਲਾਂ ਵਿਚੋਂ ਇਕ ਮੁੱਲ ਵਾਪਿਸ ਕਰਦੀਆਂ ਹਨ:

14 > 6

15 <= 6

ਉਪਰੋਕਤ ਪਹਿਲੀ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਮੁਲਾਂਕਣ ਤੋਂ ਬਾਅਦ ਸਾਨੂੰ true ਨਤੀਜਾ ਦੇਵੇਗੀ ਜਦੋਂ ਕਿ ਦੂਸਰੀ ਐਕਸਪ੍ਰੈਸ਼ਨ false ਨਤੀਜਾ ਪ੍ਰਦਾਨ ਕਰੇਗੀ।

5.5 ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ/ਹਰਾਰਕੀ (PRECEDENCE/ HIERARCHY AND ASSOCIATIVITY OF OPERATORS)

ਉਹ ਕ੍ਰਮ ਜਾਂ ਤਰਜੀਹ (order or priority) ਜਿਸ ਵਿਚ ਕਿਸੇ ਦਿੱਤੀ ਗਈ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਆਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ (evaluation) ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉੱਚ ਤਰਜੀਹ ਵਾਲੇ ਆਪਰੇਟਰ ਘੱਟ ਤਰਜੀਹ ਵਾਲੇ ਆਪਰੇਟਰਾਂ ਤੋਂ ਪਹਿਲਾਂ ਆਪਣਾ ਕੰਮ ਕਰਦੇ ਹਨ। ਸਧਾਰਨ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਆਪਰੇਟਰਾਂ ਦੇ ਮੁਲਾਂਕਣ ਦਾ ਕੰਮ ਜਿਸ ਕ੍ਰਮ ਵਿੱਚ ਉਹਨਾਂ ਨੂੰ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਆਪਰੈਂਡਾ ਉੱਤੇ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ, ਭਾਗ (division) ਕਰਨ ਦਾ ਕੰਮ ਘਟਾਓ (subtraction) ਤੋਂ ਪਹਿਲਾਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਭਾਗ ਓਪਰੇਟਰ ਨੂੰ ਘਟਾਓ ਆਪਰੇਟਰ ਨਾਲੋਂ ਵੱਧ ਤਰਜੀਹ ਦਿੱਤੀ ਜਾਂਦੀ ਹੈ। ਬਰੈਕਟਾਂ (Parentheses) ( ) ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਅਸੀਂ ਆਪਰੇਟਰਾਂ ਦੇ ਕੰਮ ਕਰਨ ਦੀ ਤਰਜੀਹ ਨੂੰ ਬਦਲ ਸਕਦੇ ਹਾਂ। ਆਮ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਆਪਰੇਟਰਾਂ ਦੀ ਤਰਜੀਹ ਘਟਦੇ ਕ੍ਰਮ ਵਿੱਚ ਅਤੇ ਉਹਨਾਂ ਦੀ ਐਸੋਸੀਏਟੀਵਿਟੀ (Associativity) ਸੰਬੰਧੀ ਜਾਣਕਾਰੀ ਹੇਠਾਂ ਟੇਬਲ ਵਿਚ ਦਿੱਤੀ ਗਈ ਹੈ:

ਲੜੀ ਨੰ:	ਆਮ ਵਰਤੇ ਜਾਂਦੇ ਆਪਰੇਟਰਜ਼	ਐਸੋਸੀਏਟੀਵਿਟੀ
1.	. ( ) [ ]	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
2.	- ++ - - ! ~ (type) casting	ਸੱਜੇ ਤੋਂ ਖੱਬੇ
3.	* / %	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
4.	+ -	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
5.	< <= > > = instanceof	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
6.	== !=	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
7.	&&	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
8.	||	ਖੱਬੇ ਤੋਂ ਸੱਜੇ
9.	? :	ਸੱਜੇ ਤੋਂ ਖੱਬੇ
10.	= *= /= %- += -=	ਸੱਜੇ ਤੋਂ ਖੱਬੇ

ਹੇਠਾਂ ਦਿੱਤੀ ਉਦਾਹਰਨ ਇਹ ਦਰਸਾਉਂਦੀ ਹੈ ਕਿ ਓਪਰੇਟਰ ਦੀ ਤਰਜੀਹ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਅਰਥਮੈਟਿਕ ਐਕਸਪ੍ਰੈਸ਼ਨਜ਼ ਦਾ ਮੁਲਾਂਕਣ ਕਿਵੇਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ:

a = 5 * 4 / 4 + 8 – 9 / 3; (* ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 20 / 4 + 8 – 9 / 3; (/ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 5 + 8 – 9 / 3; (/ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 5 + 8 - 3; (+ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 13 - 3; (- ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਵੇਗਾ)

a = 10; ਐਕਸਪ੍ਰੈਸ਼ਨ ਦਾ ਨਤੀਜਾ

ਉਹ ਕ੍ਰਮ ਜਿਸ ਵਿੱਚ ਇੱਕੋ ਤਰਜੀਹ ਵਾਲੇ ਆਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਐਸੋਸੀਏਟੀਵਿਟੀ (Associativity) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਉਦਾਹਰਨ ਲਈ: ਜੋੜ ਅਤੇ ਘਟਾਓ ਆਪਰੇਟਰਾਂ ਦੀ ਤਰਜੀਹ ਇੱਕੋ ਜਿਹੀ ਹੁੰਦੀ ਹੈ। ਹਾਲਾਂਕਿ, ਜੋੜ ਅਤੇ ਘਟਾਓ ਦੇ ਮੁਲਾਂਕਣ ਦਾ ਕ੍ਰਮ ਉਹਨਾਂ ਦੇ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ ਮੌਜੂਦ ਹੋਣ ਤੇ ਕ੍ਰਮ ਉਪਰ ਨਿਰਭਰ ਕਰਦਾ ਹੈ। ਕਿਸੇ ਓਪਰੇਟਰ ਦੀ ਐਸੋਸੀਏਟੀਵਿਟੀ ਜਾਂ ਤਾਂ ਖੱਬੇ ਤੋਂ ਸੱਜੇ ਜਾਂ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਹੋ ਸਕਦੀ ਹੈ। ਖੱਬੇ ਤੋਂ ਸੱਜੇ ਐਸੋਸੀਏਟੀਵਿਟੀ ਵਾਲੇ ਓਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ ਖੱਬੇ ਸਿਰੇ ਤੋਂ ਸ਼ੁਰੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਕਿ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਐਸੋਸੀਏਟੀਵਿਟੀ ਵਾਲੇ ਓਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ ਸੱਜੇ ਸਿਰੇ ਤੋਂ ਸ਼ੁਰੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ।

ਉਦਾਹਰਣ ਲਈ:

a = b = c = 8;

ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰ ਦੀ ਐਸੋਸੀਏਟੀਵਿਟੀ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਹੁੰਦੀ ਹੈ। ਇਸਦਾ ਮਤਲਬ ਇਹ ਹੈ ਕਿ ਇਸ ਐਕਸਪ੍ਰੈਸ਼ਨ ਵਿਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ 8 ਮੁੱਲ c ਵੇਰੀਏਬਲ ਨੂੰ ਅਸਾਈਨ ਕੀਤਾ ਜਾਵੇਗਾ, ਫਿਰ c ਦਾ ਮੁੱਲ b ਨੂੰ ਅਸਾਈਨ ਕੀਤਾ ਜਾਵੇਗਾ, ਅਤੇ ਅੰਤ ਵਿੱਚ b ਵੇਰੀਏਬਲ ਦਾ ਮੁੱਲ a ਨੂੰ ਅਸਾਈਨ ਕੀਤਾ ਜਾਵੇਗਾ। ਇਸ ਤਰ੍ਹਾਂ ਇਸ ਅਸਾਈਨਮੈਂਟ ਸਟੇਟਮੈਂਟ ਵਿਚ = ਆਪਰੇਟਰ ਸੱਜੇ ਤੋਂ ਖੱਬੇ ਵੱਲ ਜਾਂਦੇ ਹੋਏ ਆਪਣੇ ਕੰਮ ਪੂਰਾ ਕਰਨਗੇ।

5.6 ਟਾਈਪ ਕਨਵਰਜ਼ਨ (TYPE CONVERSION)

ਜਾਵਾ ਵਿਚ ਜਰੂਰਤ ਅਨੁਸਾਰ ਕਿਸੇ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦਾ ਮੁੱਲ ਕਿਸੇ ਖਾਸ ਵੱਖਰੀ ਟਾਈਪ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਜਦੋਂ ਕਿਸੇ ਇਕ ਕਿਸਮ ਦੇ ਮੁੱਲ ਨੂੰ ਕਿਸੇ ਦੂਸਰੀ ਕਿਸਮ ਦੇ ਮੁੱਲ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਸਨੂੰ ਟਾਈਪ ਕਨਵਰਜ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਵਿਚ ਇਹ ਕਨਵਰਜ਼ਨ ਦੋ ਤਰੀਕਿਆਂ ਨਾਲ ਹੋ ਸਕਦੀ ਹੈ:

1. ਇੰਪਲੀਸਿਟ (ਪ੍ਰਤੱਖ) ਕਨਵਰਜ਼ਨ Implicit Conversion or Widening Conversion
2. ਐਕਸਪਲੀਸਿੱਟ (ਸਪਸ਼ਟ) ਕਨਵਰਜ਼ਨ Explicit Conversion or Narrowing Conversion

5.6.1 ਇੰਪਲੀਸਿਟ ਕਨਵਰਜ਼ਨ ਜਾਂ ਵਾਈਡਨਿੰਗ ਕਨਵਰਜ਼ਨ (Implicit Conversion or Widening Conversion):

ਇਸ ਕਿਸਮ ਦੀ ਕਨਵਰਜ਼ਨ ਆਟੋਮੈਟਿਕ (automatic) ਹੁੰਦੀ ਹੈ। ਇਸ ਕਿਸਮ ਦੀ ਕਨਵਰਜ਼ਨ ਲਈ ਅਸੀਂ ਅਸਾਈਨਮੈਂਟ (=) ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਇਸ ਕਿਸਮ ਦੀ ਕਨਵਰਜ਼ਨ ਦੀ ਵਰਤੋਂ ਉਸ ਸਮੇਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਛੋਟੀ ਡਾਟਾ ਟਾਈਪ ਵਾਲੇ ਆਪਰੈਂਡ ਨੂੰ ਵੱਡੀ ਡਾਟਾ ਟਾਈਪ ਵਿੱਚ ਬਦਲਿਆ ਜਾਂਦਾ ਹੈ। ਜੇਕਰ ਡਾਟਾ ਕਨਵਰਜ਼ਨ ਵਿਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਦੋਵੇਂ ਡਾਟਾ ਆਈਟਮਾਂ ਅਨੁਕੂਲ (compatible) ਹੋਣ ਅਤੇ ਟਾਰਗੇਟ ਵੇਰੀਏਬਲ ਦੀ ਡਾਟਾ ਟਾਈਪ ਸੋਰਸ ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਵੱਡੀ ਹੋਵੇ, ਤਾਂ ਜਾਵਾ ਆਪਣੇ ਆਪ ਸੋਰਸ ਟਾਈਪ ਨੂੰ ਟਾਰਗੇਟ ਟਾਈਪ ਵਿੱਚ ਬਦਲ ਦਿੰਦਾ ਹੈ। ਇਸ ਕਿਸਮ ਦੇ ਕਨਵਰਜ਼ਨ ਵਿੱਚ ਜਾਣਕਾਰੀ ਦਾ ਕੋਈ ਨੁਕਸਾਨ ਨਹੀਂ (no loss of information) ਹੁੰਦਾ। ਉਦਾਹਰਨ ਲਈ: int ਡਾਟਾ ਟਾਈਪ ਦਾ ਮੁੱਲ (double) ਡਾਟਾ ਟਾਈਪ ਦੇ ਇੱਕ ਵੇਰੀਏਬਲ ਨੂੰ ਦਿੱਤਾ ਜਾ ਸਕਦਾ ਹੈ ਕਿਉਂਕਿ (double) ਟਾਈਪ int ਟਾਈਪ ਤੋਂ ਵੱਡੀ ਹੁੰਦੀ ਹੈ। ਆਟੋਮੈਟਿਕ ਕਨਵਰਜ਼ਨ ਲਈ ਅੱਗੇ ਦਿੱਤੀ ਉਦਾਹਰਣ ਦੇਖੋ:

```
double n;
n = 5;
```

ਇਸ ਉਦਾਹਰਨ ਵਿੱਚ ਇੰਪਲੀਸੈੱਟ ਕਨਵਰਜ਼ਨ ਹੋਵੇਗਾ, ਕਿਉਂਕਿ int ਟਾਈਪ ਦਾ ਮੁੱਲ (5) ਆਪਣੇ ਆਪ double ਟਾਈਪ ਦੇ ਮੁੱਲ ਵਿੱਚ ਬਦਲ ਜਾਵੇਗਾ ਤਾਂ ਜੋ ਇਸਨੂੰ double ਟਾਈਪ ਦੇ ਵੇਰੀਏਬਲ n ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾ ਸਕੇ, ਭਾਵ ਵੇਰੀਏਬਲ n ਦਾ ਮੁੱਲ 5.0d

ਬਣ ਜਾਵੇਗਾ।

5.6.2 ਐਕਸਪਲੀਸਿੱਟ ਕਨਵਰਜ਼ਨ ਜਾਂ ਨੈਰੋਇੰਗ ਕਨਵਰਜ਼ਨ ਜਾਂ ਟਾਈਪ ਕਾਸਟਿੰਗ (Explicit Conversion or Narrowing Conversion or Type Casting):

ਜੇਕਰ ਟਾਰਗੇਟ ਟਾਈਪ ਸੋਰਸ ਟਾਈਪ ਤੋਂ ਛੋਟਾ ਹੈ, ਤਾਂ ਕਨਵਰਜ਼ਨ ਆਪਣੇ ਆਪ ਨਹੀਂ ਹੁੰਦੀ। ਉਦਾਹਰਨ ਲਈ, (double) ਟਾਈਪ ਦਾ ਮੁੱਲ int ਟਾਈਪ ਵੇਰੀਏਬਲ ਵਿੱਚ ਸਟੋਰ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਅਜਿਹੇ ਕਨਵਰਜ਼ਨ ਲਈ Java ਇੱਕ ਤਕਨੀਕ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜਿਸਨੂੰ ਟਾਈਪ ਕਾਸਟਿੰਗ (Type Casting) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।ਇਹ ਜ਼ਬਰਦਸਤੀ (forceful) ਕਨਵਰਜ਼ਨ ਹੁੰਦੀ ਹੈ। ਇਸ ਕਿਸਮ ਦੀ ਕਨਵਰਜ਼ਨ ਲਈ ਅਸੀਂ (caste) ਆਪਰੇਟਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਇਸ ਕਿਸਮ ਦੇ ਕਨਵਰਜ਼ਨ ਵਿੱਚ ਕਨਵਰਜ਼ਨ ਪੂਰੀ ਹੋਣ ਤੋਂ ਬਾਅਦ ਜਾਣਕਾਰੀ ਦਾ ਕੋਈ ਨੁਕਸਾਨ (loss of information) ਹੋ ਵੀ ਸਕਦਾ ਹੈ ਜਾਂ ਨਹੀਂ ਵੀ।

ਇਸ ਕਿਸਮ ਦੀ ਕਾਸਟਿੰਗ ਲਈ ਸਿੰਟੈਕਸ ਇਸ ਪ੍ਰਕਾਰ ਹੈ:

(**data type**) variable ਜਾਂ expression

ਉਹ ਡਾਟਾ ਟਾਈਪ ਜਿਸ ਵਿੱਚ ਕਨਵਰਜ਼ਨ ਕੀਤੀ ਜਾਣੀ ਹੈ, ਉਸਨੂੰ ਬਰੈਕਟਾਂ ਵਿੱਚ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਕਨਵਰਟ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਸੋਰਸ ਮੁੱਲ ਨੂੰ ਬਰੈਕਟਾਂ ਤੋਂ ਬਾਅਦ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ। ਟਾਈਪ ਕਾਸਟਿੰਗ ਦੀ ਉਦਾਹਰਣ ਇਸ ਪ੍ਰਕਾਰ ਹੈ:

double n=45.5;

int a=(**int**) n;

ਇਸ ਉਦਾਹਰਨ ਵਿੱਚ (double) ਡਾਟਾ ਟਾਈਪ ਨੂੰ int ਡਾਟਾਟਾਈਪ ਵਿੱਚ ਬਦਲਣ ਲਈ ਟਾਈਪ ਕਾਸਟਿੰਗ ਕੀਤੀ ਗਈ ਹੈ। ਇਸ ਵਿਚ int ਟਾਈਪ ਵੇਰੀਏਬਲ 'a' ਦਾ ਮੁ ਵੇਗਾ ਅਤੇ .5 ਖਤਮ ਕਰ ਦਿਤਾ ਜਾਵੇਗਾ, ਭਾਵ ਜਾਣਕਾਰੀ ਦਾ ਨੁਕਸਾਨ ਹੋਵੇਗਾ।

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਡਾਟਾ ਟਾਈਪਸ ਇੱਕ ਖਾਸ ਕਿਸਮ, ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ ਅਤੇ ਡਾਟਾ ਉਪਰ ਕੀਤੇ ਜਾ ਸਕਣ ਵਾਲੇ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਨਿਰਧਾਰਤ ਕਰਦੀਆਂ ਹਨ॥
2. ਪ੍ਰਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਬਿਲਟ-ਇਨ (built-in) ਡਾਟਾ ਟਾਈਪਸ ਵਜੋਂ ਵੀ ਜਾਨਿਆ ਜਾਂਦਾ ਹੈ।
3. JAVA ਚਾਰ ਕਿਸਮਾਂ ਦੀਆਂ ਇੰਟੀਜ਼ਰ ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਸਪੋਰਟ ਕਰਦਾ ਹੈ: byte, short, int ਅਤੇ long ਟਾਈਪਸ।
4. ਫਲੋਟਿੰਗ ਪੁਆਇੰਟ ਡਾਟਾਟਾਈਪ ਦੀ ਵਰਤੋਂ ਰੀਅਲ ਨੰਬਰਾਂ ਜਿਵੇਂ ਕਿ: 3.14, 74.5884569, +29.05, -524836.458 ਆਦਿ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
5. ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਸਿੰਗਲ ਕੋਟਸ (') ਵਿੱਚ ਬੰਦ ਇੱਕ ਸਿੰਗਲ ਯੂਨੀਕੋਡ ਅੱਖਰ (single Unicode character enclosed in single quotes) ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
6. boolean ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ ਵੇਰੀਏਬਲਾਂ ਵਿਚ ਬੂਲੀਅਨ ਮੁੱਲਾਂ ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਡਾਟਾ ਟਾਈਪ ਸਿਰਫ਼ ਦੋ ਮੁੱਲਾਂ ਨੂੰ ਸਵੀਕਾਰ ਕਰਦਾ ਹੈ: **true ਜਾਂ false** ਮੁੱਲ।
7. ਨਾਨ-ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪ ਨੂੰ ਯੂਜ਼ਰ ਦੁਆਰਾ ਪਰਿਭਾਸ਼ਿਤ (user-defined) ਡਾਟਾ ਟਾਈਪਸ ਜਾਂ ਰੈਫਰੈਂਸ (reference) ਡਾਟਾ ਟਾਈਪਸ ਵਜੋਂ ਵੀ ਜਾਨਿਆ ਜਾਂਦਾ ਹੈ।
8. ਵੇਰੀਏਬਲ ਇੱਕ ਆਈਡੈਂਟੀਫਾਇਰ ਹੁੰਦਾ ਹੈ।ਜੋ ਇੱਕ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ (memory location) ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।
9. ਅਸੀਂ ਵੇਰੀਏਬਲ ਡਿਕਲੇਰੇਸ਼ਨ ਸਮੇਂ ਉਸ ਨੂੰ ਇੱਕ ਮੁੱਲ ਵੀ ਅਸਾਈਨ (assign) ਕਰ ਸਕਦੇ ਹਾਂ, ਇਸ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਵੇਰੀਏਬਲ ਇਨੀਸ਼ੀਅਲਾਈਜੇਸ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

10. ਆਪਰੈਂਡਜ਼ ਉਹ ਡਾਟਾ ਆਈਟਮਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜਿਨ੍ਹਾਂ ਉਪਰ ਆਪਰੇਟਰਜ਼ ਆਪਣਾ ਕੰਮ ਕਰ ਸਕਦੇ ਹੁੰਦੇ ਹਨ।

11. ਆਪਰੇਟਰਾਂ ਅਤੇ ਆਪਰੈਂਡਾਂ ਦੇ ਇੱਕ ਵੇਲੀਡ ਸਮੂਹ (vaid combination) ਨੂੰ ਐਸਕਪ੍ਰੈਸ਼ਨ/ਸਮੀਕਰਨ (Expression) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

12. ਅਰਿਥਮੈਟਿਕ ਆਪਰੇਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਅਰਿਥਮੈਟਿਕ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਕਰਵਾਉਣ ਲਈ ਕਿਤੀ ਜਾਂਦੀ ਹੈ, ਜਿਵੇਂ ਕਿ: ਜੋੜ, ਘਟਾਓ, ਗੁਣਾ, ਭਾਗ ਆਦਿ।

13. ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਤੁਲਨਾਤਮਕ (comparison) ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਆਪਰੈਂਡਜ਼ ਵਿਚਕਾਰ ਰਿਲੇਸ਼ਨਸ਼ਿਪ ਟੈਸਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

14. ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ ਬੁਲੀਅਨ (Boolean) ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

15. ਸ਼ਾਰਟਹੈਂਡ ਆਪਰੇਟਰਜ਼ ਸੈਲਫ-ਅਸਾਈਨਮੈਂਟ ਸਟੇਟਮੈਂਟਜ਼ ਵਿਚ ਲਾਭਦਾਇਕ ਹੁੰਦੇ ਹਨ।

16. ਇਨਕਰੀਮੈਂਟ (++) ਅਤੇ ਡਿਕਰੀਮੈਂਟ (--) ਆਪਰੇਟਰ ਯੂਨਰੀ ਆਪਰੇਟਰ ਹਨ।

17. ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰ ਇੱਕ ਟਰਨਰੀ ਆਪਰੇਟਰ ਹੈ।

18. ਆਪਰੇਟਰਾਂ ਦੇ ਮੁਲਾਂਕਣ ਦਾ ਕ੍ਰਮ ਜਿਸ ਵਿੱਚ ਉਹਨਾਂ ਨੂੰ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਆਪਰੈਂਡਾ ਉੱਤੇ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

19. ਉਹ ਕਮ ਜਿਸ ਵਿੱਚ ਇੱਕੋ ਤਰਜੀਹ ਵਾਲੇ ਆਪਰੇਟਰਾਂ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਐਸੋਸੀਏਟੀਵਿਟੀ (Associativity) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

20. ਜਦੋਂ ਕਿਸੇ ਇਕ ਕਿਸਮ ਦੇ ਮੁੱਲ ਨੂੰ ਕਿਸੇ ਦੂਸਰੀ ਕਿਸਮ ਦੇ ਮੁੱਲ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਸਨੂੰ ਟਾਈਪ ਕਨਵਰਜ਼ਨ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ਅਭਿਆਸ

ਪ੍ਰ:1 ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ

1. ....ਇੱਕ ਖਾਸ ਕਿਸਮ, ਮੁੱਲਾਂ ਦੀ ਰੇਂਜ ਅਤੇ ਡਾਟਾ ਉਪਰ ਕੀਤੇ ਜਾ ਸਕਣ ਵਾਲੇ ਆਪਰੇਸ਼ਨਾਂ ਨੂੰ ਨਿਰਧਾਰਤ ਕਰਦੀਆਂ ਹਨ।

ੳ. ਵੇਰੀਏਬਲ (Variable)   ਅ. ਐਕਸਪ੍ਰੈਸ਼ਨ (Expression)

ੲ. ਆਪਰੈਂਡ (Operand)   ਸ. ਡਾਟਾ ਟਾਈਪ (Data Type)

II. ਫਲੋਟਿੰਗ-ਪੁਆਇੰਟ ਡਾਟਾਟਾਈਪ ਦੀ ਵਰਤੋਂ .... ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ੳ. ਇੰਟੀਜ਼ਰਜ਼ (Integers)   ਅ. ਕਰੈਕਟਰ (Character)

ੲ. ਰੀਅਲ ਨੰਬਰਜ਼ (Real number)   ਸ. ਬੂਲੀਅਨ (Boolen)

III. ....ਇੱਕ ਆਈਡੈਂਟੀਫਾਇਰ ਹੁੰਦਾ ਹੈ ਜੋ ਇੱਕ ਮੈਮੋਰੀ ਲੋਕੇਸ਼ਨ (memory location) ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

ੳ. ਵੇਰੀਏਬਲ (Variable)   ਅ. ਲਿਟਰਲ (Literal)

ੲ. ਇੰਟੀਜ਼ਰ (Integer)   ਸ. ਆਪਰੇਟਰ (Operator)

IV. ....ਉਹ ਡਾਟਾ ਆਈਟਮਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜਿਨ੍ਹਾਂ ਉਪਰ ਆਪਰੇਟਰਜ਼ ਆਪਣਾ ਕੰਮ ਕਰ ਸਕਦੇ ਹੁੰਦੇ ਹਨ।

ੳ. ਆਪਰੈਂਡਜ਼ (Operands)   ਅ. ਲਿਟਰਲ (Literal)

ੲ. ਡਾਟਾ ਟਾਈਪ (Data Type)   ਸ. ਆਪਰੇਟਰ (Operator)

V. ....ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਆਪਰੈਂਡਜ਼ ਵਿਚਕਾਰ ਰਿਲੇਸ਼ਨਸ਼ਿਪ ਟੈਸਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ੳ. ਅਰਿਥਮੈਟਿਕ (Arithmetic)   ਅ. ਯੂਨਰੀ (Unary)

ੲ. ਰਿਲੇਸ਼ਨਲ (Relational) ਸ. ਟਰਨਰੀ (Ternary)

VI . ਇਨਕਰੀਮੈਂਟ (++) ਅਤੇ ਡਿਕਰੀਮੈਂਟ (--) ਆਪਰੇਟਰ .... ਆਪਰੇਟਰ ਹਨ।

ੳ. ਯੂਨਰੀ (Unary) ਅ. ਬਾਇਨਰੀ (Binary)

ੲ. ਟਰਨਰੀ (Ternary) ਸ. ਇਹਨਾਂ ਵਿਚੋਂ ਕੋਈ ਨਹੀਂ

VII . ਜਦੋਂ ਕਿਸੇ ਇਕ ਕਿਸਮ ਦੇ ਮੁੱਲ ਨੂੰ ਕਿਸੇ ਦੂਸਰੀ ਕਿਸਮ ਦੇ ਮੁੱਲ ਵਿਚ ਤਬਦੀਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਸਨੂੰ .... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਡਾਟਾ ਕਨਵਰਜ਼ਨ (Data Conversion) ਅ. ਟਾਈਪ ਕਨਵਰਜ਼ਨ (Type Conversion)

ੲ. ਵੈਲਿਯੂ ਕਨਵਰਜ਼ਨ (Value Conversion) ਸ. ਆਪਰੇਟਰ ਕਨਵਰਜ਼ਨ (Operator Conversion)

ਪ੍ਰ:2 ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ।

I. JAVA ਚਾਰ ਕਿਸਮਾਂ ਦੀਆਂ int ਡਾਟਾ ਟਾਈਪਸ ਨੂੰ ਸਪੋਰਟ ਕਰਦਾ ਹੈ: byte, short, int ਅਤੇ .... ਟਾਈਪਸ।

II. ਕਰੈਕਟਰ ਡਾਟਾ ਟਾਈਪ ਦੀ ਵਰਤੋਂ........................ ਵਿੱਚ ਬੰਦ ਇੱਕ ਸਿੰਗਲ ਯੂਨੀਕੋਡ ਅੱਖਰ (single Unicode character) ਨੂੰ ਸਟੋਰ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

III. ਆਪਰੇਟਰਾਂ ਅਤੇ ਅਪਰੈਂਡਾਂ ਦੇ ਇੱਕ ਵੇਲੀਡ ਸਮੂਹ (vaid combination) ਨੂੰ .... ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

IV. ਲਾਜੀਕਲ ਆਪਰੇਟਰਜ਼ ਨੂੰ .... ਆਪਰੇਟਰਜ਼ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

V. ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰ ਇੱਕ .... ਆਪਰੇਟਰ ਹੈ।

VI. ਆਪਰੇਟਰਾਂ ਦੇ ਮੁਲਾਂਕਣ ਦਾ ਕ੍ਰਮ ਜਿਸ ਵਿੱਚ ਉਹਨਾਂ ਨੂੰ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਆਪਰੈਂਡਾ ਉੱਤੇ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ .... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ਪ੍ਰ:3 ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ:

I. ਡਾਟਾ ਟਾਈਪਸ (Data Types) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।

ii. ਜਾਵਾ ਵਿਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਵੱਖ-ਵੱਖ ਪ੍ਰੀਮੀਟਿਵ ਡਾਟਾ ਟਾਈਪਸ (Primitive Data Type) ਦੇ ਨਾਂ ਲਿਖੋ।

iii. ਵੇਰੀਏਬਲਜ਼ (Variables) ਕੀ ਹੁੰਦੇ ਹਨ ?

iv. ਆਪਰੈਂਡਜ਼ (Operands) ਕੀ ਹੁੰਦੇ ਹਨ ?

v. ਜਾਵਾ ਦੇ ਅਰਿੱਥਮੈਟਿੱਕ ਆਪਰੇਟਰਜ਼ (Arithmetic Operators) ਬਾਰੇ ਲਿਖੋ।

vi. ਅਸਾਈਨਮੈਂਟ ਆਪਰੇਟਰਜ਼ (assigment Operators) ਕੀ ਹੁੰਦੇ ਹਨ ?

vii. ਤੁਸੀਂ ਇੰਕਰੀਮੈਂਟ ਅਤੇ ਡਿਕਰੀਮੈਂਟ (Increment and Decrement Operators) ਆਪਰੇਟਰਜ਼ ਬਾਰੇ ਕੀ ਜਾਣਦ ਹੋ ?

viii. ਜਾਵਾ ਵਿਚ ਆਪਰੇਟਰਾਂ ਦੀ ਦਰਜਾਬੰਦੀ (Precedence of Operators) ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ?

ix. ਟਾਈਪ ਕਨਵਰਜ਼ਨ (Type Conversion) ਕੀ ਹੁੰਦੀ ਹੈ ?

ਪ੍ਰ:4 ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ:

i. ਐਕਸਪ੍ਰੈਸ਼ਨ (Expression) ਕੀ ਹੁੰਦੀ ਹੈ ? ਜਾਵਾ ਵਿਚ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀਆਂ ਐਕਸਪ੍ਰੈਸ਼ਨਾਂ ਦਾ ਵਰਨਣ ਕਰੋ।

ii. ਰਿਲੇਸ਼ਨਲ ਆਪਰੇਟਰਜ਼ (Relational Operators) ਕੀ ਹੁੰਦੇ ਹਨ ? ਢੁੱਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਵਰਨਣ ਕਰੋ।

iii. ਲਾਜੀਕਲ ਆਪਰੇਟਰਾਂ (Logical Operators) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ ? ਢੁੱਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਵਰਨਣ ਕਰੋ।

iv. ਜਾਵਾ ਵਿਚ ਇਕ ਪ੍ਰੋਗਰਾਮ ਤਿਆਰ ਕਰੋ ਜੋ ਜਾਵਾ ਦੇ ਅਰਿਥਮੈਟਿਕ ਆਰਪੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੋਵੇ।

v. ਅਸੀਂ ਜਾਵਾ ਵਿਚ ਕੰਡੀਸ਼ਨਲ ਆਪਰੇਟਰ (Conditional Operator) ਦੀ ਵਰਤੋਂ ਕਿਸ ਤਰ੍ਹਾਂ ਕਰਾਂਗੇ ? ਢੁਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਸਹਿਤ ਵਰਨਣ ਕਰੋ।।

vi. ਟਾਈਪ ਕਨਵਰਜ਼ਨ (Type Conversion) ਕੀ ਹੈ ? ਟਾਈਪ ਕਨਵਰਜ਼ਨ ਦੀਆਂ ਵੱਖ ਵੱਖ ਕਿਸਮਾਂ ਦਾ ਵਰਨਣ ਕਰੋ।

ਜਾਵਾ ਵਿਚ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ
(Control Statements in Java)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

6.1 ਜਾਣ ਪਛਾਣ (Introduction)

6.2 ਕੰਡੀਸ਼ਨਲ ਸਟੇਟਮੈਂਟਸ (Conditional Statements): if else, switch case

6.3 ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਸ (Looping Statements): for, while, do while

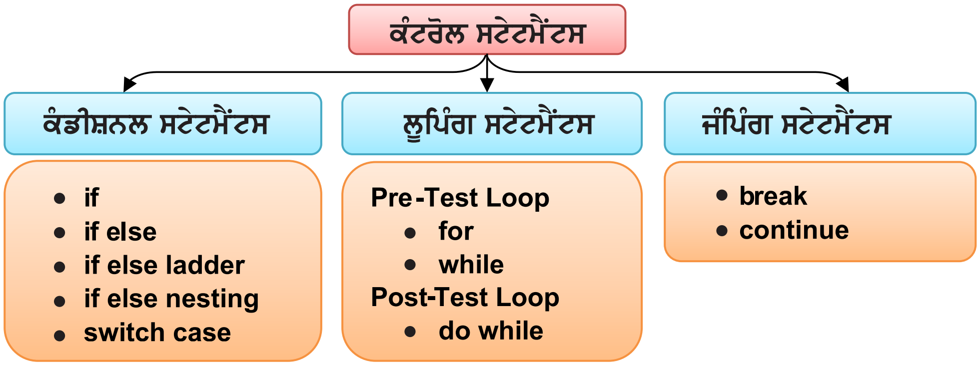
6.4 ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟਸ (Jumping Statements) : break, continue

6.1 ਜਾਣ ਪਛਾਣ (INTRODUCTION)

ਸੋਰਸ ਕੋਡ ਫਾਈਲਾਂ ਵਿਚ ਲਿਖੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਆਮ ਤੌਰ 'ਤੇ ਇਕ ਲੜੀ ਵਿਚ ਉੱਪਰ ਤੋਂ ਹੇਠਾਂ ਠੀਕ ਉਸੇ ਕ੍ਰਮ ਵਿਚ ਲਾਗੂ (exeute) ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਜਿਸ ਕ੍ਰਮ ਵਿੱਚ ਉਹ ਲਿਖੀਆਂ ਹੁੰਦੀਆਂ ਹਨ। ਹਾਲਾਂਕਿ, ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ ਜਿਵੇਂ ਕਿ: ਫੈਸਲੇ ਲੈਣ (decision making) ਵਾਲੀਆਂ ਸਟੇਟਮੈਂਟਸ, ਲੂਪਿੰਗ (looping), ਅਤੇ ਜੰਪਿੰਗ (jumping) ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੇ ਇਸ ਪ੍ਰਵਾਹ (flow of execution) ਨੂੰ ਕੰਟਰੋਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਸਟੇਟਮੈਂਟਸ ਸਾਡੇ ਪ੍ਰੋਗਰਾਮ ਵਿਚਲੇ ਕੋਡ ਦੇ ਖਾਸ ਬਲਾਕਾਂ ਨੂੰ ਦਿੱਤੀ ਗਈ ਕੰਡੀਸ਼ਨ ਅਨੁਸਾਰ ਚਲਾਉਣ ਦੇ ਯੋਗ ਬਣਾਉਂਦੀਆਂ ਹਨ॥ਇਸ ਪਾਠ ਵਿਚ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਫੈਸਲਾ ਲੈਣ ਵਾਲੀਆਂ ਸਟੇਟਮੈਂਟਸ (if, if-else, switch case ਆਦਿ), ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਸ (for, while, do-while), ਅਤੇ ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟਸ (break, continue) ਦਾ ਵਰਨਣ ਕੀਤਾ ਗਿਆ ਹੈ।

ਜਦੋਂ ਵੀ ਅਸੀਂ ਸਟੇਟਮੈਂਟਸ ਦੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਫਲੋਅ (exection flow) ਨੂੰ ਬਦਲਣਾ ਚਾਹੁੰਦੇ ਹਾਂ, ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ। ਇਹ ਸਟੇਟਮੈਂਟਸ ਕੁੱਝ ਟੈਸਟ ਕੰਡੀਸ਼ਨ ਦੇ ਆਧਾਰ 'ਤੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਫਲੋਅ ਨੂੰ ਬਦਲਦੀਆਂ ਹਨ। ਕੰਟਰੋਲ ਫਲੋਅ ਦੇ ਵੱਖ-ਵੱਖ ਤਰੀਕਿਆਂ ਦੇ ਆਧਾਰ 'ਤੇ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਤਿੰਨ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਵੰਡਿਆ ਜਾ ਸਕਦਾ ਹੈ:

1. ਕੰਡੀਸ਼ਨਲ (ਫੈਸਲਾ ਲੈਣ ਵਾਲੇ) ਸਟੇਟਮੈਂਟਸ (Conditional (Decision Making) Statements)
2. ਲੂਪਿੰਗ (ਆਇਟ੍ਰੇਟਿਵ) ਸਟੇਟਮੈਂਟਸ (Looping (Iterative) Statements)
3. ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟਸ (Jumping Statements)



ਚਿਤਰ: 6.1 ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ ਦੀਆਂ ਕਿਸਮਾਂ

ਆਓ ਹੁਣ ਇਹਨਾਂ ਸਟੇਟਮੈਂਟਸ ਸੰਬੰਧੀ ਵਿਸਥਾਰ ਵਿੱਚ ਸਮਝਣਾ ਸ਼ੁਰੂ ਕਰਦੇ ਹਾਂ:

6.2 ਕੰਡੀਸ਼ਨਲ (ਫੈਂਸਲਾ ਲੈਣ ਵਾਲੇ) ਸਟੇਟਮੈਂਟਸ (Conditional (decision Making)statements)

ਇਹਨਾਂ ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਫੈਸਲੇ ਲੈਣ (Decision-Making) ਦੇ ਉਦੇਸ਼ ਲਈ ਜਾਂ ਮਲਟੀ-ਵੇਅ ਸਿਲੈਕਸ਼ਨ (Multi-Way Slection) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਅਸੀਂ ਕਿਸੇ ਕੰਡੀਸ਼ਨ ਦੇ ਨਤੀਜੇ ਦੇ ਆਧਾਰ 'ਤੇ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਲਾਗੂ ਕਰਨ ਲਈ ਫੈਸਲੇ ਲੈ ਸਕਦੇ ਹਾਂ। ਇਹ ਸਟੇਟਮੈਂਟਸ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਫਲੋਅ (execution flow) ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਲਈ ਟੈਸਟ-ਕੰਡੀਸ਼ਨ (ਬੂਲੀਅਨ ਐਕਸਪ੍ਰੈਸ਼ਨ) ਦਾ ਮੁਲਾਂਕਣ ਕਰਦੇ ਹਨ। ਕੰਡੀਸ਼ਨ ਦਾ ਨਤੀਜਾ ਇਹ ਨਿਰਧਾਰਤ ਕਰਦਾ ਹੈ ਕਿ ਸਟੇਟਮੈਂਟਾਂ ਦੇ ਕਿਹੜੇ ਬਲਾਕ ਨੂੰ ਲਾਗੂ ਕੀਤਾ ਜਾਵੇਗਾ। ਇਸੇ ਲਈ ਇਹਨਾਂ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਫੈਸਲਾ ਲੈਣ ਵਾਲੇ (Decision-Making) ਸਟੇਟਮੈਂਟ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਬਾਂਚਿੰਗ (Branching) ਸਟੇਟਮੈਂਟਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਪ੍ਰੋਗਰਾਮ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੌਰਾਨ ਕਿਸੇ ਇੱਕ ਜਾਂ ਦੂਜੀ ਬ੍ਰਾਂਚ (ਸ਼ਾਖਾ) ਦੀ ਚੋਣ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਕੰਡੀਸ਼ਨਲ ਫਲੋਅ ਸਟੇਟਮੈਂਟਸ ਹੇਠਾਂ ਦਿੱਤੇ ਗਏ ਹਨ:

1. ਕੇਵਲ if ਸਟੇਟਮੈਂਟ
2. if-else ਸਟੇਟਮੈਂਟ
3. if-else ਲੈਡਰ (ladder) ਸਟੇਟਮੈਂਟ
4. if-else ਨੈਸਟਿੰਗ (nesting) ਸਟੇਟਮੈਂਟ
5. switch case ਸਟੇਟਮੈਂਟ

6.2.1 ਕੇਵਲ if ਸਟੇਟਮੈਂਟ (Only if Statement):

ਇਹ if-else ਸਟੇਟਮੈਂਟ ਦਾ ਸਭ ਤੋਂ ਸਰਲ (simplest) ਰੂਪ ਹੈ।if ਸਟੇਟਮੈਂਟ ਵਿਚ ਸਭ ਤੋਂ ਪਹਿਲਾਂ ਦਿੱਤੀ ਗਈ boolean_expression (ਟੈਸਟ ਕੰਡੀਸ਼ਨ)ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਜੇਕਰ ਇਸ ਟੈਸਟ ਕੰਡੀਸ਼ਨ ਦਾ ਨਤੀਜਾ true ਪ੍ਰਾਪਤ ਹੁੰਦਾ ਹੈ ਸਿਰਫ ਤਾਂ ਹੀ if ਨਾਲ ਸੰਬੰਧਤ ਸਟੇਟਮੈਂਟਾਂ ਦੇ ਸੈੱਟ ਨੂੰ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜੇਕਰ ਇਸ ਟੈਸਟ ਕੰਡੀਸ਼ਨ ਦਾ ਨਤੀਜਾ false ਪ੍ਰਾਪਤ ਹੁੰਦਾ ਹੈ ਤਾਂ if ਸਟੇਟਮੈਂਟ ਖਤਮ (terminate) ਹੋ ਜਾਂਦੀ ਹੈ ਇਸ ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਕਰਨ ਲਈ ਹੇਠਾਂ ਦਿੱਤਾ ਸਿੰਟੈਕਸ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ:

ਸਿੰਟੈਕਸ Syntax:

```
          true
if (boolean_expression)
Statement;

Next Statement after if structure;
```

OR

```
          true
if (boolean_expression)
{
        Block of Statements;
}
Next Statement after if block of statements;
```

ਸਿੰਟੈਕਸ ਵਿੱਚ ਦਿੱਤੇ boolean-expresion ਜਿਸ ਨੂੰ ਟੈਸਟ ਕੰਡੀਸ਼ਨ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ, ਨੂੰ ਬਰੈਕਟ ਵਿੱਚ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ।ਇਹਨਾਂ ਬਰੈਕਟਾਂ ਕਾਰਨ ਹੀ ਇਸ boolean-expresion (ਟੈਸਟ ਕੰਡੀਸ਼ਨ) ਦਾ ਮੁਲਾਂਕਣ ਪਹਿਲਾਂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।ਅਕਸਰ if ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਲਈ ਵਰਤੀ ਜਾਣ ਵਾਲੀ ਬੂਲੀਅਨ ਐਕਸਪ੍ਰੈਸ਼ਨ ਰਿਲੇਸ਼ਨ ਜਾਂ ਅਤੇ ਲਾਜ਼ੀਕਲ ਆਪਰੇਟਰਾਂ ਦੀ ਵਰਤੋਂ ਨਾਲ ਬਣਾਈ ਜਾਂਦੀ ਹੈ। ਜੇਕਰ ਇਸ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦਾ true ਨਤੀਜਾ ਆਉਂਦਾ ਹੈ ਤਾਂ 'Block of statements' ਨੂੰ ਲਾਗੂ ਕੀਤਾ ਜਾਵੇਗਾ ਨਹੀਂ ਤਾਂ ਸਟੇਟਮੈਂਟ if ਨਾਲ ਜੁੜੇ "Black of statements" ਨੂੰ ਨਜ਼ਰ ਅੰਦਾਜ਼ ਕਰ ਦਿੱਤਾ ਜਾਵੇਗਾ ਅਤੇ ਐਗਜੀਕਿਊਸ਼ਨ ਕੰਟਰੋਲ if ਸਟੇਟਮੈਂਟ ਦੇ ਬਲਾਕ ਤੋਂ ਬਾਅਦ ਵਾਲੇ ਸਟੇਟਮੈਂਟ ਨੂੰ ਪਾਸ ਕਰ ਦਿੱਤਾ ਜਾਵੇਗਾ।ਅੱਗੇ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਸੰਬਧੀ ਉਦਾਹਰਣ ਨੂੰ ਦਰਸਾ ਰਿਹਾ ਹੈ।

ਪ੍ਰੋਗਰਾਮ 6.1: ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ ਜੋ ਵਿਦਿਆਰਥੀ ਨੂੰ "Excellent Score!!" ਮੈਸੇਜ ਸਿਰਫ ਤਾਂ ਹੀ ਦਿਖਾਏ ਜੇਕਰ ਉਸ ਵੱਲੋਂ ਪ੍ਰਾਪਤ ਅੰਕ 80% ਤੋਂ ਵੱਧ ਹੋਣ।

cs1.java

```
class cs1
{
    public static void main(String args[])
    {
        int math=85, sci=90;
        int total=math+sci;
        double per=total/200.0*100.0;

        if(per>80)
        {
        System.out.println("Excellent Score!!");
        }
        System.out.println("Your Percentage Marks are: "+per);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.1 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਉਟਪੁੱਟ:

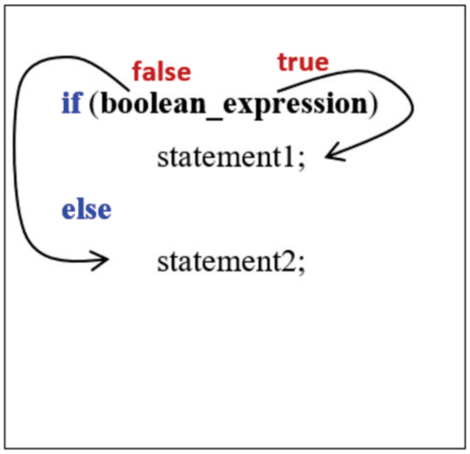
C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac cs1.java

D:\Java Programs>java cs1
Excellent Score!!
Your Percentage Marks are: 87.5

D:\Java Programs>
```

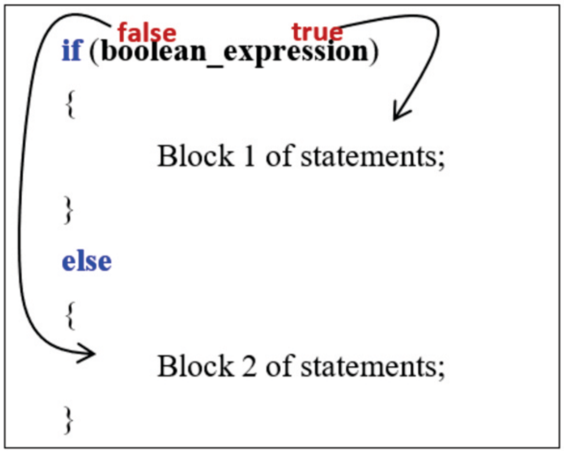
6.2.2 if else ਸਟੇਟਮੈਂਟ (if else Statement):

ਇਸ ਕੰਡੀਸ਼ਨਲ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟ ਵਿੱਚ if ਸਟੇਟਮੈਂਟ ਨਾਲ ਸੰਬੰਧਿਤ "Statement" ਜਾਂ "block of statements" ਉਦੋਂ ਹੀ ਲਾਗੂ (execute) ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਦਿੱਤੇ ਗਏ boolean_expression (ਟੈਸਟ ਕੰਡੀਸ਼ਨ) ਦਾ ਨਤੀਜਾ true ਪ੍ਰਾਪਤ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਕਿ else ਬਲਾਕ ਵਿੱਚਲੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਉਦੋਂ ਹੀ ਲਾਗੂ (execute) ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜਦੋਂ boolean_expression ਦਾ ਨਤੀਜਾ false ਹੁੰਦਾ ਹੈ। if else ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਸੰਬੰਧੀ ਸਿੰਟੈਕਸ ਹੇਠਾਂ ਦਿੱਤਾ ਗਿਆ ਹੈ:

ਸਿੰਟੈਕਸ (Syntax):



OR



ਜਿਵੇਂ ਕਿ ਸਿੰਟੈਕਸ ਵਿੱਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ, ਜੇਕਰ boolean_expression (ਟੈਸਟ ਕੰਡੀਸ਼ਨ) ਦਾ ਨਤੀਜਾ true ਹੋਵੇਗਾ, ਤਾਂ "statements" ਜਾਂ "Block 1 of statements" ਨੂੰ ਲਾਗੂ (execute) ਕੀਤਾ ਜਾਵੇਗਾ, ਨਹੀਂ ਤਾਂ (ਜੇਕਰ boolean_expression ਦਾ ਨਤੀਜਾ false ਹੋਵੇ "statements2" ਜਾਂ "Block 2 of Statements" ਨੂੰ ਲਾਗੂ ਕੀਤਾ ਜਾਵੇਗਾ। ਕਿਸੇ ਵੀ ਹਾਲਤ ਵਿੱਚ ਦੋਵੇਂ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਲਾਗੂ ਨਹੀਂ ਕੀਤਾ ਜਾਵੇਗਾ।

ਪ੍ਰੋਗਰਾਮ 6.2: ਜਾਵਾ ਵਿਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ ਜੋ ਵਿਦਿਆਰਥੀ ਦੁਆਰਾ ਪ੍ਰਾਪਤ ਪ੍ਰਤੀਸ਼ਤ ਅੰਕਾਂ ਅਨੁਸਾਰ "ਪਾਸ" ਜਾਂ "ਫੇਲ" ਨਤੀਜਾ ਦਰਸਾਉਂਦਾ ਹੋਵੇ।

cs2.java

```
class cs2
{
    public static void main(String args[])
    {
        int math=35, sci=50;
        int total=math+sci;
        double per=total/200.0*100.0;
        if(per>35)
        {
            System.out.println("Congrates!! You are Pass");
        }
        else
        {
        System.out.println("Sorry!! You are Fail");
        }
        System.out.println("Your Percentage Marks are: "+per);

    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.2 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਉਟਪੁੱਟ :

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac cs2.java

D:\Java Programs>java cs2
Congrates!! You are Pass
Your Percentage Marks are: 42.5

D:\Java Programs>_
```

6.2.3 if else ਲੈਡਰ ਸਟੇਟਮੈਂਟ (if else ladderStatement:)

ਇਹ ਮਲਟੀਪਲ if-else ਸਟੇਟਮੈਂਟਾਂ ਦੀ ਇੱਕ ਲੜੀ (chain) ਹੁੰਦੀ ਹੈ। ਇਹ ਉਹਨਾਂ if-else ਸਟੇਟਮੈਂਟਸ ਨੂੰ ਚਲਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ ਜਿੱਥੇ else ਭਾਗ ਵਿੱਚ ਇਕ ਹੋਰ if-else ਸਟੇਟਮੈਂਟ ਦਾ ਸੈੱਟ ਹੁੰਦਾ ਹੈ ਅਤੇ ਹਰੇਕ else ਭਾਗ ਲਈ ਸਾਨੂੰ ਇਕ ਹੋਰ if-else ਸਟੇਟਮੈਂਟ ਦਾ ਮੁਲਾਂਕਣ ਕਰਨਾ ਪੈਂਦਾ ਹੈ। ਕਈ if ਸਟੇਟਮੈਂਟਸ ਦੇ ਬਲਾਕਸ ਵਿਚੋਂ ਸਿਰਫ ਉਸ ਇਕ ਵਿਸ਼ੇਸ਼ if ਸਟੇਟਮੈਂਟਸ ਨਾਲ ਸੰਬੰਧਤ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਹੀ ਲਾਗੂ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜਿਸ if ਸਟੇਟਮੈਂਟਸ ਨਾਲ ਸੰਬੰਧਤ ਕੰਡੀਸ਼ਨ true ਹੁੰਦੀ ਹੈ। ਜੇਕਰ ਸਾਰੀਆਂ if ਸਟੇਟਮੈਂਟਸ ਦੀਆਂ ਕੰਡੀਸ਼ਨਾਂ false ਹੋਣ ਤਾਂ ਅੰਤਿਮ else ਬਲਾਕ ਵਿੱਚਲੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਲਾਗੂ ਕਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ if ਅਤੇ else ਸਟੇਟਮੈਂਟ ਦੇ ਵਿਚਕਾਰ ਕਿੰਨੇ ਮਰਜ਼ੀ ਵਾਰ else if ਬਲਾਕਾਂ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ।

ਸਿੰਟੈਕਸ Syntax:

```
if (boolean_expression1)
statement 1;
else if (boolean_expression2)
        statement 2;
        …….
        …….
        else
        statement n;
```

OR

```
if (boolean_expression1)
{
        Block 1 of statements;
}
else if (boolean_expression2)
        {
        Block 2 of statements;
        }
        …….
        …….
        else
        {
        Block n of statements;
        }
```

ਪ੍ਰੋਗਰਾਮ 6.3: ਵਿਦਿਆਰਥੀ ਦੇ ਪ੍ਰਾਪਤ ਅੰਕਾਂ ਅਨੁਸਾਰ ਗ੍ਰੇਡ ਦਿਖਾਉਣ ਲਈ ਇੱਕ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ। ।ਜੇਕਰ ਅੰਕ>=80 ਹੋਣ ਤਾਂ ਗ੍ਰੇਡ A ਦਿਖਾਵੇ, ਜੇਕਰ ਅੰਕ>=60 ਅਤੇ ਅੰਕ>=80 ਹੋਣ ਤਾਂ ਗ੍ਰੇਡ B ਦਿਖਾਵੇ, ਜੇਕਰ ਅੰਕ>=40 ਅਤੇ ਅੰਕ>=60 ਹੋਣ ਤਾਂ ਗ੍ਰੇਡ C, ਨਹੀਂ ਤਾਂ ਗ੍ਰੇਡ D ਦਿਖਾਵੇ।

cs3.java

```java
class cs3
{
    public static void main(String ar[])
    {
        int math=35, sci=50;
        int total=math+sci;
        double per=total/200.0*100.0;
        if(per>=80)
        System.out.println("Grade A");
        else if(per>=60)
            System.out.println("Grade B");
            else if(per>=40)
                System.out.println("Grade C");
                else
                System.out.println("Grade D");

        System.out.println("Your Percentage Marks are:  "+per);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6 .3 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਊਟਪੁੱਟ :

```
C:\Windows\system32\cmd.exe
D:\Java Programs>javac cs3.java

D:\Java Programs>java cs3
Grade C
Your Percentage Marks are: 42.5
```

6.2.4 if-else ਨੈਸਟਿੰਗ ਸਟੇਟਮੈਂਟ (if-else nesting statement):

if-else ਨੈਸਟਿੰਗ ਇੱਕ if ਜਾਂ if-else ਸਟੇਟਮੈਂਟ ਹੀ ਹੁੰਦੀ ਹੈ ਜੋ ਕਿਸੇ ਹੋਰ if-else ਜਾਂ if-else ਸਟੇਟਮੈਂਟ ਅੰਦਰ ਰੱਖੀ ਗਈ ਹੁੰਦੀ ਹੈ। ਨੈਸਟਡ if-else ਸਟੇਟਮੈਂਟਾਂ ਵਿੱਚ ਮਲਟੀਪਲ if ਅਤੇ/ਜਾਂ if-else ਸਟੇਟਮੈਂਟਾਂ ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ if ਜਾਂ if-else ਸਟੇਟਮੈਂਟ ਨੂੰ ਕਿਸੇ ਹੋਰ if ਜਾਂ else ਜਾਂ if-else ਦੇ ਅੰਦਰ ਲਿਖਣਾ if-else ਨੈਸਟਿੰਗ ਸਟੇਟਮੈਂਟ ਅਖਵਾਉਂਦਾ ਹੈ। ਅੰਦਰੂਨੀ if ਜਾਂ if-else ਨੂੰ ਬਾਹਰੀ if ਸਟੇਟਮੈਂਟ ਦੀ boolean_expression (ਟੈਸਟ ਕੰਡੀਸ਼ਨ) ਦੇ ਮੁਲਾਂਕਣ ਤੋਂ ਬਾਅਦ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ।

ਸਿੰਟੈਕਸ Syntax:

```
if(boolean_expression1)
{
    if(boolean_expression2)
    {
     Block 1 of Statements;
    }
    else
    {
     Block 2 of Statements;
    }
}
else
{
    Block 3 of statements;
}
```

```
if(boolean_expression1)
{
    Block 1 of Statements;
}
else
{
    if(boolean_expression2)
    {
     Block 2 of Statements;
    }
    else
    {
     Block 3 of Statements;
    }
}
```

```
if(boolean_expression1)
{
    if(boolean_expression2)
    {
     Block 1 of Statements;
    }
    else
    {
     Block 2 of Statements;
    }
}
else
{
    if(boolean_expression3)
    {
     Block 3 of Statements;
    }
    else
    {
     Block 4 of Statements;
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.4: ਤਿੰਨ ਨੰਬਰਾਂ ਵਿੱਚੋਂ ਸਭ ਤੋਂ ਵੱਡਾ ਨੰਬਰ ਲੱਭਣ ਲਈ ਜਾਵਾ ਵਿਚ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ।

```
cs4.java
class cs4 {
    public static void main(String args[])
    {
        int a=45,b=56,c=34;
        if(a>b)
        {
            if(a>c)
                System.out.println("Largest Number is:" + a);
            else
                System.out.println("Largest Number is:" + c);
        }
        else
        {
            if(b>c)
                System.out.println("Largest Number is:" + b);
            else
                System.out.println("Largest Number is:" + c);
        }
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.4: ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation) , ਐਗਜੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਉਟਪੁੱਟ

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac cs4.java

D:\Java Programs>java cs4
Largest Number is:56

D:\Java Programs>_
```

6.2.5 switch-case ਸਟੇਟਮੈਂਟ (switch-case Statement) :

ਇਹ ਇੱਕ ਮਲਟੀ-ਵੇਅ (ਬਹੁ-ਪੱਖੀ) ਬ੍ਰਾਂਚਿੰਗ ਸਟੇਟਮੈਂਟ ਹੈ। switch-case ਸਟੇਟਮੈਂਟ if-else ਲੈਡਰ ਦੀ ਤਰ੍ਹਾਂ ਹੁੰਦੀ ਹੈ। ਇਹ ਸਾਡੀ ਤਰਜੀਹ (preference) ਉਪਰ ਨਿਰਭਰ ਕਰਦਾ ਹੈ ਕਿ ਅਸੀਂ ਇਹਨਾਂ ਦੋਹਾਂ ਸਟੇਟਮੈਂਟਾਂ ਵਿਚੋਂ ਕਿਹੜੀ ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ।switch-case ਸਟੇਟਮੈਂਟ ਥੋੜ੍ਹਾ ਵਧੇਰੇ ਕੁਸ਼ਲ (efficient) ਹੁੰਦੀ ਹੈ ਅਤੇ ਇਸਨੂੰ ਪੜ੍ਹਨਾ ਅਤੇ ਸਮਝਣਾ ਆਸਾਨ ਹੁੰਦਾ ਹੈ।

switch-case ਸਟੇਟਮੈਂਟ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਮੁੱਲ ਦੇ ਅਧਾਰ ਤੇ ਆਪਣੇ ਕੋਡ ਦੇ ਵੱਖ-ਵੱਖ case ਲੇਬਲਾਂ ਵਿੱਚੋਂ ਇੱਕ ਲੇਬਲ ਨੂੰ ਕੰਟਰੋਲ ਪਾਸ ਕਰਨ ਦਾ ਸਭ ਤੋਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਤਰੀਕਾ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ। ਇਹ ਐਕਸਪ੍ਰੈਸ਼ਨ byte, short, int, String, ਜਾਂ char ਡਾਟਾ ਟਾਈਪ ਦਾ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ। case ਸਟੇਟਮੈਂਟਾਂ ਵਿੱਚ ਦਰਸਾਏ ਗਏ ਮੁੱਲ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੀ ਡਾਟਾ ਟਾਈਪ ਦੇ ਅਨੁਕੂਲ (type compatible) ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ। ਹਰੇਕ case ਮੁੱਲ ਇੱਕ ਕਾਂਸਟੈਂਟ ਮੁੱਲ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।switch-case ਵਿੱਚ ਡੁਪਲੀਕੇਟ case ਮੁੱਲਾਂ ਦੀ ਇਜਾਜ਼ਤ ਨਹੀਂ ਹੁੰਦੀ।

ਸਿੰਟੈਕਸ (Syntax):

```
switch (expression)
{
        case constant1:
                statements1; break:
        case constant2:
                statements2; break:
        …
        …
        case constant n:
                statements n; break:
        default :
                statements;
}
```

ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਮੁੱਲ ਦੀ ਤੁਲਨਾ case ਸਟੇਟਮੈਂਟਾਂ ਨਾਲ ਦਿੱਤੇ ਹਰੇਕ ਕਾਂਸਟੈਂਟ ਮੁੱਲ ਨਾਲ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜੇਕਰ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦਾ ਮੁਲਾਂ ਕਿਸੇ ਵੀ case ਕਾਂਸਟੈਂਟ ਦੇ ਮੁੱਲ ਨਾਲ ਮੇਲ ਖਾਂਦਾ ਹੈ, ਤਾਂ ਉਸ case ਸਟੇਟਮੈਂਟ ਤੋਂ ਬਾਅਦ ਵਾਲੇ ਕੋਡ ਨੂੰ ਚਲਾਇਆ ਜਾਵੇਗਾ। ਜੇਕਰ ਕੋਈ ਵੀ case ਕਾਂਸਟੈਂਟ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਮੁੱਲ ਨਾਲ ਮੇਲ ਨਹੀਂ ਖਾਂਦਾ, ਤਾਂ default ਸਟੇਟਮੈਂਟ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ। ਹਾਲਾਂਕਿ default ਸਟੇਟਮੈਂਟ ਆਪਸ਼ਨਲ ਹੁੰਦੀ ਹੈ। ਜੇਕਰ ਕੋਈ case ਮੇਲ ਨਹੀਂ ਖਾਂਦਾ ਅਤੇ ਕੋਈ default ਸਟੇਟਮੈਂਟ ਮੌਜੂਦ ਨਹੀਂ ਹੁੰਦੀ, ਤਾਂ switch-case ਵਿਚ ਮੌਜੂਦ ਕਿਸੇ ਵੀ ਸਟੇਟਮੈਂਟ ਉਪਰ ਕੋਈ ਕਾਰਵਾਈ ਨਹੀਂ ਕੀਤੀ ਜਾਂਦੀ।

switch ਸਟੇਟਮੈਂਟ ਅੰਦਰ break ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਸਟੇਟਮੈਂਟਾਂ ਦੀ ਲੜੀ ਦੇ ਕ੍ਰਮ ਨੂੰ ਖਤਮ ਕਰਕੇ ਕੰਟਰੋਲ ਨੂੰ switch ਸਟੇਟਮੈਂਟ ਵਿਚੋਂ ਬਾਹਰ ਲਿਆਉਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜਦੋਂ ਇੱਕ break ਸਟੇਟਮੈਂਟ ਨੂੰ ਐਗਜ਼ੀਕਿਊਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਕੰਟਰੋਲ ਸਵਿੱਚ ਸਟੇਟਮੈਂਟ ਤੋਂ ਬਾਅਦ ਵਾਲੀ ਲਾਈਨ ਵਿੱਚ ਟ੍ਰਾਂਸਫਰ ਕਰ ਦਿਤਾ ਜਾਂਦਾ ਹੈ।

ਪ੍ਰੋਗਰਾਮ 6.5: ਯੂਜ਼ਰ ਦੁਆਰਾ ਨਿਰਧਾਰਤ ਦਿਨ ਦੀ ਸੰਖਿਆ (number of the day) ਅਨੁਸਾਰ ਦਿਨ ਦਾ ਨਾਮ ਪ੍ਰਿੰਟ ਕਰਨ ਲਈ ਸਵਿਚਹ ਚਓਸਏ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ।

cs5.java

```java
class cs5
{
    public static void main(String args[])
    {
        int day=2;
        switch(day)
        {
        case 1: System.out.println("1st Day is Monday"); break;
        case 2: System.out.println("2nd Day is Tuesday"); break;
        case 3: System.out.println("3rd Day is Wednesday"); break;
        case 4: System.out.println("4th Day is Thursday"); break;
        case 5: System.out.println("5th Day is Friday"); break;
        case 6: System.out.println("6th Day is Saturday"); break;
        case 7: System.out.println("7th Day is Sunday"); break;
        default: System.out.println("Wrong value of the day");
        }
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.5: ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਊਟਪੁੱਟ :

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac cs5.java

D:\Java Programs>java cs5
2nd Day is Tuesday

D:\Java Programs>java cs5
```

6.3 ਲੂਪਿੰਗ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ (LOOPING CONTROL STATEMENTS)

ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਆਈਟਰੇਟੇਵ (Iterative) ਸਟੇਟਮੈਂਟਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਕਈ ਵਾਰ ਅਜਿਹੀਆਂ ਸਥਿਤੀਆਂ ਆ ਜਾਂਦੀਆਂ ਹਨ ਜਦੋਂ ਸਾਨੂੰ ਸਟੇਟਮੈਂਟਾਂ ਦੇ ਇੱਕ ਬਲਾਕ ਨੂੰ ਕਈ ਵਾਰ ਚਲਾਉਣ ਦੀ ਲੋੜ ਪੈਂਦੀ ਹੈ। ਅਜਿਹੀਆਂ ਸਥਿਤੀਆਂ ਵਿੱਚ ਲੂਪਸ ਸਾਨੂੰ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਦੁਹਰਾਉਣ (repeat) ਦਾ ਇੱਕ ਤਰੀਕਾ ਪ੍ਰਦਾਨ ਕਰਦੀਆਂ ਹਨ। ਇੱਕ ਲੂਪ ਵਾਰ-ਵਾਰ ਹਦਾਇਤਾਂ ਦੇ ਇੱਕੋ ਸੈੱਟ ਨੂੰ ਉਸ ਸਮੇਂ ਤੱਕ ਲਾਗੂ ਕਰਦੀ ਰਹਿੰਦੀ ਹੈ ਜਦੋਂ ਤੱਕ ਲੂਪ ਨੂੰ ਖਤਮ ਕਰਨ ਵਾਲੀ ਕੰਡੀਸ਼ਨਲ (termination condition) ਪੂਰੀ ਨਹੀਂ ਹੋ ਜਾਂਦੀ। ਜਾਵਾ ਲੂਪਸ ਨੂੰ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੋ ਸ਼੍ਰੇਣੀਆਂ ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ:

6.3.1 ਪ੍ਰੀ-ਟੈਸਟ ਲੂਪਜ਼ (Pre-Test Loops)

- ❖ for ਲੂਪ
- ❖ while ਲੂਪ

6.3.2 ਪੋਸਟ ਟੈਸਟ ਲੂਪ (Post-Test Loop)

- ❖ do-while ਲੂਪ

ਕਿਸੇ ਵੀ ਲੂਪ ਵਿੱਚ ਆਮ ਤੌਰ 'ਤੇ ਲੂਪ ਦੇ ਬਾਡੀ ਭਾਗ (body of the loop) ਦੇ ਨਾਲ ਹੇਠ ਲਿਖੇ ਤਿੰਨ ਐਕਸਪ੍ਰੈਸ਼ਨਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਲੂਪ ਦੀ ਬਾਡੀ ਵਿਚ ਉਹ ਸਟੇਟਮੈਂਟਾਂ ਹੁੰਦੀਆਂ ਹਨ ਜਿਹਨਾਂ ਨੂੰ ਦੁਹਰਾਉਣ ਲਈ ਲੂਪ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। :

- ❖ **Expression 1 (ਐਕਸਪੈਸ਼ਨ 1):** ਇਸ ਵਿੱਚ ਕਾਊਂਟਰ ਵੇਰੀਏਬਲ ਦਾ ਸ਼ੁਰੂਆਤੀ ਮੁੱਲ (Initialization of Counter Variable) ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ ਜੋ ਲੂਪ ਨੂੰ ਕੰਟਰੋਲ ਕਰਦਾ ਹੈ।
- ❖ **Expression 2 (ਐਕਸਪ੍ਰੈਸ਼ਨ 2):** ਇਸ ਵਿੱਚ boolen expresion ਸ਼ਾਮਲ ਹੁੰਦੀ ਹੈ ਜੋ ਲੂਪ ਨੂੰ ਖਤਮ ਕਰਨ ਵਾਲੀ ਕੰਡੀਸ਼ਨ (Termination Condition) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦਾ ਹੈ।
- ❖ **Expression 3 (ਐਕਸਪ੍ਰੈਸ਼ਨ 3):** ਇਸ ਵਿੱਚ ਸਟੈਪ ਮੁੱਲ (Step Value) ਸ਼ਾਮਲ ਹੁੰਦਾ ਹੈ ਜਿਸ ਵਿਚ ਕਾਊਂਟਰ ਵੇਰੀਏਬਲ ਦੇ ਮੁੱਲ ਨੂੰ ਵਧਾਇਆ ਜਾਂ ਘਟਾਇਆ ਜਾਂਦਾ ਹੈ।

6.3.1 ਪ੍ਰੀ-ਟੈਸਟ ਲੂਪਸ (Pre-Test Loops):

ਪ੍ਰੀ-ਟੈਸਟ ਲੂਪਸ ਨੂੰ ਐਂਟਰੀ-ਕੰਟਰੋਲਡ (Entry-Controlled) ਲੂਪਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਲੂਪਸ ਵਿੱਚ ਲੂਪ ਦੀ ਬਾਡੀ ਨੂੰ ਲਾਗੂ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ ਕੰਟਰੋਲ ਕੰਡੀਸ਼ਨਾਂ ਦੀ ਜਾਂਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। 'for' ਅਤੇ 'while' ਜਾਵਾ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਐਂਟਰੀ-ਕੰਟਰੋਲਡ ਲੂਪਸ ਹਨ। ਇਹਨਾਂ ਲੂਪਸ ਦੀ ਵਿਆਖਿਆ ਹੇਠਾਂ ਦਿੱਤੀ ਗਈ ਹੈ:

for ਲੂਪ : ਇਹ ਲੂਪ ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਸ ਵਿਚੋਂ ਸਭ ਤੋਂ ਵੱਧ ਵਰਤਿਆ ਜਾਣ ਵਾਲਾ ਲੂਪ ਹੈ। ਇਹ ਪ੍ਰੀ-ਟੈਸਟ ਆਈਟਿਵ ਕੰਟਰੋਲ ਸਟਰਕਚਰ ਸਾਨੂੰ ਇੱਕ ਅਜਿਹੀ ਲੂਪ ਲਿਖਣ ਦੀ ਸਹੂਲਤ ਦਿੰਦਾ ਹੈ ਜਿਹਨਾਂ ਨੂੰ ਅਸੀਂ ਇੱਕ ਖਾਸ ਗਿਣਤੀ ਲਈ ਚਲਾ ਸਕਦੇ (that needs to be executed for a specific number of times) ਹਾਂ। for ਲੂਪ ਵਿਚ ਸਟੇਟਮੈਂਟਾਂ ਉਦੋਂ ਤੱਕ ਲਾਗੂ ਕੀਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ ਜਦੋਂ ਤੱਕ ਲੂਪ ਦੀ ਦਿੱਤੀ ਗਈ ਟੈਸਟ ਕੰਡੀਸ਼ਨ true ਰਹਿੰਦੀ ਹੈ।

ਸਿੰਟੈਕਸ (Syntax):

```
for(initialization;boolean_expression;step)
{
        Body of loop;
}
```

ਜੇਕਰ ਸਿਰਫ਼ ਇੱਕ ਸਟੇਟਮੈਂਟ ਨੂੰ ਹੀ ਲੂਪ ਦੀ ਬਾਡੀ ਵੱਜੋਂ ਦੁਹਰਾਇਆ ਜਾਣਾ ਹੋਵੇ, ਤਾਂ ਸਾਨੂੰ ਲੂਪ ਲਈ ਕਰਲੀ ਬਰੇਸਿਸ (curly braces) ਦੀ ਜਰੂਰਤ ਨਹੀਂ ਪੈਂਦੀ।

for' ਲੂਪ ਦਾ ਕੰਮ ਕਰਨਾ (Working of 'for' loop):

1. ਜਦੋਂ ਲੂਪ ਪਹਿਲੀ ਵਾਰ ਸ਼ੁਰੂ ਹੁੰਦਾ ਹੈ ਤਾਂ ਲੂਪ ਦੇ ਸ਼ੁਰੂਆਤੀ ਹਿੱਸੇ (initialization) ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ। ਆਮ ਤੌਰ 'ਤੇ ਇਹ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਹੁੰਦੀ ਹੈ ਜੋ ਲੂਪ ਦੇ ਕੰਟਰੋਲ ਵੇਰੀਏਬਲ ਦਾ ਮੁੱਲ ਸੈੱਟ ਕਰਦੀ ਹੈ। ਇਹ ਕੰਟਰੋਲ ਵੇਰੀਏਬਲ ਇੱਕ ਕਾਊਂਟਰ (counter) ਵਜੋਂ ਕੰਮ ਕਰਦਾ ਹੈ ਜੋ ਲੂਪ ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਦਾ ਕੰਮ ਕਰਦਾ ਹੈ। ਇੱਥੇ ਇਹ ਯਾਦ ਰੱਖਣਾ ਮਹੱਤਵਪੂਰਨ ਰਹੇਗਾ ਕਿ ਸ਼ੁਰੂਆਤੀ ਐਕਸਪ੍ਰੈਸ਼ਨ (initalization expression) ਨੂੰ ਕੇਵਲ ਇੱਕ ਵਾਰ ਹੀ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ।
2. ਇਸ ਤੋਂ ਬਾਅਦ ਕੰਡੀਸ਼ਨ (ਬੂਲੀਅਨ ਐਕਸਪ੍ਰੈਸ਼ਨ) ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਐਕਸਪ੍ਰੈਸ਼ਨ ਆਮ ਤੌਰ 'ਤੇ ਲੂਪ ਕੰਟਰੋਲ ਵੇਰੀਏਬਲ ਦੀ ਇਕ ਟਾਰਗੇਟ ਮੁੱਲ ਨਾਲ ਜਾਂਚ ਕਰਦੀ ਹੈ। ਜੇਕਰ ਇਸ ਐਕਸਪੈਸ਼ਨ ਦਾ ਨਤੀਜਾ true ਹੋਵੇ, ਤਾਂ ਲੂਪ ਦੀ ਬਾਡੀ ਵਿਚਲੀਆਂ ਸਟੇਟਮੈਂਟਸ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ। ਜੇਕਰ ਇਹ false ਹੋਵੇ, ਤਾਂ ਲੂਪ ਨੂੰ ਟਰਮੀਨੇਟ (terminate) ਕਰ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ।

3. ਲੂਪ ਦੀ ਬਾਡੀ ਉਪਰ ਇਕ ਵਾਰ ਕੰਮ ਕਰਨ ਤੋਂ ਬਾਅਦ ਲੂਪ ਦੇ ਸਟੈਂਪ (step) ਭਾਗ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਆਮ ਤੌਰ 'ਤੇ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਹੁੰਦੀ ਹੈ ਜੋ ਲੂਪ ਦੇ ਕੰਟਰੋਲ ਵੇਰੀਏਬਲ ਨੂੰ ਵਧਾਉਣ ਜਾਂ ਘਟਾਉਣ ਦਾ ਕੰਮ ਕਰਦਾ ਹੈ।
4. ਸਟੈਪ ਭਾਗ ਉਪਰ ਕੰਮ ਕਰਨ ਤੋਂ ਬਾਅਦ ਲੂਪ ਨੂੰ ਫਿਰ ਦੁਹਰਾਇਆ ਜਾਂਦਾ ਹੈ, ਪਹਿਲਾਂ ਬੂਲੀਅਨ ਐਕਸਪ੍ਰੈਸ਼ਨ (ਕੰਡੀਸ਼ਨ) ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਫਿਰ ਲੂਪ ਦੇ ਬਾਡੀ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ, ਅਤੇ ਫਿਰ ਹਰ ਪਾਸ ਦੇ ਨਾਲ ਸਟੈਪ ਐਕਸਪੈਸ਼ਨ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਪ੍ਰਕਿਰਿਆ ਉਦੋਂ ਤੱਕ ਦੁਹਰਾਈ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਤੱਕ ਬੂਲੀਅਨ ਐਕਸਪ੍ਰੈਸ਼ਨ (ਕੰਡੀਸ਼ਨ) false ਨਹੀਂ ਹੋ ਜਾਂਦੀ।

ਪ੍ਰੋਗਰਾਮ 6.6: for ਲੂਪ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਪਹਿਲੇ n ਕੁਦਰਤੀ ਨੰਬਰਾਂ ਨੂੰ ਪ੍ਰਿੰਟ ਕਰਨ ਲਈ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ।

cs6.java

```
class cs6
{
    public static void main(String args[])
    {
        int i,n;
        n=7;
        for(i=1;i<=n;i++)
        {
            System.out.println("i=" + i);
        }
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.6 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਊਟਪੁੱਟ:

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac cs6.java

D:\Java Programs>java cs6
i=1
i=2
i=3
i=4
i=5
i=6
i=7

D:\Java Programs>
```

While ਲੂਪ:

ਇਹ ਵੀ ਪ੍ਰੀ-ਟੈਸਟ ਲੂਪ ਹੈ। ਇਹ ਲੂਪ ਆਪਣੀ ਬਾਡੀ ਦੀਆਂ ਸਟੇਟਮੈਂਟਸ ਨੂੰ ਚਲਾਉਣ ਤੋਂ ਪਹਿਲਾਂ boolean_expression (ਕੰਡੀਸ਼ਨ) ਦੀ ਜਾਂਚ ਕਰਦੀ ਹੈ।ਲੂਪ ਦੀ ਬਾਡੀ ਨੂੰ ਉਦੋਂ ਤੱਕ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਤੱਕ ਦਿਤੀ ਗਈ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨ true ਰਹਿੰਦੀ ਹੈ। ਜਦੋਂ ਕੰਡੀਸ਼ਨ false ਹੋ ਜਾਂਦੀ ਹੈ ਤਾਂ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਕੰਟਰੋਲ ਲੂਪ ਤੋਂ ਤੁਰੰਤ ਬਾਅਦ ਵਾਲੀ ਲਾਈਨ 'ਤੇ ਚਲਾ ਜਾਂਦਾ ਹੈ। ਜੇਕਰ ਇੱਕ while ਲੂਪ ਨੂੰ ਕੰਟਰੋਲ ਕਰਨ ਵਾਲੀ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪੈਸ਼ਨ ਸ਼ੁਰੂ ਵਿੱਚ ਹੀ false ਹੋ ਜਾਵੇ, ਤਾਂ ਲੂਪ ਦੀ ਬਾਡੀ ਨੂੰ ਇਕ ਵਾਰ ਵੀ ਨਹੀਂ ਚਲਾਇਆ ਜਾਂਦਾ। ਇਸ ਤਰ੍ਹਾਂ while ਲੂਪ ਦੀ ਬਾਡੀ ਲਈ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੀ ਘੱਟੋ-ਘੱਟ ਗਿਣਤੀ (minimum number of exections) ਜ਼ੀਰੋ ਹੋਵੇਗੀ। while ਲੂਪ ਦੀ ਵਰਤੋਂ ਦਾ ਜਨਰਲ ਸਿੰਟੈਕਸ ਹੇਠਾਂ ਦਿੱਤਾ ਗਿਆ ਹੈ:

ਸਿੰਟੈਕਸ (Syntax) :

```
while(boolean_expression)
{
        Body of loop;
}
```

ਪ੍ਰੋਗਰਾਮ 6.7: while ਲੂਪ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ n ਤੋਂ 1 ਤੱਕ ਦੇ ਜਿਸਤ (Even) ਅੰਕਾਂ ਨੂੰ ਪ੍ਰਿੰਟ ਕਰਨ ਲਈ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ।

cs7.java

```java
class cs7
{
    public static void main(String args[])
    {
        int n;
        n=7;
        while(n>=1)
        {
            if(n%2==0)
            System.out.println(n);

        n--;
        }
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.7 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Exection) ਅਤੇ ਆਉਟਪੁੱਟ:

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac cs7.java

D:\Java Programs>java cs7
6
4
2

D:\Java Programs>_
```

6.3.2 ਪੋਸਟ ਟੈਸਟ ਲੂਪ (Post-Test Loop) :

ਕਈ ਵਾਰ ਲੂਪ ਦੀ ਬਾਡੀ ਨੂੰ ਘੱਟੋ-ਘੱਟ ਇੱਕ ਵਾਰ ਚਲਾਉਣਾ ਫਾਇਦੇਮੰਦ ਹੁੰਦਾ ਹੈ, ਭਾਵੇਂ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨ ਸ਼ੁਰੂ ਵਿਚ false ਹੀ ਕਿਉਂ ਨਾ ਹੋਵੇ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਕਈ ਵਾਰ ਅਜਿਹਾ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਅਸੀਂ ਲੂਪ ਦੇ ਸ਼ੁਰੂਆਤ ਦੀ ਬਜਾਏ ਲੂਪ ਦੇ ਅੰਤ ਵਿੱਚ ਟਰਮੀਨੇਸ਼ਨ ਕੰਡੀਸ਼ਨ ਦੀ ਜਾਂਚ ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ। ਅਜਿਹੀਆਂ ਸਥਿਤੀਆਂ ਵਿੱਚ ਪੋਸਟ ਟੈਸਟ ਲੂਪਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਪੋਸਟ-ਟੈਸਟ ਲੂਪ ਨੂੰ ਐਗਜ਼ਿਟ ਕੰਟਰੋਲਡ (Exit-Controlled) ਲੂਪ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਵਿੱਚ do while ਲੂਪ ਇੱਕੋ ਇੱਕ ਐਗਜ਼ਿਟ ਕੰਟਰੋਲਡ ਲੂਪ ਹੈ।

do while ਲੂਪ:

do while ਲੂਪ ਵਿੱਚ ਸਟੇਟਮੈਂਟਾਂ ਉਦੋਂ ਤੱਕ ਚਲਾਈਆਂ ਜਾਂਦੀਆਂ ਹਨ ਜਦੋਂ ਤੱਕ ਲੂਪ ਦੀ ਕੰਡੀਸ਼ਨ true ਰਹਿੰਦੀ ਹੈ।do-while ਲੂਪ ਹਮੇਸ਼ਾ ਆਪਣੀ ਬਾਡੀ ਨੂੰ ਘੱਟੋ-ਘੱਟ ਇੱਕ ਵਾਰ ਚਲਾਉਂਦਾ ਹੈ, ਕਿਉਂਕਿ ਇਸਦਾ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨ ਲੂਪ ਦੇ ਹੇਠਲੇ ਪਾਸੇ ਲੂਪ ਦੀ ਬਾਡੀ ਤੋਂ ਬਾਅਦ ਲਿਖਿਆ ਜਾਂਦਾ ਹੈ। ਇਸੇ ਕਾਰਨ do-while ਲੂਪ ਵਿਚ ਲੂਪ ਦੀ ਬਾਡੀ ਲਈ ਐਕਜ਼ੀਕਿਊਸ਼ਨ ਦੀ ਘੱਟੋ-ਘੱਟ ਗਿਣਤੀ (minimum number of exections) ਇੱਕ ਹੋਵੇਗੀ। do-while ਲੂਪ ਵਿਚ ਹਰ ਵਾਰ ਜਦੋਂ ਲੂਪ ਨੂੰ ਦੁਹਰਾਇਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਪਹਿਲਾਂ ਲੂਪ ਦੀ ਬਾਡੀ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਫਿਰ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦਾ ਮੁਲਾਂਕਣ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਜੇਕਰ ਇਹ ਐਕਸਪ੍ਰੈਸ਼ਨ true ਨਤੀਜਾ ਦਿੰਦੀ ਹੈ ਤਾਂ ਲੂਪ ਨੂੰ ਦੁਹਰਾਇਆ ਜਾਵੇਗਾ। ਨਹੀਂ ਤਾਂ, ਲੂਪ ਟਰਮੀਨੇਟ ਹੋ ਜਾਵੇਗੀ।do-while ਦਾ ਜਨਰਲ ਸਿੰਟੈਕਸ ਇਸ ਤਰ੍ਹਾਂ ਹੈ:

ਸਿੰਟੈਕਸ (Syntax) :

```
do
{
        ....
        statements;
        ....
} while(boolean_expression);
```

while ਅਤੇ do-while ਲੂਪਸ ਵਿਚਕਾਰ ਇੱਕੋ-ਇੱਕ ਮੁੱਖ ਅੰਤਰ ਇਹ ਹੈ ਕਿ do-while ਵਿਚਲੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਹਮੇਸ਼ਾ ਘੱਟੋ-ਘੱਟ ਇੱਕ ਵਾਰ ਲਾਗੂ ਹੋਣਗੀਆਂ, ਭਾਵੇਂ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦਾ ਨਤੀਜਾ ਪਹਿਲੀ ਵਾਰ false ਹੀ ਕਿਉਂ ਨਾ ਹੋਵੇ। ਜਦੋਂ ਕਿ while ਲੂਪ ਵਿਚ ਜੇਕਰ boolean_expression (ਕੰਡੀਸ਼ਨ) ਪਹਿਲੀ ਵਾਰ false ਨਤੀਜਾ ਦਿੰਦੀ ਹੈ ਤਾਂ ਲੂਪ ਦੀਆਂ ਸਟੇਟਮੈਂਟਸ ਨੂੰ ਕਦੇ ਵੀ ਲਾਗੂ ਨਹੀਂ ਕੀਤਾ ਜਾਵੇਗਾ। ਅਸਲ ਵਿਚ do-while ਦੀ ਵਰਤੋਂ while ਲੂਪ ਨਾਲੋਂ ਘੱਟ ਹੁੰਦੀ ਹੈ।

ਪ੍ਰੋਗਰਾਮ 6.8: do while ਲੂਪ ਦੀ ਵਰਤੋਂ ਕਰਕੇ 1 ਤੋਂ n ਤੱਕ odd ਨੰਬਰਾਂ ਨੂੰ ਪ੍ਰਿੰਟ ਕਰਨ ਲਈ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ।

cs8.java

```
class cs8
{
    public static void main(String args[])
    {
        int i,n;
        n=7;
        i=1;
        while(i<=n)
        {
            if(i%2==1)
            System.out.println(i);

        i++;
        }
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.8 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਊਟਪੁੱਟ:

C:\Windows\System32\cmd.exe

```
D:\Java Programs>javac cs8.java

D:\Java Programs>java cs8
1
3
5
7

D:\Java Programs>_
```

6.4 ਜੰਪਿੰਗ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ (JUMPING CONTROL STATEMENTS)

ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟਾਂ ਦੀ ਵਰਤੋਂ ਪ੍ਰੋਗਰਾਮ ਦੇ ਆਮ ਪ੍ਰਵਾਹ (flow) ਨੂੰ ਬਦਲਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹਨਾਂ ਸਟੇਟਮੈਂਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਅਸੀਂ ਐਗਜ਼ੀਕਿਉਸ਼ਨ ਕੰਟਰੋਲ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਇੱਕ ਸਥਾਨ ਤੋਂ ਕਿਸੇ ਹੋਰ ਸਥਾਨ ਤੇ ਟਾਂਸਫਰ ਕਰ ਸਕਦੇ ਹਾਂ। ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਜਾਣਦੇ ਹਾਂ ਕਿ ਲੂਪ ਵਿਚਲੀਆਂ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਉਦੋਂ ਤੱਕ ਦੁਹਰਾਇਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਤੱਕ ਕਿ ਲੂਪ ਦੀ ਕੰਡੀਸ਼ਨ false ਨਹੀਂ ਹੋ ਜਾਂਦੀ ਪਰ ਕਈ ਵਾਰ ਲੂਪ ਦੇ ਅੰਦਰ ਕੁਝ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਛੱਡਣਾ ਜਾਂ ਟੈਸਟ ਐਕਸਪੈਸ਼ਨਾਂ ਦੀ ਜਾਂਚ ਕੀਤੇ ਬਿਨਾਂ ਲੂਪ ਨੂੰ ਤੁਰੰਤ ਬੰਦ ਕਰਨਾ ਫਾਇਦੇਮੰਦ ਹੁੰਦਾ ਹੈ। ਅਜਿਹੇ ਮਾਮਲਿਆਂ ਵਿੱਚ break ਅਤੇ continue ਸਟੇਟਮੈਂਟਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। break ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ switch ਸਟੇਟਮੈਂਟ ਵਿਚ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਕੰਟਰੋਲ ਨੂੰ ਬਾਹਰ ਲਿਆਉਣ ਲਈ ਵੀ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੇ ਆਮ ਪ੍ਰਵਾਹ ਨੂੰ ਬਦਲਣ ਲਈ ਜਾਵਾ ਵਿੱਚ ਬਣਾਇਆਂ ਗਈਆਂ ਜੰਪਿੰਗ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟ ਹੇਠਾਂ ਦਿੱਤੀਆਂ ਗਈਆਂ ਹਨ:

- break
- continue

6.4.1 Break ਸਟੇਟਮੈਂਟ (break statement):

break ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਲੂਪ ਜਾਂ ਸਵਿੱਚ ਸਟੇਟਮੈਂਟ ਨੂੰ ਟਰਮੀਨੇਟ (terminate) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। break ਸਟੇਟਮੈਂਟ ਨਾਲ ਲੂਪ ਜਾਂ ਸਵਿੱਚ ਸਟੇਟਮੈਂਟ ਨੂੰ ਟਰਮੀਨੇਟ ਕਰਨ ਤੋਂ ਬਾਅਦ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਕੰਟਰੋਲ ਇਹਨਾਂ ਸਟੇਟਮੈਂਟਸ ਤੋਂ ਤੁਰੰਤ ਬਾਅਦ ਅਗਲੀ ਲਾਈਨ ਉਪਰ ਟ੍ਰਾਂਸਫਰ ਕਰ ਦਿਤਾ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਿੰਗ ਵਿੱਚ break ਸਟੇਟਮੈਂਟ ਨੂੰ ਹੇਠ ਲਿਖੇ ਦੋ ਮੁੱਖ ਤਰੀਕਿਆਂ ਨਾਲ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ:

1. ਜਦੋਂ ਲੂਪ ਦੇ ਅੰਦਰ break ਸਟੇਟਮੈਂਟ ਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਲੂਪ ਤੁਰੰਤ ਬੰਦ ਹੋ ਜਾਂਦੀ ਹੈ ਅਤੇ ਐਗਜ਼ੀਕਿਉਸ਼ਨ ਕੰਟਰੋਲ ਲੂਪ ਤੋਂ ਬਾਅਦ ਅਗਲੀ ਸਟੇਟਮੈਂਟ ਤੇ ਚਲਾ ਜਾਂਦਾ ਹੈ।
2. ਇਸਦੀ ਵਰਤੋਂ (switch) ਸਟੇਟਮੈਂਟ ਵਿੱਚ case ਨੂੰ ਟਰਮੀਨੇਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ਪ੍ਰੋਗਰਾਮ 6 .9 ਲੂਪ ਵਿੱਚ (break), ਸਟੇਮੈਟ ਦੀ ਵਰਤੋ:

cs9.java

```
class cs9
{
    public static void main(String args[])
    {
        int i;
        for(i=1;i<=7;i++)
        {
            if(i==4)
            break;

        System.out.println(i);
        }
    }
}
```

ਪ੍ਰੋਗਰਾਮ 6.9 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compiliation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਉਟਪੁੱਟ:

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac cs9.java

D:\Java Programs>java cs9
1
2
3

D:\Java Programs>
```

ਜਿਵੇਂ ਕਿ ਪ੍ਰੋਗਰਾਮ 6.9 ਵਿੱਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ, break ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਅਸੀਂ ਕੰਡੀਸ਼ਨਲ ਐਕਸਪ੍ਰੈਸ਼ਨ ਅਤੇ ਲੂਪ ਦੇ ਬਾਡੀ ਭਾਗ ਵਿੱਚਲੇ ਕਿਸੇ ਵੀ ਕੋਡ ਨੂੰ ਬਾਈਪਾਸ (bypass) ਕਰਦੇ ਹੋਏ, ਲੂਪ ਨੂੰ ਤੁਰੰਤ ਸਮਾਪਤ ਕਰ ਸਕਦੇ ਹਾਂ। ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਜਦੋਂ i ਦਾ ਮੁੱਲ 4 ਹੋ ਜਾਂਦਾ ਹੈ ਤਾਂ break ਸਟੇਟਮੈਂਟ ਲੂਪ ਦੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਨੂੰ ਖਤਮ ਕਰ ਦਿੰਦੀ ਹੈ ਅਤੇ ਇਹ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਕੰਟਰੋਲ ਨੂੰ ਲੂਪ ਤੋਂ ਬਾਹਰ ਲੈ ਆਉਂਦੀ ਹੈ।

6.4.2 Continue ਸਟੇਟਮੈਂਟ (continue statement) :

ਕਈ ਵਾਰ ਲੂਪ ਤੋਂ ਬਾਹਰ ਨਿਕਲੇ ਬਿਨਾਂ ਲੂਪ ਦੇ ਅੰਦਰ ਕੁਝ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਛੱਡਣਾ ਫਾਇਦੇਮੰਦ ਹੁੰਦਾ ਹੈ continue ਸਟੇਟਮੈਂਟ ਮੌਜੂਦਾ ਦੁਹਰਾਅ (current teration) ਦੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਨੂੰ ਜੰਪ ਕਰਕੇ ਅਗਲਾ ਦੁਹਰਾਅ ਸ਼ੁਰੂ ਕਰਨ ਲਈ ਵਾਪਸ ਲੂਪ ਦੇ ਸ਼ੁਰੂ ਵਿੱਚ ਚਲੀ ਜਾਂਦੀ ਹੈ, ਜਿਵੇਂ ਕਿ ਪ੍ਰੋਗਰਾਮ 6.10 ਵਿਚ ਦਿਖਾਇਆ ਗਿਆ ਹੈ:

ਪ੍ਰੋਗਰਾਮ 6.10: ਲੂਪ ਵਿੱਚ continue ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ।

```java
cs10.java
class cs10
{
	public static void main(String args[])
	{
		int i;
		for(i=1;i<=7;i++)
		{
			if(i%2==0)
			continue;

		System.out.println(i);
		}
	}
}
```

ਪ੍ਰੋਗਰਾਮ 6.10 ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ (Compilation), ਐਗਜ਼ੀਕਿਊਸ਼ਨ (Execution) ਅਤੇ ਆਊਟਪੁੱਟ:

```
C:\Windows\System32\cmd.exe
D:\Java Programs>javac cs10.java

D:\Java Programs>java cs10
1
3
5
7

D:\Java Programs>_
```

ਪ੍ਰੋਗਰਾਮ 6.10 ਵਿੱਚ ਜਦੋਂ 'i' ਦਾ ਮੁੱਲ 2 ਨਾਲ ਪੂਰਾ-ਪੂਰਾ ਵੰਡਿਆ (divisible by 2) ਜਾਵੇਗਾ ਜਦੋਂ i ਦਾ ਮੁੱਲ 2, 4 ਅਤੇ 6 ਹੋਵੇਗਾ), ਤਾਂ continue ਸਟੇਟਮੈਂਟ ਆਪਣੀ ਭੂਮਿਕਾ ਨਿਭਾਵੇਗਾ ਅਤੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਨੂੰ ਜੰਪ ਕਰਕੇ ਲੂਪ ਦੇ ਸ਼ੁਰੂ ਵਿਚ (ਸਟੈਪ ਭਾਗ ਉਪਰ) ਚਲਾ ਜਾਵੇਗਾ, ਪਰ 'i' ਦੇ ਬਾਕੀ ਮੁੱਲਾਂ ਲਈ (ਜਦੋਂ i ਦਾ ਮੁੱਲ 1, 3, 5 ਅਤੇ 7 ਹੋਵੇਗਾ) ਲੂਪ ਸੁਚਾਰੂ ਢੰਗ ਨਾਲ print() ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ i ਦੇ ਮੁੱਲ ਨੂੰ ਦਰਸ਼ਾਵੇਗਾ।

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਜਦੋਂ ਵੀ ਅਸੀਂ ਸਟੇਟਮੈਂਟਸ ਦੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਫਲੋਅ (execution flow) ਨੂੰ ਬਦਲਣਾ ਚਾਹੁੰਦੇ ਹਾਂ, ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ।
2. ਕੰਡੀਸ਼ਨਲ ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਫੈਸਲੇ ਲੈਣ (Decision-Making) ਦੇ ਉਦੇਸ਼ ਲਈ ਜਾਂ ਮਲਟੀ ਵੇਅ ਸਿਲੈਕਸ਼ਨ (Multi-Way Selection) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।
3. ਕੰਡੀਸ਼ਨਲ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਬ੍ਰਾਂਚਿੰਗ (Branching) ਸਟੇਟਮੈਂਟਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਪ੍ਰੋਗਰਾਮ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੌਰਾਨ ਕਿਸੇ ਇੱਕ ਜਾਂ ਦੂਜੀ ਬ੍ਰਾਂਚ (ਸ਼ਾਖਾ) ਦੀ ਚੋਣ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
4. if ਜਾਂ if-else ਸਟੇਟਮੈਂਟ ਨੂੰ ਕਿਸੇ ਹੋਰ if ਜਾਂ if-else ਜਾਂ if-else ਦੇ ਅੰਦਰ ਲਿਖਣਾ if-else ਨੈਸਟਿਡ ਸਟੇਟਮੈਂਟ ਅਖਵਾਉਂਦਾ ਹੈ।
5. switch-case ਇੱਕ ਮਲਟੀ-ਵੇਅ (ਬਹੁ-ਪੱਖੀ) ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟ ਹੈ।
6. switch-case ਸਟੇਟਮੈਂਟ ਇੱਕ ਐਕਸਪ੍ਰੈਸ਼ਨ ਦੇ ਮੁੱਲ ਦੇ ਅਧਾਰ ਤੇ ਆਪਣੇ ਕੋਡ ਦੇ ਵੱਖ-ਵੱਖ case ਲੇਬਲਾਂ ਵਿੱਚੋਂ ਇੱਕ ਲੇਬਲ ਨੂੰ ਕੰਟਰੋਲ ਪਾਸ ਕਰਨ ਦਾ ਸਭ ਤੋਂ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਤਰੀਕਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।
7. ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ ਆਈਟਰੇਟਿਵ (Iterative) ਸਟੇਟਮੈਂਟਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
8. ਪ੍ਰੀ-ਟੈਸਟ ਲੂਪਸ ਨੂੰ ਐਂਟਰੀ-ਕੰਟਰੋਲਡ (Entry-Controlled) ਲੂਪਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਲੂਪਸ ਵਿੱਚ ਲੂਪ ਦੀ ਬਾਡੀ ਨੂੰ ਲਾਗੂ ਕਰਨ ਤੋਂ ਪਹਿਲਾਂ ਕੰਟਰੋਲ ਕੰਡੀਸ਼ਨਾਂ ਦੀ ਜਾਂਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
9. 'for' ਅਤੇ 'while' ਜਾਵਾ ਵਿੱਚ ਵਰਤੀ ਜਾਣ ਵਾਲੀ ਐਂਟਰੀ-ਕੰਟਰੋਲਡ ਲੂਪਸ ਹਨ।

10. for ਲੂਪ ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਸ ਵਿਚੋਂ ਸਭ ਤੋਂ ਵੱਧ ਵਰਤੀ ਜਾਣ ਵਾਲੀ ਲੂਪ ਹੈ॥ ਇਹ ਪ੍ਰੀ-ਟੈਸਟ ਆਈਟ੍ਰਿਵ ਕੰਟਰੋਲ ਸਟਰਕਚਰ ਸਾਨੂੰ ਇੱਕ ਅਜਿਹੀ ਲੂਪ ਲਿਖਣ ਦੀ ਸਹੂਲਤ ਦਿੰਦੀ ਹੈ ਜਿਹਨਾਂ ਨੂੰ ਅਸੀਂ ਇੱਕ ਖਾਸ ਗਿਣਤੀ ਲਈ ਚਲਾ ਸਕਦੇ (that needs to be executed for a specific number of times) ਹਾਂ।

11. while ਲੂਪ ਦੀ ਬਾਡੀ ਲਈ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੀ ਘੱਟੋ-ਘੱਟ ਗਿਣਤੀ (minimum number of execution) ਜ਼ੀਰੋ ਹੁੰਦੀ ਹੈ।

12. ਪੋਸਟ-ਟੈਸਟ ਲੂਪ ਨੂੰ ਐਗਜ਼ਿਟ-ਕੰਟਰੋਲਡ (Exit-Controlled) ਲੂਪ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

13. ਜਾਵਾ ਵਿੱਚ do while ਲੂਪ ਇੱਕੋ ਇੱਕ ਐਗਜ਼ਿਟ-ਕੰਟਰੋਲਡ ਲੂਪ ਹੈ।

14. do while ਲੂਪ ਹਮੇਸ਼ਾ ਆਪਣੀ ਬਾਡੀ ਨੂੰ ਘੱਟੋ-ਘੱਟ ਇੱਕ ਵਾਰ ਚਲਾਉਂਦਾ ਹੈ।

15. ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਅਸੀਂ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਕੰਟਰੋਲ ਨੂੰ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਇੱਕ ਸਥਾਨ ਤੋਂ ਕਿਸੇ ਹੋਰ ਸਥਾਨ ਤੇ ਟ੍ਰਾਂਸਫਰ ਕਰ ਸਕਦੇ ਹਾਂ।

16. break ਅਤੇ continue ਜਾਵਾ ਵਿੱਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟਸ ਹਨ।

17. break ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਲੂਪ ਜਾਂ ਸਵਿੱਚ ਸਟੇਟਮੈਂਟ ਨੂੰ ਟਰਮੀਨੇਟ (terminate) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

18. continue ਸਟੇਟਮੈਂਟ ਮੌਜੂਦਾ ਦੁਹਰਾਅ (current iteration) ਦੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਨੂੰ ਜੰਪ ਕਰਕੇ ਅਗਲਾ ਦੁਹਰਾਅ ਸ਼ੁਰੂ ਕਰਨ ਲਈ ਵਾਪਸ ਲੂਪ ਦੇ ਸ਼ੁਰੂ ਵਿੱਚ ਚਲੀ ਜਾਂਦੀ ਹੈ।

ਅਭਿਆਸ

ਪ੍ਰ: I ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ :

i. ਸਟੇਟਮੈਂਟਸ ਦੇ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਫਲੋਅ (execution flow) ਨੂੰ ਬਦਲਣ ਲਈ .... ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।

ੳ. ਕੰਪਾਊਂਡ (Compound) ਅ. ਕੰਟਰੋਲ (Control)

ੲ. ਤੁਲਨਾਤਾਮਕ (Comparative) ਸ. ਇਹਨਾਂ ਵਿਚੋਂ ਕੋਈ ਨਹੀਂ

ii. ....ਸਟੇਟਮੈਂਟਸ ਦੀ ਵਰਤੋਂ ਫੈਸਲੇ ਲੈਣ (Decision-Making) ਦੇ ਉਦੇਸ਼ ਲਈ ਜਾਂ ਮਲਟੀ-ਵੇਅ ਸਿਲੈਕਸ਼ਨ (Multi-Way Selection) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ॥

ੳ. ਲੂਪਿੰਗ (Looping) ਅ. ਆਇਟ੍ਰੇਟਿਵ (Iterative)

ੲ. ਕੰਡੀਨਲ (Conditional) ਸ. ਜੰਪਿੰਗ (Jumping)

iii. .... ਇੱਕ ਮਲਟੀ-ਵੇਅ (ਬਹੁ-ਪੱਖੀ) ਬ੍ਰਾਂਚਿੰਗ ਸਟੇਟਮੈਂਟ ਹੈ।

ੳ. switch ਅ. if-else

ੲ. while ਸ. do-while

iv. ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਾਂ ਨੂੰ .... ਸਟੇਟਮੈਂਟਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ੳ. ਆਈਟ੍ਰੇਟਿਵ (Iterative) ਅ. ਸਿਲੈਕਸ਼ਨ (Selection)

ੲ. ਫੈਂਸਲਾ ਲੈਣ ਵਾਲੇ (Decision-Making) ਸ. ਜੰਪਿੰਗ (Jumping)

v. ਹੇਠ ਲਿਖਿਆਂ ਵਿਚੋਂ ਕਿਹੜਾ ਲੂਪ ਜਾਵਾ ਵਿਚ ਪੋਸਟ ਟੈਸਟ ਲੂਪ ਦੀ ਉਦਾਹਰਣ ਹੈ ?

ੳ. for loop   ਅ. while loop

ੲ. do-while   ਸ. ਇਹਨਾਂ ਵਿੱਚੋਂ ਕੋਈ ਨਹੀਂ

vi. 'for' ਅਤੇ 'while' ਲੂਪਸ ਜਾਵਾ ਵਿਚ .... ਲੂਪਸ ਹੁੰਦੇ ਹਨ ॥

ੳ. ਐਂਟਰੀ ਕੰਟਰੋਲਡ (entry-controlled)   ਅ. ਐਗਜ਼ਿਟ ਕੰਟਰੋਲਡ (exit-controlled)

ੲ. ਪੋਸਟ ਟੈਸਟ (Post-test)   ਸ. ਇਹਨਾਂ ਵਿਚੋਂ ਕੋਈ ਨਹੀਂ

vii. break ਅਤੇ continue ਜਾਵਾ ਵਿੱਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ .... ਸਟੇਟਮੈਂਟਸ ਹਨ।

ੳ. ਆਇਟ੍ਰੇਟਿਵ (Iterative)   ਅ. ਸਿਲੈਕਸ਼ਨ (Selection)

ੲ. ਫੈਸਲਾ ਲੈਣ ਵਾਲੇ (Decison-Making)   ਸ. ਜੰਪਿੰਗ (Jumping)

ਪ੍ਰ: II ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ:

i. ਪ੍ਰੀ-ਟੈਸਟ ਲੂਪਸ ਨੂੰ .... ਲੂਪਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ii. if ਜਾਂ if-else ਸਟੇਟਮੈਂਟ ਨੂੰ ਕਿਸੇ ਹੋਰ if ਜਾਂ else ਜਾਂ if-else ਦੇ ਅੰਦਰ ਲਿਖਣਾ ....ਸਟੇਟਮੈਂਟ ਅਖਵਾਉਂਦਾ ਹੈ।

iii. ਲੂਪ ਦੀ ਬਾਡੀ ਲਈ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੀ ਘੱਟੋ-ਘੱਟ ਗਿਣਤੀ (minimum number of executions) ....ਹੁੰਦੀ ਹੈ।

iv. .... ਸਟੇਟਮੈਂਟ ਦੀ ਵਰਤੋਂ ਲੂਪ ਜਾਂ ਸਵਿੱਚ ਸਟੇਟਮੈਂਟ ਨੂੰ ਟਰਮੀਨੇਟ (terminate) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

v. .... ਲੂਪ ਹਮੇਸ਼ਾ ਆਪਣੀ ਬਾਡੀ ਨੂੰ ਘੱਟੋ-ਘੱਟ ਇੱਕ ਵਾਰ ਚਲਾਉਂਦਾ ਹੈ।

vi. .... ਸਟੇਟਮੈਂਟ ਜਾਵਾ ਵਿੱਚ ਵਰਤੀ ਜਾਣ ਵਾਲੀ ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟਸ ਦੀ ਉਦਾਹਰਣ ਹੈ।

ਪ੍ਰ: III ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ (Control Statements) ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।

ii. ਜਾਵਾ ਵਿਚ ਵਰਤੀਆਂ ਜਾਣ ਵਾਲੀਆਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀਆਂ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ ਦੇ ਨਾਂ ਲਿਖੋ।।

iii. ਜਾਵਾ ਵਿਚ ਕੰਡੀਸ਼ਨਲ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ (conditional controls statements) ਕੀ ਹੁੰਦੀਆਂ ਹਨ ?

iv. ਜਾਵਾ ਵਿਚ if-else ਸਟੇਟਮੈਂਟ ਦਾ ਸਿੰਟੈਕਸ ਲਿਖੋ।

v. ਜਾਵਾ ਦੇ switch-case ਸਟੇਟਮੈਂਟਸ ਬਾਰੇ ਤੁਸੀਂ ਕੀ ਜਾਣਦੇ ਹੋ ?

vi. ਆਇਟ੍ਰੇਟਿਵ ਸਟੇਟਮੈਂਟਸ (Iterative Statements) ਕੀ ਹੁੰਦੀਆਂ ਹਨ ?

vii. Pre-test ਲੂਪਸ ਕੀ ਹੁੰਦੇ ਹਨ ? ਉਦਾਹਰਣ ਲਿਖੋ।

viii. while ਅਤੇ do-while ਲੂਪਸ ਦੀ ਤੁਲਨਾ ਕਰੋ।

ix. ਜੰਪਿੰਗ ਸਟੇਟਮੈਂਟਸ (Jumping Statements) ਕੀ ਹੁੰਦੀਆਂ ਹਨ ?

ਪ੍ਰ: IV ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ (Control Statements) ਕੀ ਹੁੰਦੀਆਂ ਹਨ ? ਜਾਵਾ ਵਿੱਚ ਵਰਤੀਆਂ ਜਾਂਦੀਆਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀਆਂ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟਸ ਦੀ ਸੰਖੇਪ ਜਾਣਕਾਰੀ ਦਿਓ।

ii. ਫੈਸਲਾ ਲੈਣ ਵਾਲੀਆਂ ਸਟੇਟਮੈਂਟਸ (Decision-Making Statements) ਕਿਹੜੀਆਂ ਹੁੰਦੀਆਂ ਹਨ ? ਵਰਨਣ ਕਰੋ।

iii. ਜਾਵਾ ਵਿਚ 3 ਅੰਕਾਂ ਵਿਚੋਂ ਸਭ ਤੋਂ ਵੱਡਾ ਅੰਕ ਪਤਾ ਕਰਨ ਲਈ ਇਕ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ।

iv. ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਸ (looping statements) ਕੀ ਹੁੰਦੀਆਂ ਹਨ ? ਲੂਪਿੰਗ ਸਟੇਟਮੈਂਟਾਂ ਦੀਆਂ ਵੱਖ ਵੱਖ ਸ਼੍ਰੇਣੀਆਂ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।

ਕਲਾਸ ਅਤੇ ਆਬਜੈਕਟ
(Class and Object)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

7.1 ਕਲਾਸ (Class)

7.2 ਆਬਜੈਕਟ (Object)

7.3 ਸਟੈਟਿਕ ਮੈਂਬਰ (Static Member)

7.4 ਕੰਸਟਰਕਟਰ (Constructor)

7.5 ਜਾਵਾ ਵਿੱਚ ਓਵਰਲੋਡਿੰਗ -ਮੈਥਡ ਅਤੇ ਕੰਸਟਰਕਟਰ (Methods and Constructors)

ਜਾਣ ਪਛਾਣ (INTRODUCTION) :

BASIC, C ਵਰਗੀਆਂ ਭਾਸ਼ਾਵਾਂ ਨੂੰ ਪ੍ਰੋਸੀਜ਼ਰਲ ਭਾਸ਼ਾਵਾਂ (Procedural Languages) ਮੰਨਿਆ ਜਾਂਦਾ ਹੈ ਜਿਹਨਾਂ ਵਿੱਚ ਹਰੇਕ ਪ੍ਰੋਗਰਾਮ ਫੰਕਸ਼ਨਾਂ ਦੇ ਆਪਸੀ ਸੰਬਧਾਂ ਤੋਂ ਵੱਧ ਕੁਝ ਨਹੀਂ ਹੁੰਦਾ। ਇਹਨਾਂ ਭਾਸ਼ਾਵਾਂ ਵਿੱਚ ਫੰਕਸ਼ਨਾਂ ਅਤੇ ਡੇਟਾ ਮੈਂਬਰਾਂ ਵਿੱਚ ਆਪਸੀ ਤਾਲਮੇਲ (data binding) ਦੀ ਘਾਟ ਹੁੰਦੀ ਹੈ। ਇਸ ਕਿਸਮ ਦੀ ਪ੍ਰੋਗਰਾਮਿੰਗ ਇੱਕ ਮਜਬੂਤ ਐਪਲੀਕੇਸ਼ਨ ਡਿਜ਼ਾਈਨ ਕਰਨ ਵਿੱਚ ਅਸਫ਼ਲ ਰਹਿੰਦੀ ਹੈ। ਇਸ ਕਮੀ ਨੂੰ ਦੂਰ ਕਰਨ ਲਈ, JAVA (ਜਾਵਾ) ਦੁਆਰਾ ਆਬਜੈਕਟ ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਪੈਰਾਡਾਈਮ (Object Oriented Programming paradigm) ਮੁਹੱਇਆ ਕਰਵਾਇਆ ਜਾਂਦਾ ਹੈ ਜਿਸ ਨੂੰ OOP ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇੱਥੇ "ਆਬਜੈਕਟਾਂ" ਦਾ ਅਰਥ ਹੈ ਇੱਕ ਅਸਲ-ਸੰਸਾਰ ਦਾ ਕੋਈ ਤੱਤ ਜਿਵੇਂ ਕਿ ਪੈੱਨ, ਕੁਰਸੀ, ਮੇਜ਼, ਕੰਪਿਊਟਰ, ਘੜੀ, ਆਦਿ। ਇੱਕ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ, ਕਲਾਸ ਅਤੇ ਆਬਜੈਕਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਡਿਜ਼ਾਈਨ ਕਰਨ ਦੀ ਇੱਕ ਵਿਧੀ ਹੁੰਦੀ ਹੈ। ਇਹ ਪ੍ਰੋਗਰਾਮਿੰਗ ਤਕਨੀਕ ਕੁਝ ਸੰਕਲਪ, ਜਿਵੇਂ ਕਿ ਇਨਹੈਰੀਟੈਂਸ (Inheritance), ਪੌਲੀਮੋਰਫਿਜ਼ਮ (Polymorphism), ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) ਆਦਿ ਪ੍ਰਦਾਨ ਕਰਕੇ ਸੌਫਟਵੇਅਰ ਡਿਵੈਲਪਮੈਂਟ ਅਤੇ ਰੱਖ-ਰਖਾਵ ਨੂੰ ਆਸਾਨ ਬਣਾਉਂਦੀ ਹੈ। ਆਉ, ਕਲਾਸ ਅਤੇ ਆਬਜੈਕਟ ਨੂੰ ਸਮਝ ਕੇ OPPs ਬਾਰੇ ਹੋਰ ਜਾਣਕਾਰੀ ਹਾਸਿਲ ਕਰੀਏ।

7.1 ਕਲਾਸ (CLASS):

ਜਾਵਾ ਵਿੱਚ ਕਲਾਸ ਇੱਕ ਯੂਜ਼ਰ ਪਰਿਭਾਸ਼ਿਤ ਪ੍ਰੋਟੋਟਾਈਪ (prototype) ਹੈ ਜਿਸ ਤੋਂ ਆਬਜੈਕਟ ਬਣਾਏ ਜਾਂਦੇ ਹਨ। ਇਹ ਉਹਨਾਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਜਾਂ ਮੈਥਡਜ਼ ਦੇ ਸਮੂਹ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ ਜੋ ਇੱਕ ਕਿਸਮ ਦੇ ਸਾਰੇ ਆਬਜੈਕਟਾਂ ਲਈ ਸਾਂਝੇ ਹੁੰਦੇ ਹਨ। ਆਮ ਤੌਰ ਤੇ ਕਲਾਸ ਨੂੰ ਇੱਕ ਅਜਿਹੇ ਟੈਂਪਲੇਟ (template)/ਬਲੂਪ੍ਰਿੰਟ (blueprint) ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਉਸ ਕਲਾਸ ਦੇ ਆਬਜੈਕਟਸ ਦੁਆਰਾ ਮੁਹੱਇਆ ਕਰਵਾਏ ਜਾਂਦੇ ਵਿਵਹਾਰ (behaviuour) ਅਤੇ/ਜਾਂ ਸਥੀਤੀ (state) ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ "ਜਾਵਾ ਵਿੱਚ ਕਲਾਸ ਇਹ ਨਿਰਧਾਰਤ ਕਰਦੀ ਹੈ ਕਿ ਕੋਈ ਆਬਜੈਕਟ ਕਿਵੇਂ ਵਿਵਹਾਰ (behaviour) ਕਰੇਗਾ ਅਤੇ ਆਬਜੈਕਟ ਵਿੱਚ ਕੀ-ਕੀ ਗੁਣ (Properties) ਸ਼ਾਮਿਲ ਹੋਣਗੇ।" ਇੱਕ ਕਲਾਸ ਵਿੱਚ ਕਿਸੇ ਵੀ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਇੱਕ ਆਬਜੈਕਟ ਨਾਲ ਸਬੰਧਤ ਫੀਲਡਜ਼ (fields) ਅਤੇ ਮੈਥਡਜ਼ (Methods) ਸ਼ਾਮਿਲ ਹੁੰਦੇ ਹਨ। ਅਸੀਂ ਕਲਾਸ ਅਤੇ ਆਬਜੈਕਟਸ ਦੇ ਇਸ ਸੰਕਲਪ ਨੂੰ ਇੱਕ ਚਿੱਤਰ ਦੇ ਰੂਪ ਵਿੱਚ ਇਸ ਤਰ੍ਹਾਂ ਸਮਝ ਸਕਦੇ ਹਾਂ :

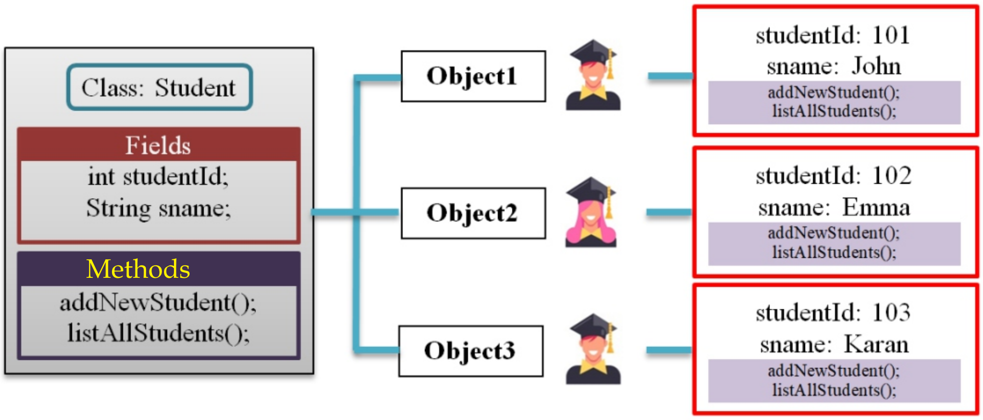


Fig 7.1 Concept of Class and Objects

ਜਾਵਾ ਵਿੱਚ ਤਿਆਰ ਕੀਤੀ ਹਰੇਕ ਕਲਾਸ ਵਿੱਚ ਹੇਠਾਂ ਦਿੱਤੇ ਕੁਝ ਭਾਗ (Components) ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ:

7.1.1 ਜਾਵਾ ਕਲਾਸ ਦੇ ਮੂਲ ਭਾਗ (Basic Components of Java Class):

- ਮੋਡੀਫਾਇਰ (Modifier): ਮੋਡੀਫਾਇਰ ਉਹ ਕੀਅ-ਵਰਡ ਹੁੰਦੇ ਹਨ ਜੋ ਕਲਾਸ ਦੇ ਮੈਂਬਰਾਂ ਤੱਕ ਪਹੁੰਚ ਕਰਨ ਦੇ ਵੱਖ-ਵੱਖ ਅਧਿਕਾਰਾਂ ਦਾ ਵਰਣਨ ਕਰਦੇ ਹਨ। ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਕਲਾਸ ਪਬਲਿਕ (public) ਜਾਂ ਜਾਵਾ ਦੀ ਮੁੱਢਲੀ ਐਕਸੈੱਸ ਕਿਸਮ (default access) ਫਰੈਂਡਲੀ (friengly) ਹੋ ਸਕਦੀ ਹੈ।
- ਕਲਾਸ ਕੀਵਰਡ (class keyword) : class ਕੀਅ-ਵਰਡ, ਕਲਾਸ ਡਿਫਾਇਨ (defiene) ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।
- ਕਲਾਸ ਦਾ ਨਾਮ (class name): ਇਹ ਕਲਾਸ ਦੀ ਪਹਿਚਾਣ ਕਰਵਾਉਣ ਵਾਲਾ ਨਾਮ ਹੁੰਦਾ ਹੈ। ਕਲਾਸ ਦਾ ਇਹ ਨਾਮ ਅੰਗ੍ਰੇਜ਼ੀ ਦੇ ਇੱਕ ਵੱਡੇ ਅੱਖਰ ਨਾਲ ਸ਼ੁਰੂ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ। ਜਾਵਾ ਨਾਮਕਰਨ ਨਿਯਮਾਂ (Naming Ruels) ਦੇ ਅਨੁਸਾਰ ਹਰੇਕ ਕਲਾਸ ਦੇ ਨਾਮ ਦਾ ਪਹਿਲਾ ਅੱਖਰ ਅੰਗ੍ਰੇਜ਼ੀ ਵਰਨਮਾਲਾ ਦਾ ਵੱਡਾ ਅੱਖਰ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।
- ਬਾਡੀ (body): ਕਲਾਸ ਬਾਡੀ ਬਰੈਕਟਾਂ ਨਾਲ ਦਰਸ਼ਾਈ ਜਾਂਦੀ ਹੈ ਜਿਵੇਂ ਕਿ { }। ਜਾਵਾ ਕਲਾਸ ਦੀ ਬਾਡੀ ਦੇ ਹਿੱਸੇ ਵਿੱਚ ਦੋ ਕਿਸਮਾਂ ਦੇ ਐਲੀਮੈਂਟਸ (ਤੱਤ) ਸ਼ਾਮਿਲ ਹੋ ਸਕਦੇ ਹਨ।
  - ਫੀਲਡਸ (Fields)
  - ਮੈਥਡਜ਼ (Methods)

ਜਾਵਾ ਕਲਾਸ ਦੇ ਇਹਨਾਂ ਸਾਰੇ ਭਾਗਾਂ ਵਿੱਚੋਂ ਹਰੇਕ ਹਿੱਸੇ ਦਾ ਆਪਣਾ ਮਹੱਤਵ ਹੁੰਦਾ ਹੈ। ਇੱਕ ਕਲਾਸ ਬਣਾਉਣ ਦਾ ਆਮ ਸਿੰਟੈਕਸ (ਤਰੀਕਾ) ਹੇਠਾਂ ਦਿੱਤਾ ਗਿਆ ਹੈ:

```
[Modifier] class ClassName
{
Java Class body
// Fields
// Methods
}
```

ਨੋਟ: ਉਪਰੋਕਤ ਸਿੰਟੈਕਸ ਵਿੱਚ [ ] ਬਰੈਕਟਾਂ ਵਿਚ ਲਿਖਿਆ **Modifier** ਆਪਸ਼ਨਲ ਭਾਗ ਨੂੰ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ।

Example Program 7.1:

Fields OR Member Variables

```
public class CompApp
{
int ch1,ch2;                          //Instance Variables
static String title;                  // Class Variables
```

Methods

```
    void showTitle()
    {
    int a=10;                         // Local Variables
    System.out.println("Title of Chap. is "+title);
    ------------------------------
    }
}
```

7.1.2 ਕਲਾਸ ਵਿੱਚ ਫੀਲਡਸ (Fields Member Variables in Class) :

ਕਲਾਸ ਦੇ ਅੰਦਰ ਪ੍ਰਭਾਸ਼ਿਤ ਵੇਰੀਏਬਲਸ ਨੂੰ ਫੀਲਡ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਵੱਖ-ਵੱਖ ਐਕਸੈਸ ਸਪੈਸੀਫਾਇਰ (ਪਹੁੰਚ ਨਿਰਧਾਰਤ ਕਰਨ ਵਾਲੇ ਮੋਡੀਫਾਇਰ) ਜਿਵੇਂ ਕਿ ਪ੍ਰਾਈਵੇਟ (private), ਪਬਲਿਕ (public), ਪ੍ਰੋਟੈਕਟਿਡ (protected) ਆਦਿ ਨਾਲ ਫੀਲਡ ਘੋਸ਼ਿਤ ਕਰ ਸਕਦੇ ਹਾਂ। ਅਸੀਂ ਫੀਲਡਜ਼ ਨੂੰ ਹੇਠ ਲਿਖੀਆਂ ਕਿਸਮਾਂ ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕਰ ਸਕਦੇ ਹਾਂ:

- ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲ (Instance Variables) : ਇਹ ਉਹ ਵੇਰੀਏਬਲ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿਸੇ ਆਬਜੈਕਟ ਦੇ ਅੰਦਰੂਨੀ ਵੇਰੀਏਬਲਜ਼ ਹੁੰਦੇ ਹਨ ਅਤੇ ਜਿਨ੍ਹਾਂ ਨੂੰ ਕਿਸੇ ਵੀ ਮੈਥਡ, ਕੰਸਟਰਕਟਰ ਜਾਂ ਬਲਾਕ ਦੇ ਅੰਦਰੋਂ ਐਕਸੈੱਸ (ਪਹੁੰਚ ਕਰਨਾ) ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਜਦੋਂ ਆਬਜੈਕਟ ਦਾ ਸਕੋਪ (scope) ਖਤਮ ਹੋ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਹਨਾਂ ਵੇਰੀਏਬਲਜ਼ ਦੀ ਹੋਂਦ ਵੀ ਨਸ਼ਟ ਹੋ ਜਾਂਦੀ ਹੈ।
- ਕਲਾਸ ਵੇਰੀਏਬਲ (Class Variables) : ਕਲਾਸ ਵੇਰੀਏਬਲ ਜਾਂ ਸਟੈਟਿਕ ਵੇਰੀਏਬਲ, ਇੱਕ ਕਲਾਸ ਵਿੱਚ ਸਟੈਟਿਕ (static) ਕੀਅ-ਵਰਡ ਨਾਲ ਪ੍ਰਭਾਸ਼ਿਤ ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਇਹ ਉਂਝ ਤਾਂ ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲਾਂ ਦੇ ਸਮਾਨ ਹੀ ਹੁੰਦੇ ਹਨ ਪਰੰਤੂ ਜਦੋਂ ਪ੍ਰੋਗਰਾਮ ਸ਼ੁਰੂ ਹੁੰਦਾ ਹੈ ਉਸ ਸਮੇਂ ਹੋਂਦ ਵਿੱਚ ਆਉਂਦੇ ਹਨ ਅਤੇ ਪ੍ਰੋਗਰਾਮ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਖਤਮ ਹੋਣ ਤੇ ਨਸ਼ਟ ਹੁੰਦੇ ਹਨ। ਇਹਨਾਂ ਦਾ ਸਕੋਪ (ਪਹੁੰਚ ਕਰਨ ਦਾ ਦਾਇਰਾ) ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲ ਤੋਂ ਮੁੱਖ ਤੌਰ ਤੇ ਵੱਖਰਾ ਹੁੰਦਾ ਹੈ। ਕਲਾਸ ਵੇਰੀਏਬਲ ਨੂੰ ਕਲਾਸ ਦੇ ਨਾਂ ਨਾਲ ਸਿੱਧਾ ਹੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ ਜਦੋਂ ਕਿ ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲ ਦੀ ਵਰਤੋਂ ਕਲਾਸ ਦੇ ਆਬਜੈਕਟਾਂ ਦੁਆਰਾ ਹੀ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

7.1.3 ਕਲਾਸ ਵਿੱਚ ਮੈਥਡਜ਼ Methods in Class:

ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਹਦਾਇਤਾਂ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦਾ ਹੈ ਜੋ ਇੱਕ ਖਾਸ ਕੰਮ ਕਰਨ ਯੋਗ ਹੁੰਦਾ ਹੈ। ਇਹ ਪ੍ਰੋਗਰਾਮ ਕੋਡ ਦੀ ਮੁੜ ਵਰਤੋਂ (reusability) ਯੋਗਤਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਅਸੀਂ ਇੱਕ ਗੁੰਝਲਦਾਰ ਸਮੱਸਿਆ ਨੂੰ ਛੋਟੇ ਹਿੱਸਿਆਂ ਵਿੱਚ ਵੰਡ ਸਕਦੇ ਹਾਂ ਜਿਸਨੂੰ ਮਡਿਊਲ (Module) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਸਾਡੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸਮਝਣ ਵਿੱਚ ਆਸਾਨ ਅਤੇ ਮੁੜ ਵਰਤੋਂ ਯੋਗ ਬਣਾਉਂਦਾ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਦੋ ਮੁੱਖ ਕਿਸਮਾਂ ਦੇ ਮੈਥਡ ਵਰਤੇ ਜਾਂਦੇ ਹਨ:

- ਸਟੈਂਡਰਡ ਲਾਇਬ੍ਰੇਰੀ ਮੈਥਡਜ਼ (Standard Library Methods) : ਇਸ ਕਿਸਮ ਦੇ ਮੈਥਡਜ਼ ਨੂੰ ਪ੍ਰੀ-ਪਰਿਭਾਸ਼ਿਤ (Pre-defined) ਮੈਥਡਜ਼ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਇਹ ਬਿਲਟ-ਇਨ ਮੈਥਡਜ਼ ਪਹਿਲਾਂ ਹੀ ਜਾਵਾ-ਲਾਇਬ੍ਰੇਰੀ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤੇ ਗਏ ਹੁੰਦੇ ਹਨ। ਇਸ ਕਿਸਮ ਦੇ ਮੈਥਡਜ਼ ਦੀ ਉਦਾਹਰਨ java.io.printStream ਕਲਾਸ ਅਧੀਨ ਆਉਂਦਾ print() ਮੈਥਡ ਹੋ ਸਕਦਾ ਹੈ ਜੋ ਡਬਲ ਕੋਟਸ ਅੰਦਰ ਲਿਖੀ ਗਈ ਸਟਰਿੰਗ ਨੂੰ ਪ੍ਰਿੰਟ ਕਰਦਾ ਹੈ। ਇਸਦੇ ਨਾਲ ਹੀ sqrt() ਮੈਥਡ, Math ਕਲਾਸ ਦਾ ਇੱਕ ਹੋਰ ਮੈਥਡ ਹੈ ਜੋ ਇੱਕ ਆਰਗੂਮੈਂਟ ਦੇ ਤੌਰ ਤੇ ਦਿੱਤੇ ਗਏ ਕਿਸੇ ਵੀ ਖਾਸ ਨੰਬਰ ਦਾ ਵਰਗਮੂਲ (square root) ਪਤਾ ਕਰਕੇ ਵਾਪਸ ਕਰਦਾ ਹੈ।

- **ਯੂਜ਼ਰ-ਪਰਿਭਾਸ਼ਿਤ ਮੈਥਡ (User-Defined Method) :** ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਆਪਣੀਆਂ ਲੋੜਾਂ ਦੇ ਆਧਾਰ ਤੇ ਆਪਣਾ ਮੈਥਡ ਬਣਾ ਸਕਦੇ ਹਾਂ। ਯੂਜ਼ਰ ਦੁਆਰਾ ਤਿਆਰ ਕੀਤੇ ਗਏ ਸਾਰੇ ਮੈਥਡਜ਼ ਨੂੰ "ਯੂਜ਼ਰ ਪਰਿਭਾਸ਼ਿਤ ਮੈਥਡ" (User-Defined Method) ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਆਪਣੇ ਆਬਜੈਕਟਸ ਨਾਲ ਸੰਬੰਧਤ ਵੱਖ ਵੱਖ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਸਾਰੇ ਕੰਮਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ ਜਾਵਾ ਕਲਾਸ ਵਿੱਚ ਇੱਕ ਤੋਂ ਵੱਧ ਮੈਥਡ ਬਣਾ ਸਕਦੇ ਹਾਂ।

  ਯੂਜ਼ਰ ਲੋੜੀਂਦੇ ਕੰਮਾਂ ਨੂੰ ਕਰਨ ਲਈ ਸਟੇਟਮੈਂਟਸ ਅਤੇ ਲੋਕਲ ਵੇਰੀਏਬਲਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਲੋੜ ਅਨੁਸਾਰ ਮੈਥਡਜ਼ ਡਿਜ਼ਾਈਨ ਕਰ ਸਕਦਾ ਹੈ।ਅਸੀਂ ਇੱਕ ਮੈਥਡ ਵਿੱਚ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਲੋਕਲ ਵੇਰੀਏਬਲਜ਼ ਨੂੰ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਪਰਿਭਾਸ਼ਿਤ ਕਰ ਸਕਦੇ ਹਾਂ:
  - ਲੋਕਲ ਵੇਰੀਏਬਲ (Local variable) : ਇਸ ਕਿਸਮ ਦੇ ਵੇਰੀਏਬਲ ਮੈਥਡਜ਼ ਦੇ ਅੰਦਰ ਪਰਿਭਾਸ਼ਿਤ ਅਸਥਾਈ ਵੇਰੀਏਬਲ (temporary variables) ਹੁੰਦੇ ਹਨ।ਇਹਨਾਂ ਨੂੰ ਉਸ ਮੈਥਡ ਦੇ ਅੰਦਰ ਹੀ ਡਿਕਲੇਅਰ (ਘੋਸ਼ਿਤ) ਅਤੇ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ (ਮੁੱਢਲਾ ਮੁੱਲ ਦੇਣਾ) ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਮੈਥਡ ਦੇ ਲਾਗੂ ਹੋਣ ਤੋਂ ਬਾਅਦ ਉਹਨਾਂ ਨੂੰ ਗਾਰਵੇਜ਼ ਕੁਲੈਕਸ਼ਨ (ਕੰਮ ਪੂਰਾ ਹੋਣ ਉਪਰੰਤ ਰੋਕੀ ਗਈ ਮੈਮਰੀ ਨੂੰ ਖਾਲੀ ਕਰਨ ਦੀ ਪ੍ਰੀਕਿਰਆ) ਯੋਗ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਵੇਰੀਏਬਲ ਉਹਨਾਂ ਮੈਥਡਜ਼ ਤੋਂ ਬਾਹਰ ਪਹੁੰਚਯੋਗ ਨਹੀਂ ਹੁੰਦੇ ਜਿਹਨਾਂ ਵਿੱਚ ਇਹਨਾਂ ਨੂੰ ਘੋਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੁੰਦਾ ਹੈ।

7.1.4 ਐਕਸੈਸ ਮੋਡੀਫਾਇਰ (Access Modifiers) :

ਜਾਵਾ ਵਿੱਚ ਐਕਸੈਸ ਮੋਡੀਫਾਇਰ, ਇੱਕ ਫੀਲਡ, ਮੈਥਡ, ਕੰਸਟਰਕਟਰ ਜਾਂ ਕਲਾਸ ਦੀ ਪਹੁੰਚਯੋਗਤਾ (Accessibility) ਜਾਂ ਦਾਇਰੇ (Scope) ਨੂੰ ਦਰਸਾਉਂਦੇ ਹਨ॥ ਅਸੀਂ ਵੱਖ ਵੱਖ ਮੋਡੀਫਾਇਰਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਤੱਤਾਂ (elements) ਦੇ ਐਕਸੈਸ ਲੈਵਲ (Access level) ਨੂੰ ਬਦਲ ਸਕਦੇ ਹਾਂ।ਜਾਵਾ ਵਿੱਚ ਮੁੱਖ ਤੌਰ ਤੇ ਚਾਰ ਐਕਸੈਸ ਮੋਡੀਫਾਇਰ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

- **ਪ੍ਰਾਇਵੇਟ (Private) :** ਇਸ ਨਾਲ ਨਿਰਧਾਰਿਤ ਤੱਤ ਸਿਰਫ਼ ਉਸੇ ਕਲਾਸ ਦੇ ਅੰਦਰ ਪਹੁੰਚਯੋਗ ਹੁੰਦੇ ਹਨ।
- **ਡਿਫਾਲਟ (Default) :** ਇਸ ਨਾਲ ਨਿਰਧਾਰਿਤ ਤੱਤ ਕੇਵਲ ਉਸੇ ਕਲਾਸ ਅਤੇ ਉਸੇ ਹੀ ਪੈਕੇਜ ਵਿੱਚ ਪਹੁੰਚਯੋਗ ਹੁੰਦੇ ਹਨ।
- **ਪ੍ਰੋਟੈਕਟਿਡ (Protected) :** ਇਸ ਨਾਲ ਨਿਰਧਾਰਿਤ ਤੱਤ ਕੇਵਲ ਉਸੇ ਕਲਾਸ, ਉਸੇ ਪੈਕਜ਼ ਅਤੇ ਸਿਰਫ਼ ਸਬ-ਕਲਾਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪੈਕੇਜ ਤੋਂ ਬਾਹਰ ਪਹੁੰਚਯੋਗ ਹੋ ਸਕਦੇ ਹਨ।
- **ਪਬਲਿਕ (Public) :** ਇਸ ਨਾਲ ਨਿਰਧਾਰਿਤ ਤੱਤ, ਕਲਾਸ ਅਤੇ ਪੈਕੇਜ ਤੋਂ ਬਾਹਰ ਕਿਤੇ ਵੀ ਪਹੁੰਚਯੋਗ ਹੋ ਸਕਦੇ ਹਨ।

ਨੋਟ : ਪੈਕੇਜ਼ ਇੱਕੋ ਤਰ੍ਹਾਂ ਦੀਆਂ ਕਲਾਸਾਂ, ਇੰਟਰਫੇਜ਼ਿਜ਼ ਅਤੇ ਸਬ-ਪੈਕੇਜਾਂ ਦਾ ਸਮੂਹ ਹੁੰਦਾ ਹੈ। ਇਸ ਨੂੰ ਫਾਈਲ ਸਿਸਟਮ ਦੇ ਇੱਕ ਫੋਲਡਰ ਦੀ ਤਰ੍ਹਾਂ ਮੰਨਿਆ ਜਾ ਸਕਦਾ ਹੈ।

Access Modifier	Within Class	Within Package	Outside Package by Subclass only	Outside Package
Private	Yes	No	No	No
Default	Yes	Yes	No	No
Protected	Yes	Yes	Yes	No
Public	Yes	Yes	Yes	Yes

Table 7.1 ਜਾਵਾਂ ਵਿੱਚ ਐਕਸੈਸ ਮਾਡੀਫਾਇਰ

ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ, ਜਾਵਾ ਕਲਾਸ ਦੇ ਸਾਧਾਰਨ ਪ੍ਰੋਗਰਾਮ ਅਤੇ ਇਸਦੇ ਭਾਗਾਂ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ :

```
class FindArea
{
        float radius, area;                //Fields
        void getRadius(float r)            //Method
        {
                radius=r;
        }
```

```
void showArea ()                    //Method
        {
                final float PIE=(float)3.143;
                area=PIE*radius*radius;
                System.out.println("The Area of Circle is: "+area);
        }
}
```

7.2 ਆਬਜੈਕਟਸ (OBJECTS) :

ਜਾਵਾ ਆਬਜੈਕਟ ਜਾਵਾ ਕਲਾਸ ਦਾ ਇੱਕ ਮੈਂਬਰ ਹੁੰਦਾ ਹੈ ਜਿਸ ਨੂੰ ਇੰਸਟੈਂਸ (instance) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਹਰੇਕ ਜਾਵਾ ਆਬਜੈਕਟ ਦੀ ਆਪਣੀ ਇੱਕ ਪਛਾਣ (identity), ਇੱਕ ਵਿਵਹਾਰ (behavour) ਅਤੇ ਇੱਕ ਸਟੇਟ (state) ਹੁੰਦੀ ਹੈ। ਪਛਾਣ ਹਰੇਕ ਆਬਜੈਕਟ ਨੂੰ ਨਿਰਧਾਰਤ ਕੀਤੀ ਅੰਦਰੂਨੀ ਆਈ. ਡੀ. (ID) ਹੁੰਦੀ ਹੈ, ਕਿਸੇ ਆਬਜੈਕਟ ਦੀ ਸਟੇਟ ਫੀਲਡਜ਼ (ਵੇਰੀਏਬਲ) ਵਿੱਚ ਸਟੋਰ ਕੀਤੀ ਜਾਂਦੀ ਹੈ, ਜਦੋਂ ਕਿ ਮੈਥਡਜ਼ (ਫੰਕਸ਼ਨ) ਆਬਜੈਕਟ ਉੱਪਰ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਵੱਖ-ਵੱਖ ਆਪ੍ਰੇਸ਼ਨਜ਼ (Operations) ਨੂੰ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਦੇ ਹਨ। ਪ੍ਰੋਗਰਾਮ ਦੇ ਲਾਗੂਕਰਨ (Execution) ਸਮੇਂ ਟੈਂਪਲੇਟਸ (templates) ਤੋਂ ਆਬਜੈਕਟ ਬਣਾਏ ਜਾਂਦੇ ਹਨ, ਇਨ੍ਹਾਂ ਟੈਪਲੇਟਸ ਨੂੰ ਕਲਾਸ (Class) ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਹਰੇਕ ਕਲਾਸ ਆਬਜੈਕਟ-ਓਰੀਐਂਟਡ ਪ੍ਰੋਗਰਾਮਿੰਗ ਦੀ ਇੱਕ ਬੁਨਿਆਦੀ ਇਕਾਈ (Unit) ਹੁੰਦੀ ਹੈ ਅਤੇ ਰੀਅਲ ਲਾਇਫ ਦੀਆਂ ਵਸਤੂਆਂ/ ਤੱਤਾਂ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ। ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਜਾਣਦੇ ਹਾਂ, ਇੱਕ ਆਮ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਹੁਤ ਸਾਰੇ ਆਬਜੈਕਟ ਬਣਾਉਂਦਾ ਹੈ, ਜੋ ਕਿ ਮੈਥਡਜ਼ ਦੁਆਰਾ ਇੰਟਰੈਕਟ ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਜੇਕਰ ਅਸੀਂ ਰੀਅਲ ਲਾਇਫ ਦੇ ਆਧਾਰ ਤੇ ਵਿਚਾਰ ਕਰੀਏ, ਤਾਂ ਅਸੀਂ ਆਪਣੇ ਆਲੇ-ਦੁਆਲੇ ਬਹੁਤ ਸਾਰੀਆਂ ਵਸਤੂਆਂ ਜਿਵੇਂ ਕਿ ਕੁੱਤੇ, ਕਾਰਾਂ, ਮਨੁੱਖ ਆਦਿ ਦੇਖ ਸਕਦੇ ਹਾਂ। ਇਨ੍ਹਾਂ ਸਾਰੀਆਂ ਵਸਤੂਆਂ (ਆਬਜੈਕਟਸ) ਦੀ ਇੱਕ ਅਵਸਥਾ (state) ਅਤੇ ਇੱਕ ਵਿਵਹਾਰ (behaviour) ਹੁੰਦਾ ਹੈ। ਆਬਜੈਕਟ ਦੀ ਕੁੱਤੇ ਰੂਪ ਵਿੱਚ ਇੱਕ ਉਦਾਹਰਣ ਵਜੋਂ ਇਸਦੀਆਂ ਕੁਝ ਅਵਸਥਾਵਾਂ ਹੋ ਸਕਦੀਆਂ ਹਨ ਜਿਵੇਂ ਨਾਮ, ਨਸਲ, ਰੰਗ ਅਤੇ ਵਿਵਹਾਰ ਵਜੋਂ ਭੌਂਕਣਾ, ਪੂੰਛ ਹਿਲਾਉਣਾ, ਦੌੜਨਾ ਆਦਿ। ਇਸੇ ਤਰ੍ਹਾਂ ਸਾਫਟਵੇਅਰ ਡਿਵੈਲਪਮੈਂਟ ਵਿੱਚ ਮੈਥਡਜ਼ ਇੱਕ ਵਸਤੂ ਦੀ ਅੰਦਰੂਨੀ ਸਟੇਟ (ਫੀਲਜ਼) ਤੇ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਕੰਮ ਨੂੰ ਦਰਸ਼ਾਉਂਦੇ ਹਨ ਅਤੇ ਦੋ ਆਬਜੈਕਟਾਂ ਦੇ ਆਪਸੀ ਸੰਚਾਰ ਦੇ ਤਰੀਕੇ ਵੀ ਮੁਹੱਇਆ ਕਰਵਾਉਂਦੇ ਹਨ। ਇੱਕ ਆਬਜੈਕਟ ਦੀਆਂ ਦੋ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹੋ ਸਕਦੀਆਂ ਹਨ:

- **ਸਟੇਟ (State) :** ਇਹ ਕਿਸੇ ਆਬਜੈਕਟ ਦੇ ਹਰੇਕ ਫੀਲਡ (field) ਵਿੱਚ ਸਟੋਰ ਕੀਤੇ ਡੇਟਾ (ਮੁੱਲ) ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ।
- **ਵਿਵਹਾਰ (Behaviour) :** ਮੈਥਡ ਦੇ ਰੂਪ ਵਿੱਚ ਦਿੱਤੇ ਗਏ ਇੱਕ ਆਬਜੈਕਟ ਦੀ ਕਾਰਜਕੁਸ਼ਲਤਾ (functionality) ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ਜਿਵੇਂ ਕਿ readvalue(), displayValue() ਆਦਿ।

ਅਸੀਂ ਬਾਅਦ ਵਿੱਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੇ ਰੂਪ ਵਿੱਚ ਇਹਨਾਂ ਸਾਰੀਆਂ ਟਰਮਜ਼ (terms) ਨੂੰ ਦੇਖਾਂਗੇ ਅਤੇ ਹਰ ਇੱਕ ਦੀ ਮਹੱਤਤਾ ਨੂੰ ਸਮਝਣ ਦੀ ਕੋਸ਼ਿਸ਼ ਕਰਾਂਗੇ।

7.2.1 ਜਾਵਾ ਵਿੱਚ new ਕੀਵਰਡ ਦੀ ਵਰਤੋਂ (Use of new Keyword in Java) :

ਜਾਵਾ ਵਿੱਚ new ਕੀਅ-ਵਰਡ ਕਲਾਸ ਦਾ ਇੰਸਟੈਂਸ (instance) ਜਿਸ ਨੂੰ ਆਬਜੈਕਟ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ, ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ ਇਹ ਕੀਅ-ਵਰਡ, ਇੱਕ ਨਵੇਂ ਆਬਜੈਕਟ ਲਈ ਮੈਮੋਰੀ ਨਿਰਧਾਰਤ (allocate) ਕਰਕੇ ਉਸ ਮੈਮੋਰੀ ਦਾ ਰੈਫਰੈਂਸ (reference) ਵਾਪਸ ਕਰਦਾ ਹੈ। ਅਸੀਂ new ਕੀਅ-ਵਰਡ ਦੀ ਮਦਦ ਨਾਲ ਆਬਜੈਕਟ ਦਾ ਐਰੇ (array) ਵੀ ਬਣਾ ਸਕਦੇ ਹਾਂ। new ਕੀਅ-ਵਰਡ ਦੀਆਂ ਕੁਝ ਮੁੱਖ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ:

- ਇਹ ਇੱਕ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।
- ਇਹ ਰਨਟਾਈਮ (runtime) ਤੇ ਮੈਮੋਰੀ ਨਿਰਧਾਰਤ ਕਰਦਾ ਹੈ।
- ਇਹ ਸਾਰੇ ਆਬਜੈਕਟਸ ਲਈ ਹੀਪ ਖੇਤਰ (heap area) ਵਿੱਚੋਂ ਮੈਮੋਰੀ ਨਿਰਧਾਰਿਤ ਕਰਦਾ ਹੈ।
- ਇਹ ਆਬਜੈਕਟ ਨਾਲ ਸੰਬੰਧਤ ਕੰਸਟਰਕਟਰ (Constructor) ਨੂੰ ਚਲਾਉਂਦਾ (invoke) ਹੈ।

7.2.2 ਜਾਵਾ ਵਿੱਚ ਆਬਜੈਕਟ ਬਣਾਉਣਾ (Creating Object in Java):

ਆਬਜੈਕਟਸ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਦੇ ਇੱਕ ਬੁਨਿਆਦੀ ਬਿਲਡਿੰਗ ਬਲਾਕ (building block) ਹੁੰਦੇ ਹਨ। ਅਸੀਂ ਇੱਕ ਆਬਜੈਕਟ ਬਣਾਏ ਬਿਨਾਂ ਕਿਸੇ ਵੀ ਕਲਾਸ ਦੇ ਮੈਥਡਜ਼ ਅਤੇ ਫੀਲਡਜ਼ ਦੀ ਵਰਤੋਂ ਨਹੀਂ ਕਰ ਸਕਦੇ। ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਬਣਾਉਣ ਦੇ ਕਈ ਤਰੀਕੇ ਹਨ। ਆਮ ਤੌਰ ਤੇ ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਬਣਾਉਣ ਲਈ new ਕੀਵਰਡ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ। ਇਹ ਨਵੇਂ ਬਣਾਏ ਗਏ ਆਬਜੈਕਟ ਲਈ ਮੈਮੋਰੀ ਨਿਰਧਾਰਤ (allocate) ਕਰਦਾ ਹੈ ਅਤੇ ਉਸ ਆਬਜੈਕਟ ਨਾਲ ਸੰਬੰਧਤ ਮੈਮੋਰੀ ਦਾ ਰੈਫਰੈਂਸ (reference) ਵੀ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਇੱਕ ਆਬਜੈਕਟ ਬਣਾਉਣ ਲਈ ਸਿੰਟੈਕਸ ਇਸ ਪ੍ਰਕਾਰ ਹੈ:

new ਕੀਵਰਡ ਦੀ ਵਰਤੋਂ ਨਾਲ ਆਬਜੈਕਟ ਬਣਾਉਣ ਦਾ ਸਿੰਟੈਕਸ :

```
ClassName Object_Name=new ClassName ([argument list-if any]);
Or
ClassName Object_Name;
Object_Name =new ClassName ([argument list-if any]);
```

ਉਦਾਹਰਣ ਲਈ :

```
FindArea obj=new FindArea();
Or
FindArea obj;
obj=new FindArea();
```

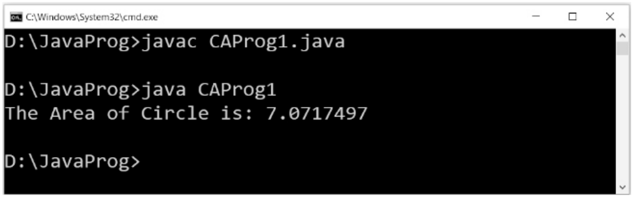
ਹੁਣ ਅਸੀਂ ਕਲਾਸ ਅਤੇ ਆਬਜੈਕਟਸ ਦੀ ਧਾਰਨਾ ਨੂੰ ਪੂਰੀ ਤਰਾਂ ਸਮਝ ਲਿਆ ਹੈ। ਆਉ ਜਾਵਾ ਵਿੱਚ ਕਲਾਸ ਅਤੇ ਆਬਜੈਕਟਸ ਨੂੰ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਨ ਲਈ ਇੱਕ ਸਧਾਰਨ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਬਣਾਈਏ।

ਪ੍ਰੋਗਰਾਮ 7.2 (CAProg1.java) ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਕਲਾਸ ਦੀ ਰਚਨਾ ਅਤੇ ਵਰਤੋਂ ਲਈ ਪ੍ਰੋਗਰਾਮ

```
class FindArea
{
        float radius, area;                         //Fields
        void getRadius(float r)                     //Method
        {
                radius=r;
        }
        void showArea ()                            //Method
        {
        final float PIE=(float)3.143;  //using type casting
        area=PIE*radius*radius;
        System.out.println("The Area of Circle is: "+area);
```

```
        }
}//end of Find Area Class
class CAProg1
{
        public static void main(String arg[])        {
                FindAreaobj=new FindArea();                //creating object
                obj.getRadius(1.5f);
                obj.showArea();
        }
}
```

ਪ੍ਰੋਗਰਾਮ 7.2 (CAProg1.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ



```
C:\Windows\System32\cmd.exe
D:\JavaProg>javac CAProg1.java

D:\JavaProg>java CAProg1
The Area of Circle is: 7.0717497

D:\JavaProg>
```

7.3 ਸਟੈਟਿਕ ਮੈਂਬਰਜ਼ (ਕਲਾਸ ਵੇਰੀਏਬਲਜ਼) (STATIC MEMBERS (CLASS VARIABLES)) :

ਸਟੈਟਿਕ ਮੋਡੀਫਾਇਰ ਤੋਂ ਭਾਵ ਹੈ ਉਹ ਇਕਾਈ (entity) ਜਿਸ ਤੇ ਇਸ ਨੂੰ ਲਾਗੂ ਕੀਤਾ ਗਿਆ ਹੈ, ਕਲਾਸ ਦੇ ਕਿਸੇ ਖਾਸ ਇਨਸਟੈਂਸ (instance) ਤੋਂ ਬਾਹਰ ਉਪਲਬਧ ਹੈ। ਇਸਦਾ ਅਰਥ ਇਹ ਵੀ ਹੈ ਕਿ ਸਟੈਟਿਕ ਮੈਥਡ ਜਾਂ ਫੀਲਡ ਉਸ ਖਾਸ ਕਲਾਸ ਦਾ ਹਿੱਸਾ ਹਨ ਨਾ ਕਿ ਕਿਸੇ ਆਬਜੈਕਟ ਦਾ ਹਿੱਸਾ ਹਨ। ਕਲਾਸ ਲੋਡਿੰਗ ਦੇ ਸਮੇਂ ਮੈਮੋਰੀ ਅਜਿਹੇ ਫੀਲਡ ਜਾਂ ਮੈਥਡ ਨੂੰ ਨਿਰਧਾਰਤ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇੱਕ ਸਟੈਟਿਕ ਮੋਡੀਫਾਇਰ ਦੀ ਵਰਤੋਂ ਮੈਮੋਰੀ ਨੂੰ ਬਚਾ ਕੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਵਧੇਰੇ ਕੁਸ਼ਲ ਬਣਾਉਂਦਾ ਹੈ। ਇੱਕ ਸਟੈਟਿਕ ਫੀਲਡ ਸਾਰੇ ਕਲਾਸ ਆਬਜੈਕਟਸ ਵਿੱਚ ਸਾਂਝੇ ਤੌਰ ਤੇ ਮੌਜੂਦ ਹੁੰਦਾ ਹੈ ਅਤੇ ਇਸਨੂੰ ਕਲਾਸ ਦੇ ਕਿਸੇ ਵੀ ਆਬਜੈਕਟ ਤੋਂ ਬਿਨਾਂ ਵਰਤਿਆ (call) ਜਾਂਦਾ ਹੈ। ਇੱਕ ਸਟੈਟਿਕ ਮੋਡੀਫਾਇਰ ਦੀ ਵਰਤੋਂ ਨੂੰ ਹੇਠਾਂ ਦਿੱਤੇ ਪ੍ਰੋਗਰਾਮ ਦੇ ਰੂਪ ਵਿੱਚ ਸਮਝਿਆ ਜਾ ਸਕਦਾ ਹੈ:

ਪ੍ਰੋਗਰਾਮ 7.3 (CAProg2.java) ਸਟੈਟਿਕ ਵੇਰੀਏਬਲ/ਕਲਾਸ ਵੇਰੀਏਬਲ ਲਈ ਪ੍ਰੋਗਰਾਮ

```
class StaticDataMember
{
        int Variable=10;
        static int Static_Member=10;
        void showValue()
        {
                System.out.println("Variable: "+Variable);
                System.out.println("Static_Member: "+Static_Member);
```

```
		}
		void incValue()
		{
			Variable++;
			Static_Member++;
		}
	}
	class CAProg2 {
		public static void main(String arg[]) {
			StaticDataMember obj1=new StaticDataMember();
			obj1.showValue();
			obj1.incValue();
			obj1.showValue();
			obj1.incValue();
			StaticDataMember obj2=new StaticDataMember();
			obj2.showValue();
			obj2.incValue();
			obj1.showValue();
		}
	}
```

ਪ੍ਰੋਗਰਾਮ 7.3 (CAProg2.java) ਦੀ ਕੰਪਾਈਲੇਸ਼ਨ, ਐਗਜੀਕਿਊਸ਼ਨ ਅਤੇ ਆਊਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg2.java

D:\JavaProg>java CAProg2
Non_Static: 10
Static_Member: 10
Non_Static: 11
Static_Member: 11
Non_Static: 10
Static_Member: 12
Non_Static: 12
Static_Member: 13

D:\JavaProg>
```

ਦਿੱਤੀ ਗਈ ਉਦਾਹਰਨ ਵਿੱਚ ਅਸੀਂ ਕਲਾਸ ਵਿੱਚ ਸਟੈਟਿਕ ਅਤੇ ਇੰਸਟੈਂਸ ਡੇਟਾ ਫੀਲਡ ਦੀ ਮਹੱਤਤਾ ਨੂੰ ਸਪਸ਼ਟ ਰੂਪ ਵਿੱਚ ਦੇਖ ਸਕਦੇ ਹਾਂ। ਅਸੀਂ ਦਿੱਤੀ ਗਈ ਕਲਾਸ StaticDataMember ਵਿੱਚ Variable ਅਤੇ Static_Member ਨਾਮ ਦੇ ਦੋ ਵੱਖ-ਵੱਖ ਵੇਰੀਏਬਲ ਘੋਸ਼ਿਤ (declare) ਕੀਤੇ ਹਨ। ਅਸੀਂ ਦੋ ਮੈਥਡ ਬਣਾਏ ਹਨ, showValue(); ਨਾਮ ਦਾ ਮੈਥਡ ਇਹਨਾਂ ਦੋਵਾਂ ਫੀਲਡਜ਼ ਦੇ ਮੁੱਲਾਂ ਨੂੰ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਨ ਲਈ ਅਤੇ incValue(); ਨਾਮ ਦਾ ਮੈਥਡ ਇਹਨਾਂ ਦੋਵਾਂ ਵੇਰੀਏਬਲਾਂ ਦੇ ਮੁੱਲਾਂ ਵਿਚ ਇੰਕਰੀਮੈਂਟ ਕਰਨ ਲਈ ਬਣਾਇਆ ਹੋਇਆ ਹੈ। ਅਸੀਂ obj1 ਅਤੇ obj2 ਨਾਮ ਦੇ ਦੋ ਵੱਖ-ਵੱਖ ਆਬਜੈਕਟ ਬਣਾਏ ਹਨ। ਅਸੀਂ ਇਹਨਾਂ ਦੋਵਾਂ ਆਬਜੈਕਟਸ ਦੇ ਨਾਲ ਦਿੱਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਕਿਸੇ ਖਾਸ ਕ੍ਰਮ ਵਿੱਚ ਚਲਾਉਂਦੇ ਹਾਂ। ਅਸੀਂ ਦੇਖ ਸਕਦੇ ਹਾਂ ਕਿ ਕਲਾਸ ਦਾ Satic_Member ਉਸ ਆਬਜੈਕਟ ਤੋਂ ਮੁਕਤ ਹੈ ਜਿਸ ਨਾਲ ਇਸਨੂੰ ਚਲਾਇਆ ਜਾਂਦਾ ਹੈ। ਸਟੈਟਿਕ ਵੇਰੀਏਬਲ ਦਾ ਮੁੱਲ ਸਾਰੇ ਆਬਜੈਕਟਸ ਵਿੱਚ ਸਾਂਝੇ ਤੌਰ ਤੇ ਵਰਤਿਆ ਜਾ ਰਿਹਾ ਹੈ। ਦੂਜੇ ਪਾਸੇ, Variable (Non_Static) (ਇਨਸਟੈਂਸ

ਵੇਰੀਏਬਲ) ਆਬਜੈਕਟ ਨਾਲ ਜੁੜਿਆ ਹੋਇਆ ਹੈ। ਜਦੋਂ ਵੱਖ-ਵੱਖ ਆਬਜੈਕਟਸ ਨਾਲ ਇਸ ਵੇਰੀਏਬਲ ਨੂੰ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਹ ਉਸ ਆਬਜੈਕਟ ਦੇ ਸੰਬੰਧਤ ਵੱਖ-ਵੱਖ ਮੁੱਲ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਦਾ ਹੈ। ਇਹ ਹੀ ਸਟੈਟਿਕ ਵੇਰੀਏਬਲ ਦਾ ਮੁੱਖ ਕੰਮ ਹੁੰਦਾ ਹੈ ਜੋ ਫੀਲਡ ਨੂੰ ਇੱਕੋ ਕਲਾਸ ਦੇ ਸਾਰੇ ਆਬਜੈਕਟਸ ਵਿੱਚ ਸਾਂਝਾ ਕਰਦਾ ਹੈ।

7.4 ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰ (CONSTRUCTORS IN JAVA)

ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਇੱਕ ਵਿਸ਼ੇਸ਼ ਮੈਥਡ ਹੁੰਦਾ ਹੈ ਜੋ ਆਬਜੈਕਟ ਦੇ ਮੈਂਬਰਾਂ ਨੂੰ ਸ਼ੁਰੂਆਤੀ ਮੁੱਲ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ।ਕੰਸਟਰਕਟਰ ਉਸ ਸਮੇਂ ਆਪਣੇ ਆਪ ਕਾਲ (Call) ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਕਿਸੇ ਸੰਬੰਧਤ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਸਦੀ ਵਰਤੋਂ ਆਬਜੈਕਟ ਦੀਆਂ ਪ੍ਰਾਪਰਟੀਜ਼(Fields) ਲਈ ਸ਼ੁਰੂਆਤੀ ਮੁੱਲ ਸੈੱਟ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਜਾਵਾ ਵਿੱਚ, ਕੰਸਟਰਕਟਰ ਸਟੇਟਮੈਂਟਸ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦਾ ਹੈ ਜੋ ਮੈਥਡ ਵਰਗਾ ਹੀ ਹੁੰਦਾ ਹੈ, ਪਰ ਕੁਝ ਖਾਸ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਰੱਖਦਾ ਹੈ। ਕੰਸਟਰਕਟਰ ਸਵੈਚਲਿਤ ਹੁੰਦਾ ਹੈ। ਜਦੋਂ ਕਲਾਸ ਦਾ ਇੱਕ ਆਬਜੈਕਟ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ ਤਾਂ ਆਪਣੇ ਆਪ ਹੀ ਸੰਬੰਧਤ ਕੰਸਟਰਕਟਰ ਕਾਲ (invoke) ਹੋ ਜਾਂਦਾ ਹੈ। ਕੰਸਟਰਕਟਰ ਨੂੰ ਕਾਲ ਕਰਨ ਸਮੇਂ ਮੈਮੋਰੀ ਵਿੱਚ ਆਬਜੈਕਟ ਲਈ ਮੈਮਰੀ ਨਿਰਧਾਰਤ (allocate) ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਹ ਇੱਕ ਖਾਸ ਕਿਸਮ ਦਾ ਮੈਥਡ ਹੈ ਜੋ ਮੁੱਖ ਤੌਰ 'ਤੇ ਆਬਜੈਕਟ ਨੂੰ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ (initialize) ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ।ਅਸੀਂ ਮੈਥਡ ਅਤੇ ਕੰਸਟਰਕਟਰ ਵਿਚਲੇ ਅੰਤਰਾਂ ਨੂੰ ਹੇਠਾਂ ਦਰਸ਼ਾਏ ਅਨੁਸਾਰ ਸਮਝ ਸਕਦੇ ਹਾਂ:

ਮੈਥਡ	ਕੰਸਟਰਕਟਰ
ਮੈਥਡ ਨੂੰ ਇਸ ਦੁਆਰਾ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਕੰਮ ਦੇ ਅਨੁਸਾਰ ਨਾਮ ਦਿੱਤਾ ਜਾ ਸਕਦਾ ਹੈ।	ਕੰਸਟਰਕਟਰ ਦਾ ਨਾਮ ਹਮੇਸ਼ਾ ਉਸ ਕਲਾਸ ਦਾ ਨਾਮ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਇਸ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੈ।
ਮੈਥਡ ਕਿਸੇ ਵੀ ਕਿਸਮ ਦੇ ਇੱਕ ਮੁੱਲ ਨੂੰ ਵਾਪਸ ਕਰ ਸਕਦਾ ਹੈ ਹਾਲਾਂਕਿ ਇੱਕ ਫੰਕਸ਼ਨ ਨੂੰ ਕਿੰਨੇ ਵੀ ਗਿਣਤੀ ਦੇ ਮੁੱਲ ਆਰਗੂਮੈਂਟਸ (arguments) ਦੇ ਤੌਰ ਤੇ ਦਿੱਤੇ ਜਾ ਸਕਦੇ ਹਨ।	ਕੰਸਟਰਕਟਰ ਕਿਸੇ ਵੀ ਕਿਮਸ ਦਾ ਕੋਈ ਮੁੱਲ ਵਾਪਸ ਨਹੀਂ ਕਰ ਸਕਦਾ। ਹਾਲਾਂਕਿ ਕਿਸੇ ਵੀ ਕਿਸਮ ਅਤੇ ਗਿਣਤੀ ਦੇ ਮੁੱਲ ਕੰਸਟਰਕਟਰ ਨੂੰ ਆਰਗੂਮੈਂਟ ਦੇ ਤੌਰ ਤੇ ਮੈਥਡ ਦੀ ਤਰ੍ਹਾਂ ਹੀ ਦਿੱਤੇ ਜਾ ਸਕਦੇ ਹਨ।
ਇੱਕ ਮੈਥਡ ਨੂੰ ਸਪੱਸ਼ਟ ਤੌਰ ਤੇ ਕਿਸੇ ਵੀ ਸਮੇਂ ਅਤੇ ਜਿੰਨੇ ਵਾਰ ਚਾਹੋ, ਕਾਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।	ਕੰਸਟਰਕਟਰਾਂ ਨੂੰ ਸਿਰਫ ਇੱਕ ਵਾਰ ਆਬਜੈਕਟ ਬਣਾਉਣ ਦੇ ਸਮੇਂ ਕਾਲ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਟੇਬਲ 7.2. ਮੈਥਡ ਅਤੇ ਕੰਸਟਰਕਟਰ ਵਿੱਚ ਅੰਤਰ

7.4.1 ਕੰਸਟਰਕਟਰਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਦੇ ਸਮੇਂ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਨਿਯਮ (Rules to be followed while defining constructors):

ਜਾਵਾ ਕਲਾਸ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਬਣਾਉਣ ਲਈ ਕੁਝ ਨਿਯਮ ਤੈਅ ਕੀਤੇ ਗਏ ਹਨ। ਅਸੀਂ ਇਹਨਾਂ ਵਿੱਚੋਂ ਕੁਝ ਦੀ ਵਿਆਖਿਆ ਇਸ ਤਰ੍ਹਾਂ ਕਰ ਸਕਦੇ ਹਾਂ:

- ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਦਾ ਉਹੀ ਨਾਮ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਜੋ ਕਿ ਕਲਾਸ ਦਾ ਨਾਮ ਹੈ, ਜਿਸ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਘੋਸ਼ਿਤ (declare) ਕੀਤਾ ਗਿਆ ਹੈ।
- ਕੰਸਟਰਕਟਰ ਦੇ ਅਕਸੈੱਸ (access) ਨੂੰ ਨਿਯੰਤਰਿਤ ਕਰਨ ਲਈ ਕੰਸਟਰਕਟਰ ਘੋਸ਼ਣਾ (declaration) ਦੌਰਾਨ ਐਕਸੈਂਸ ਮੋਡੀਫਾਇਰ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।
- ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਐਬਸਟਰੈਕਟ (abstract), ਫਾਈਨਲ (final), ਸਟੈਟਿਕ (static) ਜਾਂ ਸਿੰਕਰੋਨਾਈਜ਼ਡ (Synchronized) ਨਹੀਂ ਹੋ ਸਕਦਾ ਹੈ।
- ਕੰਸਟਰਕਟਰ ਦੀ ਕੋਈ ਵੀ ਰਿਟਰਨ (return) ਕਿਸਮ ਨਹੀਂ ਹੋਣੀ ਚਾਹੀਦੀ, ਇੱਥੋਂ ਤੱਕ ਕਿ ਵੋਆਇਡ (void) ਕਿਸਮ ਵੀ ਨਹੀਂ।

7.4.2 ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰਾਂ ਦੀਆਂ ਕਿਸਮਾਂ (Types of Constructors in Java) :

ਕੰਸਟਰਕਟਰ ਬਾਰੇ ਇਹ ਸਭ ਜਾਣਕਾਰੀਆਂ ਹਾਸਲ ਕਰਨ ਤੋਂ ਬਆਦ ਇਸ ਦੀਆਂ ਕਿਸਮਾਂ ਬਾਰੇ ਚਰਚਾ ਕਰਨ ਦਾ ਸਹੀ ਸਮਾਂ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਮੁੱਖ ਤੌਰ ਤੇ ਦੋ ਕਿਸਮਾਂ ਦੇ ਕੰਸਟਰਕਟਰ ਵਰਤੇ ਜਾਂਦੇ ਹਨ:

1. ਡਿਫਾਲਟ (Default) ਕੰਸਟਰਕਟਰ (ਨੋ-ਆਰਗੂਮੈਂਟਸ ਕੰਸਟਰਕਟਰ): ਇੱਕ ਕੰਸਟਰਕਟਰ ਜਿਸਦਾ ਕੋਈ ਪੈਰਾਮੀਟਰ ਨਹੀਂ ਹੁੰਦਾ, ਡਿਫਾਲਟ ਕੰਸਟਰਕਟਰ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਜੇਕਰ ਅਸੀਂ ਕਲਾਸ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਨਹੀਂ ਕਰਦੇ, ਤਾਂ ਕੰਪਾਈਲਰ ਕਲਾਸ ਲਈ ਇੱਕ ਡਿਫਾਲਟ ਕੰਸਟਰਕਟਰ (ਬਿਨਾਂ ਆਰਗੂਮੈਂਟਸ ਦੇ) ਆਪਣੇ ਆਪ ਬਣਾਉਂਦਾ ਹੈ ਅਤੇ ਜੇਕਰ ਅਸੀਂ ਆਰਗੂਮੈਂਟਸ ਜਾਂ ਬਿਨਾਂ ਆਰਗੂਮੈਂਟਸ ਦੇ ਕੰਸਟਰਕਟਰ ਬਣਾ ਦਿੰਦੇ ਹਾਂ ਤਾਂ ਕੰਪਾਈਲਰ ਡਿਫਾਲਟ ਕੰਸਟਰਕਟਰ ਨਹੀਂ ਬਣਾਉਂਦਾ। ਡਿਫਾਲਟ ਕੰਸਟਰਕਟਰ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਡਿਫਾਲਟ ਮੁੱਲਾਂ ਦੇ ਨਾਲ ਕਿਸੇ ਵੀ ਅਣ-ਸ਼ੁਰੂਆਤੀ ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲ (Uninitialized Instance Variable) ਨੂੰ ਇਨੀਸ਼ੀਅਲਾਈਜ਼ ਕਰਦਾ ਹੈ:

Type	Default Value
boolean	false
byte	0
short	0
int	0
long	0L
char	\u0000
float	0.0f
double	0.0d
object	Reference null

ਟੇਬਲ 7.3 : ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲ ਦੇ ਪੂਰਵ-ਨਿਰਧਾਰਿਤ ਮੁੱਲ

2. ਪੈਰਾਮੀਟਰਾਈਜ਼ਡ ਕੰਸਟਰਕਟਰ (Parameterized Constructor) : ਕੰਸਟਰਕਟਰ ਜਿਸ ਵਿੱਚ ਪੈਰਾਮੀਟਰ ਪਾਸ ਕੀਤੇ ਹੁੰਦੇ ਹਨ, ਨੂੰ ਪੈਰਾਮੀਟਰਾਈਜ਼ਡ ਕੰਸਟਰਕਟਰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਜੇਕਰ ਅਸੀਂ ਕਲਾਸ ਦੇ ਫੀਲਡ ਨੂੰ ਆਪਣੇ ਕਿਸੇ ਖਾਸ ਮੁੱਲਾਂ ਨਾਲ ਸ਼ੁਰੂ (initialize) ਕਰਨਾ ਚਾਹੁੰਦੇ ਹਾਂ ਤਾਂ ਪੈਰਾਮੀਟਰਾਈਜ਼ਡ ਕੰਸਟਰਕਟਰ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।ਉਦਾਹਰਣ ਲਈ:

ਪ੍ਰੋਗਰਾਮ 7.4 (CAProg5.java) ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਕੰਸਟਰਕਟਰ ਦੀ ਵਰਤੋਂ ਲਈ ਪ੍ਰੋਗਰਾਮ

```
class Constructors
{
int no1,no2,total;
Constructors()//default constructor
{
no1=no2=2;
}
Constructors(int inp1)//parameterized constructor
{
no1=no2=inp1;
}
void show()//method
{
```

```
                total=no1+no2;
System.out.println("Sum is: "+total);
}
} //end of class
class CAProg5
{
        public static void main(String arg[])
        {
          Constructors obj1=new Constructors();//call to default costructor
          obj1.show();
          Constructors obj2=new Constructors(5);//call to default costructor
          obj2.show();
        }
}
```

Creating object without passing any argument

Creating object passing argument(s) to Constructor

ਪ੍ਰੋਗਰਾਮ 7.4 (CAProg5.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

ਅਸੀਂ ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਨੂੰ ਵੀ ਬਤੌਰ ਆਰਗੂਮੈਂਟ ਇੱਕ ਕੰਸਟਰਕਟਰ ਨੂੰ ਪਾਸ ਕਰ ਸਕਦੇ ਹਾਂ, ਅਜਿਹੇ ਕੰਸਟਰਕਟਰ ਨੂੰ **ਕਾਪੀ ਕੰਸਟਰਕਟਰ (Copy Constructor)** ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਤਰ੍ਹਾਂ ਦੇ ਕੰਸਟਰਕਟਰ ਉਸ ਸਥਿਤੀ ਵਿੱਚ ਲਾਭਦਾਇਕ ਹੁੰਦੇ ਹਨ ਜਦੋਂ ਅਸੀਂ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਕਿਸੇ ਮੌਜੂਦਾ ਆਬਜੈਕਟ ਦੀ ਕਾਪੀ ਬਣਾਉਣਾ ਚਾਹੁੰਦੇ ਹਾਂ। ਉਦਾਹਰਣ ਲਈ:

```
Constructors(Constructor  obj)                    //Copy Constructor
{
        no1=obj.no1;
}
```

7.5 ਜਾਵਾ ਵਿੱਚ ਓਵਰਲੋਡਿੰਗ (OVERLOADING IN JAVA)

ਓਵਰਲੋਡਿੰਗ ਵੱਖ-ਵੱਖ ਮੈਥਡਜ਼ ਜਾਂ ਕੰਸਟਰਕਟਰਾਂ ਨੂੰ ਇੱਕੋ ਨਾਮ ਰੱਖਣ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦੀ ਹੈ ਪਰੰਤੂ ਹਰੇਕ ਕਿਸਮ ਵਿੱਚ ਇਨੱਪੁਟ ਪੈਰਾਮੀਟਰ ਜਾਂ ਇਨਪੁੱਟ ਪੈਰਾਮੀਟਰਜ਼ ਦੀ ਡਾਟਾ ਕਿਸਮ (data type) ਜਾਂ ਦੋਵੇਂ ਵੱਖਰੇ ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ। ਓਵਰਲੋਡਿੰਗ, ਕੰਪਾਈਲ-ਟਾਈਮ (ਜਾਂ ਸਥਿਰ-static) ਪੋਲੀਮੋਰਫਿਜ਼ਮ (polymorphism) ਨਾਲ ਸਬੰਧਤ ਹੈ। ਓਵਰਲੋਡਿੰਗ ਕਦੇ ਵੀ ਕਿਸੇ ਮੈਥਡ ਦੀ ਰਿਟਰਨ ਟਾਇਪ ਦੇ ਅਧਾਰ ਤੇ ਨਹੀਂ ਹੋ ਸਕਦੀ। ਅਸੀਂ ਓਵਰਲੋਡਿੰਗ ਨੂੰ ਹੇਠ ਲਿਖੀਆਂ

ਦੋ ਕਿਸਮਾਂ ਵਿੱਚ ਸ਼੍ਰੇਣੀਬੱਧ ਕਰ ਸਕਦੇ ਹਾਂ:

- ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading)
- ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ (Constructor Overloading)

7.5.1 ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading in Java)

ਕਈ ਬਾਰ ਸਾਨੂੰ ਆਰਗੂਮੈਂਟਸ ਦੀ ਸੰਖਿਆ ਦੇ ਅਧਾਰ ਤੇ ਇੱਕ ਕਲਾਸ ਦੇ ਇੱਕ ਆਮ ਵਿਵਹਾਰ (behaviour) ਨਾਲ ਜੁੜੇ ਕਈ ਅਲੱਗ-ਅਲੱਗ ਓਪਰੇਸ਼ਨਜ਼ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਦੀ ਲੋੜ ਹੋ ਸਕਦੀ ਹੈ।ਅਸੀਂ ਇਸ ਮੰਤਵ ਲਈ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (method overloading) ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ। ਇਸ ਸਥਿਤੀ ਵਿੱਚ, ਕਈ ਮੈਥਡਜ਼ ਦਾ ਨਾਮ ਇੱਕੋ ਹੁੰਦਾ ਹੈ ਪਰੰਤੂ ਪੈਰਾਮੀਟਰਜ਼ ਦੀ ਸੰਖਿਆ ਵਿੱਚ ਜਾਂ ਪੈਰਾਮੀਟਰਜ਼ ਦੀ ਡਾਟਾ ਕਿਸਮ ਵਿੱਚ ਅੰਤਰ ਹੁੰਦਾ ਹੈ।ਮੈਥਡ ਬਣਾਉਣ ਦੇ ਅਜਿਹੇ ਤਰੀਕੇ ਨੂੰ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (method overloading) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।ਇਹ ਪ੍ਰੋਗਰਾਮ ਦੀ ਪੜ੍ਹਨਯੋਗ ਨੂੰ ਵਧਾਉਂਦਾ ਹੈ। ਮੰਨ ਲਓ ਕਿ ਅਸੀਂ ਦਿੱਤੀਆਂ ਗਈਆਂ ਸੰਖਿਆਵਾਂ ਨੂੰ ਜੋੜਨਾ ਹੈ ਪਰੰਤੂ ਆਰਗੂਮੈਂਟ ਦੀ ਗਿਣਤੀ ਕੋਈ ਵੀ ਹੋ ਸਕਦੀ ਹੈ।। ਅਜਿਹੀ ਸਥਿਤੀ ਵਿੱਚ, ਜੇਕਰ ਅਸੀਂ ਦੋ ਪੈਰਾਮੀਟਰਜ਼ ਲਈ sumTwo(int,int) ਅਤੇ ਤਿੰਨ ਪੈਰਾਮੀਟਰਜ਼ ਲਈ sumThree(int,int,int) ਆਦਿ ਮੈਥਡ ਬਣਾਉਂਦੇ ਹਾਂ ਤਾਂ ਇਹ ਸਾਡੇ ਲਈ ਅਤੇ ਦੂਜੇ ਪ੍ਰੋਗਰਾਮਰਜ਼ ਲਈ ਸਮਝਣਾ ਮੁਸ਼ਕਿਲ ਹੋ ਸਕਦਾ ਹੈ।। ਇਹਨਾਂ ਦੇ ਨਾਮ ਵਿੱਚ ਅੰਤਰ ਹੋਣ ਕਾਰਨ ਮੈਥਡ ਦੇ ਵਿਵਹਾਰ ਨੂੰ ਕੁਸ਼ਲਤਾ ਨਾਲ ਵਰਤਣ ਦੀ ਜ਼ਰੂਰਤ ਹੋ ਸਕਦੀ ਹੈ।ਅਸੀਂ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇਸ ਸਮੱਸਿਆ ਨੂੰ ਦੂਰ ਕਰ ਸਕਦੇ ਹਾਂ।

ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਕਈ ਤਰੀਕਿਆਂ ਨਾਲ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ ਜਿਵੇਂ ਕਿ :

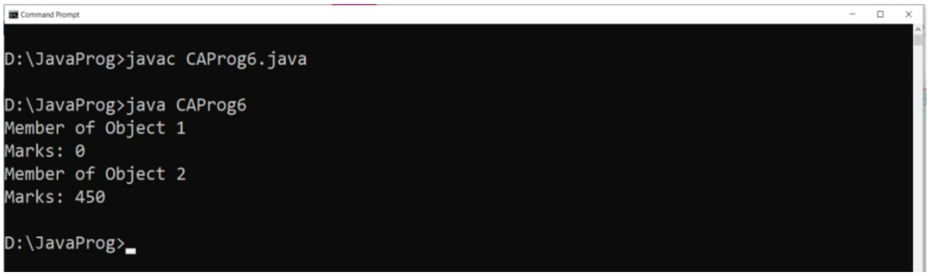
1. ਆਰਗੂਮੈਂਟਾਂ ਦੀ ਗਿਣਤੀ ਦੇ ਆਧਾਰ ਤੇ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading by changing number of arguments) : ਇਸ ਕਿਸਮ ਦੀ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਵਿੱਚ ਅਸੀਂ ਇੱਕੋ ਨਾਮ ਦੇ ਦੋ ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਮੈਥਡ ਬਣਾ ਸਕਦੇ ਹਾਂ ਪਰ ਆਰਗੂਮੈਂਟਸ ਦੀ ਗਿਣਤੀ ਵੱਖਰੀ ਹੋ ਸਕਦੀ ਹੈ। ਉਦਾਹਰਨ ਲਈ: ਅਸੀਂ ਕਲਾਸ ਦੇ ਫੀਲਡਜ਼ ਨੂੰ ਕੋਈ ਡਿਫਾਲਟ ਮੁੱਲ ਨਿਰਧਾਰਤ (assign) ਕਰਨ ਲਈ ਪਹਿਲਾ ਮੈਥਡ setValue() ਘੋਸ਼ਿਤ (declare) ਕਰ ਸਕਦੇ ਹਾਂ ਪਰ ਉਸੇ ਨਾਮ ਦੇ ਨਾਲ ਇੱਕ ਹੋਰ ਮੈਥਡ ਜਿਸ ਵਿੱਚ ਆਰਗੂਮੈਂਟਸ ਦੀ ਗਿਣਤੀ ਵੱਖਰੀ ਹੋਵੇ, ਨਾਲ ਵੀ ਕਲਾਸ ਫੀਲਡ ਨੂੰ ਦਿੱਤੇ ਜਾਣ ਵਾਲੇ ਮੁੱਲਾਂ ਨੂੰ ਨਿਰਧਾਰਤ ਕਰ ਸਕਦੇ ਹਾਂ।ਆਓ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਇਸ ਤਰ੍ਹਾਂ ਦੀ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਨੂੰ ਵੇਖੀਏ।

ਪ੍ਰੋਗਰਾਮ 7.5 (CAProg6.java) ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਲਈ ਪ੍ਰੋਗਰਾਮ

```
class CAProg6 {
    int marks;
    void setValues()        {
        marks=0;
    }
    void setValues(int inp1)     { //Method Overloading
        marks=inp1;
    }
    void show()             {
    System.out.println("Marks: "+marks);
    }
    public static void main(String arg[])        {
        CAProg6 obj1=new CAProg6();
        CAProg6 obj2=new CAProg6();
```

```
                obj1.setValues();
                obj2.setValues(450);
                System.out.println("Member of Object 1");
                obj1.show();
                System.out.println("Member of Object 2");
                obj2.show();
        }
}
```

ਪ੍ਰੋਗਰਾਮ 7.5 (CAProg6.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ



ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਆਉਟਪੁੱਟ ਵਿੱਚ ਵੇਖ ਸਕਦੇ ਹਾਂ, ਜਦੋਂ ਅਸੀਂ setValues ਮੈਥਡ ਨੂੰ ਕੋਈ ਵੀ ਆਰਗੂਮੈਂਟ ਪਾਸ ਨਹੀਂ ਕੀਤਾ ਤਾਂ ਐਗਜ਼ੀਕਿਊਸ਼ਨ (execution) ਲਈ ਬਿਨਾਂ ਆਰਗੂਮੈਂਟ ਦੇ ਮੈਥਡ ਦੀ ਚੋਣ ਹੋਈ ਅਤੇ 0 ਮੁੱਲ marks ਵੇਰੀਏਬਲ ਨੂੰ ਅਸਾਇਨ (assign) ਕੀਤਾ ਗਿਆ। ਜਦੋਂ ਅਸੀਂ setValue ਮੈਥਡ ਨੂੰ ਆਰਗੂਮੈਂਟ ਦੇ ਤੌਰ ਤੇ 450 ਮੁੱਲ ਪਾਸ ਕੀਤਾ ਤਾਂ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਲਈ ਇੱਕ ਆਰਗੂਮੈਂਟ ਵਾਲਾ ਮੈਥਡ ਚੁਣਿਆ ਗਿਆ ਅਤੇ Marks ਵੇਰੀਏਬਲ ਨੂੰ ਆਰਗੂਮੈਂਟ ਦੇ ਤੌਰ ਤੇ ਦਿੱਤਾ ਗਿਆ ਮੁੱਲ ਅਸਾਇਨ (assign) ਹੋਇਆ ਹੈ। ਇਹ ਤਰੀਕਾ ਆਰਗੂਮੈਂਟਸ ਦੀ ਸੰਖਿਆ ਦੇ ਆਧਾਰ ਤੇ ਕੀਤੀ ਜਾਣ ਵਾਲੀ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

2. **ਆਰਗੂਮੈਂਟਸ ਦੀ ਡਾਟਾ ਕਿਸਮ ਨੂੰ ਬਦਲ ਕੇ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading by changing data type of arguments) :** ਇਸ ਕਿਸਮ ਦੇ ਮੈਥਡ ਓਵਰਲੋਡ ਵਿੱਚ, ਸਾਡੇ ਕੋਲ ਇੱਕੋ ਨਾਮ ਦੇ ਨਾਲ ਕਈ ਮੈਥਡ ਹੋ ਸਕਦੇ ਹਨ ਪਰੰਤੂ ਆਰਗੂਮੈਂਟ ਦੀ ਡਾਟਾ ਕਿਸਮ ਵੱਖਰੀ ਹੁੰਦੀ ਹੈ। ਉਦਾਹਰਨ ਲਈ: ਅਸੀਂ ਕਲਾਸ ਦੇ ਵੱਖ-ਵੱਖ ਫੀਲਡਜ਼ ਨੂੰ ਵੱਖ-ਵੱਖ ਮੁੱਲ (different values) ਨਿਰਧਾਰਤ ਕਰਨ ਲਈ setValues ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ।

ਪ੍ਰੋਗਰਾਮ 7.6 (CAProg7.java) ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਲਈ ਪ੍ਰੋਗਰਾਮ

```
class CAProg7
{
int marks;
String name;
void setValues(int inp1)      {
        marks=inp1;
        name="";
}
void setValues(String inp2) { // method overloading
        marks=0;
        name=inp2;
```

```
    }
    void show()   {
    System.out.println("Marks: "+marks);
    System.out.println("Name: "+name);
    }
    public static void main(String arg[])
    {
            CAProg7 obj1=new CAProg7();
            CAProg7 obj2=new CAProg7();
            obj1.setValues("Shivpreet");
            obj2.setValues(450);
            obj1.show( );
            obj2.show( );
            }
    }
```

ਪ੍ਰੋਗਰਾਮ 7.6 (CAProg7.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg7.java

D:\JavaProg>java CAProg7
Member of Object 1
Marks: 0
Name: Shivpreet
Member of Object 2
Marks: 450
Name:

D:\JavaProg>_
```

3. **ਆਰਗੂਮੈਂਟਸ ਦੀ ਸੰਖਿਆ ਅਤੇ ਆਰਗੂਮੈਂਟਸ ਦੀ ਡਾਟਾ ਕਿਸਮ ਦੋਵਾਂ ਨੂੰ ਬਦਲ ਕੇ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading by changing both,number of arguments and data type of arguments) :** ਜਿਵੇਂ ਕਿ ਇਸਦੇ ਨਾਮ ਤੋਂ ਹੀ ਸਪਸ਼ਟ ਹੁੰਦਾ ਹੈ, ਅਸੀਂ ਇੱਕੋ ਨਾਮ ਦੇ ਪਰੰਤੂ ਆਰਗੂਮੈਂਟਜ਼ ਦੀ ਵੱਖ-ਵੱਖ ਸੰਖਿਆ ਦੇ ਨਾਲ-ਨਾਲ ਵੱਖ-ਵੱਖ ਡਾਟਾ ਕਿਸਮਾਂ ਨਾਲ ਕਈ ਮੈਥਡ ਡਿਕਲੇਅਰ ਕਰਕੇ ਵੀ ਮੈਥਡਜ਼ ਨੂੰ ਓਵਰਲੋਡ ਕਰ ਸਕਦੇ ਹਾਂ।ਅਸੀਂ ਐਪਲੀਕੇਸ਼ਨ ਦੀ ਲੋੜ ਅਨੁਸਾਰ ਇਸ ਕਿਸਮ ਦੀ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ। ਅਸੀਂ ਪਿਛਲੀਆਂ ਉਦਾਹਰਣਾਂ ਵਾਂਗ ਬਦਲਾਵ ਕਰਕੇ ਹੇਠਾਂ ਦਰਸ਼ਾਏ ਅਨੁਸਾਰ ਮੈਥਡ ਬਣਾ ਸਕਦੇ ਹਾਂ ਜਿਵੇਂ ਕਿ:

```
void setValues(String inp1)
{
……………….
}
void setValues(int inp1,String inp2)//Method Overloading
{
……………….
}
```

4. **ਪੈਰਾਮੀਟਰਾਂ ਵਿੱਚ ਡੇਟਾ ਕਿਸਮਾਂ ਦੇ ਕ੍ਰਮ ਦੇ ਅਧਾਰ ਤੇ ਮੈਥਡ ਓਬਰਲੋਡਿੰਗ (Based on the sequence of data types in parameters):** ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਦੀ ਇਸ ਕਿਸਮ ਦੇ ਅੰਦਰ ਓਵਰਲੋਡਿੰਗ ਪੈਰਾਮੀਟਰਜ਼ ਦੀਆਂ ਡੇਟਾ ਕਿਸਮਾਂ ਦੇ ਕ੍ਰਮ ਉੱਪਰ ਨਿਰਭਰ ਕਰਦੀ ਹੈ।

ਅਸੀਂ ਵੱਖ-ਵੱਖ ਆਰਗੂਮੈਂਟਸ ਨਾਲ ਪਿਛਲੀਆਂ ਉਦਾਹਰਣਾਂ ਵਿੱਚ ਅੱਗੇ ਦਰਸ਼ਾਏ ਅਨੁਸਾਰ ਮੈਥਡ ਬਣਾ ਕੇ ਇਸ ਕਿਸਮ ਦੇ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰ ਸਕਦੇ ਹਾਂ:

```
void setValues(String inp1, int inp2)
{
……………..
}
void setValues(int inp1,String inp2)//Method Overloading
{
……………..
}
```

7.4.2 ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ (Constructor Overloading in Java) :

ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਨੂੰ ਵੀ ਓਵਰਲੋਡ ਕਰ ਸਕਦੇ ਹਾਂ ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਵਿੱਚ ਪੜ੍ਹ ਚੁੱਕੇ ਹਾਂ। ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਨੂੰ ਵੱਖ-ਵੱਖ ਤਰੀਕਿਆਂ ਦੇ ਨਾਲ ਇੱਕ ਤੋਂ ਵੱਧ ਕੰਸਟਰਕਟਰ ਬਣਾਉਣ ਦੀ ਧਾਰਨਾ (concept) ਦੇ ਰੂਪ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਤਾਂ ਜੋ ਹਰ ਕੰਸਟਰਕਟਰ ਇੱਕ ਵੱਖਰਾ ਕੰਮ ਕਰ ਸਕੇ। ਕੰਸਟਰਕਟਰ ਦੀ ਵਰਤੋਂ ਫੀਲਡਜ਼ ਨੂੰ ਸ਼ੁਰੂ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਕਈ ਵਾਰ, ਸਾਨੂੰ ਆਰਗੂਮੈਂਟਸ ਦੇ ਵੱਖ-ਵੱਖ ਸੁਮੇਲ ਵਿੱਚ ਵੱਖ-ਵੱਖ ਵੇਰੀਏਬਲਾਂ ਨੂੰ ਵੱਖਰਾ ਮੁੱਲ ਦੇਣ ਦੀ ਲੋੜ ਪੈਂਦੀ ਹੈ। ਉਸ ਸਥਿਤੀ ਵਿੱਚ, ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਲੋੜੀਂਦਾ ਕੰਮ ਕਰਨ ਲਈ ਵਧੇਰੇ ਸੁਵਿਧਾਜਨਕ ਬਣ ਜਾਂਦਾ ਹੈ। ਮੈਥਡਜ਼ ਵਾਂਗ ਹੀ ਅਸੀਂ ਹੇਠ ਲਿਖੇ ਤਰੀਕਿਆਂ ਨਾਲ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਕਰ ਸਕਦੇ ਹਾਂ:

1. **ਆਰਗੂਮੈਂਟਸ ਦੀ ਗਿਣਤੀ ਨੂੰ ਬਦਲ ਕੇ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ (Constructor Overloading by changing number of arguments) :** ਇਸ ਕਿਸਮ ਦੀ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਵਿੱਚ ਅਸੀਂ ਵੱਖ-ਵੱਖ ਆਰਗੂਮੈਂਟਸ ਦੀ ਗਿਣਤੀ ਵਾਲੇ ਦੋ ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਕੰਸਟਰਕਟਰ ਬਣਾ ਸਕਦੇ ਹਾਂ। ਉਦਾਹਰਨ ਲਈ: ਅਸੀਂ ਇੱਕ ਕੰਸਟਰਕਟਰ ਵਿੱਚ ਕਲਾਸ ਦੇ ਸਾਰੇ ਫੀਲਡਜ਼ ਨੂੰ ਇਕਹਰਾ ਮੁੱਲ (Same value) ਅਤੇ ਦੂਜੇ ਕੰਸਟਰਕਟਰ ਵਿੱਚ ਵੱਖ-ਵੱਖ ਫੀਲਡਜ਼ ਲਈ ਵੱਖ-ਵੱਖ ਮੁੱਲ (different values) ਨਿਰਧਾਰਤ ਕਰ ਸਕਦੇ ਹਾਂ।

ਪ੍ਰੋਗਰਾਮ 7.7 (CAProg8.java) ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਲਈ ਪ੍ਰੋਗਰਾਮ

```
class CAProg8
{
        int studentId;
        CAProg8()    { //Default Constructor
        studentId=101;
        }
        CAProg8(int sid)     { //Overloading Constructor
        studentId=sid;
        }
void show() //Method
{
System.out.println("Your student ID is: "+studentId);
}
public static void main(String arg[])
{
        CAProg8 obj1=new CAProg8();
        CAProg8 obj2=new CAProg8(2462);
```

```
        System.out.println("Member of Object 1");
        obj1.show();
        System.out.println("------------\n Member of Object 2");
        obj2.show();
    }
}
```

ਪ੍ਰੋਗਰਾਮ 7.7 (CAProg8.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਊਟਪੁੱਟ

```
D:\JavaProg>javac CAProg8.java

D:\JavaProg>java CAProg8
Member of Object 1
Your student ID is: 101
---------------------
 Member of Object 2
Your student ID is: 2462

D:\JavaProg>
```

2. **ਆਰਗੂਮੈਂਟਸ ਦੀ ਡੇਟਾ ਕਿਸਮ ਨੂੰ ਬਦਲ ਕੇ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ (Constructor Overloading by changing data type of arguments) :** ਇਸ ਕਿਸਮ ਦੇ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਵਿੱਚ ਅਸੀਂ ਵੱਖ-ਵੱਖ ਡੇਟਾ ਕਿਸਮਾਂ ਦੇ ਆਰਗੂਮੈਂਟ ਦੇ ਕੇ ਮਲਟੀਪਲ ਕੰਸਟਰਕਟਰ ਬਣਾ ਸਕਦੇ ਹਾਂ। ਉਦਾਹਰਨ ਲਈ: ਅਸੀਂ ਇੱਕ ਕੰਸਟਰਕਟਰ ਵਿੱਚ ਗਣਿਤਕ ਅੰਕ ਅਤੇ ਦੂਜੇ ਵਿੱਚ ਸਟ੍ਰਿੰਗ ਮੁੱਲ ਅਸਾਇਨ ਕਰਨ ਲਈ ਦੋ ਕੰਸਟਰਕਟਰ ਘੋਸ਼ਿਤ ਕਰ ਸਕਦੇ ਹਾਂ। ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਵਿੱਚ ਪਹਿਲਾਂ ਹੀ ਪੜ੍ਹ ਚੁੱਕੇ ਹਾਂ। ਅਸੀਂ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਕੰਸਟਰਕਟਰਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇਸ ਕਿਸਮ ਦੇ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰ ਸਕਦੇ ਹਾਂ:

```
CAProg8(int inp1)
{
……………..
}
CAProg8 (String inp2) //constructor overloading with different
types of arguments
{
……………..
}
```

3. **ਆਰਗੂਮੈਂਟਸ ਦੀ ਗਿਣਤੀ ਅਤੇ ਆਰਗੂਮੈਂਟਾਂ ਦੀ ਡਾਟਾ ਕਿਸਮ ਦੋਵਾਂ ਨੂੰ ਬਦਲ ਕੇ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ (Constructor Overloading by changing both, number of arguments and data type of arguments) :** ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਇਸਦੇ ਨਾਮ ਤੋਂ ਹੀ ਸਮਝ ਸਕਦੇ ਹਾਂ, ਕਿਸੇ ਵੀ ਕੰਸਟਰਕਟਰ ਦੇ ਵੱਖ-ਵੱਖ ਗਿਣਤੀ ਵਿੱਚ ਆਰਗੂਮੈਂਟਸ ਦੇ ਨਾਲ-ਨਾਲ ਵੱਖ-ਵੱਖ ਡਾਟਾ ਕਿਸਮ ਦੇ ਆਰਗੂਮੈਂਟਸ ਦੇ ਕੇ ਅਸੀਂ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਕਰ ਸਕਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ: ਅਸੀਂ ਅਜਿਹੇ ਮਲਟੀਪਲ ਕੰਸਟਰਕਟਰ ਘੋਸ਼ਿਤ ਕਰ ਸਕਦੇ ਹਾਂ ਜਿੰਨਾਂ ਵਿੱਚੋਂ ਕਿਸੇ ਇਕ ਵਿੱਚ ਕੇਵਲ ਇੱਕ ਗਣਿਤਕ ਅੰਕ ਦਾ ਆਰਗੂਮੈਂਟ ਅਤੇ ਦੂਸਰੇ ਵਿੱਚ ਇੱਕ ਗਣਿਤਕ ਅੰਕ ਅਤੇ ਇੱਕ ਸਟ੍ਰਿੰਗ ਆਰਗੂਮੈਂਟ ਅਤੇ/ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਸਟ੍ਰਿੰਗ ਆਰਗੂਮੈਂਟ ਆਦਿ। ਜਿਵੇਂ ਅਸੀਂ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਵਿੱਚ ਪਹਿਲਾਂ ਹੀ ਪੜ੍ਹ ਚੁੱਕੇ ਹਾਂ, ਅਸੀਂ ਅੱਗੇ ਦਿੱਤੇ ਅਨੁਸਾਰ ਕੰਸਟਰਕਟਰਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਕੇ ਇਸ ਕਿਸਮ ਦੀ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਕਰ ਸਕਦੇ ਹਾਂ:

```
CAProg8(int inp1)
{
……………..
}
CAProg8 (int inp1,String inp2) //Constructor Overloadig
{
……………..
}
```

4. **ਪੈਰਾਮੀਟਰਸ ਵਿੱਚ ਡਾਟਾ ਕਿਸਮਾਂ ਦੇ ਕ੍ਰਮ ਦੇ ਆਧਾਰ ਤੇ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ (Based on the sequence of data types in parameters) :** ਇਸ ਕਿਸਮ ਦੀ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਪੈਰਾਮੀਟਰਜ਼ ਦੀਆਂ ਡਾਟਾ ਕਿਸਮਾਂ ਦੇ ਕ੍ਰਮ ਦੇ ਆਧਾਰ ਤੇ ਓਵਰਲੋਡਿੰਗ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਵਿੱਚ ਅਧਿਐਨ ਕੀਤਾ ਹੈ, ਅਸੀਂ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਕੰਸਟਰਕਟਰਜ਼ ਪਰਿਭਾਸ਼ਿਤ ਕਰਕੇ ਇਸ ਕਿਸਮ ਦੀ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ:

```
CAProg8(String inp1, int inp2)
{
……………..
}
CAProg8 (int inp1,String inp2) //construction overloading
{
……………..
}
```

7.5.3 ਜਾਵਾ ਵਿੱਚ ਓਵਰਲੋਡਿੰਗ ਦੇ ਫਾਇਦੇ (Advantages of Overloading in Java):

ਜਾਵਾ ਵਿੱਚ ਓਵਰਲੋਡਿੰਗ ਦੇ ਕਈ ਲਾਭ ਹੁੰਦੇ ਹਨ:

1. ਜਾਵਾ ਵਿੱਚ ਓਵਰਲੋਡਿੰਗ, ਕੋਡ ਦੀ ਮੁੜ ਵਰਤੋਂ ਯੋਗਤਾ (re-usability) ਅਤੇ ਪੜ੍ਹਨਯੋਗਤਾ (readability) ਵਿੱਚ ਸੁਧਾਰ ਕਰਦੀ ਹੈ।
2. ਜਾਵਾ ਵਿੱਚ ਓਵਰਲੋਡਿੰਗ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਡੇਟਾ ਦੇ ਨਾਲ ਇੱਕ ਸਮਾਨ ਕਿਸਮ ਦੇ ਮੈਥਡਜ਼ ਨੂੰ ਕਾਲ ਕਰਨ ਲਈ ਲਚਕਤਾ (flexibility) ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ।
3. ਓਵਰਲੋਡਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਕਈ ਨਾਵਾਂ ਦੀ ਬਜਾਏ ਇੱਕ ਨਾਮ ਦੇ ਮੈਥਡ ਨੂੰ ਯਾਦ ਰੱਖਣਾ ਆਸਾਨ ਹੁੰਦਾ ਹੈ।ਅਸੀਂ ਵੱਖ-ਵੱਖ ਮਾਮਲਿਆਂ ਵਿੱਚ ਫੀਲਡਜ਼ ਦੇ ਡਿਫਾਲਟ ਮੁੱਲਾਂ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨ ਲਈ ਵੱਖ-ਵੱਖ ਤਰੀਕਿਆਂ ਨਾਲ ਆਬਜੈਕਟ ਵੀ ਬਣਾ ਸਕਦੇ ਹਾਂ।
4. ਜਾਵਾ ਮੈਥਡਜ਼ ਦੇ ਨਾਮਕਰਨ ਵਿੱਚ ਇਕਸਾਰਤਾ ਲਿਆਉਣ ਲਈ ਓਵਰਲੋਡਿੰਗ ਦੁਆਰਾ ਅਹਿਮ ਭੂਮਿਕਾ ਨਿਭਾਈ ਜਾ ਸਕਦੀ ਹੈ।
5. ਅਸੀਂ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਇੰਸਟੈਂਸ (instance) ਬਣਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਦੌਰਾਨ ਇੱਕ ਆਬਜੈਕਟ ਨੂੰ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦਾ ਡੇਟਾ ਪਾਸ ਕਰ ਸਕਦੇ ਹਾਂ।

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਕਲਾਸ ਨੂੰ ਇੱਕ ਅਜਿਹੇ ਟੈਂਪਲੇਟ (template) ਬਲੂਪ੍ਰਿੰਟ (blueprint) ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ ਜੋ ਉਸ ਕਲਾਸ ਦੇ ਆਬਜੈਕਟਸ ਦੁਆਰਾ ਮੁਹੱਈਆ ਕਰਵਾਏ ਜਾਂਦੇ ਵਿਵਹਾਰ (behaviour) ਅਤੇ/ਜਾਂ ਸਥੀਤੀ (state) ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ।
2. ਮੋਡੀਫਾਇਰ ਉਹ ਕੀਅ-ਵਰਡ ਹੁੰਦੇ ਹਨ ਜੋ ਕਲਾਸ ਦੇ ਮੈਂਬਰਾਂ ਤੱਕ ਪਹੁੰਚ ਕਰਨ ਦੇ ਵੱਖ-ਵੱਖ ਅਧਿਕਾਰਾਂ ਦਾ ਵਰਣਨ ਕਰਦੇ ਹਨ।
3. ਕਲਾਸ ਦੇ ਅੰਦਰ ਪ੍ਰਭਾਸ਼ਿਤ ਵੇਰੀਏਬਲਸ ਨੂੰ ਫੀਲਡਜ਼ (field) ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
4. ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲ ਉਹ ਵੇਰੀਏਬਲ ਹੁੰਦੇ ਹਨ ਜੋ ਕਿਸੇ ਆਬਜੈਕਟ ਦੇ ਅੰਦਰੂਨੀ ਵੇਰੀਏਬਲਜ਼ ਹੁੰਦੇ ਹਨ ਅਤੇ ਜਿਨ੍ਹਾਂ ਨੂੰ ਕਿਸੇ ਵੀ ਮੈਥਡ, ਕੰਸਟਰਕਟਰ ਜਾਂ ਬਲਾਕ ਦੇ ਅੰਦਰੋਂ ਐਕਸੈੱਸ (ਪਹੁੰਚ ਕਰਨਾ) ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।
5. ਕਲਾਸ ਵੇਰੀਏਬਲ ਜਾਂ ਸਟੈਟਿਕ ਵੇਰੀਏਬਲ, ਇੱਕ ਕਲਾਸ ਵਿੱਚ ਸਟੈਟਿਕ (static) ਕੀਅ-ਵਰਡ ਨਾਲ ਪ੍ਰਭਾਸ਼ਿਤ ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਇਹ ਉਂਝ ਤਾਂ ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲਾਂ ਦੇ ਸਮਾਨ ਹੀ ਹੁੰਦੇ ਹਨ ਪਰੰਤੂ ਇਹ ਉਸ ਸਮੇਂ ਹੋਂਦ ਵਿੱਚ ਆਉਂਦੇ ਹਨ ਜਦੋਂ ਪ੍ਰੋਗਰਾਮ ਸ਼ੁਰੂ ਹੁੰਦਾ ਹੈ ਅਤੇ ਪ੍ਰੋਗਰਾਮ ਖਤਮ ਹੋਣ ਤੇ ਇਹ ਨਸ਼ਟ ਹੋ ਜਾਂਦੇ ਹਨ।
6. ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਹਦਾਇਤਾਂ ਦਾ ਇੱਕ ਸਮੂਹ ਹੁੰਦਾ ਹੈ ਜੋ ਇੱਕ ਖਾਸ ਕੰਮ ਕਰਨ ਯੋਗ ਹੁੰਦਾ ਹੈ। ਇਹ ਪ੍ਰੋਗਰਾਮ ਕੋਡ ਦੀ ਮੁੜ ਵਰਤੋਂ ਯੋਗਤਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।
7. ਜਾਵਾ ਆਬਜੈਕਟ, ਜਾਵਾ ਕਲਾਸ ਦਾ ਇੱਕ ਮੈਂਬਰ ਹੁੰਦਾ ਹੈ (ਜਿਸਨੂੰ ਇੰਸਟੈਂਸ (instance) ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ)। ਹਰ ਆਬਜੈਕਟ ਦੀ ਇੱਕ ਪਛਾਣ (identity), ਵਿਵਹਾਰ (behaviour) ਅਤੇ ਸਟੇਟ (state) ਹੁੰਦੀ ਹੈ।
8. ਜਾਵਾ ਵਿੱਚ new ਕੀਅ-ਵਰਡ ਕਲਾਸ ਦਾ ਇੰਸਟੈਂਸ (instance) ਜਿਸ ਨੂੰ ਆਬਜੈਕਟ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ, ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।
9. ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਇੱਕ ਵਿਸ਼ੇਸ਼ ਮੈਥਡ ਹੁੰਦਾ ਹੈ ਜੋ ਆਬਜੈਕਟ ਦੇ ਮੈਂਬਰਾਂ ਨੂੰ ਸ਼ੁਰੂਆਤੀ ਮੁੱਲ ਪ੍ਰਦਾਨ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਕੰਸਟਰਕਟਰ ਉਸ ਸਮੇਂ ਆਪਣੇ ਆਪ ਕਾਲ (call) ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਕਿਸੇ ਸੰਬੰਧਤ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ।
10. ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਦਾ ਉਹੀ ਨਾਮ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਜੋ ਕਿ ਕਲਾਸ ਦਾ ਨਾਮ ਹੈ, ਜਿਸ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਘੋਸ਼ਿਤ (declare) ਕੀਤਾ ਗਿਆ ਹੈ।
11. ਇੱਕ ਕੰਸਟਰਕਟਰ ਜਿਸਦਾ ਕੋਈ ਪੈਰਾਮੀਟਰ ਨਹੀਂ ਹੁੰਦਾ, ਡਿਫਾਲਟ ਕੰਸਟਰਕਟਰ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
12. ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (method overloading) ਵਿੱਚ, ਕਈ ਮੈਥਡਜ਼ ਦਾ ਇੱਕੋ ਨਾਮ ਹੁੰਦਾ ਹੈ ਪਰੰਤੂ ਪੈਰਾਮੀਟਰਜ਼ ਦੀ ਸੰਖਿਆ ਵਿੱਚ ਜਾਂ ਪੈਰਾਮੀਟਰਜ਼ ਦੀ ਡਾਟਾ ਕਿਸਮ ਵਿੱਚ ਅੰਤਰ ਹੁੰਦਾ ਹੈ।
13. ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਨੂੰ ਵੀ ਓਵਰਲੋਡ ਕਰ ਸਕਦੇ ਹਾਂ, ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਮੈਥਡ ਓਵਰਲਿੰਗ ਵਿੱਚ ਕਰਦੇ ਹਾਂ।

ਅਭਿਆਸ

ਪ੍ਰਸ਼ਨ: 1 ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ

i. ਜਾਵਾ ਕਲਾਸ ਦੇ ਇੰਸਟੈਂਸ ਨੂੰ ਕੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ?

ੳ. ਮੈਥਡ (Method)   ਅ. ਫੀਲਡ (Field)

ੲ. ਆਬਜੈਕਟ (Object)   ਸ. ਕੰਸਟਰਕਟਰ (Constructor)

ii. ਇੱਕੋ ਨਾਮ ਪਰੰਤੂ ਵੱਖ-ਵੱਖ ਆਰਗੂਮੈਂਟਸ ਦੀ ਗਿਣਤੀ ਨਾਲ ਕਈ ਮੈਥਡਜ਼ ਡਿਕਲੇਅਰ (Declare) ਕਰਨ ਨੂੰ ਕੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ?

ੳ. ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading) ਅ. ਡੈਕਲੇਰੇਸ਼ਨ (Declaration)
ੲ. ਇਨਹੈਰੀਟੈਂਸ (Inheritance) ਸ. ਇਹਨਾਂ ਵਿੱਚੋਂ ਕੋਈ ਨਹੀਂ

iii. .... ਆਟੋਮੈਟਿਕ ਹੀ ਕਾਲ (invoked) ਹੋ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਅਸੀਂ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਉਂਦੇ ਹਾਂ।
ੳ. ਫੀਲਡ (Field) ਅ. ਮੈਥਡ (Method)
ੲ. ਸਟੈਟਿਕ ਮੈਂਬਰ (Static Member) ਸ. ਕੰਸਟਰਕਟਰ (Constructor)

iv. ਇੱਕ ਕੰਸਟਰਕਟਰ ਜਿਸਦਾ ਕੋਈ ਪੈਰਾਮੀਟਰ ਨਹੀਂ ਹੁੰਦਾ, ਨੂੰ ਕੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ ?
ੳ. ਡਿਫਾਲਟ ਕੰਸਟਰਕਟਰ (Default Constructor)
ਅ. ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ (Constructor Overloading)
ੲ. ਪੈਰਾਮੀਟਰਾਈਜ਼ਡ ਕੰਸਟਰਕਟਰ (Parameterized Constructor)
ਸ. ਕਮਜ਼ੋਰ ਕੰਸਟਰਕਟਰ (Weak Constructor)

v. ਕੰਸਟਰਕਟਰ.............. ਨਹੀਂ ਹੋ ਸਕਦਾ।
ੳ. ਐਬਸਟਰੈਕਟ (Abstract) ਅ. ਫਾਈਨਲ (Final)
ੲ. ਸਥਿਰ (Static) ਸ. ਉਪਰੋਕਤ ਸਾਰੇ

ਪ੍ਰਸ਼ਨ: 2 ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ:

i. ਇੱਕ ਕਲਾਸ ਨੂੰ ਇਸਦੇ ਸਾਰੇ ਆਬਜੈਕਟਸ ਦੇ .... ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।
ii. .... ਕਲਾਸ ਦੇ ਅੰਦਰ ਘੋਸ਼ਿਤ (decalred) ਵੇਰੀਏਬਲ ਹੁੰਦੇ ਹਨ।
iii. .... ਕੀਅ-ਵਰਡ ਜਾਵਾ ਵਿੱਚ ਆਬਜੈਕਟ ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।
iv. ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ .... ਜਾਂ .... ਦੇ ਅਧਾਰ ਤੇ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।
v. ਇੱਕ ਕਲਾਸ ਵਿੱਚ ਮਲਟੀਪਲ ਕੰਸਟਰਕਟਰ ਘੋਸ਼ਿਤ (declare) ਕਰਨ ਨੂੰ .... ਕਿਹਾ ਜਾਂਦਾ ਹੈ।

ਪ੍ਰਸ਼ਨ :3 ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ:

i. ਜਾਵਾ ਵਿੱਚ ਕਲਾਸ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।
ii. ਜਾਵਾ ਕਲਾਸ ਦੇ ਮੁੱਖ ਭਾਗ ਕਿਹੜੇ ਹਨ ?
iii. ਕਲਾਸ ਵਿੱਚ ਫੀਲਡ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
iv. ਆਬਜੈਕਟ ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ?
v. ਇੰਸਟੈਂਸ ਵੇਰੀਏਬਲ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ ?
vi. ਕਲਾਸ ਵੇਰੀਏਬਲ ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ?
vii. ਜਾਵਾ ਵਿੱਚ ਮੈਥਡਜ਼ ਕੀ ਹੁੰਦੇ ਹਨ ?
viii. ਪ੍ਰਾਈਵੇਟ ਅਤੇ ਪ੍ਰੋਟੈਕਟੀਡ ਮੋਡੀਫਾਈਰ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
ix. ਇੱਕ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਕਿਵੇਂ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ?
x. ਕੰਸਟਰਕਟਰ ਕੀ ਹੁੰਦਾ ਹੈ ?
xi. ਮੈਥਡਜ਼ ਅਤੇ ਕੰਸਟਰਕਟਰਜ਼ ਵਿੱਚ ਅੰਤਰ ਦੱਸੋ।

ਪ੍ਰਸ਼ਨ:4 ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ:

i. ਕਲਾਸ ਕੀ ਹੈ ? ਕਲਾਸ ਵਿੱਚ ਫੀਲਡਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
ii. ਕੰਸਟਰਕਟਰ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।। ਢੁਕਵੀਂ ਉਦਾਹਰਣ ਦੇ ਨਾਲ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਕੰਸਟਰਕਟਰ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
iii. ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ? ਇਸ ਦੇ ਕੋਈ ਦੋ ਤਰੀਕਿਆਂ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।
iv. ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਕਿਵੇਂ ਹੁੰਦੀ ਹੈ। ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਕੰਸਟਰਕਟਰ ਓਵਰਲੋਡਿੰਗ ਦੀ ਵਿਆਖਿਆ ਕਰਨ ਲਈ ਪ੍ਰੋਗਰਾਮ ਲਿਖੋ।

ਸਟ੍ਰਿੰਗਸ ਅਤੇ ਰੈਪਰ ਕਲਾਸਾਂ
(Strings & Wrapper Classes)

8

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

8.1 ਸਟ੍ਰਿੰਗਸ (Strings)
8.2 ਸਟ੍ਰਿੰਗਸ ਉੱਪਰ ਕੰਮ ਕਰਨਾ (Performing operations on Strings)
8.3 ਰੈਪਰ ਕਲਾਸਿਜ਼ (Wrapper Classes)

ਜਾਣ ਪਛਾਣ (INTRODUCTION)

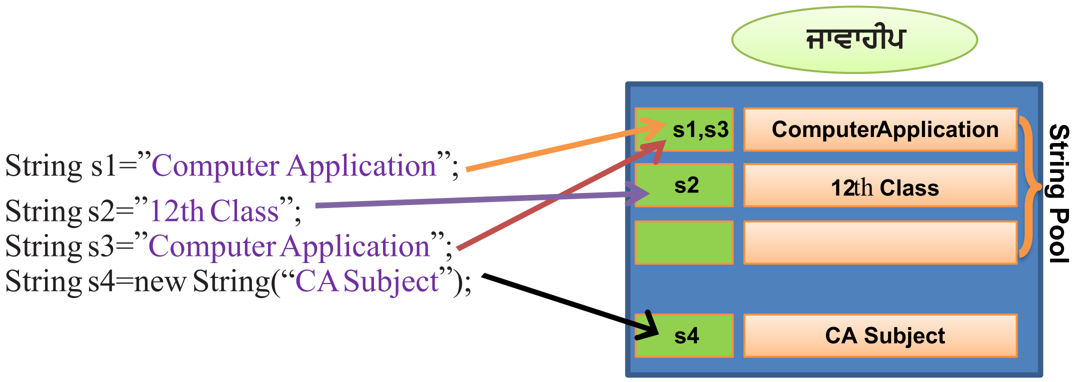
ਸਟ੍ਰਿੰਗਸ ਹਰ ਖੇਤਰ ਦਾ ਇੱਕ ਜ਼ਰੂਰੀ ਹਿੱਸਾ ਹੁੰਦੇ ਹਨ ਕਿਉਂਕਿ ਅਸੀਂ ਆਪਣੇ ਡੇਟਾ ਨੂੰ ਡਿਜ਼ੀਟਲ ਤੌਰ ਤੇ ਪ੍ਰੋਸੈੱਸ ਕਰਨ ਲਈ ਉਸਨੂੰ ਸਟ੍ਰਿੰਗ ਦੇ ਰੂਪ ਵਿੱਚ ਸਟੋਰ ਕਰ ਸਕਦੇ ਹਾਂ। ਇਸ ਲਈ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ, ਸਟ੍ਰਿੰਗ ਇੱਕ ਆਮ ਤੌਰ ਤੇ ਵਰਤੀ ਜਾਣ ਵਾਲੀ ਡਾਟਾ ਕਿਸਮ ਹੈ ਜੋ ਸਾਨੂੰ ਕਿਸੇ ਵੀ ਵਸਤੂ ਨਾਲ ਸੰਬੰਧਤ ਕੋਈ ਡਾਟਾ ਜਾਂ ਹਰ ਉਸ ਮੁੱਲ ਨੂੰ ਸਟੋਰ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦੀ ਹੈ ਜੋ ਸ਼ਬਦਾਂ (words), ਵਾਕਾਂ (sentences), ਫ਼ੋਨ ਨੰਬਰਾਂ (phone numbers) ਜਾਂ ਇੱਥੋਂ ਤੱਕ ਕਿ ਸਿਰਫ਼ ਨਿਯਮਤ ਨੰਬਰਾਂ ਨਾਲ ਸਬੰਧਤ ਹੋਵੇ। ਇਸ ਅਧਿਆਏ ਵਿੱਚ ਅਸੀਂ ਸਟ੍ਰਿੰਗਸ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦੀਆਂ ਕਈ ਧਾਰਨਾਵਾਂ (concepts) ਅਤੇ ਸਟਰਿੰਗਸ ਉੱਪਰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕੰਮ ਕਰਨ ਦੇ ਕਈ ਤਰੀਕੇ ਵੀ ਸਿੱਖਾਂਗੇ। ਆਉ ਸਟ੍ਰਿੰਗਸ ਦੀ ਵਰਤੋਂ ਨੂੰ ਵਿਸਥਾਰ ਵਿੱਚ ਸਮਝੀਏ।

8.1 ਸਟ੍ਰਿੰਗਸ (STRINGS)

ਅੱਖਰਾਂ (Character) ਦੇ ਕ੍ਰਮ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਸਟ੍ਰਿੰਗਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਜਾਵਾ ਪ੍ਰੋਗ੍ਰਾਮਿੰਗ ਭਾਸ਼ਾ ਵਿੱਚ ਇੱਕ ਸਟ੍ਰਿੰਗ ਅਸਲ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਹੁੰਦਾ ਹੈ, ਜਿਸ ਵਿੱਚ ਉਹ ਸਾਰੇ ਮੈਥਡਜ਼ ਸ਼ਾਮਿਲ ਹੁੰਦੇ ਹਨ ਜੋ ਸਟ੍ਰਿੰਗਸ ਉੱਪਰ ਕੁਝ ਖਾਸ ਕੰਮ ਜਾਂ ਆਪ੍ਰੇਸ਼ਨ ਕਰਵਾਉਣ ਲਈ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ। ਉਦਾਹਰਨ ਲਈ, ਇੱਕ ਸਟ੍ਰਿੰਗ ਦੀ ਲੰਬਾਈ ਪਤਾ ਕਰਨ ਲਈ length() ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ ਅਸੀਂ char ਡੇਟਾ ਕਿਸਮ ਦੀ ਐਰੇ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਵੀ ਸਟ੍ਰਿੰਗਸ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ।ਕਿਉਂਕਿ ਐਰੇ ਬਦਲਨਯੋਗ ਹੁੰਦੇ ਹਨ, ਜਿਸ ਤੋਂ ਭਾਵ ਹੈ ਕਿ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੌਰਾਨ ਐਰੇ ਦਾ ਮੁੱਲ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਦੂਜੇ ਪਾਸੇ, ਸਟ੍ਰਿੰਗਸ ਨਾ-ਬਦਲਨ ਯੋਗ ਹੁੰਦੇ ਹਨ, ਜਿਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਉਹਨਾਂ ਦੇ ਮੁੱਲ ਜਾਵਾ ਪ੍ਰੋਗ੍ਰਾਮਿੰਗ ਵਿੱਚ ਕਦੇ ਵੀ ਬਦਲੇ ਨਹੀਂ ਜਾ ਸਕਦੇ। ਜਦੋਂ ਅਸੀਂ ਇੱਕ ਸਟ੍ਰਿੰਗ ਦਾ ਮੁੱਲ ਬਦਲਦੇ ਹਾਂ (ਉਦਾਹਰਨ ਲਈ ਇੱਕ ਹੋਰ ਅੱਖਰ ਨੂੰ ਸਟ੍ਰਿੰਗ ਵਿੱਚ ਜੋੜਨਾ, ਇਸਨੂੰ ਛੋਟੇ ਭਾਗਾਂ ਵਿੱਚ ਵੰਡਨਾ, ਦੋ ਅੱਖਰਾਂ ਦੀ ਆਪਸੀ ਬਦਲੀ ਕਰਨਾ, ਆਦਿ) ਤਾਂ ਅਸੀਂ ਅਸਲ ਵਿੱਚ ਸਟ੍ਰਿੰਗ ਦੀ ਇੱਕ ਨਵੀਂ ਅਤੇ ਵੱਖਰੀ ਕਾਪੀ ਬਣਾ ਰਹੇ ਹੁੰਦੇ ਹਾਂ:

ਸਟ੍ਰਿੰਗ ਆਬਜੈਕਟ ਬਣਾਉਣ ਦੇ ਦੋ ਤਰੀਕੇ ਹੁੰਦੇ ਹਨ:

- **ਸਟ੍ਰਿੰਗਸ ਲਿਟਰਲ ਦੁਆਰਾ (By string literal) :** ਜਾਵਾ ਸਟ੍ਰਿੰਗ ਲਿਟਰਲ ਨੂੰ ਡਬਲ ਕੋਟਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇਸ ਤਰ੍ਹਾਂ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ॥
  For Example : String s1= “Computer Application”;
- **new ਕੀਅ ਵਰਡ ਦੀ ਵਰਤੋਂ ਦੁਆਰਾ (By new keyboard) :** ਜਾਵਾ ਸਟ੍ਰਿੰਗ “new” ਕੀਅ ਵਰਡ ਵਰਤ ਕੇ ਇਸ ਤਰ੍ਹਾਂ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।
  For example : String s2=new String (“CA Subject”);
  ਆਉ ਇੱਕ ਚਿੱਤਰ ਦੇ ਰੂਪ ਵਿੱਚ ਜਾਵਾ ਵਿੱਚ ਇਹਨਾਂ ਦੋਵੇਂ ਕਿਸਮਾਂ ਦੇ ਸਟ੍ਰਿੰਗਜ਼ ਵਿੱਚ ਅੰਤਰ ਨੂੰ ਸਮਝੀਏ:



8. 1.1 ਸਟ੍ਰਿੰਗਸ ਦੇ ਲਾਭ (Advantages of String) :

- String ਕਲਾਸ ਸਾਨੂੰ ਸਟ੍ਰਿੰਗ ਆਬਜੈਕਟ ਬਣਾਉਣ ਲਈ ਇੱਕ ਸਟ੍ਰਿੰਗ ਲਾਇਬ੍ਰੇਰੀ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ ਜੋ ਸਟ੍ਰਿੰਗਸ ਨੂੰ ਘੋਸ਼ਿਤ (declare) ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦੀ ਹੈ। ਸਟ੍ਰਿੰਗ ਦੀ ਸੀਮਾ (boundary) ਦਾ ਪ੍ਰਬੰਧਨ ਵੀ ਇਸੇ ਕਲਾਸ ਲਾਇਬ੍ਰੇਰੀ ਦੇ ਅੰਦਰ ਆਪਣੇ ਆਪ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
- String ਕਲਾਸ ਸਾਨੂੰ ਇਨ-ਬਿਲਟ ਸਟ੍ਰਿੰਗ ਮੈਥਡਜ਼ ਦੀ ਵੱਡੀ ਲਾਇਬ੍ਰੇਰੀ ਪ੍ਰਦਾਨ ਕਰਦੀ ਹੈ ਜੋ ਸਟ੍ਰਿੰਗ ਉੱਪਰ ਕੰਮ ਕਰਨਾ ਬਹੁਤ ਆਸਾਨ ਬਣਾਉਂਦੀ ਹੈ ।
- String ਕਲਾਸ ਬਹੁਤ ਸਾਰੇ ਡੇਟਾ ਸਟਰਕਚਰ (data structure) ਜਿਵੇਂ ਕਿ (tree) ਅਤੇ ਐਰੇਜ਼ (arrays) ਲਈ ਅਧਾਰ ਵਜੋਂ ਕੰਮ ਕਰਦੀ ਹੈ।
- ਸਟ੍ਰਿੰਗਸ ਸਾਨੂੰ ਘੱਟ ਸਮੇਂ (time) ਅਤੇ ਗੁੰਝਲਤਾ (complexity) ਨਾਲ ਕੋਈ ਵੀ ਵੱਡੀ ਸਮੱਸਿਆਵਾਂ ਨੂੰ ਹੱਲ ਕਰਨ ਲਈ ਬਹੁਤ ਮਦਦਗਾਰ ਸਟ੍ਰਿੰਗ ਐਲਗੋਰਿਥਮ (string algorithm) ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਨ।
- String ਕਲਾਸ ਦਾ StringBuffer ਕਲਾਸ ਨਾਲ ਨਜ਼ਦੀਕੀ ਰਿਸ਼ਤਾ ਹੁੰਦਾ ਹੈ ਜੋ ਕਿ ਵਧਣਯੋਗ (growing), ਜੋੜਨਯੋਗ (adding), ਬਦਲਣਯੋਗ (changable) ਅਤੇ ਹੋਰ ਕਿਸਮਾਂ ਦੇ ਸਟ੍ਰਿੰਗਸ ਤੇ ਕੰਮ ਕਰਨ ਲਈ ਇੱਕ ਕੁਸ਼ਲ (efficient) ਵਿਧੀ (mechanism) ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ।

8.1.2 ਸਟ੍ਰਿੰਗਸ ਦੀਆਂ ਸੀਮਾਵਾਂ (Limitations of Strings) :

- ਜਾਵਾ ਵਿੱਚ ਸਟ੍ਰਿੰਗ ਨਾ-ਬਦਲਣਯੋਗ ਹੁੰਦੇ ਹਨ। ਭਾਵ ਉਹਨਾਂ ਨੂੰ ਸੋਧਿਆ ਜਾਂ ਬਦਲਿਆ ਨਹੀਂ ਜਾ ਸਕਦਾ ਹੈ ॥
- ਸਟ੍ਰਿੰਗ ਆਮ ਤੌਰ ਤੇ ਸਟ੍ਰਿੰਗ ਮੁੱਲ ਦੇ ਇਨਪੁਟ ਅਤੇ ਆਉਟਪੁੱਟ ਵਰਗੇ ਆਪ੍ਰੇਸ਼ਨਜ਼ ਕਰਨ ਵਿੱਚ ਹੌਲੀ ਹੁੰਦੇ ਹਨ ॥
- ਅਸੀਂ String ਕਲਾਸ ਨੂੰ ਇਨਹੈਰੀਟ (inherit) ਨਹੀਂ ਕਰ ਸਕਦੇ ਜਿਸਦਾ ਮਤਲਬ ਹੈ ਕਿ ਸਟ੍ਰਿੰਗ ਕਲਾਸ ਵਿੱਚ ਬਣਾਏ ਗਏ ਮੈਥਡਜ਼ ਦੀ ਓਵਰਰਾਈਡਿੰਗ ਸੰਭਵ ਨਹੀਂ ਹੈ ॥

8.2 ਸਟ੍ਰਿੰਗਸ ਤੇ ਕੰਮ ਕਰਨਾ (PERFORMING OPERATIONS ON STRINGS)

ਜਾਵਾ ਸਟ੍ਰਿੰਗਸ ਤੇ ਵੱਖ-ਵੱਖ ਕੰਮ ਕਰਨ ਲਈ ਵੱਖ-ਵੱਖ ਮੈਥਡ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਅਸੀਂ ਵੱਖ-ਵੱਖ ਮੈਥਡਜ਼ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਨਵੀਂ ਸਟ੍ਰਿੰਗ ਬਣਾ ਸਕਦੇ ਹਾਂ ਅਤੇ ਕਈ ਹੋਰ ਸਟ੍ਰਿੰਗ ਮੈਥਡਜ਼ ਦੀ ਮਦਦ ਨਾਲ ਦਿੱਤੀ ਗਈ ਸਟ੍ਰਿੰਗ ਉੱਪਰ ਕੰਮ ਕਰ ਸਕਦੇ ਹਾਂ। ਆਉ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੇ ਰੂਪ ਵਿੱਚ ਸਟ੍ਰਿੰਗ ਦੀ ਮੁਢਲੀ ਵਰਤੋਂ ਸਿੱਖੀਏ।

ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਇੱਕ ਨਵਾਂ ਸਟ੍ਰਿੰਗ ਵੇਰੀਏਬਲ ਬਣਾਉਣ ਅਤੇ ਉਸ ਉੱਪਰ ਵੱਖ-ਵੱਖ ਕੰਮ ਕਰਨ ਦੇ ਤਰੀਕੇ ਦਿਖਾਉਂਦਾ ਹੈ।

```
class CAProg9
{
        public static void main(String arg[])
        {
```

```
//First way of using String
        String str1 = "Computer Application";
        System.out.println(str1);
//Second way of using String
        String str2=new String("Punjab School Education Board");
        System.out.println(str2);
//Third way of using String
        String str3;
        str3=new String("SAS Nagar");
        System.out.println(str3);
//Fourth way of using String
        char[] strarray = { 'M','o','h','a','l','i' };
        String str4 = new String(strarray);
        System.out.println(str4);
    }
}
```

ਪ੍ਰੋਗਰਾਮ 8.1 (CAProg9.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

ਇਹ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀਆਂ ਸਟ੍ਰਿੰਗਸ ਘੋਸ਼ਣਾ (declare) ਕਰਨ ਦੇ ਤਰੀਕੇ ਦਰਸਾਉਂਦਾ ਹੈ। ਅਸੀਂ ਸਟ੍ਰਿੰਗ ਹੈਂਡਲਿੰਗ (ਸਟ੍ਰਿੰਗਜ਼ ਉੱਪਰ ਕੰਮ ਕਰਨਾ) ਮੈਥਡਜ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਸਟ੍ਰਿੰਗਜ਼ ਉੱਪਰ ਕੁੱਝ ਹੋਰ ਆਪ੍ਰੇਸ਼ਨਜ਼ ਵੀ ਕਰਵਾ ਸਕਦੇ ਹਾਂ। ਆਓ ਸਟਰਿੰਗ ਹੈਂਡਲਿੰਗ ਮੈਥਡਜ਼ ਦੇ ਮਹੱਤਵ ਅਤੇ ਉਦਾਹਰਣਾਂ ਨੂੰ ਸਮਝੀਏ।

8.2.1 ਸਟ੍ਰਿੰਗ ਹੈਂਡਲਿੰਗ ਮੈਥਡਜ਼ (String Handling Methods) :

ਜਾਵਾ ਸਟ੍ਰਿੰਗਸ ਉੱਤੇ ਆਪ੍ਰੇਸ਼ਨ ਲਾਗੂ ਕਰਨ ਦੇ ਕਈ ਵੱਖ-ਵੱਖ ਤਰੀਕੇ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ ਜਿਨ੍ਹਾਂ ਨੂੰ ਸਟ੍ਰਿੰਗ ਮੈਥਡਜ਼ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹਨਾਂ ਮੈਥਡਜ਼ ਦੀ ਵਰਤੋਂ ਸਟ੍ਰਿੰਗ ਤੁਲਨਾ ਕਰਨਾ (), ਸਟ੍ਰਿੰਗ ਇਕੱਠਾ ਕਰਨਾ, ਸਟ੍ਰਿੰਗਸ ਕਾਪੀ ਕਰਨਾ, ਸਟ੍ਰਿੰਗਜ਼ ਦਾ ਕੋਈ ਖਾਸ ਹਿੱਸਾ ਪ੍ਰਾਪਤਾ ਕਰਨਾ, ਸਟ੍ਰਿੰਗਜ਼ ਨੂੰ ਛੋਟੇ ਜਾਂ ਵੱਡੇ ਅੱਖਰਾਂ ਵਿੱਚ ਬਦਲਣਾ ਆਦਿ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਮੈਥਡਜ਼ ਜਿਵੇਂ compare(), concat(), equals(), split(), length(), replace(), compareTo() ਆਦਿ ਅਜਿਹੇ ਮੈਥਡਜ਼ ਦੀਆਂ ਕੁਝ ਉਦਾਹਰਣਾਂ ਹਨ। ਕੁੱਝ ਸਭ ਤੋਂ ਵੱਧ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਸਟਿੰਗ ਮੈਥਡਜ਼ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ:

1. **charAt( ) :** ਇਹ ਮੈਥਡ ਇੱਕ ਸਟ੍ਰਿੰਗ ਵਿੱਚ ਦਿੱਤੇ ਕਿਸੇ ਖਾਸ ਇੰਡੈਕਸ (index) ਉੱਪਰ ਮੌਜੂਦ ਇੱਕ ਅੱਖਰ ਨੂੰ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਸਟ੍ਰਿੰਗ ਵਿੱਚ ਪਹਿਲੇ ਅੱਖਰ ਦਾ ਇੰਡੈਕਸ ਹਮੇਸ਼ਾ ਜ਼ੀਰੋ ਨਾਲ ਸ਼ੁਰੂ ਹੁੰਦਾ ਹੈ, ਦੂਸਰਾ ਅੱਖਰ 1 ਅਤੇ ਇਸ ਤਰ੍ਹਾਂ ਹੀ ਬਾਕੀ ਅੱਖਰਾਂ ਦਾ ਇੰਡੈਕਸ ਤੈਅ ਹੁੰਦਾ ਹੈ। ਇਹ ਮੈਥਡ char ਡੇਟਾ ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦੀ ਹੈ। ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੀ ਗਈ ਹੈ।

ਸਿੰਟੈਕਸ (Syntax) :

Char_Variable= String_Identifier.charAt(Int_index);

ਉਦਾਹਰਨ (Example) :

```
String Str = "Computer";
char result = Str.charAt(1);
System.out.println(result);
```

	0	1	2	3	4	5	6	7
Str.	C	O	M	P	U	T	E	R

ਆਉਟਪੁੱਟ (Output) :

o

ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ charAt() ਮੈਥਡ ਦੇ ਲਾਗੂਕਰਨ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ

```
class CAProg12
{
        public static void main(String arg[])
        {
        String Str = "Computer";
        char result = Str.charAt(1);
        System.out.println(result);
        }
}
```

ਪ੍ਰੋਗਰਾਮ 8.2 (CAProg.12.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg12.java

D:\JavaProg>java CAProg12
o

D:\JavaProg>
```

ਇਸੇ ਤਰ੍ਹਾਂ ਅਸੀਂ ਆਪਣੇ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮਾਂ ਵਿੱਚ ਸਾਰੇ ਸਟ੍ਰਿੰਗ ਮੈਥਡਜ਼ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ।

2. **indexOf():** ਇਹ ਮੈਥਡ charAt() ਫੰਕਸ਼ਨ ਦੁਆਰਾ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਕੰਮ ਤੋਂ ਉਲਟ ਕੰਮ ਕਰਦਾ ਹੈ। ਇਹ ਇੱਕ ਸਟ੍ਰਿੰਗ ਦੇ ਅੰਦਰ ਦਿੱਤੇ ਗਏ ਅੱਖਰ ਦਾ ਇੰਡੈਕਸ ਪਤਾ ਕਰਕੇ ਵਾਪਸ ਦਿੰਦਾ ਹੈ। ਜਾਵਾ ਸਟਿੰਗ ਵਿੱਚ ਅੱਖਰਾਂ ਦਾ ਇੰਡੈਕਸ ਹਮੇਸ਼ਾ ਜ਼ੀਰੋ ਨਾਲ ਸ਼ੁਰੂ ਹੁੰਦਾ ਹੈ। ਇਹ ਮੈਥਡ int ਡੇਟਾ ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੀ ਗਈ ਹੈ।

Syntax :

Char_Variable= String_Identifier.indexOf(Char_Literal);

ਉਦਾਹਰਨ (Example) :

```
String Str = "Computer Application";
int location = Str.indexOf('o');
System.out.println(location);
```

ਆਉਟਪੁੱਟ (Output) :

1

3. length() : ਇਹ ਮੈਥਡ ਕਿਸੇ ਵੀ ਸਟਰਿੰਗ ਦੀ ਲੰਬਾਈ ਪਤਾ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ।ਜਾਵਾ ਸਟ੍ਰਿੰਗ ਦੀ ਲੰਬਾਈ ਸਟ੍ਰਿੰਗ ਦੇ ਅੰਦਰ ਖਾਲੀ ਥਾਵਾਂ ਸਮੇਤ ਅੱਖਰਾਂ ਜਾਂ ਚਿੰਨ੍ਹਾਂ ਦੀ ਗਿਣਤੀ ਦੇ ਬਰਾਬਰ ਹੁੰਦੀ ਹੈ॥ ਇਹ ਮੈਥਡ int ਡਾਟਾ ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ।ਇਸ ਮੈਥਡ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

ਸਿੰਟੈਕਸ (Syntax) :

Int_Variable= String_Identifier.length();

ਉਦਾਹਰਨ (Example) :

```
String Str = "Computer Application";
int count = Str.length();
System.out.println(count);
```

ਆਉਟਪੁੱਟ (Output)

20

4. toLowerCase() : ਇਹ ਜਾਵਾ ਮੈਥਡ ਸਟਰਿੰਗ ਦੇ ਸਾਰੇ ਅੱਖਰਾਂ ਨੂੰ ਲੋਅਰਕੇਸ ਵਿੱਚ ਬਦਲ ਕੇ ਨਤੀਜੇ ਦੇ ਤੌਰ ਤੇ ਵਾਪਸ ਕਰਦਾ ਹੈ।ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹ ਮੈਥਡ ਸਟ੍ਰਿੰਗ ਦੇ ਸਾਰੇ ਅੱਖਰਾਂ ਨੂੰ ਛੋਟੇ ਅੱਖਰਾਂ ਵਿੱਚ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੈਥਡ ਸਟ੍ਰਿੰਗ ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ॥ ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

ਸਿੰਟੈਕਸ (Syntax) :

String_Variable= String_Identifier.toLowerCase();

ਉਦਾਹਰਨ (Example)

```
String Str = "Computer Application";
String str_result = Str.toLowerCase ();
System.out.println(str_result);
```

ਆਉਟਪੁੱਟ (Output) :

computer application

5. toUpperCase() : ਇਹ ਜਾਵਾ ਮੈਥਡ ਸਟਰਿੰਗ ਨੂੰ ਵੱਡੇ ਅੱਖਰਾਂ ਵਿੱਚ ਬਦਲ ਕੇ ਨਤੀਜੇ ਦੇ ਤੌਰ ਤੇ ਵਾਪਸ ਕਰਦਾ ਹੈ। । ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹ ਮੈਥਡ ਸਟ੍ਰਿੰਗ ਦੇ ਸਾਰੇ ਅੱਖਰਾਂ ਨੂੰ ਅੰਗ੍ਰੇਜ਼ੀ ਵਰਨਮਾਲਾ ਦੇ ਵੱਡੇ ਅੱਖਰਾਂ ਵਿੱਚ ਤਬਦੀਲ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ।ਇਹ ਮੈਥਡ String ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ॥ ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਅੱਗੇ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

ਸਿੰਟੈਕਸ (Syntax) :

String_Variable= String_Identifier.toUpperCase();

ਉਦਾਹਰਨ (Example) :

```
String Str = "Computer Application";
String str_result = Str.toUpperCase ();
System.out.println(str_result);
```

ਆਉਟਪੁੱਟ (Output) :

COMPUTER APPLICATION

6. trim() : ਜਾਵਾ ਸਟ੍ਰਿੰਗ ਵਿੱਚ trim() ਮੈਥਡ ਸਟਰਿੰਗ ਤੋਂ ਪਹਿਲਾਂ ਅਤੇ/ਜਾਂ ਬਾਅਦ ਵਿੱਚ ਛੱਡੀ ਗਈ ਖਾਲੀ ਥਾਂ (spaces) ਦੀ ਜਾਂਚ ਕਰਦਾ ਹੈ। ਜੇਕਰ ਕੋਈ ਵੀ ਖਾਲੀ ਥਾਂ (ਸਪੇਸ) ਮੌਜੂਦ ਹੁੰਦੀ ਹੈ ਤਾਂ ਇਹ ਮੈਥਡ ਉਸ ਸਪੇਸ ਨੂੰ ਹਟਾ ਦਿੰਦਾ ਹੈ ਅਤੇ ਬਾਕੀ ਦਾ ਅਸਲ ਸਟ੍ਰਿੰਗ ਮੂਲ ਰੂਪ ਵਿੱਚ ਵਾਪਸ ਕਰਦਾ ਹੈ। trim() ਮੈਥਡ ਕਦੇ ਵੀ ਸਟ੍ਰਿੰਗ ਦੇ ਵਿਚਕਾਰ ਦੀ ਕੋਈ ਵੀ ਸਪੇਸ ਨੂੰ ਖਤਮ ਨਹੀਂ ਕਰਦਾ। ਇਹ ਮੈਥਡ string ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ।ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

ਸਿੰਟੈਕਸ (Syntax) :

String_Variable= String_Identifier.trim();

ਉਦਾਹਰਨ (Example) :

```
String Str = "  Computer Application   ";
String str_result = Str.trim();
System.out.println("["+str_result+"]");
```

ਆਉਟਪੁੱਟ (Output) :

[COMPUTER APPLICATION]

7. subString() : ਜਾਵਾ ਮੈਥਡ substring() ਅੰਕਾਂ ਦੇ ਵਿੱਚ ਦਿੱਤੇ ਗਏ ਆਰਗੂਮੈਂਟਸ ਦੁਆਰਾ ਦਰਸਾਏ ਅਨੁਸਾਰ ਕਿਸੇ ਵੀ ਸਟ੍ਰਿੰਗ ਦਾ ਇੱਕ ਖਾਸ ਹਿੱਸਾ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਦੋ ਵੱਖ-ਵੱਖ ਤਰੀਕਿਆਂ ਨਾਲ ਵਰਤ ਸਕਦੇ ਹਾਂ। ਜੇਕਰ ਅਸੀਂ ਕੇਵਲ ਸ਼ੁਰੂ ਦਾ ਇੰਡੈਕਸ (ਕੇਵਲ ਇੱਕ ਆਰਗੂਮੈਂਟ) ਦਿੰਦੇ ਹਾਂ, ਤਾਂ ਇਹ ਮੈਥਡ ਉਸ ਇੰਡੈਕਸ ਤੋਂ ਲੈ ਕੇ ਬਾਕੀ ਸਾਰੇ ਅੱਖਰ ਵਾਪਸ ਕਰ ਦਿੰਦਾ ਹੈ। ਪਰੰਤੂ ਜੇਕਰ ਅਸੀਂ ਦੋ ਆਰਗੂਮੈਂਟਸ ਦੇ ਰੂਪ ਵਿੱਚ ਸ਼ੁਰੂ ਦਾ ਇੰਡੈਕਸ ਅਤੇ ਅੰਤਿਮ ਇੰਡੈਕਸ ਆਰਗੂਮੈਂਟ ਦੇ ਤੌਰ ਤੇ ਪਾਸ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਇਹ ਮੈਥਡ ਸ਼ੁਰੂ ਵਾਲੇ ਇੰਡੈਕਸ ਅਤੇ ਅੰਤਿਮ ਇੰਡੈਕਸ ਵਿਚਕਾਰ ਸਟ੍ਰਿੰਗ ਵਿੱਚ ਮੌਜੂਦ ਅੱਖਰ ਨਤੀਜੇ ਦੇ ਤੌਰ ਤੇ ਵਾਪਸ ਕਰਦਾ ਹੈ। ।ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਸ਼ੁਰੂ ਦਾ ਇੰਡੈਕਸ ਦੇਣਾ ਇਸ ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਲਈ ਹਮੇਸ਼ਾ ਜਰੂਰੀ ਹੁੰਦਾ ਹੈ । ਪਰੰਤੂ ਅੰਤਿਮ ਇੰਡੈਕਸ ਦੇਣਾ ਸਾਡੇ ਲਈ ਆਪਸ਼ਨਲ (ਗੈਰ-ਲਾਜ਼ਮੀ) ਹੋ ਸਕਦਾ ਹੈ। ਇਹ ਮੈਥਡ String ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ।

ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

Syntax :

String_Variable= String_Identifier.subString(Int_Begin_Index);
String_Variable= String_Identifier.subString(begin_Index,end_Index);

Example :

```
String Str = "Personality";
String str_result1 = Str.substring(3);
System.out.println(str_result1);
String str_result2=str.substring(3,8);
System.out.println(str_result2);
```

Output :

sonality
sonal

8. concat() : ਇਹ ਮੈਥਡ ਇੱਕ ਸਟ੍ਰਿੰਗ ਦੇ ਅੰਤ ਵਿੱਚ ਆਰਗੂਮੈਂਟ ਦੇ ਰੂਪ ਵਿੱਚ ਦਿੱਤੀ ਗਈ ਸਟ੍ਰਿੰਗ ਨੂੰ ਜੋੜਦਾ ਹੈ ਅਤੇ ਇੱਕ ਸੰਯੁਕਤ (combined) ਸਟ੍ਰਿੰਗ ਨਤੀਜੇ ਦੇ ਤੌਰ ਤੇ ਵਾਪਿਸ ਕਰਦਾ ਹੈ ll ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ, ਇਹ ਮੈਥਡ ਕਿਸੇ ਵੀ ਮੌਜੂਦਾ ਸਟ੍ਰਿੰਗ ਨਾਲ ਇੱਕ ਦਿੱਤੇ ਗਏ ਸਟ੍ਰਿੰਗ ਨੂੰ ਜੋੜਦਾ ਹੈ। । ਇਹ ਮੈਥਡ String ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ।

Syntax :

String_Variable= String_Identifier.concat(String_Value);

Example :

```
String Str = "Computer";
String str_result = Str.concat("Application");
System.out.println(str_result);
```

Output :

ComputerApplication

9. replace() : ਜਾਵਾ ਮੈਥਡ replace() ਕਿਸੇ ਵੀ old_string ਵਿਚਲੇ ਕਿਸੇ ਵੀ ਖਾਸ ਸਟ੍ਰਿੰਗ ਨੂੰ ਦਿੱਤੇ ਗਏ ਸਟ੍ਰਿੰਗ ਨਾਲ ਸਾਰੀਆਂ ਥਾਵਾਂ ਤੇ ਬਦਲਣ ਉਪਰੰਤ new_string ਦੇ ਰੂਪ ਵਿੱਚ ਨਤੀਜੇ ਵਜੋਂ ਵਾਪਸ ਕਰਦਾ ਹੈ। । ਇਹ ਸਟ੍ਰਿੰਗ ਮੈਥਡ ਦਿੱਤੀ ਗਈ ਮੋਜੂਦਾ ਸਟ੍ਰਿੰਗ ਵਿੱਚ char ਮੁੱਲਾਂ ਦੀ ਇੱਕ ਲੜੀ ਨੂੰ Replace ਕਰਦਾ ਹੈ। ਇਹ ਮੈਥਡ String ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ।ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ।

Syntax :

String_Variable= String_Identifier.replace(old_string,new_string);

Example :

```
String Str = "Computer was a good and was a modern device";
String str_result = Str.replace("was", "is");
System.out.println(str_result);
```

Output :

Computer is a good and is a modern device

10. String.valueOf() : ਜਾਵਾ ਵਿੱਚ ਇਹ ਮੈਥਡ ਵੱਖ-ਵੱਖ ਡਾਟਾ-ਕਿਸਮਾਂ ਦੇ ਮੁੱਲਾਂ ਨੂੰ ਸਟ੍ਰਿੰਗ ਮੁੱਲ ਵਿੱਚ ਬਦਲਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ ll ਅਸੀਂ ਇਸ ਮੈਥਡ ਦੁਆਰਾ ਡਾਟਾ ਕਿਸਮਾਂ ਜਿਵੇਂ ਕਿ int, long, Boolean, character, float, double, object ਨੂੰ ਸਟ੍ਰਿੰਗ ਡਾਟਾ ਕਿਸਮ ਵਿੱਚ ਬਦਲ ਸਕਦੇ ਹਾਂ। ਇਹ ਮੈਥਡ String ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ।

Syntax :

String_Variable= String.valueOf(Any_DataType_Variable);

Example :

ਨੋਟ : valueOf() ਮੈਥਡ string ਕਲਾਸ ਦਾ static ਮੈਥਡ ਹੈ ਜਿਸ ਕਾਰਨ ਇਸ ਮੈਥਡ ਨੂੰ ਅਸੀਂ ਸਿੱਧਾ ਕਲਾਸ ਦੇ ਨਾਮ ਨਾਲ ਵਰਤ ਸਕਦੇ ਹਾਂ।

Output :

101

```
String str_result = String.valueOf(101);
System.out.println(str_result.length());
```

11. equals() : ਜਾਵਾ ਵਿੱਚ equals() ਮੈਥਡ ਦੋ ਸਟ੍ਰਿੰਗਜ਼ ਦੀ ਤੁਲਨਾ ਕਰਦਾ ਹੈ ਅਤੇ ਜੇਕਰ ਦੋਵੇਂ ਸਟ੍ਰਿੰਗ ਬਿਲਕੁਲ ਬਰਾਬਰ ਹਨ ਤਾਂ true, ਨਹੀਂ ਤਾਂ false ਮੁੱਲ ਵਾਪਿਸ ਕਰਦਾ ਹੈ। ਇਹ ਸਟ੍ਰਿੰਗ ਤੁਲਨਾ ਕਰਨ ਸਮੇਂ ਕੇਸ ਸੰਵੇਦਨਸ਼ੀਲਤਾ (case sensitive) ਦੇ ਆਧਾਰ ਤੇ ਨਤੀਜਾ ਤਿਆਰ ਕਰਦਾ ਹੈ। ਜਦੋਂ ਦੋਵੇਂ ਸਟ੍ਰਿੰਗਜ਼ ਦੇ ਸਾਰੇ ਅੱਖਰ ਇੱਕੋ ਜਿਹੇ ਹੁੰਦੇ ਹਨ ਪਰੰਤੂ ਕੇਸ ਵੱਖਰੇ ਹੁੰਦੇ ਹਨ ਤਾਂ ਵੀ ਇਹ ਮੈਥਡ boolean ਮੁੱਲ false ਵਾਪਸ ਕਰਦਾ ਹੈ। । ਜ਼ਿਆਦਾਤਰ ਮਾਮਲਿਆਂ ਵਿੱਚ, ਇਹ ਫੰਕਸ਼ਨ ਕੰਡੀਸ਼ਨਲ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟ ਜਿਵੇਂ ਕਿ if-else ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। । ਇਹ ਮੈਥਡ Boolean ਡਾਟਾ ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ, ਜਿਸਤੋਂ ਭਾਵ ਹੈ true ਜਾਂ false ਮੁੱਲ। ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

Syntax :

String_Variable.equals(Any_String_Variable or Constant);

Example :

```
String str1 = “Computer”;
String str2 = “COMPUTER”;
if(str1.equals(str2))
System.out.println(“Strings are same”);
else
 System.out.println(“Strings are different”);
```

Output :

Strings are different

12. equalsIgnoreCase() : ਜਾਵਾ ਮੈਥਡ equalsIgnoreCase() ਵੀ ਦੋ ਸਟ੍ਰਿੰਗਜ਼ ਦੀ ਤੁਲਨਾ ਕਰਦਾ ਹੈ, ਅਤੇ ਜੇਕਰ ਸਟ੍ਰਿੰਗ ਬਰਾਬਰ ਹਨ ਤਾਂ true, ਨਹੀਂ ਤਾਂ false ਮੁੱਲ ਵਾਪਿਸ ਕਰਦਾ ਹੈ। ਇਹ ਸਟ੍ਰਿੰਗ ਤੁਲਨਾ ਕਰਨ ਸਮੇਂ ਕੇਸ ਸੰਵੇਦਨਸ਼ੀਲਤਾ (case sensitive) ਦੇ ਆਧਾਰ ਤੇ ਨਤੀਜਾ ਤਿਆਰ ਨਹੀਂ ਕਰਦਾ ਬਲਕਿ ਕੇਵਲ ਅੱਖਰ ਇਕ ਸਮਾਨ (same) ਹੋਣ ਨੂੰ ਤਰਜੀਹ ਦਿੰਦਾ ਹੈ। ਜਦੋਂ ਦੋਵੇਂ ਸਟ੍ਰਿੰਗਜ਼ ਦੇ ਸਾਰੇ ਅੱਖਰ ਇੱਕੋ ਜਿਹੇ ਹੁੰਦੇ ਹਨ ਪਰੰਤੂ ਕੇਸ ਵੱਖਰੇ ਹੁੰਦੇ ਹਨ ਤਾਂ ਵੀ ਇਹ ਮੈਥਡ true ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਜ਼ਿਆਦਾਤਰ ਮਾਮਲਿਆਂ ਵਿੱਚ, ਇਹ ਫੰਕਸ਼ਨ ਕੰਡੀਸ਼ਨਲ ਕੰਟਰੋਲ ਸਟੇਟਮੈਂਟ ਜਿਵੇਂ ਕਿ if-else ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੈਥਡ Boolean ਡਾਟਾ ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ, ਜਿਸਤੋਂ ਭਾਵ ਹੈ true ਜਾਂ false ਮੁੱਲ। ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ।

Syntax :

String_Variable.equalsIgnoreCase(Any_String_Variable or Constant);

Example :

```
String str1 = “Computer”;
String str2 = “COMPUTER”;
if(str1.equalsIgnoreCase(str2))
System.out.println(“Strings are same”);
else
 System.out.println(“Strings are different”);
```

Output :

Strings are same

8.2.2 ਜਾਵਾ ਵਿੱਚ StringBuffer ਕਲਾਸ (StringBuffer class in Java)

ਜਾਵਾ ਵਿੱਚ StringBuffer ਕਲਾਸ ਨੂੰ ਇੱਕ ਪਰਿਵਰਤਨਸ਼ੀਲ (mutable) ਸਟ੍ਰਿੰਗ ਆਬਜੈਕਟ ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਪਹਿਲਾਂ ਪੜ੍ਹ ਚੁੱਕੇ ਹਾਂ ਕਿ String ਗੈਰ ਪਰਿਵਰਤਨਸ਼ੀਲ (immutable) ਹੁੰਦਾ ਹੈ। ਇੱਕ ਵਾਰ ਇੱਕ String ਘੋਸ਼ਿਤ (declare) ਹੋ ਜਾਣ ਤੋਂ ਬਾਅਦ ਇਸਨੂੰ ਬਦਲਿਆ ਨਹੀਂ ਜਾ ਸਕਦਾ ਹੈ, ਦੂਜੇ ਪਾਸੇ StringBuffer ਕਲਾਸ ਵਿੱਚ ਸਟੋਰ ਕੀਤੇ ਸਟਿੰਗ ਨੂੰ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਜਾਣਾ ਵਿੱਚ StringBuffer ਕਲਾਸ ਦੀ ਵਰਤੋਂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹੋ ਸਕਦੀ ਹੈ।

Syntax:

StringBuffer str = new StringBuffer();

ਇਸ ਤਰੀਕੇ ਨਾਲ ਬਣਾਇਆ StringBuffer ਕਲਾਸ ਦਾ ਹਰੇਕ ਆਬਜੈਕਟ ਮੁੜ-ਰਾਖਵਾਂਕਰਣ (reallocation) ਕੀਤੇ ਬਿਨ੍ਹਾਂ 16 ਅੱਖਰਾਂ ਲਈ ਮੈਮਰੀ ਰਾਖਵੀਂ ਰੱਖਦਾ ਹੈ।

StringBuffer str = new StringBuffer(20);

ਇਸ ਤਰੀਕੇ ਰਾਹੀਂ ਬਣਾਇਆ StringBuffer ਕਲਾਸ ਦਾ ਹਰੇਕ ਆਬਜੈਕਟ ਆਰਗੂਮੈਂਟ ਦੇ ਤੌਰ ਤੇ ਦਿੱਤੇ ਗਏ ਅੰਕ, ਜੋ ਸਟ੍ਰਿੰਗ ਦੇ ਸਟੋਰੇਜ਼ ਲਈ ਵਰਤੇ ਜਾਣ ਵਾਲੇ ਬਫਰ ਦਾ ਆਕਾਰ ਬਿਆਨ ਕਰਦਾ ਹੈ, ਦੇ ਅਨੁਸਾਰ ਮੈਮਰੀ ਰਾਖਵੀਂ ਰੱਖਦਾ ਹੈ।

StringBuffer str = new StringBuffer("ComputerApplication");

ਇਸ ਤਰੀਕੇ ਨਾਲ StringBuffer ਕਲਾਸ ਦਾ ਬਣਾਇਆ ਹਰੇਕ ਆਬਜੈਕਟ ਇੱਕ ਸਟ੍ਰਿੰਗ ਆਰਗੂਮੈਂਟ ਦੇ ਰੂਪ ਵਿੱਚ ਦਿੱਤੇ ਗਏ ਸਟ੍ਰਿੰਗ ਨੂੰ StringBuffer ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਲਈ ਸ਼ੁਰੂਆਤੀ ਮੁੱਲ ਦੇ ਤੌਰ ਤੇ ਸੈੱਟ ਕਰਦਾ ਹੈ।ਉਦਾਹਰਨ

```
class CAProg11
{
        public static void main(String arg[])
        {
        StringBuffer str1=new StringBuffer();
        str1.insert(0,"Computer");
        System.out.println(str1);
        StringBuffer str2=new StringBuffer(25);
        str2.insert(0,"Application");
        System.out.println(str2);
        StringBuffer str3=new StringBuffer("PSEB");
        System.out.println(str3);
        }
}
```

ਪ੍ਰੋਗਰਾਮ 8.3 (CAProg.11.java) StringBuffer ਕਲਾਸ ਦੀ ਵਰਤੋਂ ਕਰਨਾ :

```
D:\JavaProg>javac CAProg11.java

D:\JavaProg>java CAProg11
Computer
Application
PSEB

D:\JavaProg>
```

8.2.3 StringBuffer ਕਲਾਸ ਦੇ ਵੱਖ ਵੱਖ ਮੈਥਡਜ਼ (StringBuffer Class Methods) :

ਉੱਪਰ ਦਰਸ਼ਾਏ ਗਏ ਸਾਰੇ ਸਟ੍ਰਿੰਗ ਮੈਥਡਜ਼ ਨੂੰ StringBuffer ਕਲਾਸ ਆਬਜੈਕਟ ਨਾਲ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਅਸੀਂ StringBuffer ਕਲਾਸ ਆਬਜੈਕਟ ਵਿੱਚ ਸਟੋਰ ਕੀਤੇ ਸਟ੍ਰਿੰਗਜ਼ ਉੱਪਰ ਕੁਝ ਹੋਰ ਆਪ੍ਰੇਸ਼ਨ ਲਾਗੂ ਕਰਨ ਲਈ ਕਈ ਹੋਰ String Buffer ਮੈਥਡਜ਼ ਦੀ ਵਰਤੋਂ ਵੀ ਕਰ ਸਕਦੇ ਹਾਂ ਜੋ ਕਿ String ਕਲਾਸ ਉੱਪਰ ਲਾਗੂ ਨਹੀਂ ਹੁੰਦੇ ਆਓ ਅਜਿਹੇ ਮੈਥਡਜ਼ ਬਾਰੇ ਵਿਸਥਾਰ ਵਿੱਚ ਚਰਚਾ ਕਰੀਏ।

1. **reverse() :** ਇਹ ਮੈਥਡ StringBuffer ਆਬਜੈਕਟ ਦੇ ਅੰਦਰ ਸਟੋਰ ਕੀਤੇ ਸਟਰਿੰਗ ਨੂੰ ਉਲਟਾਉਂਦਾ ਹੈ ਅਤੇ ਨਤੀਜੇ ਦੇ ਤੌਰ ਤੇ ਵਾਪਸ ਕਰਦਾ ਹੈ। ਇਹ ਮੈਥਡ ਸਟ੍ਰਿੰਗ ਵਿੱਚ ਆਖਰੀ ਇੰਡੈਕਸ ਉੱਪਰ ਮੌਜੂਦ ਅੱਖਰ ਨੂੰ ਸਭ ਤੋਂ ਪਹਿਲੇ ਇੰਡੈਕਸ, ਭਾਵ ਜ਼ੀਰੋ ਦੇ ਤੌਰ ਤੇ ਲਗਾ ਕੇ ਸਟ੍ਰਿੰਗ ਵਾਪਸ ਕਰਦਾ ਹੈ। । ਇਹ ਉਸ ਆਬਜੈਕਟ ਵਿੱਚ ਸਟੋਰ ਸਟ੍ਰਿੰਗ ਮੁੱਲ ਨੂੰ ਮੂਲ ਰੂਪ ਵਿੱਚ ਹੀ ਬਦਲ ਦਿੰਦਾ ਹੈ, ਜਿਸ ਨਾਲ reverse() method ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। । ਇਹ ਮੈਥਡ StringBuffer ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ।ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ:

   Syntax :

   String_Variable= StringBuffer_Object.reverse ();

   Example :

```
StringBuffer Str = new StringBuffer("Computer");
StringBuffer result = Str.reverse();
System.out.println(result);
System.out.println(Str);
```

   Output :

   retupmoC
   retupmoC

2. **append() :** ਇਹ ਮੈਥਡ ਆਰਗੂਮੈਂਟ ਦੇ ਤੌਰ ਤੇ ਦਿੱਤੀ ਗਈ ਇੱਕ ਸਟ੍ਰਿੰਗ ਨੂੰ ਮੌਜੂਦਾ ਸਟ੍ਰਿੰਗ ਦੇ ਨਾਲ ਜੋੜਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। append() ਮੈਥਡ ਨੂੰ ਕਈ ਤਰੀਕਿਆਂ ਰਾਹੀਂ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ, ਜਿਵੇਂ append(char), append(boolean), append(int), append(float), append (double) ਆਦਿ॥ ਇਹ ਉਸ ਆਬਜੈਕਟ ਵਿੱਚ ਸਟੋਰ ਸਟਿੰਗ ਮੁੱਲ ਨੂੰ ਮੂਲ ਰੂਪ ਵਿੱਚ ਹੀ ਬਦਲ ਦਿੰਦਾ ਹੈ, ਜਿਸ ਨਾਲ append ( ) ਮੈਥਡ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੈਥਡ StringBuffer ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ।ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ। ।

   Syntax :

   String_Variable= StringBuffer_Object.append ();

   Example :

```
StringBuffer Str = new StringBuffer("Computer");
Str.append("Application");
System.out.println(Str);
```

Output :

ComputerApplication

3. insert() : ਇਹ ਮੈਥਡ ਇੱਕ ਮੌਜੂਦਾ StringBuffer ਆਬਜੈਕਟ ਦੇ ਵਿਚ ਦਿੱਤੇ ਗਏ ਸਟ੍ਰਿੰਗ ਨੂੰ ਦਾਖਲ (insert) ਕਰ ਦਿੰਦਾ ਹੈ। ਇਹ ਇੱਕ ਅੰਕ ਅਤੇ ਇੱਕ ਸਟ੍ਰਿੰਗ ਦੇ ਰੂਪ ਵਿੱਚ ਦੋ ਆਰਗੂਮੈਂਟ ਸਵੀਕਾਰ ਕਰਦਾ ਹੈ। ਪਹਿਲਾ ਪੈਰਾਮੀਟਰ ਉਸ ਇੰਡੈਕਸ ਨੂੰ ਦਰਸ਼ਾਉਂਦਾ ਹੈ, ਜਿਸ ਜਗ੍ਹਾ ਤੇ ਦਿੱਤਾ ਗਿਆ ਸਟ੍ਰਿੰਗ ਸ਼ਾਮਿਲ ਕੀਤਾ ਜਾਵੇਗਾ ਅਤੇ ਦੂਸਰਾ ਪੈਰਾਮੀਟਰ ਜੋ ਕਿ ਸਟ੍ਰਿੰਗ ਦੇ ਰੂਪ ਵਿੱਚ ਹੁੰਦਾ ਹੈ, ਸਟ੍ਰਿੰਗਬਫਰ ਆਬਜੈਕਟ ਵਿੱਚ ਸ਼ਾਮਲ ਕੀਤੇ ਜਾਣ ਵਾਲੇ ਸਟ੍ਰਿੰਗ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ਇਹ ਉਸ ਆਬਜੈਕਟ ਵਿੱਚ ਸਟੋਰ ਸਟ੍ਰਿੰਗ ਮੁੱਲ ਨੂੰ ਮੂਲ ਰੂਪ ਵਿੱਚ ਹੀ ਬਦਲ ਦਿੰਦਾ ਹੈ, ਜਿਸ ਨਾਲ insert() method ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਮੈਥਡ StringBuffer ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ॥ ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

Syntax :

String_Variable= StringBuffer_Object.insert (Index,String_Value);

Example :

```
StringBuffer Str = new StringBuffer("Computer PSEB");
Str.insert(9,"Application");
System.out.println(Str);
```

Output :

Computer ApplicationPSEB

4. capacity() : ਇਹ ਮੈਥਡ StringBuffer ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਦੀ ਮੌਜੂਦਾ ਸਮਰੱਥਾ (capacity) ਦੱਸਦਾ ਹੈ। ਅਸੀਂ StringBuffer ਆਬਜੈਕਟ ਦੀ ਸਮਰੱਥਾ ਆਬਜੈਕਟ ਬਣਾਉਣ ਸਮੇਂ ਨਿਰਧਾਰਤ ਕਰ ਸਕਦੇ ਹਾਂ। ਇਸ ਸਥਿਤੀ ਵਿੱਚ ਜੇਕਰ ਆਰਗੂਮੈਂਟ ਨਹੀਂ ਦਿੱਤਾ ਜਾਂਦਾ ਤਾਂ ਇੱਕ ਖਾਲੀ ਕੰਸਟਰਕਟਰ ਐਗਜ਼ੀਕਿਉਟ ਹੁੰਦਾ ਜੋ 16 ਅੱਖਰਾਂ ਲਈ ਮੈਮਰੀ ਰਾਖਵੀਂ (reserves) ਕਰਦਾ ਹੈ। ਇਹ ਮੈਥਡ int ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ।

ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ।

Syntax:

Int_Variable= StringBuffer_Object.capacity ();

Example :

```
StringBuffer Str = new StringBuffer("Computer Application");
System.out.println(Str.capacity);
```

Output :

36

ਇੱਥੇ StringBuffer ਕੰਸਟਰਕਟਰ ਨੂੰ ਆਰਗੂਮੈਂਟ ਦੇ ਰੂਪ ਵਿੱਚ 20 ਕਰੈਕਟਰਜ਼ ਵਾਲਾ ਸਟ੍ਰਿੰਗ ਦਿੱਤਾ ਗਿਆ ਹੈ ਅਤੇ 16 ਹੋਰ ਸਪੇਸਜ਼ ਆਪਣੇ ਆਪ ਰਾਖਵੀਆਂ ਸੈਟ ਕੀਤੀਆਂ ਗਈਆਂ ਹਨ। ਇਸ ਲਈ ਉਕਤ ਪ੍ਰੋਗਰਾਮ 36 ਮੁੱਲ ਆਉਟਪੁੱਟ ਦੇ ਤੌਰ ਤੇ ਦਰਸ਼ਾ ਰਿਹਾ ਹੈ।

5. delete() : ਜਾਵਾ delete() ਮੈਥਡ StringBuffer ਆਬਜੈਕਟ ਵਿੱਚੋਂ ਕਿਸੇ ਵੀ ਖਾਸ ਸਟਿੰਗ ਨੂੰ ਡਿਲੀਟ (delete) ਕਰ ਦਿੰਦਾ ਹੈ। ਅਸੀਂ ਇਸ ਮੈਥਡ ਵਿੱਚ ਦੋ ਮੁੱਲ ਆਰਗੂਮੈਂਟ ਦੇ ਤੌਰ ਤੇ ਪ੍ਰਦਾਨ ਕਰਦੇ ਹਾਂ। ਜਿਸ ਵਿੱਚ ਪਹਿਲਾ ਆਰਗੂਮੈਂਟ ਸਟਿੰਗ ਦਾ ਸ਼ੁਰੂਆਤੀ ਇੰਡੈਕਸ ਅਤੇ ਦੂਜਾ ਆਰਗੂਮੈਂਟ ਸਟ੍ਰਿੰਗ ਦੀ ਆਖਰੀ ਲੋਕੇਸ਼ਨ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ ਜਿਸ ਨੂੰ ਅਸੀਂ ਮਿਟਾਉਣਾ ਚਾਹੁੰਦੇ ਹਾਂ। ਇਹ ਮੈਥਡ StringBuffer ਕਿਸਮ ਦਾ ਮੁੱਲ ਵਾਪਸ ਕਰਦਾ ਹੈ। ।ਅਸੀਂ ਇਸ ਮੈਥਡ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਅੱਗੇ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ।

Syntax:

String_Variable= StringBuffer_Object.delete(Start_index,End_Index);

Example:

```
StringBuffer Str = new StringBuffer("ComputerApplication");
Str.delete(3,15);
System.out.println(Str);
```

Output :

Comtion

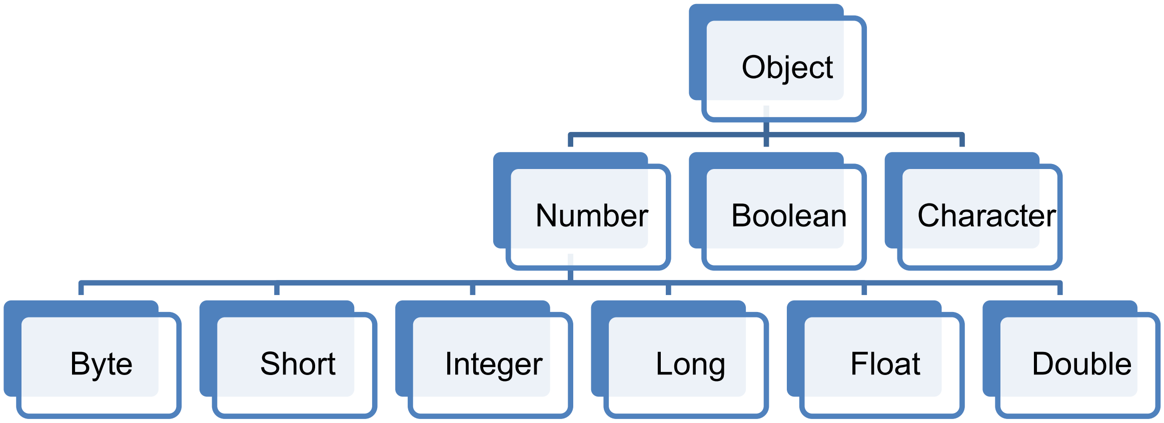
8.3 ਜਾਵਾ ਰੈਪਰ ਕਲਾਸਾਂ (JAVA WRAPPER CLASSES)

ਜਾਵਾ ਵਿੱਚ ਰੈਪਰ (Wrapper) ਕਲਾਸ ਉਹ ਕਲਾਸ ਹੁੰਦੀ ਹੈ ਜਿਸਦੇ ਆਬਜੈਕਟ ਵਿੱਚ ਮੁੱਢਲੀਆਂ ਡੇਟਾ ਕਿਸਮਾਂ (primitive data types) ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ। ਜਦੋਂ ਅਸੀਂ ਇੱਕ ਰੈਪਰ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਬਣਾਉਂਦੇ ਹਾਂ ਤਾਂ ਇਸ ਵਿੱਚ ਇੱਕ ਫੀਲਡ ਹੁੰਦਾ ਹੈ। ਇਸ ਫੀਲਡ ਵਿੱਚ ਅਸੀਂ ਉਸ ਖਾਸ ਮੁੱਢਲੀ ਡੇਟਾ ਕਿਸਮਾਂ ਦਾ ਕੋਈ ਵੀ ਮੁੱਲ ਨੂੰ ਸਟੋਰ ਕਰ ਸਕਦੇ ਹਾਂ।ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇੱਕ ਮੁੱਢਲੀ ਡਾਟਾ ਕਿਸਮ ਦਾ ਕੋਈ ਵੀ ਮੁੱਲ ਇੱਕ ਰੈਪਰ ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਵਿੱਚ ਸਟੋਰ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਕਿਸੇ ਵੀ ਅਜਿਹੇ ਆਬਜੈਕਟ ਵਿੱਚ ਮੁੱਢਲੀ ਡਾਟਾ ਕਿਸਮ ਤੋਂ ਆਬਜੈਕਟ ਕਿਸਮ ਵਿੱਚ ਆਟੋਮੈਟਿਕ ਰੁਪਾਂਤਰਣ (automatic conversion) ਨੂੰ ਆਟੋਬਾਕਸਿੰਗ (autoboxing) ਅਤੇ ਇਸ ਤੋਂ ਉਲਟ ਰੁਪਾਂਤਰਣ ਨੂੰ ਅਨਬਾਕਸਿੰਗ (unboxing) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।

java.lang ਪੈਕੇਜ ਦੀਆਂ ਅੱਠ ਕਲਾਸਾਂ ਜਾਵਾ ਵਿੱਚ ਰੈਪਰ ਕਲਾਸਾਂ ਵਜੋਂ ਜਾਣੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।ਇਹ ਰੈਪਰ ਕਲਾਸਾਂ ਇਸ ਪ੍ਰਕਾਰ ਹਨ:

ਮੁੱਢਲੀ ਕਿਸਮ (Primitive Type)	ਰੈਪਰ ਕਲਾਸ (Wrapper class)
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

ਅਸੀਂ ਇਹਨਾਂ ਸਾਰੇ ਵਰਗਾਂ ਨੂੰ ਨਿਮਨ ਅਨੁਸਾਰ ਇੱਕ ਲੜੀ ਵਿੱਚ ਦਰਸਾ ਸਕਦੇ ਹਾਂ:



8.3.1 ਰੈਪਰ ਕਲਾਸਾਂ ਦੀ ਜਰੂਰਤ Need of Wrapper Classes

1. ਇਹ ਕਲਾਸਾਂ ਮੁੱਢਲੇ ਡੇਟਾ ਕਿਸਮਾਂ ਨੂੰ ਆਬਜੈਕਟਸ ਵਿੱਚ ਬਦਲਦੀਆਂ ਹਨ। ਅਜਿਹੇ ਆਬਜੈਕਟ ਉਹਨਾਂ ਖਾਸ ਸਥਿਤੀਆਂ ਵਿੱਚ ਲੋੜੀਂਦੇ ਹੁੰਦੇ ਹਨ, ਜਦੋਂ ਆਬਜੈਕਟਸ ਨੂੰ ਇੱਕ ਆਰਗੂਮੈਂਟ ਵਜੋਂ ਕਿਸੇ ਮੈਥਡ ਨੂੰ ਭੇਜਿਆ ਜਾਣਾ ਹੋਵੇ ਅਤੇ ਉਹਨਾਂ ਉੱਪਰ ਕੋਈ ਖਾਸ ਆਪ੍ਰੇਸ਼ਨ ਲਾਗੂ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇ।
2. java.util ਪੈਕੇਜ ਵਿੱਚ ਮੌਜੂਦ ਕਲਾਸਾਂ ਸਿਰਫ ਆਬਜੈਕਟਸ ਨੂੰ ਹੈਂਡਲ ਕਰਦੀਆਂ ਹਨ ਅਤੇ ਇਸ ਸਥਿਤੀ ਵਿੱਚ ਰੈਪਰ ਕਲਾਸਾਂ ਬਹੁਤ ਲਾਹੇਵੰਦ ਹੁੰਦੀਆਂ ਹਨ।
3. ਕੁਲੈਕਸ਼ਨ ਫਰੇਮਵਰਕ (Collection framework) ਵਿੱਚ ਡਾਟਾ ਸਟਰਕਚਰ, ਜਿਵੇਂ ਕਿ ArrayList ਅਤੇ Vector, ਸਿਰਫ ਆਬਜੈਕਟ ਸਟੋਰ ਕਰਦੇ ਹਨ ਅਤੇ ਮੁੱਢਲੀਆਂ ਕਿਸਮਾਂ ਤੇ ਕੰਮ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ।
4. ਮਲਟੀਥ੍ਰੈਡਿੰਗ (multithreading) ਵਿੱਚ ਸਿੰਕ੍ਰੋਨਾਈਜੇਸ਼ਨ (synchronization) ਲਾਗੂ ਕਰਨ ਲਈ ਵੀ ਇੱਕ ਆਬਜੈਕਟ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ॥

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਸਟ੍ਰਿੰਗ ਆਮ ਤੌਰ ਤੇ ਵਰਤੀ ਜਾਣ ਵਾਲੀ ਡਾਟਾ ਕਿਸਮ ਹੈ ਜੋ ਸਾਨੂੰ ਕਿਸੇ ਵੀ ਵਸਤੂ ਨਾਲ ਸੰਬੰਧਤ ਕੋਈ ਡਾਟਾ ਜਾਂ ਹਰ ਉਹ ਮੁੱਲ ਨੂੰ ਸਟੋਰ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ ਦਿੰਦੀ ਹੈ ਜੋ ਸ਼ਬਦਾਂ (words), ਵਾਕਾਂ (sentences), ਫ਼ੋਨ ਨੰਬਰਾਂ (phone numbers) ਜਾਂ ਇੱਥੋਂ ਤੱਕ ਕਿ ਸਿਰਫ਼ ਨਿਯਮਤ ਨੰਬਰਾਂ ਨਾਲ ਸਬੰਧਤ ਹੋਵੇ।
2. ਅੱਖਰਾਂ (characters) ਦੇ ਕ੍ਰਮ ਨੂੰ ਦਰਸਾਉਣ ਲਈ ਸਟ੍ਰਿੰਗਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।
3. ਸਟ੍ਰਿੰਗ ਅਸਲ ਵਿੱਚ ਇੱਕ ਆਬਜੈਕਟ ਹੁੰਦਾ ਹੈ, ਜਿਸ ਵਿੱਚ ਉਹ ਸਾਰੇ ਮੈਥਡਜ਼ ਸ਼ਾਮਿਲ ਹੁੰਦੀਆਂ ਹਨ ਜੋ ਸਟ੍ਰਿੰਗਸ ਉੱਪਰ ਕੁਝ ਖਾਸ ਕੰਮ ਜਾਂ ਆਪ੍ਰੇਸ਼ਨ ਲਗਾਉਣ ਲਈ ਵਰਤੇ ਜਾ ਸਕਦੇ ਹਨ।
4. ਜਾਵਾ ਸਟ੍ਰਿੰਗਸ ਉਪਰ ਵੱਖ-ਵੱਖ ਕੰਮ ਕਰਨ ਲਈ ਵੱਖ-ਵੱਖ ਮੈਥਡ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਜਿਨ੍ਹਾਂ ਨੂੰ ਸਟ੍ਰਿੰਗ ਮੈਥਡਜ਼ ਕਿਹਾ ਜਾਂਦਾ ਹੈ।
5. ਸਟ੍ਰਿੰਗ ਗੈਰ-ਪਰਿਵਰਤਨਸ਼ੀਲ (immutable) ਆਬਜੈਕਟ ਹੁੰਦੇ ਹਨ।
6. ਜਾਵਾ ਵਿੱਚ StringBuffer ਕਲਾਸ ਨੂੰ ਇੱਕ ਪਰਿਵਰਤਨਸ਼ੀਲ (mutable) ਸਟ੍ਰਿੰਗ ਆਬਜੈਕਟ ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।
7. ਜਾਵਾ ਵਿੱਚ ਰੈਪਰ (Wrapper) ਕਲਾਸ ਉਹ ਕਲਾਸ ਹੁੰਦੀ ਹੈ ਜਿਸਦੇ ਆਬਜੈਕਟ ਵਿੱਚ ਮੁੱਢਲੀਆਂ ਡੇਟਾ ਕਿਸਮਾਂ (primitive data types) ਸ਼ਾਮਲ ਹੁੰਦੀਆਂ ਹਨ।
8. java.util ਪੈਕੇਜ ਵਿੱਚ ਮੌਜੂਦ ਕਲਾਸਾਂ ਸਿਰਫ ਆਬਜੈਕਟਸ ਨੂੰ ਹੈਂਡਲ ਕਰਦੀਆਂ ਹਨ।
9. ਕਲੈਕਸ਼ਨ ਫਰੇਮਵਰਕ (Collection framework) ਵਿੱਚ ਡਾਟਾ ਸਟਰਕਚਰ ਜਿਵੇਂ ਕਿ ArrayList ਅਤੇ Vector, ਸਿਰਫ ਆਬਜੈਕਟ ਸਟੋਰ ਕਰਦੇ ਹਨ ਅਤੇ ਮੁੱਢਲੀਆਂ ਕਿਸਮਾਂ ਤੇ ਕੰਮ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ।

ਅਭਿਆਸ

ਪ੍ਰਸ਼ਨ : 1 ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ :

1. ਇੱਕ ਸਟ੍ਰਿੰਗ ਇਹਨਾਂ ਵਿਚੋਂ ਕਿਸ ਢੰਗ ਨਾਲ ਬਣਾਈ ਜਾ ਸਕਦੀ ਹੈ ?
   ੳ. ਸਟਰਿੰਗ ਲਿਟਰਲ (String Literal) ਦੁਆਰਾ
   ਅ. new ਕੀਅ-ਵਰਡ ਦੁਆਰਾ (By new Keyword)
   ੲ. ਦੋਵੇਂ ੳ ਅਤੇ ਅ
   ਸ. ਇਹਨਾਂ ਵਿੱਚੋਂ ਕੋਈ ਨਹੀਂ
2. ....ਮੈਥਡ ਇੱਕ ਸਟ੍ਰਿੰਗ ਵਿੱਚ ਦਿੱਤੇ ਗਏ ਇੰਡੈਕਸ ਤੇ ਮੌਜੂਦ ਅੱਖਰ ਵਾਪਸ ਕਰਦਾ ਹੈ।

ੳ. IndexOf()　　ਅ. charAt()

ੲ. length()　　ਸ. trim()

3. ....ਮੈਥਡ ਬਿਨਾਂ ਕੇਸ ਸੰਵੇਦਨਸ਼ੀਲਤਾ (case sensitivity) ਤੋਂ ਦੋ ਸਟਿੰਗਜ਼ ਦੀ ਤੁਲਨਾ ਕਰਦਾ ਹੈ।

ੳ. compare()　　ਅ. equals()

ੲ. equalsIgnoreCase()　　ਸ. ਉਪਰੋਕਤ ਸਾਰੇ

4. ਮੁੱਢਲੇ ਡੇਟਾ ਕਿਸਮਾਂ ਦੇ ਆਬਜੈਕਟ........... ਕਲਾਸਾਂ ਵਿੱਚ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ।

ੳ. ਡਿਫਾਲਟ ਕਲਾਸ (Default Class)　　ਅ. ਰੈਪਰ ਕਲਾਸ (Wrapper Class)

ੲ. ਫਾਈਨਲ ਕਲਾਸ (Final Class)　　ਸ. ਸਟੈਟਿਕ ਕਲਾਸ (Static Class)

5. ਕਿਹੜਾ ਇੱਕ ਮੁੱਢਲੀ ਡੇਟਾ ਕਿਸਮ (primitive data type) ਦੀ ਉਦਾਹਰਨ ਨਹੀਂ ਹੈ ?

ੳ. char　　ਅ. int

ੲ. short　　ਸ. Double

ਪ੍ਰਸ਼ਨ:2 ਖਾਲੀ ਥਾਵਾਂ ਭਰੋ:

I. .... ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਇਸਦੇ ਲਾਗੂ ਕਰਨ ਦੇ ਦ੍ਰਿਸ਼ਟੀਕੋਣ ਅਨੁਸਾਰ ਪਰਿਵਰਤਨਸ਼ੀਲ (mutable) ਹੁੰਦਾ ਹੈ।

ii .... ਮੈਥਡ ਸਟ੍ਰਿੰਗ ਦੇ ਕਿਸੇ ਵੀ ਸਿਰੇ ਤੋਂ ਸਾਰੀਆਂ ਖਾਲੀ ਥਾਂਵਾਂ ਨੂੰ ਹਟਾ ਦਿੰਦਾ ਹੈ।

iii .... ਅੱਖਰਾਂ ਦੇ ਕ੍ਰਮ (Sequence) ਦੇ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

iv. StringBuffer ਆਬਜੈਕਟ ਤੋਂ ਕਿਸੇ ਖਾਸ ਸਟ੍ਰਿੰਗ ਨੂੰ ਮਿਟਾਉਣ ਲਈ .... ਮੈਥਡ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।

v. .... ਮੁੱਢਲੀਆਂ ਡੇਟਾ ਕਿਸਮਾਂ (primitive data types) ਨੂੰ ਆਬਜੈਕਟ ਦੇ ਰੂਪ ਵਿੱਚ ਬਦਲਦਾ ਹੈ॥

ਪ੍ਰਸ਼ਨ :3 ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਸਟ੍ਰਿੰਗਸ ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ?

ii. ਕਿਸੇ ਆਬਜੈਕਟ ਨੂੰ ਸਟ੍ਰਿੰਗ ਲਿਟਰਲ ਵਜੋਂ ਬਣਾਉਣ ਦਾ ਤਰੀਕਾ ਦੱਸੋ।

iii. ਸਟ੍ਰਿੰਗ ਦੇ ਕੀ ਲਾਭ ਹਨ ?

iv. ਸਟ੍ਰਿੰਗ ਦੀਆਂ ਸੀਮਾਵਾਂ (limitations) ਦੀ ਵਿਆਖਿਆ ਕਰੋ॥

v. ਉਦਾਹਰਨ ਦੇ ਨਾਲ to LowerCase() ਮੈਥਡ ਨੂੰ ਪਰਿਭਾਸ਼ਿਤ ਕਰੋ।

vi. equals() ਅਤੇ equalsIgnoreCase() ਵਿੱਚ ਕੀ ਅੰਤਰ ਹੁੰਦਾ ਹੈ ?

vii. ਅਸੀਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੇ ਮੁੱਲਾਂ ਨੂੰ ਸਟ੍ਰਿੰਗ ਮੁੱਲ ਵਿੱਚ ਕਿਵੇਂ ਬਦਲ ਸਕਦੇ ਹਾਂ ?

viii. StringBuffer ਕਲਾਸ ਕੀ ਹੁੰਦੀ ਹੈ ?

ix. capacity ਮੈਥਡ ਦੀ ਵਿਆਖਿਆ ਕਰੋ॥

x. ਰੈਪਰ ਕਲਾਸ ਕੀ ਹੁੰਦੀ ਹੈ ?

ਪ੍ਰਸ਼ਨ :4 ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਸਟ੍ਰਿੰਗਸ ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ? ਉਦਾਹਰਨ ਦੇ ਨਾਲ ਇੱਕ ਸਟ੍ਰਿੰਗ ਡਿਕਲੇਅਰ ਕਰਨ ਦੇ ਵੱਖ-ਵੱਖ ਤਰੀਕਿਆਂ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।

ii. ਸਟ੍ਰਿੰਗ ਹੈਂਡਲਿੰਗ (string handling) ਮੈਥਡਜ਼ ਦੀ ਵਿਆਖਿਆ ਕਰੋ। ਪ੍ਰੋਗਰਾਮ ਦੀ ਮਦਦ ਨਾਲ ਕੋਈ ਵੀ ਦੋ ਮੈਥਡਜ਼ (methods) ਦਾ ਵਰਨਣ ਕਰੋ।

iii. replace( ) ਅਤੇ subString() ਮੈਥਡਜ਼ ਨੂੰ ਢੁਕਵੀਆਂ ਉਦਾਹਰਣਾਂ ਦੇ ਕੇ ਸਮਝਾਓ।

iv. ਅਸੀਂ StringBuffer ਕਲਾਸ ਆਬਜੈਕਟ ਦੀ ਵਰਤੋਂ ਕਿਉਂ ਕਰਦੇ ਹਾਂ ? StringBuffer ਕਲਾਸ ਦੇ ਕਿਸੇ ਵੀ ਦੋ ਮੈਥਡਜ਼ ਦੀ ਵਿਆਖਿਆ ਕਰੋ॥

ਇਨਹੈਰੀਟੈਂਸ
(Inheritance)

ਇਸ ਪਾਠ ਦੇ ਉਦੇਸ਼

9.1 ਇਨਹੈਰੀਟੈਂਸ ਅਤੇ ਇਸ ਦੀਆਂ ਕਿਸਮਾਂ

9.2 ਇੰਟਰਫੇਸ

9.3 ਜਾਵਾ ਵਿਚ Super ਕੀਵਰਡ

9.4 ਜਾਵਾ ਵਿਚ ਐਬਸਟਰੈਕਸ਼ਨ

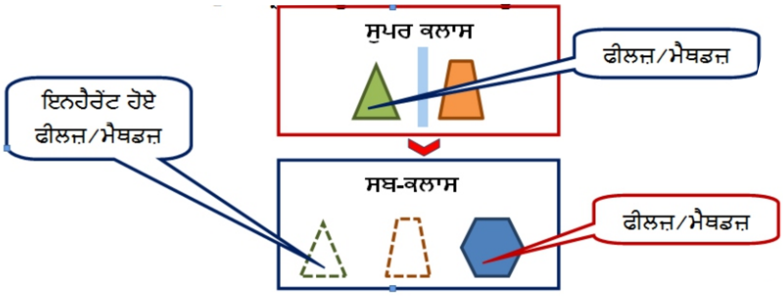
9.5 ਜਾਵਾ ਵਿਚੋਂ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ (Overriding)

9.6 Final ਕਲਾਸ, ਮੈਥਡ ਅਤੇ ਵੈਰੀਏਬਲ

9.1 ਇਨਹੈਰੀਟੈਂਸ ਨਾਲ ਜਾਣ-ਪਛਾਣ (INTRODUCTION TO INHERITANCE)

ਇਨਹੈਰੀਟੈਂਸ (Inheritance) ਇਕ ਵਿਸ਼ੇਸ਼ਤਾ ਜਾਂ ਇੱਕ ਪ੍ਰਕਿਰਿਆ ਹੁੰਦੀ ਹੈ ਜਿਸ ਵਿੱਚ ਮੌਜੂਦਾ ਕਲਾਸ ਤੋਂ ਨਵੀਆਂ ਕਲਾਸਾਂ ਬਣਾਈਆਂ ਜਾਂਦੀਆਂ ਹਨ। ਇਸ ਵਿਚ ਮੌਜੂਦਾ ਕਲਾਸ ਨੂੰ ਸੁਪਰ ਕਲਾਸ ਅਤੇ ਨਵੀਂ ਬਣੀ ਕਲਾਸ ਨੂੰ ਸਬ-ਕਲਾਸ ਨੂੰ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਧਾਰਨਾ (Concept) ਕਿਸੇ ਵੀ ਇੱਕ ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਵਿੱਚ ਦੂਜਿਆਂ ਕਲਾਸਾਂ ਵਿੱਚੋਂ ਇੱਕ ਜਾਂ ਇੱਕ ਤੋਂ ਵੱਧ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ (ਫੀਲਡਜ਼ ਅਤੇ/ਜਾਂ ਮੈਥਡਜ਼) ਨੂੰ ਸ਼ਾਮਿਲ ਕਰਨ ਦੀ ਯੋਗਤਾ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ। ਇਸ ਪ੍ਰਕਿਰਿਆ ਨੂੰ ਅਸੀਂ ਇਸ ਤਰ੍ਹਾਂ ਸਮਝ ਸਕਦੇ ਹਾਂ ਜਿਵੇਂ ਇੱਕ ਬੱਚੇ ਨੂੰ ਉਸਦੇ ਮਾਤਾ-ਪਿਤਾ ਦੇ ਗੁਣ ਵਿਰਾਸਤ ਵਿੱਚ ਮਿਲਦੇ ਹਨ। ਇਨਹੈਰੀਟੈਂਸ ਦਾ ਮੁੱਖ ਉਦੇਸ਼ ਇੱਕ ਮੌਜੂਦਾ ਕੋਡ (ਪ੍ਰੋਗਰਾਮ) ਨੂੰ ਹੋਰ ਮਜ਼ਬੂਤ ਕਰਨਾ ਅਤੇ ਮੁੜ-ਵਰਤੋਂ (reuse) ਕਰਨਾ ਹੈ।ਉਦਾਹਰਨ ਲਈ; ਜੇਕਰ "Person" ਸੁਪਰ ਕਲਾਸ ਬਣਾਈ ਜਾਂਦੀ ਹੈ ਤਾਂ "Student", "Teacher" ਅਤੇ "Staff" ਆਦਿ ਨਾਮ ਦੀਆਂ ਉਪ ਕਲਾਸਾਂ ਬਣਾ ਕੇ "Person" ਨਾਮ ਦੀ ਕਲਾਸ ਦੇ ਸਾਂਝੇ ਗੁਣਾਂ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਇਹਨਾਂ ਸਾਰਿਆਂ ਕਲਾਸਾਂ ਤੇ ਲਾਗੂ ਹੋਣ ਵਾਲੇ ਫੀਲਡਜ਼ ਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਇਸ "Person" ਨਾਮ ਦੀ ਸੁਪਰਕਲਾਸ ਵਿੱਚ ਸਾਂਝੇ ਤੌਰ ਤੇ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

ਇਨਹੈਰੀਟੈਂਸ ਵਿੱਚ ਪੇਰੈਂਟ (Parent) ਕਲਾਸ ਤੋਂ ਪ੍ਰਾਪਤ ਹੋਣ ਵਾਲੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਆਮ ਤੌਰ 'ਤੇ ਵੇਰੀਏਬਲ ਜਾਂ ਮੈਥਡਜ਼ ਆਦਿ ਹੋ ਸਕਦੇ ਹਨ। ਇਸ ਤਰ੍ਹਾਂ ਬਣਾਈ ਗਈ ਨਵੀਂ ਕਲਾਸ ਨੂੰ "ਡਰਾਇਵ ਕਲਾਸ" ਜਾਂ "ਚਾਈਲਡ ਕਲਾਸ" ਕਿਹਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਮੌਜੂਦਾ ਕਲਾਸ ਨੂੰ "ਬੇਸ ਕਲਾਸ" ਜਾਂ "ਪੇਰੈਂਟ ਕਲਾਸ" ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।ਡਰਾਈਵ ਕਲਾਸ ਨੂੰ ਅਜਿਹੀ ਸਥਿਤੀ ਵਿੱਚ ਅਸੀਂ ਬੇਸ ਕਲਾਸ ਤੋਂ ਬਣਾਈ ਗਈ ਕਲਾਸ ਕਹਿ ਸਕਦੇ ਹਾਂ।ਅਸੀਂ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਗ੍ਰਾਫਿਕਲ ਰੂਪ ਵਿੱਚ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਸਮਝ ਸਮਦੇ ਹਾਂ:



ਇਨਹੈਰੀਟੈਂਸ ਨਾਲ ਸਬੰਧਤ ਸ਼ਬਦਾਵਲੀ (Terminologies related to the Inheritance):

- ਕਲਾਸ (Class): ਇਸ ਨੂੰ ਸਾਰੇ ਆਬਜੈਕਟਸ ਦੀਆਂ ਸਾਂਝੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੇ ਸੰਗ੍ਰਹਿ ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।ਇਹ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਬਣਾਏ ਗਏ ਇਕ ਕਿਸਮ ਦੇ ਆਬਜੈਕਟਸ ਦਾ ਬਲੂਪ੍ਰਿੰਟ ਹੁੰਦਾ ਹੈ।

- **ਸਬ-ਕਲਾਸ (Sub Class) :** ਇਸ ਨੂੰ ਡਰਾਇਵਡ ਕਲਾਸ ਜਾਂ ਐਕਸਟੈਂਡਿਡ ਕਲਾਸ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਕਲਾਸ ਕਿਸੇ ਹੋਰ ਕਲਾਸ ਤੋਂ ਵੱਖ-ਵੱਖ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਜਿਵੇਂ ਵੇਰੀਏਬਲ ਜਾਂ ਮੈਥਡਜ਼ ਆਦਿ ਪ੍ਰਾਪਤ ਕਰਦੀ ਹੈ। ਸਬ ਕਲਾਸ ਵਿੱਚ ਇਸਦੀ ਸੁਪਰ ਕਲਾਸ ਤੋਂ ਇਨਹੈਰੇਂਟ ਹੋਏ ਫੀਲਡਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ ਅਤੇ ਨਾਲ ਹੀ ਇਸ ਦੇ ਆਪਣੇ ਖੁਦ ਦੇ ਬਣਾਏ ਮੈਂਬਰ ਵੀ ਸ਼ਾਮਿਲ ਹੋ ਸਕਦੇ ਹਨ।
- **ਸੁਪਰ ਕਲਾਸ (Super Class) :** ਸੁਪਰ ਕਲਾਸ ਨੂੰ ਪੇਰੈਂਟ ਕਲਾਸ ਵੀ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਉਹ ਕਲਾਸ ਹੁੰਦੀ ਹੈ ਜੋ ਉਹ ਸਾਰੀਆਂ ਸਾਂਝੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਦੀ ਹੈ ਜਿੰਨ੍ਹਾਂ ਨੂੰ ਫੀਲਡਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਦੇ ਰੂਪ ਵਿੱਚ ਸਬ ਕਲਾਸਜ਼ ਦੁਆਰਾ ਇਨਹੈਰੇਟ **(inherit)** ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
- **ਮੁੜ ਵਰਤੋਂ ਯੋਗਤਾ (Reusability):** ਜਿਵੇਂ ਕਿ ਇਸਦਾ ਨਾਮ ਹੀ ਦਰਸਾਉਂਦਾ ਹੈ, ਇਹ ਨਵੀਂ ਬਣਾਈ ਗਈ ਕਲਾਸ ਵਿੱਚ ਮੌਜੂਦਾ ਕਲਾਸ ਦੇ ਮੈਂਬਰਜ਼ ਮੁੜ ਵਰਤੋਂ ਕਰਨ ਦੀ ਇੱਕ ਤਕਨੀਕ ਹੈ। ਇਸ ਵਿੱਚ ਮੌਜੂਦਾ ਕਲਾਸ ਦੇ ਫੀਲਡਜ਼ ਜਾਂ ਮੈਥਡਜ਼ ਦੀ ਮੁੜ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਸੰਖੇਪ ਵਿੱਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹ ਕੋਡ ਨੂੰ ਦੁਬਾਰਾ ਵਰਤਣ ਦੀ ਇੱਕ ਤਕਨੀਕ ਹੁੰਦੀ ਹੈ।

ਇਹ ਸਾਰੀਆਂ ਪਰਿਭਾਸ਼ਾਵਾਂ ਜਾਵਾ ਵਿੱਚ ਇਨਹੈਰੀਟੈਂਸ ਦਾ ਮੁੱਖ ਭਾਗ ਹੁੰਦੀਆਂ ਹਨ। ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਇਨਹੈਰੀਟੈਂਸ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

ਸਿੰਟੈਕਸ

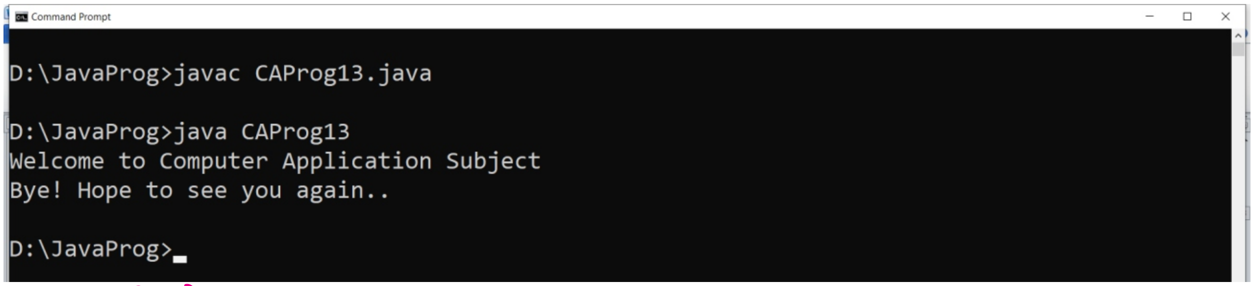
```
class derived_class extends base_class
{
//Fields
//Methods
}
```

ਨੋਟ : ਜਾਵਾ ਵਿਚ extends ਕੀਵਰਡ ਦੀ ਵਰਤੋਂ ਨਾਲ ਸੁਪਰ/ਬੇਸ ਕਲਾਸ ਦੇ ਮੈਂਬਰਾਂ ਨੂੰ ਇਨਹੈਰੇਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਪ੍ਰੋਗਰਾਮ 9.1 : ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦਾ ਤਰੀਕਾ ਦਰਸਾਉਂਦਾ ਹੈ।

```
class SupClass
{
void showStart()
{
System.out.println("Welcome to Computer Application Subject");
}
}
class SubClass extends SupClass
{
void showEnd()
{
System.out.println("Bye! Hope to see you again..");
}
}
class CAProg13
{
        public static void main(String arg[])
        {
        SubClass obj=new SubClass();
        obj.showStart();
        obj.showEnd();
        }
}
```

ਪ੍ਰੋਗਰਾਮ 9.1 (CAProg 13. Java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ



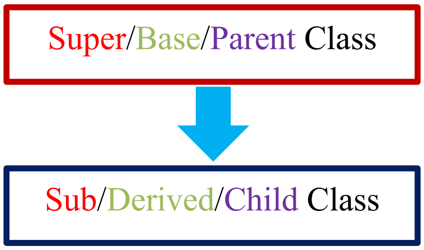
9.1.1 ਇਨਹੈਰੀਟੈਂਸ ਦੀਆਂ ਕਿਸਮਾਂ (TYPES OF INHERITANCE)

ਜਾਵਾ ਵਿਚ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਵਰਤੀ ਜਾ ਸਕਦੀ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਇਨਹੈਰੀਟੈਂਸ ਦੀਆਂ ਕੁਝ ਆਮ ਵਰਤੀਆਂ ਜਾਦੀਆਂ ਕਿਸਮਾਂ ਹੇਠਾਂ ਦਿੱਤੀਆਂ ਗਈਆਂ ਹਨ :

- ਮਲਟੀਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ (Single Inheritance)
- ਮਲਟੀਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ (Multi-level Inheritance)
- ਹਾਇਰਰਾਰਕੀਕਲ ਇਨਹੈਰੀਟੈਂਸ (Hierarchical Inheritance)
- ਹਾਇਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ (Hybrid Inheritance)

• ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ (Single Inheritance)

ਇਹ ਜਾਵਾ ਵਿੱਚ ਸਭ ਤੋਂ ਸਧਾਰਣ ਕਿਸਮ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਹੁੰਦੀ ਹੈ। ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ ਵਿੱਚ ਇੱਕ ਸਬ-ਕਲਾਸ ਕੇਵਲ ਇੱਕ ਸੁਪਰ ਕਲਾਸ ਤੋਂ ਬਣਾਈ ਜਾਂਦੀ ਹੈ। ਇਸ ਵਿੱਚ ਇੱਕ ਸਿੰਗਲ child ਕਲਾਸ parent ਕਲਾਸ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਅਤੇ ਵਿਵਹਾਰ ਨੂੰ ਇਨਹੈਰੀਟੈਂਸ ਦੇ ਰੂਪ ਵਿੱਚ ਪ੍ਰਾਪਤ ਕਰਦੀ ਹੈ। ਕਈ ਵਾਰ ਇਸਨੂੰ ਸਿੰਪਲ (simple inheritance) ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਹੇਠਾਂ ਦਰਸਾਏ ਅਨੁਸਾਰ ਗ੍ਰਾਫਿਕਲੀ ਪੇਸ਼ ਕਰ ਸਕਦੇ ਹਾਂ।



Single Inheritance in Java

ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਉੱਪਰ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ ਦੇਖ ਸਕਦੇ ਹਾਂ, ਪੇਰੈਂਟ (parent) ਕਲਾਸ ਚਾਇਲਡ (child) ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਟ ਹੁੰਦੀ ਹੋਈ ਦਰਸਾਈ ਗਈ ਹੈ। ਇੱਥੇ ਚਾਈਲਡ ਕਲਾਸ ਵਿੱਚ ਪੇਰੈਂਟ ਕਲਾਸ ਦੀਆਂ ਸਾਰੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਸ਼ਾਮਲ ਹੋਣਗੀਆ। ਪਰੰਤੂ ਚਾਇਲਡ ਕਲਾਸ ਦਾ ਕੋਈ ਵੀ ਵਿਵਹਾਰ ਜਾਂ ਵਿਸ਼ੇਸ਼ਤਾ ਪੇਰੈਂਟ ਕਲਾ ਵਿੱਚ ਇਨਹੈਰੇਟ ਨਹੀਂ ਹੋਵੇਗਾ। ਅਸੀਂ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੇ ਤੌਰ ਤੇ ਜਾਵਾ ਵਿੱਚ ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

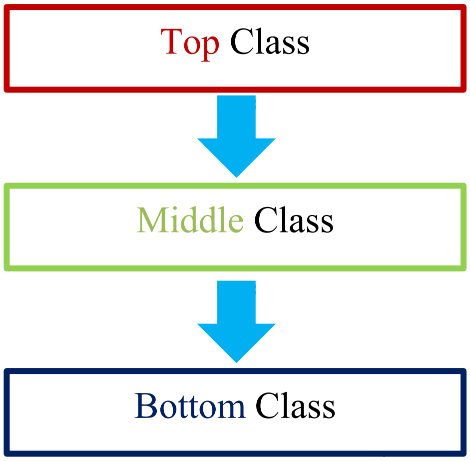
ਪ੍ਰੋਗਰਾਮ 9.2 ਹੇਠਾਂ ਦਰਸਾਏ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਲਾਗੂ ਕੀਤਾ ਗਿਆ ਹੈ:

```
class Person
{
int age;
String sname;
}
```

```
class Student extends Person
{
int sclass,rollno;
void setData(int age_Input,String sname_Input,int sclass_Input,int rollno_Input)
{
age=age_Input;
sname=sname_Input;
sclass=sclass_Input;
rollno=rollno_Input;
}
void showRecord()
{
System.out.println("Age:"+age);
System.out.println("Student Name: "+sname);
System.out.println("Student Class:"+sclass);
System.out.println("Roll No:"+rollno);
}
}
class CAProg14
{
	public static void main(String arg[])
	{
	Student obj=new Student();
	obj.setData(16,"Navleen",7,1001);
	obj.showRecord();
	}
}
```

- **ਮਲਟੀ-ਲੇਵਲ (ਬਹੁ-ਪੱਧਰੀ) ਇਨਹੈਰੀਟੈਂਸ (Multi-level Inheritance) :**

ਇਸ ਕਿਸਮ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਲੜੀ ਵਜੋਂ ਦੇਖਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਮਲਟੀ ਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ ਵਿੱਚ ਇੱਕ ਕਲਾਸ ਨੂੰ ਦੂਸਰੀ ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਕਿ ਅੱਗੇ ਇੱਕ ਹੋਰ ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਹੁੰਦੀ ਹੈ। ਇਸੇ ਪ੍ਰਕਾਰ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਮਲਟੀ ਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਸਰਲ ਸ਼ਬਦਾਂ ਵਿੱਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਉਹ ਇਨਹੈਰੀਟੈਂਸ ਜਿਸ ਵਿੱਚ ਇੱਕ ਕਲਾਸ ਵਿੱਚ ਇੱਕ ਤੋਂ ਵੱਧ ਪੇਰੇਂਟ ਕਲਾਸਾਂ ਪਰੰਤੂ ਵੱਖ-ਵੱਖ ਪੱਧਰਾਂ ਤੇ ਹੋਣ, ਨੂੰ ਮਲਟੀ ਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਸ ਕਿਸਮ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਵਿੱਚ ਇੱਕ ਸਬ-ਕਲਾਸ ਵਿੱਚ ਇੱਕੋ ਸਮੇਂ ਇੱਕ ਤੋਂ ਵੱਧ ਸੁਪਰ ਕਲਾਸਾਂ ਸ਼ਾਮਿਲ ਨਹੀਂ ਹੋ ਸਕਦੀਆਂ। ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਮਲਟੀ ਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਗ੍ਰਾਫਿਕਲ ਪੇਸ਼ਕਾਰੀ ਅੱਗੇ ਦਰਸਾਏ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ।



ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਉੱਪਰ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ ਦੇਖ ਸਕਦੇ ਹਾਂ, Top ਕਲਾਸ ਨੂੰ Middle ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਕੀਤਾ ਗਿਆ ਹੈ ਜੋ ਕਿ ਅੱਗੇ ਇੱਕ ਹੋਰ Bottom ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਹੁੰਦੀ ਹੈ। ਇੱਥੇ Bottom ਕਲਾਸ ਵਿਚ Top ਅਤੇ Middle ਕਲਾਸ ਦੀਆਂ ਸਾਰੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਇਨਹੈਰੇਟ ਹੋਣਗੀਆਂ, ਪਰੰਤੂ Middle ਕਲਾਸ ਦੀ ਕੋਈ ਵੀ ਵਿਸ਼ੇਸ਼ਤਾ Top ਕਲਾਸ ਵਿਚ ਇਨਹੈਰੇਟ ਨਹੀਂ ਹੋਵੇਗੀ ਅਤੇ ਨਾਂ ਹੀ Bottom ਕਲਾਸ ਦੀ ਕੋਈ ਵਿਸ਼ੇਸ਼ਤਾ Top ਜਾਂ Middle ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਟ ਹੋਵੇ। ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੇ ਰੂਪ ਵਿੱਚ ਹੇਠਾਂ ਲਿਖੇ ਅਨੁਸਾਰ ਮਲਟੀ ਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ ਤਰੀਕੇ ਦੀ ਉਦਾਹਰਣ ਦੇਖ ਸਕਦੇ ਹਾਂ।

ਪ੍ਰੋਗਰਾਮ 9.3 ਹੇਠਲਾ ਪ੍ਰੋਗਰਾਮ JAVA ਵਿੱਚ ਮਲਟੀ-ਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ ਦੇ ਲਾਗੂ ਕਰਨ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
class Top
{
void show()
{
System.out.println("This is a method of TOP Class");
}
}
class Middle extends Top
{
void display()
{
System.out.println("This is a method of Middle Class");
}
}
class Bottom extends Middle
{
void print()
{
System.out.println("This is a method of Bottom Class");
}
}
```

```
class CAProg16
{
        public static void main(String arg[])
        {
        Bottom obj=new Bottom();
        obj.show();
        obj.display();
        obj.print();
        }
}
```

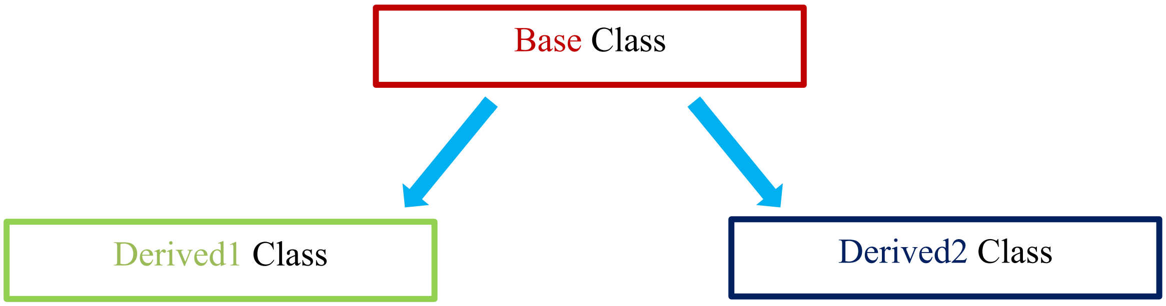
ਪ੍ਰੋਗਰਾਮ 9.3 (CAProg16.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
D:\JavaProg>javac CAProg16.java

D:\JavaProg>java CAProg16
This is a method of TOP Class
This is a method of Middle Class
This is a method of Bottom Class

D:\JavaProg>_
```

ਇੱਥੇ ਅਸੀਂ ਦੇਖ ਸਕਦੇ ਹਾਂ ਕਿ Top ਕਲਾਸ ਦੇ ਮੈਥਡ [show ()] ਅਤੇ Middle ਕਲਾਸ ਦੇ ਮੈਥਡ [display ()] Bottom ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਹੋਏ ਹਨ।ਇਹ ਸਾਰੇ ਮੈਥਡ [show(), display() and print()] ਅਸੀਂ Bottom ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਦੇ ਨਾਲ ਕਾਲ (Call) ਕਰ ਸਕਦੇ ਹਾਂ।

• ਹਾਇਰਾਰਕੀਕਲ ਇਨਹੈਰੀਟੈਂਸ (Hierarchical Inheritance or Tree Inheritance) :

ਜਦੋਂ ਦੋ ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਕਲਾਸਾਂ ਇੱਕ ਸਾਂਝੀ ਸੁਪਰ ਕਲਾਸ ਨੂੰ ਇਨਹੈਰੇਂਟ ਕਰਦੀਆਂ ਹਨ ਤਾਂ ਇਸ ਨੂੰ ਹਾਇਰਾਰਕੀਕਲ ਇਨਹੈਰੀਟੈਂਸ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਕਿਸਮ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਉਸ ਸਥਿਤੀ ਵਿੱਚ ਲਾਭਦਾਇਕ ਹੁੰਦੀ ਹੈ ਜਦੋਂ ਅਸੀਂ ਇੱਕ ਸਾਂਝੀ ਬੇਸ ਕਲਾਸ ਦੇ ਮੈਥਡ ਜਾਂ ਫੀਲਡਜ਼ ਨੂੰ ਮਲਟੀਪਲ ਸਬ ਕਲਾਸਾਂ ਵਿੱਚ ਪ੍ਰਯੋਗ ਕਰਨਾ ਹੋਵੇ, ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਵਿੱਚ ਦੇਖ ਸਕਦੇ ਹਾਂ, Deroved1 ਅਤੇ Derived2 ਕਲਾਸਾਂ Base ਕਲਾਸ ਨੂੰ ਇਨਹੈਰੇਟ ਕਰ ਰਹੀਆਂ ਹਨ।ਇਸ ਪ੍ਰਕਾਰ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਬਣਤਰ ਹੀ ਹਾਇਰਾਰਕੀਕਲ ਇਨਹੈਰੀਟੈਂਸ ਅਖਵਾਉਂਦੀ ਹੈ।



Hierarchical Inheritance in Java

ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਇੱਕ ਪ੍ਰੋਗਰਾਮ ਦੇ ਤੌਰ ਤੇ ਅੱਗੇ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹਾਇਰਾਰਕੀਕਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਉਦਾਹਰਣ ਦਰਸਾ ਸਕਦੇ ਹਾਂ।

```
class Base
{
void show()
{
System.out.println("Base Class");
}
}
class Derived1 extends Base
{
void display()
{
System.out.println("Derived 1 Class");
}
}
class Derived2 extends Base
{
void print()
{
System.out.println("Derived 2 Class");
}
}
class CAProg17
{
        public static void main(String arg[])
        {
        Derived1 obj1=new Derived1();
        obj1.show();
        obj1.display();
        Derived2 obj2=new Derived2();
        obj2.show();
        obj2.print();
        }
}
```

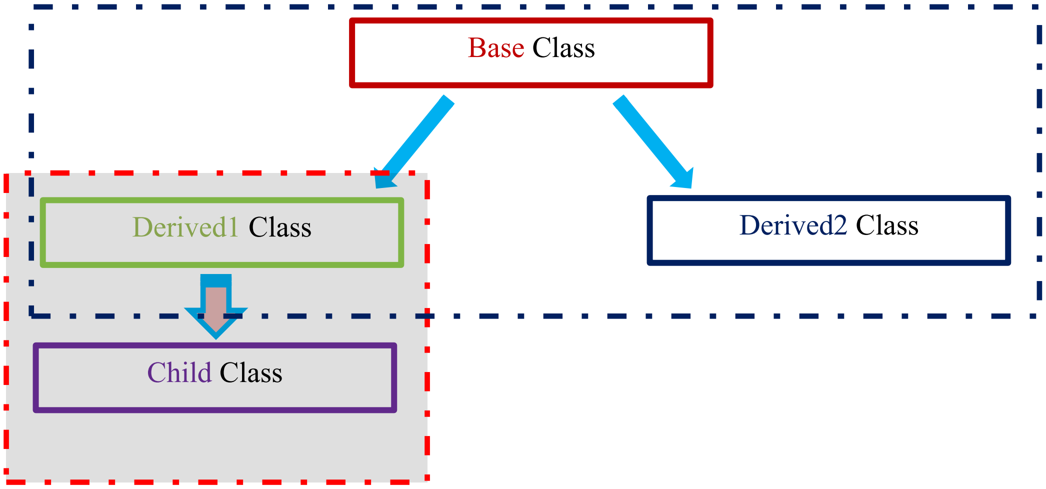
ਪ੍ਰੋਗਰਾਮ 9.4 ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਹਾਇਰਾਰਕੀਕਲ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਦਰਸਾ ਰਿਹਾ ਹੈ :

ਪ੍ਰੋਗਰਾਮ 9.4 (CAProg17.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

ਇੱਥੇ ਅਸੀਂ ਦੇਖ ਸਕਦੇ ਹਾਂ ਕਿ Base ਕਲਾਸ ਦਾ ਮੈਥਡ [show()] ਦੋਵੇਂ ਕਲਾਸਾਂ Derived1 ਅਤੇ Derived2 ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਹੋਇਆ ਹੈ। Base ਕਲਾਸ ਦੇ ਮੈਥਡ ਨੂੰ ਅਸੀਂ Derived1 ਕਲਾਸ ਜਾਂ Derived2 ਕਲਾਸ ਦੇ ਆਬਜੈਕਟ ਨਾਲ ਕਾਲ ਕਰ ਸਕਦੇ ਹਾਂ।

• ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ (Hybrid Inheritance) :

ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ ਤੋਂ ਭਾਵ ਹੈ ਕਿ ਇੱਕ ਇਨਹੈਰੀਟੈਂਸ ਅੰਦਰ ਇੱਕ ਤੋਂ ਵੱਧ ਕੋਈ ਵੀ ਹੋਰ ਇਨਹੈਰੀਟੈਂਸ ਦੇ ਰੂਪਾਂ ਨੂੰ ਸ਼ਾਮਲ ਕਰਨਾ। ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ ਜਾਵਾ ਵਿੱਚ ਕਿਸੇ ਵੀ ਦੋ ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਕਿਸਮਾਂ ਦੇ ਇਨਹੈਰੀਟੈਂਸ ਦਾ ਸੁਮੇਲ ਹੋ ਸਕਦਾ ਹੈ। ਅਸੀਂ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਦੇ ਰੂਪ ਵਿੱਚ ਸਧਾਰਨ ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਇੱਕ ਉਦਾਹਰਣ ਦੇਖ ਸਕਦੇ ਹਾਂ।



Hierarchical Inheritance in Java

ਇਸ ਚਿੱਤਰ ਵਿਚ ਅਸੀਂ ਹਾਇਰਾਰਕੀਕਲ ਅਤੇ ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਵਰਤੋਂ ਨਾਲ ਹਾਇਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਦਰਸਾਇਆ ਹੈ। ਜਾਵਾ ਵਿੱਚ ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ ਕਿਸਮ ਦੀ ਉਦਾਹਰਣ ਇਸ ਪ੍ਰਕਾਰ ਹੋ ਸਕਦੀ ਹੈ।

ਪ੍ਰੋਗਰਾਮ 9.5 : ਨਿਮਨਲਿਖਤ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੇ ਤਰੀਕੇ ਨੂੰ ਪੇਸ਼ ਕਰਦਾ ਹੈ।

```
class Base
{
void show()
{
System.out.println("Base Class");
}
}
class Derived1 extends Base
{
void display()
{
System.out.println("Derived 1 Class");
}
}
class Derived2 extends Base
{
void print()
{
System.out.println("Derived 2 Class");
}
}
class Child extends Derived1
{
void send()
{
System.out.println("This is Child Class here");
}
}
class CAProg18
{
        public static void main(String arg[])
        {
        Child obj1=new Child();
        obj1.show();
        obj1.display();
        obj1.send();
        Derived2 obj2=new Derived2();
        obj2.show();
        obj2.print();
        }   }
```

ਪ੍ਰੋਗਰਾਮ 9.5 (CAProg18.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
D:\JavaProg>javac CAProg18.java

D:\JavaProg>java CAProg18
Base Class
Derived 1 Class
This is Child Class here
Base Class
Derived 2 Class

D:\JavaProg>_
```

• ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ (Multiple Inheritance) :

ਜਾਵਾ ਕਲਾਸਾਂ ਉੱਪਰ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਇਜਾਜ਼ਤ ਨਹੀਂ ਦਿੰਦੀ ਹੈ। ਅਸੀਂ ਇੰਟਰਫੇਸ ਦੀ ਧਾਰਨਾ (Concept) ਨੂੰ ਸਮਝਣ ਤੋਂ ਬਾਅਦ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਵਿਆਖਿਆ ਕਰਾਂਗੇ।

9.2 ਜਾਵਾ ਵਿੱਚ ਇੰਟਰਫੇਸ (INTERFACES IN JAVA) :

ਜਾਵਾ ਵਿੱਚ ਇੰਟਰਫੇਸ ਐਬਸਟਰੈਕਸ਼ਨ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੀ ਇੱਕ ਵਿਧੀ ਹੁੰਦੀ ਹੈ। ਜਾਵਾ ਇੰਟਰਫੇਸ ਵਿੱਚ ਕੇਵਲ ਐਬਸਟਰੈਕਟ ਮੈਥਡ ਹੀ ਸ਼ਾਮਿਲ ਹੋ ਸਕਦੇ ਅਤੇ ਇਹਨਾਂ ਮੈਥਡਜ਼ ਦੀ ਬਾਡੀ ਨਹੀਂ ਹੁੰਦੀ। ਇੱਕ ਕਲਾਸ ਇੰਟਰਫੇਸ ਨੂੰ ਇੰਪਲੀਮੈਂਟ (implement) ਕਰਦੀ ਹੈ ਅਤੇ ਇੰਟਰਫੇਸ ਦੇ ਐਬਸਟਰੈਕਟ ਮੈਥਡਜ਼ ਨੂੰ ਇੰਨਹੈਰੇਂਟ ਕਰਦੀ ਹੈ। ਐਬਸਟਰੈਕਟ ਮੈਥਡਜ਼ ਦੇ ਨਾਲ-ਨਾਲ ਇੱਕ ਇੰਟਰਫੇਸ ਵਿੱਚ ਕੋਂਸਟੈਂਟ (constants), ਡਿਫਾਲਟ (default) ਮੈਥਡਜ਼, ਸਟੈਟਿਕ (static) ਮੈਥਡਜ਼ ਅਤੇ ਨੈਸਟਿਡ (nested) ਕਿਸਮਾਂ ਦੇ ਮੈਂਬਰ ਵੀ ਸ਼ਾਮਿਲ ਹੋ ਸਕਦੇ ਹਨ। ਜਾਵਾ ਦੇ ਨਵੇਂ ਵਰਜ਼ਨਾਂ (Java8 ਅਤੇ ਇਸ ਤੋਂ ਬਾਅਦ) ਵਿਚ ਸਿਰਫ ਡਿਫਾਲਟ (default) ਮੈਥਡਜ਼ ਅਤੇ ਸਟੈਟਿਕ (static) ਮੈਥਡਜ਼ ਦੀਆਂ ਬਾਡੀਜ਼ ਹੀ ਇੰਟਰਫੇਸ ਦੇ ਅੰਦਰ ਮੌਜੂਦ ਹੋ ਸਕਦੀਆਂ ਹਨ।

ਇੰਟਰਫੇਸ ਬਣਾਉਣ ਦਾ ਤਰੀਕਾ ਕਲਾਸ ਬਣਾਉਣ ਦੇ ਸਮਾਨ ਹੀ ਹੁੰਦਾ ਹੈ। ਕਲਾਸ ਕਿਸੇ ਆਬਜੈਕਟ ਦੇ ਗੁਣਾਂ (properties) ਅਤੇ ਵਿਹਾਰਾਂ (Behavirous) ਦਾ ਵਰਣਨ ਕਰਦੀ ਹੈ ਪਰ ਇੰਟਰਫੇਸ ਵਿੱਚ ਵਿਹਾਰ (behaviour) ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ ਜੋ ਇੱਕ ਕਲਾਸ ਦੁਆਰਾ implement ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਜਦੋਂ ਇੱਕ ਕਲਾਸ ਕਿਸੇ ਵੀ ਇੰਟਰਫੇਸ ਨੂੰ ਇੰਪਲੀਮੈਂਟ (implement) ਕਰਦੀ ਹੈ, ਤਾਂ ਇੰਟਰਫੇਸ ਦੇ ਸਾਰੇ ਮੈਥਡਜ਼ ਨੂੰ ਕਲਾਸ ਵਿੱਚ ਪਰਿਭਾਸ਼ਿਤ ਕਰਨਾ ਜ਼ਰੂਰੀ ਹੁੰਦਾ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇੰਟਰਫੇਸ ਵਿੱਚ ਐਬਸਟ੍ਰੈਕਟ ਵੇਰੀਏਬਲ ਅਤੇ ਐਬਸਟ੍ਰੈਕਟ ਮੈਥਡਜ਼ ਹੋ ਸਕਦੇ ਹਨ, ਜਿੰਨ੍ਹਾਂ ਦੀ ਬਾਡੀ (body) ਪਰਿਭਾਸ਼ਿਤ ਨਹੀਂ ਹੁੰਦੀ।

ਨੋਟ : ਐਬਸਟਰੈਕਟ (abstract) ਮੈਥਡ ਦੀ ਬਾਡੀ ਨਹੀਂ ਹੁੰਦੀ,ਇਹਨਾਂ ਨੂੰ ਸਿਰਫ਼ ਡਿਕਲੇਅਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਇਹਨਾਂ ਦੀ ਪਰਿਭਾਸ਼ਾ ਨਹੀਂ ਦਿੱਤੀ ਜਾਂਦੀ।

ਕਲਾਸ ਅਤੇ ਇੰਟਰਫੇਸ ਵਿਚਕਾਰ ਸਮਾਨਤਾਵਾਂ (Similarities between class and interface) :

- ❖ ਕਲਾਸ ਅਤੇ ਇੰਟਰਫੇਸ ਦੋਨਾਂ ਵਿੱਚ ਕਈ ਮੈਥਡ ਸ਼ਾਮਲ ਹੋ ਸਕਦੇ ਹਨ।
- ❖ ਕਲਾਸ ਅਤੇ ਇੰਟਰਫੇਸ ਦੋਵੇਂ java ਐਕਸਟੈਂਸ਼ਨ ਦੇ ਨਾਲ ਇੱਕ ਫਾਈਲ ਦੇ ਰੂਪ ਵਿੱਚ ਬਣਾਏ ਜਾ ਸਕਦੇ ਹਨ। public ਕਲਾਸ ਅਤੇ ਇੰਟਰਫੇਸ ਦਾ ਨਾਮ ਫਾਈਲ ਦੇ ਨਾਮ ਨਾਲ ਮੇਲ ਖਾਣਾ ਚਾਹੀਦਾ ਹੈ।
- ❖ ਕਲਾਸ ਅਤੇ ਇੰਟਰਫੇਸ ਦੋਵਾਂ ਦਾ ਬਾਈਟ ਕੋਡ ਕੰਪਾਇਲੇਸ਼ਨ ਤੋਂ ਬਾਅਦ .class ਫਾਈਲ ਵਿੱਚ ਸਟੋਰ ਹੁੰਦਾ ਹੈ।
- ❖ ਕਲਾਸ ਅਤੇ ਇੰਟਰਫੇਸ ਦੋਵੇਂ ਪੈਕੇਜ਼ਾਂ ਵਿੱਚ ਦਿਖਾਈ ਦਿੰਦੇ ਹਨ ਅਤੇ ਇਹਨਾਂ ਦੀ ਬਾਈਟਕੋਡ ਫਾਈਲ ਇੱਕ ਡਾਇਰੈਕਟਰੀ ਢਾਂਚੇ ਵਿੱਚ ਸ਼ਾਮਿਲ ਹੁੰਦੀ ਹੈ ਜੋ ਕਿ ਪੈਕੇਜ ਨਾਮ ਨਾਲ ਮੇਲ ਖਾਂਦੀ ਹੈ।

ਕਲਾਸ ਅਤੇ ਇੰਟਰਫੇਸ ਵਿਚਕਾਰ ਅੰਤਰ (Difference between class and interface) :

- ❖ ਅਸੀਂ ਇੰਟਰਫੇਸ ਦਾ ਆਬਜੈਕਟ ਨਹੀਂ ਬਣਾ ਸਕਦੇ।
- ❖ ਇੰਟਰਫੇਸ ਵਿੱਚ ਕੋਈ ਕੰਸਟਰਕਟਰ ਨਹੀਂ ਹੁੰਦਾ।

- ਇੰਟਰਫੇਸ ਵਿੱਚ ਸਾਰੇ ਮੈਥਡ ਐਬਸਟਰੈਕਟ ਹੁੰਦੇ ਹਨ (ਕੇਵਲ ਡਿਫਾਲਟ ਜਾਂ ਸਟੈਟਿਕ ਮੈਥਡਜ਼ ਦੀ ਬਾਡੀ ਹੋ ਸਕਦੀ ਹੈ)।
- ਇੰਟਰਫੇਸ ਵਿੱਚ ਸਿਰਫ਼ ਸਟੈਟਿਕ (static) ਅਤੇ ਫਾਇਨਲ (final) ਫੀਲਡਜ਼ ਹੀ ਹੋ ਸਕਦੇ ਹਨ।
- ਇੰਟਰਫੇਸ ਨੂੰ ਕਲਾਸ ਦੁਆਰਾ ਇੰਪਲੀਮੈਂਟ (implement) ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
- ਇੱਕ ਇੰਟਰਫੇਸ ਦੂਜੇ ਇੰਟਰਫੇਸ ਨੂੰ ਐਕਸਟੈਂਡ (extend) ਕਰ ਸਕਦਾ ਹੈ।

9.2.1 ਇੰਟਰਫੇਸ ਨੂੰ ਡਿਕਲੇਅਰ ਕਰਨਾ (DECLARING INTERFACES) :

Interface ਕੀਅ-ਬੋਰਡ ਦੀ ਵਰਤੋਂ ਇੰਟਰਫੇਸ ਘੋਸ਼ਿਤ (declare) ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇੰਟਰਫੇਸ ਘੋਸ਼ਣਾ ਕਰਨ ਦਾ ਸਿੰਟੈਕਸ ਅਤੇ ਉਦਾਹਰਣ ਹੇਠ ਲਿਖੇ ਅਨੁਸਾਰ ਹੋ ਸਕਦਾ ਹੈ।

ਸਿੰਟੈਕਸ :

```
interface InterfaceName
{
        // declaration of final & static fields
        // declaration of abstract methods
}
```

ਅਸੀਂ ਪ੍ਰੋਗਰਾਮ ਵਿਚ ਲੋੜ ਅਨੁਸਾਰ ਉਪਰੋਕਤ ਸਿੰਟੈਕਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇੰਟਰਫੇਸ ਪਰਿਭਾਸ਼ਿਤ ਕਰ ਸਕਦੇ ਹਾਂ। ਇੰਟਰਫੇਸ ਦੀ ਵਰਤੋਂ ਨੂੰ ਵਿਸਥਾਰ ਵਿੱਚ ਸਮਝਣ ਲਈ ਇੱਕ ਉਦਾਹਰਨ ਹੇਠਾਂ ਦਿੱਤੀ ਗਈ ਹੈ।

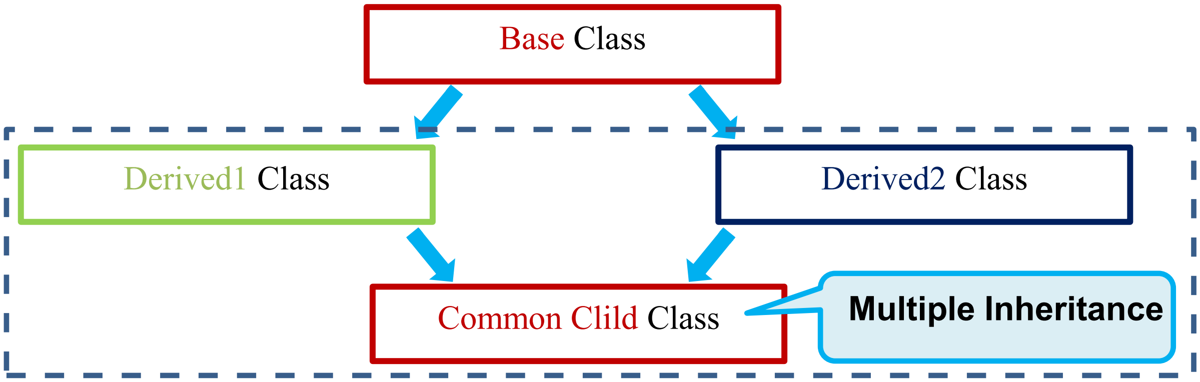
ਪ੍ਰੋਗਰਾਮ 9.6 ਹੇਠਲਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਇੰਟਰਫੇਸ ਨੂੰ ਵਰਤਣ ਦਾ ਤਰੀਕਾ ਬਿਆਨ ਕਰਦਾ ਹੈ।

```
interface Student
{
final int MinAge=18;  // final field
public void show();   // abstract method
}
class CAProg19 implements Student
{
        public void show()
        {
        System.out.println("Welcome to Student Mangement");
        System.out.println("Minimum age of Student is:"+MinAge);
        }
        public static void main(String arg[])
        {
        CAProg19 obj=new CAProg19( );
        obj.show();
        }

 }
```

ਪ੍ਰੋਗਰਾਮ 9.6 (CAProg19.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

ਜਾਵਾ ਵਿੱਚ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ (Multiple Inheritance in Java) :

- ਅਸਪਸ਼ਟਤਾ (ambiguity) ਕਾਰਨ ਜਾਵਾ ਵਿਚ ਕਲਾਸਾਂ ਨਾਲ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਇਜ਼ਾਜਤ ਨਹੀਂ ਹੁੰਦੀ। ਉਦਾਹਰਨ ਲਈ, ਆਉ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਦੇ ਹੇਠਾਂ ਦਿੱਤੇ ਚਿੱਤਰ ਤੇ ਵਿਚਾਰ ਕਰਦੇ ਹਾਂ :



ਜਾਵਾ ਵਿੱਚ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸਲੇ ਆਉਟ।

ਉਪਰੋਕਤ ਚਿੱਤਰ OOPs ਐਪਲੀਕੇਸ਼ਨ ਵਿੱਚ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। ਜੇਕਰ ਇਹ ਲੇਆਉਟ ਜਾਵਾ ਕਲਾਸਾਂ ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਐਰਰ (error) ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ ਕਿਉਂਕਿ ਕੰਪਾਈਲਰ ਇਹ ਫੈਸਲਾ ਨਹੀਂ ਕਰ ਸਕਦਾ ਕਿ Base ਕਲਾਸ ਦਾ ਕਿਹੜਾ ਮੈਥਡ Common Child ਕਲਾਸ ਵਿੱਚ ਵਰਤਿਆ ਜਾਣਾ ਹੈ। ਅਜਿਹਾ ਇਸ ਲਈ ਹੈ ਕਿ Base ਕਲਾਸ ਦੇ ਮੈਥਡ Derived1 ਅਤੇ Derived2 ਨਾਮ ਦੀਆਂ ਦੋ ਕਲਾਸਾਂ ਰਾਹੀਂ Common Child ਕਲਾਸ ਵਿੱਚ ਪ੍ਰਾਪਤ ਹੋ ਰਹੇ ਹਨ। ਇਸ ਕਰਕੇ ਜਾਵਾ ਕਲਾਸ ਪੱਧਰ ਤੇ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਦਾ ਸਮਰਥਨ ਨਹੀਂ ਕਰਦੀ, ਪਰੰਤੂ ਇੰਟਰਫੇਸ ਨਾਲ ਇਸ ਕਿਸਮ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

ਪ੍ਰੋਗਰਾਮ 9.7 ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
interface Base
{
public void show();
}
interface Derived1 extends Base {
public void display();
}
interface Derived2 extends Base {
```

```
public void print();
}
class CommonChild implements Derived1,Derived2  // Multiple Inheritance
{
        public void show()
        {
        System.out.println("Base Class");
        }
        public void display()
        {
        System.out.println("Derived 1 Class");
        }
        public void print()
        {
        System.out.println("Derived 2 Class");
        }
}
class CAProg20
{
        public static void main(String arg[])
        {
        CommonChild obj=new CommonChild();
        obj.show();
        obj.display();
        obj.print();
        }
}
```

ਪ੍ਰੋਗਰਾਮ 9.7 (CAProg20.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟਕ

ਇੱਥੇ ਅਸੀਂ ਦੇਖ ਸਕਦੇ ਹਾਂ ਕਿ ਅਸੀਂ ਇੱਕ Base ਇੰਟਰਫੇਸ ਨੂੰ ਵੱਖ-ਵੱਖ Derived1 ਅਤੇ Derived2 ਇੰਟਰਫੇਸ ਵਿੱਚ ਐਕਸਟੈਂਡ ਕੀਤਾ ਹੈ ਜੋ ਅੱਗੇ ਇੱਕ Common Child ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਹੋਏ ਹਨ। ਸਿਰਫ਼ ਇੰਟਰਫੇਸ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਹੀ ਜਾਵਾ ਵਿੱਚ ਇਸ ਤਰ੍ਹਾਂ ਦੀ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਸੰਭਵ ਹੋ ਸਕਦੀ ਹੈ।

9.3 ਜਾਵਾ ਵਿੱਚ super ਕੀਅ-ਵਰਡ (super KEYWORD IN JAVA):

super ਕੀਅ-ਵਰਡ ਸਾਨੂੰ ਸੁਪਰ ਕਲਾਸ (superclass) ਦੇ ਮੈਂਬਰਾਂ ਤੱਕ ਪਹੁੰਚ ਕਰਨ ਦੀ ਇਜ਼ਾਜਤ ਦਿੰਦਾ ਹੈ। super ਕੀਅ-ਵਰਡ ਦੀ ਵਰਤੋਂ ਸੁਪਰ ਕਲਾਸਾਂ ਅਤੇ ਸਬ ਕਲਾਸਾਂ ਵਿਚਕਾਰ ਮੈਂਬਰਾਂ ਨੂੰ ਸਪਸ਼ਟ ਤੌਰ ਤੇ ਪਹਿਚਾਨਣ ਲਈ ਉਸ ਸਮੇਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਦੋਂ ਉਹਨਾਂ ਵਿੱਚ ਇੱਕੋ ਨਾਮ ਦੇ ਮੈਥਡਜ਼ ਵਰਤੇ ਗਏ ਹੁੰਦੇ ਹਨ।ਅਸੀਂ super ਕੀਅ-ਵਰਡ ਨੂੰ ਦੋ ਵੱਖ-ਵੱਖ ਰੂਪਾਂ ਵਿੱਚ ਵਰਤ ਸਕਦੇ ਹਾਂ।

1. **super ਕੀਵਰਡ ਦੇ ਤੌਰ ਤੇ :** ਇਸ ਤਰੀਕੇ ਨਾਲ super ਕੀਅ-ਵਰਡ ਉਸ ਸਮੇਂ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ ਜਦੋਂ ਇੱਕ ਇਨਹੈਰੇਟ ਕੀਤੀ ਕਲਾਸ ਅਤੇ ਸਬ-ਕਲਾਸ ਵਿੱਚ ਇੱਕੋ ਜਿਹੇ ਡੇਟਾ ਮੈਂਬਰ ਸ਼ਾਮਿਲ ਹੁੰਦੇ ਹਨ। ਇਸ ਸਥਿਤੀ ਵਿੱਚ JVM ਲਈ ਇਹ ਸਪਸ਼ਟ ਨਹੀਂ ਹੁੰਦਾ ਕਿ ਅਸੀਂ ਕਿਹੜਾ ਮੈਂਬਰ ਵਰਤਣਾ ਚਾਹੁੰਦੇ ਹਾਂ।ਅਜਿਹੀ ਸਥਿਤੀ ਵਿੱਚ super ਕੀਅ-ਵਰਡ ਦੀ ਵਰਤੋਂ ਇਹ ਨਿਰਧਾਰਿਤ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਕਿ ਕਿਹੜੇ ਮੈਂਬਰ ਨੂੰ ਐਕਸੈੱਸ ਕੀਤਾ ਜਾਣਾ ਹੈ। super ਕੀਅ-ਵਰਡ ਬਾਰੇ ਕੁਝ ਮੁੱਖ ਗੱਲਾਂ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹਨ :
   - ❖ ਸਬ ਕਲਾਸ ਵਿੱਚ ਓਵਰਰਾਈਡ ਕੀਤੇ ਗਏ ਸੁਪਰ ਕਲਾਸ ਦੇ ਮੈਥਡ ਨੂੰ ਕਾਲ ਕਰਨ ਲਈ ਇਸ ਕੀਅ-ਵਰਡ ਦੀ ਵਰਤੋਂ ਹੁੰਦੀ ਹੈ।
   - ❖ ਜੇਕਰ ਸੁਪਰ ਕਲਾਸ ਅਤੇ ਸਬ ਕਲਾਸ ਦੋਵਾਂ ਵਿੱਚ ਇੱਕੋ ਨਾਮ ਦੇ ਫੀਲਡਜ਼ ਸ਼ਾਮਿਲ ਹੁੰਦੇ ਹਨ ਤਾਂ ਸੁਪਰ ਕਲਾਸ ਦੇ ਫੀਲਡਜ਼ ਨੂੰ ਐਕਸੈੱਸ ਕਰਨ ਲਈ ਵੀ ਇਸ ਕੀਅ-ਵਰਡ ਦੀ ਵਰਤੋਂ ਹੁੰਦੀ ਹੈ।

   ਅਸੀਂ ਹੇਠਾਂ ਦਿੱਤੀ ਉਦਾਹਰਣ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਸੁਪਰ ਕੀਵਰਡ ਦੀ ਧਾਰਨਾ (concept) ਨੂੰ ਸਮਝ ਸਕਦੇ ਹਾਂ।

ਪ੍ਰੋਗਰਾਮ 9.8 ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਕੀਵਰਡ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
class Base
{
void show()
{
System.out.println("Base class show Method");
}
}
class CAProg21 extends Base
{
        void show()
        {
        super.show(); // Accessing Base class method using super
        System.out.println("Derived class show Method");
        }
        public static void main(String arg[])
        {
        CAProg21 obj=new CAProg21();
        obj.show();
        }
}
```

ਪ੍ਰੋਗਰਾਮ 9.8 (CAProg21.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg21.java

D:\JavaProg>java CAProg21
Base class show Method
Derived class show Method

D:\JavaProg>_
```

2. ਸੁਪਰ ਮੈਥਡ (super method) : super ਕੀਅ-ਵਰਡ ਨੂੰ ਮੈਥਡ ਦੇ ਤੌਰ ਤੇ ਪੇਰੈਂਟ ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਨੂੰ ਕਾਲ (call) ਕਰਨ ਲਈ ਵੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਅਸੀਂ super() ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਪੈਰਾਮੀਟਰਾਈਜ਼ਡ ਅਤੇ ਗੈਰ ਪੈਰਾਮੀਟਰਾਈਜ਼ਡ ਕੰਸਟਰਕਟਰ ਨੂੰ ਕਾਲ (call) ਕਰ ਸਕਦੇ ਹਾਂ। ਇਹ ਮੈਥਡ ਬਹੁਤ ਮਹੱਤਵਪਰਨ ਭੂਮਿਕਾ ਨਿਭਾਉਂਦਾ ਹੈ ਕਿਉਂਕਿ ਅਸੀ ਪਹਿਲਾਂ ਹੀ ਪੜ੍ਹ ਚੁੱਕੇ ਹਾਂ ਕਿ ਜੇਕਰ ਕਿਸੇ ਕਲਾਸ ਵਿੱਚ ਪੈਰਾਮੀਟਰਜ਼ ਦੇ ਨਾਲ ਇੱਕ ਕੰਸਟਰਕਟਰ ਘੋਸ਼ਿਤ ਕਰ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਸਦਾ ਡਿਫੌਲਟ ਕੰਸਟਰਕਟਰ ਆਟੋਮੈਟਿਕਲੀ ਨਹੀਂ ਬਣਦਾ। ਇਸ ਲਈ ਜਦੋਂ ਅਸੀਂ ਇਨਹੈਰੀਟੈਂਸ ਵਿੱਚ ਕੰਸਟਰਕਟਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹਾਂ ਅਤੇ ਸਾਡੀ ਸੁਪਰ ਕਲਾਸ ਵਿੱਚ ਪੈਰਾਮੀਟ੍ਰਾਈਜ਼ਡ ਕੰਸਟਕਰਕਟਰ ਹੁੰਦਾ ਹੈ, ਤਾਂ ਇਹ ਲਾਜ਼ਮੀ ਹੋ ਜਾਂਦਾ ਹੈ ਕਿ ਕੰਸਟਰਕਟਰ ਨੂੰ ਸਬ-ਕਲਾਸ ਵਿੱਚ ਵੱਖਰੇ ਤੌਰ ਤੇ ਕਾਲ ਕੀਤਾ ਜਾਵੇ ਅਤੇ ਪੈਰਾਮੀਟਰ ਮੁੱਲਾਂ ਨੂੰ ਪਾਸ ਕੀਤਾ ਜਾਵੇ। ਇਸ ਸਥਿਤੀ ਵਿੱਚ ਇਹ ਜ਼ਰੂਰੀ ਹੁੰਦਾ ਹੈ ਕਿ ਅਸੀਂ ਸਬ-ਕਲਾਸ ਵਿੱਚ ਸੁਪਰ ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਨੂੰ ਅਲੱਗ ਤੋਂ ਕਾਲ ਕਰੀਏ ਜੋ ਕਿ super() ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਕਾਲ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। super() ਮੈਥਡ ਬਾਰੇ ਕੁਝ ਮਹੱਤਵਪੂਰਨ ਗੱਲਾਂ ਹੇਠਾਂ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ :

- super() ਮੈਥਡ ਹਮੇਸ਼ਾ ਡਰਾਇਵਡ ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਵਿੱਚ ਪਹਿਲੀ ਸਟੇਟਮੈਂਟ ਦੇ ਤੌਰ ਤੇ ਵਰਤਿਆ ਜਾਣਾ ਚਾਹੀਦਾ ਹੈ ਨਹੀਂ ਤਾਂ ਕੰਪਾਇਲੇਸ਼ਨ ਐਰਰ ਦਿਖਾਈ ਦੇਵੇਗਾ।
- ਜਦੋਂ ਅਸੀਂ ਵੱਖਰੇ ਤੌਰ ਤੇ ਸੁਪਰ ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਨੂੰ ਸਬ (ਡਰਾਇਵਡ) ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਵਿਚ ਕਾਲ ਕਰਦੇ ਹਾਂ ਤਾਂ ਜਾਵਾ ਕੰਪਾਈਲਰ ਸੁਪਰ (ਬੇਸ) ਕਲਾਸ ਦੇ ਡਿਫੌਲਟ ਕੰਸਟਰਕਟਰ ਨੂੰ ਕਾਲ ਨਹੀਂ ਕਰਦਾ।

ਅਸੀਂ ਹੁਣ ਤੱਕ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਧਾਰਨਾ ਨੂੰ ਸਮਝ ਚੁੱਕੇ ਹਾਂ। ਆਓ ਇੱਕ ਉਦਾਹਰਨ ਦੇ ਰੂਪ ਵਿੱਚ super() ਮੈਥਡ ਨੂੰ ਹੋਰ ਵਿਸਥਾਰਪੂਰਨ ਤਰੀਕੇ ਨਾਲ ਸਮਝੀਏ :

ਪ੍ਰੋਗਰਾਮ 9.9. ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਸੁਪਰ ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
class Base
{
Base(int id)// Base class constructor with parameters
{
System.out.println("Student id is: "+id);
}
}
class Derived extends Base
{
Derived(int sid)// derived class construction with parameters
{
super(sid);   // Passing parameters to base class constructor
}
void show(String sname)
{
System.out.println("Student name is: "+sname);
}
}

class CAProg22
{
        public static void main(String arg[])
        {
        Derived obj=new Derived(1001);
        obj.show("Shivpreet");
        }
}
```

ਪ੍ਰੋਗਰਾਮ 9.9 (CAProg22.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg22.java

D:\JavaProg>java CAProg22
Student id is: 1001
Student name is: Shivpreet

D:\JavaProg>_
```

ਅਸੀਂ ਇਨਹੈਰੀਟੈਂਸ ਦੀਆਂ ਵੱਖ-ਵੱਖ ਕਿਸਮਾਂ ਵਿਚ ਕਲਾਸ ਦੇ ਸਾਰੇ ਤੱਤਾਂ ਜਿਵੇਂ ਕੀ ਫੀਲਡਜ਼ (fields) ਮੈਥਡਜ਼ (methods) ਆਦਿ ਨਾਲ super ਕੀਵਰਡ ਜਾਂ super() ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕਰ ਸਕਦੇ ਹਾਂ।

9.4 ਜਾਵਾ ਵਿੱਚ ਐਬਸਟਰੈਕਸ਼ਨ (ABSTRACTION IN JAVA):

ਐਬਸਟਰੈਕਸ਼ਨ ਕਿਸੇ ਵੀ ਪ੍ਰੋਗਰਾਮ ਦੀ ਅੰਦਰੂਨੀ ਕਾਰਜ਼ਸ਼ੀਲਤਾ ਨੂੰ ਛੁਪਾਉਣਾ ਅਤੇ ਯੂਜ਼ਰ ਨੂੰ ਵਰਤੋਂ ਲਈ ਸਿਰਫ਼ ਲੋੜੀਂਦੇ ਪਹਿਲੂ ਦਿਖਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਹੁੰਦੀ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ ਕਿ ਇਹ ਯੂਜ਼ਰ ਨੂੰ ਸਿਰਫ ਜ਼ਰੂਰੀ ਤੱਤ ਹੀ ਦਿਖਾਉਂਦੀ ਹੈ ਅਤੇ ਅੰਦਰੂਨੀ ਜਟਿਲਤਾਵਾਂ (complexities) ਨੂੰ ਛੁਪਾ ਕੇ ਰੱਖਦੀ ਹੈ। ਉਦਾਹਰਨ ਲਈ : ਇੱਕ ਕਾਰ ਵਿੱਚ ਸਿਰਫ਼ ਸਟੀਅਰਿੰਗ (steering), ਐਕਸਲਰੇਟਰ (accelerator), ਬ੍ਰੇਕ (brakes) , ਗੀਅਰ (gears) ਜਾਂ ਹੋਰ ਇਲੈਕਟ੍ਰਾਨਿਕ ਕੰਟਰੋਲ (electronic control) ਵਰਗੇ ਨਿਯੰਤਰਣਾਂ ਨੂੰ ਹੀ ਯੂਜ਼ਰ ਦੀ ਪਹੁੰਚ ਵਿੱਚ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ ਅਤੇ ਇੰਜਣ ਦਾ ਅੰਦਰੂਨੀ ਕੰਮਕਾਜ ਯੂਜ਼ਰ ਤੋਂ ਛੁਪਿਆ ਰਹਿੰਦਾ ਹੈ।

ਜਾਵਾ ਵਿੱਚ ਐਬਸਟਰੈਕਟ ਕਲਾਸਾਂ ਜਾਂ ਇੰਟਰਫੇਸਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਐਬਸਟਰੈਕਸ਼ਨ ਪ੍ਰਾਪਤ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ। abstract ਕੀਅ-ਵਰਡ ਇੱਕ ਨਾਨ-ਐਕਸੈੱਸ ਮੋਡੀਫਾਇਰ (non-access modifier) ਹੁੰਦਾ ਹੈ, ਜੋ ਕਲਾਸਾਂ ਅਤੇ ਮੈਥਡਜ਼ ਲਈ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।ਅਸੀਂ ਜਾਵਾ ਵਿੱਚ ਐਬਸਟਰੈਸਕਸ਼ਨ ਨੂੰ ਹੇਠ ਲਿਖੀਆਂ ਕਿਸਮਾਂ ਵਿੱਚ ਵੰਡ ਸਕਦੇ ਹਾਂ :

- **ਐਬਸਟਰੈਕਟ ਕਲਾਸ (Abstract class) :** ਇਹ ਇੱਕ ਖਾਸ ਕਿਸਮ ਦੀ ਕਲਾਸ ਹੰਦੀ ਹੈ ਜਿਸਦੀ ਵਰਤੋਂ ਆਬਜੈਕਟ ਬਣਾਉਣ ਲਈ ਨਹੀਂ ਕੀਤੀ ਜਾ ਸਕਦੀ।ਇਸ ਕਲਾਸ ਦੇ ਮੈਂਬਰਜ਼ ਨੂੰ ਐਕਸੈੱਸ ਕਰਨ ਲਈ ਇਸ ਕਲਾਸ ਨੂੰ ਕਿਸੇ ਹੋਰ ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਟ ਕੀਤਾ ਜਾਣਾ ਜਰੂਰੀ ਹੁੰਦਾ ਹੈ।
- **ਐਬਸਟਰੈਕਟ ਮੈਥਡ (Abstract methods) :** ਇਹ ਮੈਥਡਜ਼ ਕੇਵਲ ਇੱਕ ਐਬਸਟ੍ਰੈਕਟ ਕਲਾਸ ਜਾਂ ਇੰਟਰਫੇਸ ਵਿੱਚ ਹੀ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ ਅਤੇ ਇਹਨਾਂ ਮੈਥਡਜ਼ ਦੀ ਕੋਈ ਬਾਡੀ ਉਸ ਕਲਾਸ ਵਿੱਚ ਨਹੀਂ ਹੋ ਸਕਦੀ। ਇਹਨਾਂ ਮੈਥਡਜ਼ ਦੀ ਬਾਡੀ ਸਬ-ਕਲਾਸ ਵਿੱਚ ਹੀ ਬਣਾਈ ਜਾਂਦੀ ਹੈ।(ਅਸੀਂ ਇਸ ਅਧਿਆਇ ਦੇ ਇੰਟਰਫੇਸ ਭਾਗ ਵਿੱਚ ਪਹਿਲਾਂ ਹੀ ਇਸ ਸੰਕਲਪ ਬਾਰੇ ਪੜ੍ਹ ਚੁੱਕੇ ਹਾਂ)

ਆਉ ਇੱਕ ਉਦਾਹਰਣ ਵਜੋਂ ਐਬਸਟਰੈਕਟ ਕਲਾਸ ਦੀ ਧਾਰਨਾ (concept) ਨੂੰ ਸਮਝੀਏ।

ਪ੍ਰੋਗਰਾਮ 9.0 . ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਐਬਸਟਰੈਕਟ ਕਲਾਸ ਅਤੇ ਐਬਸਟਰੈਕਟ ਮੈਥਡਜ਼ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
abstract class Base
{                          // abstract method
abstract void show();
void display()
{
System.out.println("This is not an abstract function");
}
}
class CAProg23
{
        public static void main(String arg[])
        {
        Base obj=new Base();
        obj.display();
        }
}
```

ਪ੍ਰੋਗਰਾਮ Program 9.10 (CAProg23.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg23.java
CAProg23.java:13: Base is abstract; cannot be instantiated
        Base obj=new Base();
                 ^
1 error

D:\JavaProg>_
```

ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਦੇਖ ਸਕਦੇ ਹਾਂ, ਇਸ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਐਗਜ਼ੀਕਿਊਟ (execute) ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਕਿਉਂਕਿ ਐਬਸਟਰੈਕਟ ਕਲਾਸ ਦਾ ਸਿੱਧੇ ਰੂਪ ਨਾਲ ਕੋਈ ਵੀ ਆਬਜੈਕਟ ਨਹੀਂ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਅਸੀਂ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਉਸੇ ਪ੍ਰੋਗਰਾਮ ਨੂੰ ਸਹੀ ਤਰੀਕੇ ਨਾਲ ਵਰਤ ਸਕਦੇ ਹਾਂ:

ਪ੍ਰੋਗਰਾਮ 9.11 ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਐਬਸਟਰੈਕਟ ਕਲਾਸ ਅਤੇ ਐਬਸਟਰੈਕਟ ਵਿਧੀ ਦੀ ਸਹੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
abstract class Base
{
abstract void show();//abstract method
void display()
{
System.out.println("This is not an abstract function");
}
}
class CAProg23 extends Base
{
        void show()  //defination of abstract method
        {
        System.out.println("This is an abstract function");
        }
        public static void main(String arg[])
        {
        CAProg23 obj=new CAProg23();
        obj.show();
        obj.display();
        }
}
```

ਪ੍ਰੋਗਰਾਮ 9.11 (CAProg23.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਉਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg23.java

D:\JavaProg>java CAProg23
This is an abstract function
This is not an abstract function

D:\JavaProg>_
```

ਉਪਰੋਕਤ ਦਿੱਤੀ ਸਹੀ ਉਦਾਹਰਣ ਵਿੱਚ ਅਸੀਂ ਐਬਸਟਰੈਕਟ ਕਲਾਸ ਅਤੇ ਐਬਸਟਰੈਕਟ ਮੈਥਡ ਨੂੰ ਡਿਕਲੇਅਰ ਕਰਨ ਅਤੇ ਉਹਨਾਂ ਨੂੰ ਐਕਸੈੱਸ ਕਰਨ ਦਾ ਸਹੀ ਤਰੀਕਾ ਦੇਖ ਸਕਦੇ ਹਾਂ।

9.5 ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ (METHOD OVERRIDING IN JAVA)

ਅਸੀਂ ਪਹਿਲਾਂ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਬਾਰੇ ਪੜ੍ਹ ਚੁੱਕੇ ਹਾਂ ਜਿੱਥੇ ਅਸੀਂ ਇੱਕੋ ਨਾਮ ਪਰੰਤੂ ਵੱਖ-ਵੱਖ ਆਰਗੂਮੈਂਟਸ ਜਾਂ ਵੱਖ-ਵੱਖ ਡੇਟਾ ਕਿਸਮਾਂ ਦੇ ਆਰਗੂਮੈਂਟ ਨਾਲ ਕਈ ਮੈਥਡਜ਼ ਬਣਾ ਕੇ ਅਲੱਗ-ਅਲੱਗ ਕੰਮ ਕੀਤੇ ਸਨ। ਜੇਕਰ ਸੁਪਰ ਕਲਾਸ ਵਿੱਚ ਘੋਸ਼ਿਤ ਕੀਤੇ ਗਏ ਕਿਸੇ ਵੀ ਮੈਥਡ ਨੂੰ ਸਬ-ਕਲਾਸ ਵਿੱਚ ਉਸੇ ਨਾਮ ਦਾ ਮੈਥਡ ਬਣਾ ਕੇ ਓਵਰਰਾਈਡ ਕੀਤਾ ਜਾਵੇ ਤਾਂ ਇਸਨੂੰ ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਦੂਜੇ ਸ਼ਬਦਾਂ ਵਿੱਚ, ਜੇਕਰ ਇੱਕ ਸਬ ਕਲਾਸ ਉਸ ਮੈਥਡ ਦਾ ਕੋਈ ਬਦਲਵਾਂ ਕੰਮ ਦੇ ਕੇ ਪ੍ਰਯੋਗ ਕਰੇ ਜੋ ਇਸਦੀ ਸੁਪਰ ਕਲਾਸ ਵਿੱਚੋਂ ਇੱਕ ਮੈਥਡ ਦੇ ਤੌਰ ਤੇ ਘੋਸ਼ਿਤ ਕੀਤਾ ਗਿਆ ਹੈ, ਤਾਂ ਇਸਨੂੰ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਵਜੋਂ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ। ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਲਈ ਕੁਝ ਬੁਨਿਆਦੀ ਨਿਯਮ ਹੇਠਾਂ ਲਿਖੇ ਅਨੁਸਾਰ ਹਨ:

- ਮੈਥਡ ਦਾ ਉਹੀ ਨਾਮ ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ ਜੋ ਕਿ ਪੇਰੈਂਟ ਕਲਾਸ ਵਿੱਚ ਹੈ।
- ਮੈਥਡ ਵਿੱਚ ਉਹੀ ਪੈਰਾਮੀਟਰਜ਼ (parameter) ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ ਜੋ ਪੇਰੈਂਟ ਕਲਾਸ ਵਿੱਚ ਹਨ।
- ਦੋਵੇਂ ਕਲਾਸਾਂ ਵਿੱਚ ਇੱਕ IS-A ਰੀਲੇਸ਼ਨ (ਇਨਹੈਰੀਟੈਂਸ inheritance) ਹੋਣਾ ਚਾਹੀਦਾ ਹੈ।

ਆਉ ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਨੂੰ ਇੱਕ ਉਦਾਹਰਣ ਦੇ ਰੂਪ ਵਿੱਚ ਸਮਝੀਏ।

ਪ੍ਰੋਗਰਾਮ 9.12 ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਨੂੰ ਪ੍ਰਦਰਸ਼ੀਤ ਕਰਦਾ ਹੈ।

```
class Base
{
void display()
{
System.out.println("Base class is here");
}
}
class Derived extends Base
{
void display() // method overriding
{
System.out.println("Derived class is here");
}
}
```

```
class CAProg24
{
	public static void main(String arg[])
	{
	Derived obj=new Derived();
	obj.display();
	}
}
```

ਪ੍ਰੋਗਰਾਮ Program 9.12 (CAProg24.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
D:\JavaProg>javac CAProg24.java

D:\JavaProg>java CAProg24
Derived class is here

D:\JavaProg>_
```

ਇਹ ਉਦਾਹਰਨ ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਦੀ ਵਰਤੋਂ ਦੀ ਵਿਆਖਿਆ ਕਰਦੀ ਹੈ। ਅਸੀਂ ਕਿਸੇ ਵੀ ਸਟੈਟਿਕ ਮੈਥਡ (static method) ਨੂੰ ਓਵਰਰਾਈਡ ਨਹੀਂ ਕਰ ਸਕਦੇ। ਜਾਵਾ ਵਿੱਚ main ਮੈਥਡ ਨੂੰ ਵੀ ਓਵਰਰਾਈਡ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ।

ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਅਤੇ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਵਿੱਚ ਅੰਤਰ (Difference between method overloading and method overriding):

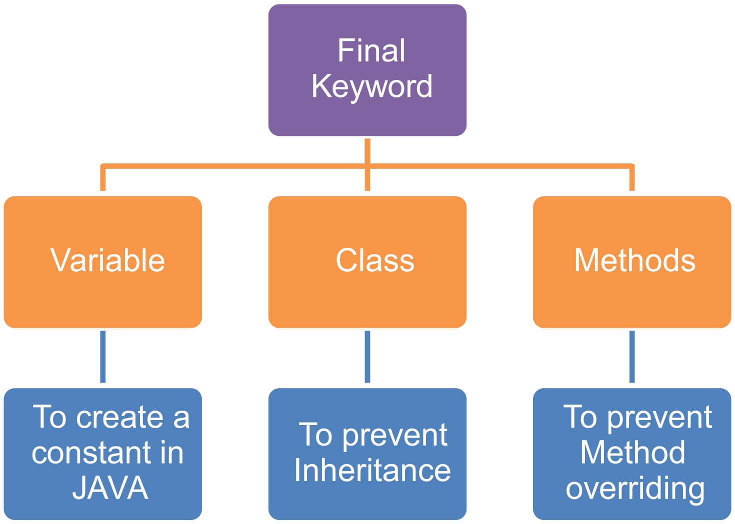
ਜਾਵਾ ਵਿੱਚ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਅਤੇ ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਵਿੱਚ ਕਈ ਅੰਤਰ ਹਨ ਜੋ ਹੇਠਾਂ ਦਰਸਾਏ ਅਨੁਸਾਰ ਹਨ:

ਲੜੀ ਨੰ:	ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ (Method Overloading)	ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ (Method Overriding)
1)	ਪ੍ਰੋਗਰਾਮ ਦੀ ਪੜ੍ਹਨਯੋਗਤਾ (readability) ਨੂੰ ਵਧਾਉਣ ਲਈ ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।	ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਦੀ ਵਰਤੋਂ ਉਹਨਾਂ ਮੈਥਡਜ਼ ਦੇ ਖਾਸ ਲਾਗੂ-ਕਰਨ ਨੂੰ ਵਰਤਣ ਲਈ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜੋ ਪਹਿਲਾਂ ਹੀ ਕਿਸੇ ਕਲਾਸ ਦੀ ਸੁਪਰ ਕਲਾਸ ਵਿੱਚ ਬਣੇ ਹੁੰਦੇ ਹਨ।
2)	ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਕਿਸੇ ਇੱਕ ਕਲਾਸ ਦੇ ਅੰਦਰ ਕੀਤੀ ਜਾਂਦੀ ਹੈ।	ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਦੋ ਵੱਖ-ਵੱਖ ਕਲਾਸਾਂ ਵਿੱਚ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਿਨ੍ਹਾਂ ਵਿੱਚ IS-A ਰਿਲੇਸ਼ਨ (ਇਨਹੈਰੀਟੈਂਸ) ਹੋਵੇ।
3)	ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਵਿੱਖ ਪੈਰਾਮੀਟਰਜ਼ ਵੱਖਰੇ ਹੋਣੇ ਚਾਹੀਦੇ ਹਨ।	ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਦੇ ਮਾਮਲੇ ਵਿੱਚ ਪੈਰਾਮੀਟਰ ਵੀ ਇੱਕੋ ਜਿਹੇ ਹੋ ਸਕਦੇ ਹਨ।

4)	ਮੈਥਡ ਓਵਰਲੋਡਿੰਗ ਕੰਪਾਈਲ ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (compile time polymorphism) ਦੀ ਉਦਾਹਰਨ ਹੈ।	ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਰਨ ਟਾਈਮ ਪੋਲੀਮੋਰਫਿਜ਼ਮ (run time polymorphism) ਦੀ ਉਦਾਹਰਨ ਹੈ।
5)	ਜਾਵਾ ਵਿੱਚ ਮੈਥਡਜ਼ ਦੀ ਓਵਰਲੋਡਿੰਗ ਮੈਥਡ ਦੀ ਰਿਟਰਨ ਟਾਇਪ ਬਦਲ ਕੇ ਨਹੀਂ ਕੀਤੀ ਜਾ ਸਕਦੀ।	ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ ਵਿੱਚ ਮੈਥਡ ਦੀ ਰਿਟਰਨ ਟਾਈਪ ਇੱਕੋ ਜਿਹੀ ਹੋਣੀ ਚਾਹੀਦੀ ਹੈ।

9.6 ਜਾਵਾ ਵਿੱਚ ਫਾਈਨਲ ਕਲਾਸ ਮੈਥਡ ਅਤੇ ਵੇਰੀਏਬਲ (FINAL CLASS, METHOD AND VARIABLES IN JAVA)

final ਕੀਅ-ਵਰਡ ਜਾਵਾ ਵਿੱਚ ਮੌਜੂਦ ਕੀਅ-ਵਰਡਸ ਵਿੱਚੋਂ ਇੱਕ ਮਹੱਤਵਪੂਰਨ ਕੀਵਰਡ ਹੁੰਦਾ ਹੈ ਜੋ ਜਾਵਾ ਵਿੱਚ ਕਿਸੇ ਵੀ ਤੱਤ ਨਾਲ ਉਹਨਾਂ ਦੀ ਵਰਤੋਂ ਨੂੰ ਸੀਮਿਤ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।ਅਸੀਂ ਇਸ ਨੂੰ ਕਲਾਸ (class) , ਮੈਥਡਜ਼ (methods), ਅਤੇ ਵੇਰੀਏਬਲ (variables), ਦੇ ਨਾਲ ਵਰਤ ਸਕਦੇ ਹਾਂ।ਜਾਵਾ final ਕੀਵਰਡ ਇੱਕ ਨਾਨ-ਐਕਸੈੱਸ ਮੋਡੀਫਾਇਰ ਹੁੰਦਾ ਹੈ ਜੋ ਕਲਾਸ, ਵੇਰੀਏਬਲ ਅਤੇ ਮੈਥਡ ਤੇ ਕੁਝ ਨਿਰਧਾਰਿਤ ਪਾਬੰਦੀ (restriction) ਲਗਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।ਜੇਕਰ ਅਸੀਂ final ਕੀਅ-ਵਰਡ ਨਾਲ ਇੱਕ ਨਾਲ ਇੱਕ ਵੇਰੀਏਬਲ ਨੂੰ ਡਿਕਲੇਅਰ ਕਰਦੇ ਹਾਂ ਤਾਂ ਅਸੀਂ ਇਸਦੇ ਮੁੱਲ ਨੂੰ ਬਦਲਣ ਤੋਂ ਰੋਕ ਸਕਦੇ ਹਾਂ। ਜੇਕਰ ਅਸੀਂ ਇੱਕ ਮੈਥਡ ਨੂੰ final ਕੀਅ-ਵਰਡ ਨਾਲ ਘੋਸ਼ਿਤ (declare) ਕਰਦੇ ਹਾਂ ਤਾਂ ਇਸਨੂੰ ਕਿਸੇ ਵੀ ਸਬ-ਕਲਾਸ ਦੁਆਰਾ ਓਵਰਰਾਈਡ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ।ਜੇਕਰ ਅਸੀਂ ਇੱਕ ਕਲਾਸ ਨੂੰ final ਕਲਾਸ ਦੇ ਤੌਰ ਤੇ ਡਿਕਲੇਅਰ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਕਲਾਸ ਨੂੰ ਇਨਹੈਰੇਟ ਹੋਣ ਤੋਂ ਰੋਕ ਸਕਦੇ ਹਾਂ।ਅਸੀਂ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਫਾਈਨਲ ਕੀਵਰਡ ਦੀ ਵਰਤੋਂ ਨੂੰ ਗ੍ਰਾਫਿਕ ਤੌਰ ਤੇ ਦਰਸਾ ਸਕਦੇ ਹਾਂ।



ਚਿੱਤਰ : final ਕੀਅ-ਵਰਡ ਦੇ ਵੱਖ-ਵੱਖ ਉਪਯੋਗ

ਆਓ ਜਾਵਾ ਵਿੱਚ ਵੱਖ-ਵੱਖ ਤੱਤਾਂ ਦੇ ਨਾਲ ਇਸ final ਕੀਅ-ਵਰਡ ਦੀ ਵਰਤੋਂ ਬਾਰੇ ਸਮਝੀਏ।

final ਵੇਰੀਏਬਲ (variable) :

ਜੇਕਰ ਅਸੀਂ ਕਿਸੇ ਵੀ ਵੇਰੀਏਬਲ ਨੂੰ final ਘੋਸ਼ਿਤ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਇਸ ਦਾ ਮੁੱਲ ਬਦਲਣਯੋਗ ਨਹੀਂ ਰਹਿੰਦਾ। ਇਹ ਵੇਰੀਏਬਲ ਇੱਕ ਕਾਂਸਟੈਂਟ (constant) ਬਣ ਜਾਵੇਗਾ। ਇਸ ਤੋਂ ਭਾਵ ਹੈ ਕਿ ਸਾਨੂੰ ਘੋਸ਼ਣਾ ਕਰਦੇ ਸਮੇਂ ਹੀ ਇੱਕ final ਵੇਰੀਏਬਲ ਨੂੰ ਸ਼ੁਰੂਆਤੀ ਮੁੱਲ ਦੇਣਾ ਪੈਂਦਾ ਹੈ ਕਿਉਂਕਿ ਉਸ ਤੋਂ ਬਾਅਦ final ਵੇਰੀਏਬਲ ਵਿੱਚ ਕੋਈ ਤਬਦੀਲੀ ਕਰਨ ਦੀ ਆਗਿਆ ਨਹੀਂ ਹੁੰਦੀ। ਜੇਕਰ final ਵੇਰੀਏਬਲ ਇੱਕ ਰੈਫਰੈਂਸ ਹੈ ਤਾਂ ਇਸ ਤੋਂ ਭਾਵ ਹੈ ਵੇਰੀਏਬਲ ਵਿੱਚ ਕਿਸੇ ਹੋਰ ਆਬਜੈਕਟ ਦਾ ਰੈਫਰੈਂਸ ਸਟੋਰ ਕਰਨ ਲਈ ਉਸਨੂੰ ਰੀ-ਬਾਉਂਡ

(re-bound) ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ, ਪਰੰਤੂ ਉਸ ਰੈਫਰੈਂਸ ਦੁਆਰਾ ਦਰਸਾਏ ਗਏ ਆਬਜੈਕਟ ਦੇ ਅੰਦਰੂਨੀ ਮੁੱਲਾਂ ਨੂੰ ਬਦਲਿਆ ਜਾ ਸਕਦਾ ਹੈ। ਉਦਾਹਰਣ ਵਜੋਂ ਅਸੀਂ final ਐਰੇ ਜਾਂ final ਕੁਲੈਕਸ਼ਨ (collection) ਵਿੱਚ ਨਵੇਂ ਐਲੀਮੈਂਟਸ (elements) ਜੋੜ ਜਾਂ ਹਟਾ ਸਕਦੇ ਹਾਂ। ਸਾਰੇ final ਐਲੀਮੈਂਟਸ ਨੂੰ ਵੱਖਰੇ ਤੌਰ ਤੇ ਦਰਸਾਉਣ ਲਈ ਅੰਡਰਸਕੋਰ ਦੀ ਵਰਤੋਂ ਕਰਦੇ ਹੋਏ ਸਾਰੇ ਵੱਡੇ ਅੱਖਰਾਂ ਵਿੱਚ ਦਰਸਾਉਣਾ ਜ਼ਿਆਦਾ ਸਪਸ਼ਟਤਾ ਪ੍ਰਦਾਨ ਕਰਦਾ ਹੈ। ਆਉ ਇੱਕ ਉਦਾਹਰਣ ਨਾਲ final ਵੇਰੀਏਬਲ ਦੀ ਵਰਤੋਂ ਨੂੰ ਸਮਝੀਏ।

ਪ੍ਰੋਗਰਾਮ 9.13 ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ final ਵੇਰੀਏਬਲ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
class CAProg25
{
        public static void main(String arg[])
        {
        final int size=40;  //final variable
        System.out.println("Size of class is: "+size);
        size=45;
        }
}
```

ਪ੍ਰੋਗਰਾਮ 9.13 (CAProg25.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg25.java
CAProg25.java:7: cannot assign a value to final variable size
        size=45;
        ^
1 error

D:\JavaProg>_
```

ਇੱਥੇ ਅਸੀਂ ਸਪਸ਼ੱਟ ਤੌਰ ਤੇ ਦੇਖ ਸਕਦੇ ਹਾਂ ਕਿ ਡਿਕਲੇਅਰੇਸ਼ਨ (declaration) ਤੋਂ ਬਾਅਦ final ਵੇਰੀਏਬਲ ਵਿੱਚ ਕਿਸੇ ਵੀ ਬਦਲਾਅ ਦੀ ਇਜਾਜ਼ਤ ਨਹੀਂ ਹੈ। ਜੇਕਰ ਅਸੀਂ ਸਟੇਟਮੈਂਟ size = 45; ਨੂੰ ਹਟਾ ਦਿੰਦੇ ਹਾਂ ਤਾਂ ਇਹ ਪ੍ਰੋਗਰਾਮ ਸਹੀ ਢੰਗ ਨਾਲ ਚੱਲੇਗਾ ਅਤੇ ਆਉਟਪੁੱਟ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਹੋਵੇਗੀ:

ਪ੍ਰੋਗਰਾਮ 9.13 (CAProg25.java) ਦੇ ਸੁਧਾਰ ਤੋਂ ਬਾਅਦ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg25.java

D:\JavaProg>java CAProg25
Size of class is: 40

D:\JavaProg>
```

ਇੱਥੇ ਅਸੀਂ ਉਕਤ ਸਟੇਟਮੈਂਟ ਨੂੰ ਹਟਾਉਣ ਤੋਂ ਬਾਅਦ ਪ੍ਰੋਗਰਾਮ ਦੀ ਆਉਟਪੁੱਟ ਦੇਖ ਸਕਦੇ ਹਾਂ।

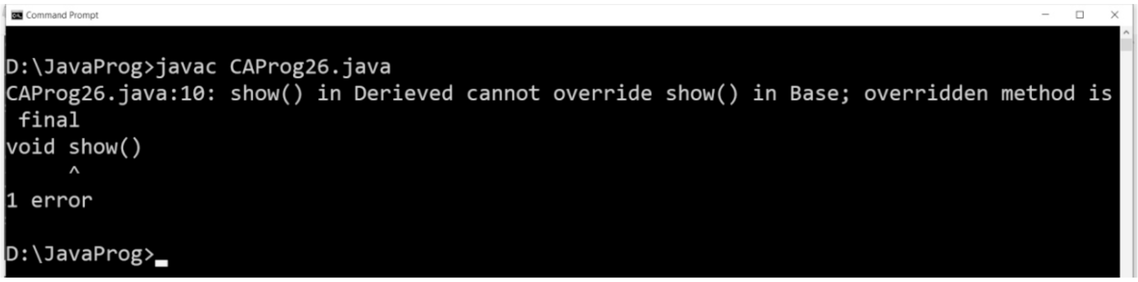
Final ਮੈਥਡ (Final Methods)

ਜਦੋਂ ਕਿਸੇ ਮੈਥਡ ਨੂੰ final ਕੀਅ-ਵਰਡ ਨਾਲ ਘੋਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਤਾਂ ਇਸ ਨੂੰ ਫਾਈਨਲ ਮੈਥਡ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇੱਕ ਫਾਇਨਲ ਮੈਥਡ ਨੂੰ ਓਵਰਰਾਈਡ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ। ਸਾਨੂੰ final ਕੀਅ-ਵਰਡ ਦੇ ਨਾਲ ਉਸ ਮੈਥਡ ਦੀ ਡਿਕਲੇਅਰੇਸ਼ਨ ਕਰਨੀ ਪੈਂਦੀ ਹੈ ਜਿਸ ਲਈ ਸਾਨੂੰ ਸਾਰੀਆਂ ਸਬ-ਕਲਾਸਾਂ ਵਿੱਚ ਇੱਕੋ ਕਾਰਜ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੀ ਲੋੜ ਹੋਵੇ। ਅਜਿਹੇ ਮੈਥਡ ਦੀ ਜ਼ਰੂਰਤ ਉਹਨਾਂ ਐਪਲੀਕੇਸ਼ਨਾਂ ਵਿੱਚ ਹੋ ਸਕਦੀ ਹੈ ਜਿੱਥੇ ਕਿਸੇ ਵੀ ਤਬਦੀਲੀ ਨੂੰ ਪੱਕੇ ਤੌਰ ਰੋਕਣ ਦੀ ਲੋੜ ਹੁੰਦੀ ਹੈ। ਅਸੀਂ final ਮੈਥਡ ਦੀ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ:

ਪ੍ਰੋਗਰਾਮ 9.14 ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਫਾਇਨਲ ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
class Base
{
final void show()   //final method
{
System.out.println("Base Function");
}
}
class Derieved extends Base
{
void show()  //can not be overridden due to final method
{
System.out.println("Derived Function");
}
}
class CAProg26
{
        public static void main(String arg[])
        {
        Derived obj=new Derived();
        obj.show();
        }
}
```

ਪ੍ਰੋਗਰਾਮ 9.14 (CAProg26.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ



```
D:\JavaProg>javac CAProg26.java
CAProg26.java:10: show() in Derieved cannot override show() in Base; overridden method is
 final
void show()
     ^
1 error

D:\JavaProg>_
```

ਜਿਵੇਂ ਕਿ ਅਸੀਂ ਦਿਖਾਈ ਗਈ ਆਉਟਪੁੱਟ ਸਕਰੀਨ ਵਿੱਚ ਦੇਖ ਸਕਦੇ ਹਾਂ, ਕਿਸੇ ਵੀ ਸਬ-ਕਲਾਸ ਵਿੱਚ ਕੋਈ ਵੀ ਫਾਇਨਲ ਮੈਥਡ ਨੂੰ ਓਵਰਰਾਈਡ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ।

Final ਕਲਾਸ (Final classes) :

ਜਦੋਂ ਇੱਕ ਕਲਾਸ ਨੂੰ final ਕੀਅ-ਵਰਡ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਫਾਈਨਲ ਘੋਸ਼ਿਤ ਕਰ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ ਤਾਂ ਇਸ ਕਲਾਸ ਨੂੰ ਕਿਸੇ ਵੀ ਸਬ ਕਲਾਸ ਦੁਆਰਾ ਐਕਸਟੈਂਡ (extende) ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ। ਇਸ ਕਿਸਮ ਦੀਆਂ ਕਲਾਸਾਂ ਕਿਸੇ ਮੌਜੂਦਾ ਸੁਪਰ ਕਲਾਸ ਵਿੱਚ ਬਣਾਏ ਗਏ ਮੈਥਡਜ਼ ਨੂੰ ਓਵਰਰਾਈਡ ਹੋਣ ਤੋਂ ਰੋਕਣ ਲਈ ਉਪਯੋਗੀ ਹੁੰਦੀਆਂ ਹਨ।ਅਸੀਂ ਫਾਇਨਲ ਕਲਾਸ ਦੀ ਇੱਕ ਉਦਾਹਰਣ ਹੇਠਾਂ ਦਿੱਤੇ ਅਨੁਸਾਰ ਦੇਖ ਸਕਦੇ ਹਾਂ :

ਪ੍ਰੋਗਰਾਮ 9.15 : ਹੇਠਾਂ ਦਿੱਤਾ ਪ੍ਰੋਗਰਾਮ ਜਾਵਾ ਵਿੱਚ ਫਾਈਨਲ ਕਲਾਸ ਦੀ ਵਰਤੋਂ ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ।

```
final class Base //Final Class
{
void show1()
{
System.out.println("Base Function");
}
}
class Derieved extends Base  //can not extended/inherited
{
void show2()
{
System.out.println("Derived Function");
}
}
class CAProg27
{
        public static void main(String arg[])
        {
        Derived obj=new Derived();
        obj.show2( );
        }
}
```

ਪ੍ਰੋਗਰਾਮ 9.15 (CAProg27.java) ਦੀ ਕੰਪਾਇਲੇਸ਼ਨ, ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਅਤੇ ਆਉਟਪੁੱਟ

```
Command Prompt

D:\JavaProg>javac CAProg27.java
CAProg27.java:8: cannot inherit from final Base
class Derieved extends Base
                       ^
CAProg27.java:16: cannot inherit from final Base
class CAProg27 extends Base
                       ^
```

ਇੱਥੇ ਅਸੀਂ ਵੇਖ ਸਕਦੇ ਹਾਂ ਕਿ ਫਾਇਨਲ ਕਲਾਸ ਨੂੰ ਇਨਹੈਰੇਂਟ ਕਰਨ ਦੀ ਆਗਿਆ ਨਹੀਂ ਹੈ। ਜੇਕਰ ਅਸੀਂ ਅਜਿਹਾ ਕਰਨ ਦੀ ਕੋਸ਼ਿਸ਼ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਇਸ ਸੰਬੰਧੀ ਐਰੇਰ ਦੇ ਰੂਪ ਵਿੱਚ ਸਾਨੂੰ ਉਕਤ ਅਨੁਸਾਰ ਮੈਸੇਜ (message) ਪ੍ਰਦਰਸ਼ਿਤ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਯਾਦ ਰੱਖਣ ਯੋਗ ਗੱਲਾਂ

1. ਇਨਹੈਰੀਟੈਂਸ (inheritance) ਇੱਕ ਅਜਿਹੀ ਵਿਸ਼ੇਸ਼ਤਾ ਜਾਂ ਪ੍ਰਕਿਰਿਆ ਹੁੰਦੀ ਹੈ ਜਿਸ ਵਿੱਚ ਮੌਜੂਦਾ ਕਲਾਸ ਤੋਂ ਨਵੀਆਂ ਕਲਾਸਾਂ ਬਣਾਈਆਂ ਜਾਂਦੀਆਂ ਹਨ।
2. ਕਲਾਸ ਨੂੰ ਸਾਰੇ ਆਬਜੈਕਟਸ ਦੀਆਂ ਸਾਂਝੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਦੇ ਸੰਗ੍ਰਹਿ ਵਜੋਂ ਪਰਿਭਾਸ਼ਿਤ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਜਾਵਾ ਪ੍ਰੋਗਰਾਮ ਵਿੱਚ ਬਣਾਏ ਗਏ ਇਕ ਕਿਸਮ ਦੇ ਆਬਜੈਕਟਸ ਦਾ ਬਲੂਪ੍ਰਿੰਟ ਹੁੰਦਾ ਹੈ।
3. ਸਬ ਕਲਾਸ ਵਿੱਚ ਇਸਦੀ ਸੁਪਰ ਕਲਾਸ ਦੇ ਇਨਹੈਰੇਂਟ ਹੋਏ ਫੀਲਡਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਸ਼ਾਮਲ ਹੁੰਦੇ ਹਨ ਅਤੇ ਨਾਲ ਹੀ ਇਸ ਦੇ ਆਪਣੇ ਖੁਦ ਦੇ ਬਣਾਏ ਮੈਂਬਰ ਵੀ ਸ਼ਾਮਿਲ ਹੋ ਸਕਦੇ ਹਨ।
4. ਸਬ ਕਲਾਸ ਨੂੰ ਡਰਾਇਵਡ ਕਲਾਸ ਜਾਂ ਐਕਸਟੈਂਡਿਡ ਕਲਾਸ ਵਜੋਂ ਵੀ ਜਾਣਿਆ ਜਾਂਦਾ ਹੈ।
5. ਸੁਪਰ ਕਲਾਸ ਉਹ ਸਾਰੀਆਂ ਸਾਂਝੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਪ੍ਰਦਰਸ਼ਿਤ ਕਰਦੀ ਹੈ ਜਿੰਨ੍ਹਾਂ ਨੂੰ ਫੀਲਡਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਦੇ ਰੂਪ ਵਿੱਚ ਸਬ ਕਲਾਸਜ਼ ਦੁਆਰਾ ਇਨਹੈਰੇਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।
6. ਮੁੜ ਵਰਤੋਂ ਯੋਗਤਾ ਨਵੀਂ ਬਣਾਈ ਗਈ ਕਲਾਸ ਵਿੱਚ ਮੌਜੂਦਾ ਕਲਾਸ ਦੇ ਮੈਥਡਜ਼ ਉਸੇ ਤਰ੍ਹਾਂ ਹੀ ਦੁਬਾਰਾ ਵਰਤੋਂ ਕਰਨ ਦੀ ਇੱਕ ਤਕਨੀਕ ਹੈ।
7. ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ (Single inheritance), ਮਲਟੀ-ਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ (Multi-level inheritance), ਹਾਇਰਾਰਕੀਕਲ ਇਨਹੈਰੀਟੈਂਸ (Hierarchical inheritance), ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ (Hybrid inheritance), ਅਤੇ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ (Multiple inheritance) ਇਨਹੈਰੀਟੈਂਸ ਦੀਆਂ ਕਿਸਮਾਂ ਹਨ।
8. ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ ਤੋਂ ਭਾਵ ਹੈ ਇੱਕ ਇਨਹੈਰੀਟੈਂਸ ਕਿਸਮ ਦੇ ਅੰਦਰ ਇੱਕ ਤੋਂ ਵੱਧ ਕੋਈ ਵੀ ਹੋਰ ਇਨਹੈਰੀਟੈਂਸ ਦੇ ਰੂਪਾਂ ਨੂੰ ਸ਼ਾਮਿਲ ਕਰਨਾ।
9. ਜਾਵਾ ਕਲਾਸਾਂ ਉੱਪਰ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਆਗਿਆ ਨਹੀਂ ਦਿੰਦੀ।
10. ਜਾਵਾ ਵਿੱਚ ਇੰਟਰਫੇਸ ਐਬਸਟਰੈਕਸ਼ਨ ਨੂੰ ਲਾਗੂ ਕਰਨ ਦੀ ਇੱਕ ਵਿਧੀ ਹੁੰਦੀ ਹੈ।
11. ਸਿਰਫ ਡਿਫਾਲਟ (default) ਮੈਥਡਜ਼ ਅਤੇ ਸਟੈਟਿਕ (static) ਮੈਥਡਜ਼ ਦੀਆਂ ਬਾਡੀਜ਼ ਹੀ ਇੰਟਰਫੇਸ ਦੇ ਅੰਦਰ ਮੌਜੂਦ ਹੋ ਸਕਦੀਆਂ ਹਨ।
12. ਅਸੀਂ ਇੰਟਰਫੇਸ ਦਾ ਆਬਜੈਕਟ ਨਹੀਂ ਬਣਾ ਸਕਦੇ ਅਤੇ ਇੰਟਰਫੇਸ ਵਿੱਚ ਕੋਈ ਵੀ ਕੰਸਟਰਕਟਰ ਸ਼ਾਮਿਲ ਨਹੀਂ ਹੁੰਦਾ।
13. ਜਾਵਾ ਵਿੱਚ ਇੰਟਰਫੇਸ ਰਾਹੀਂ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਵਰਤੀ ਜਾ ਸਕਦੀ ਹੈ।
14. ਜਾਵਾ ਵਿੱਚ ਇੰਟਰਫੇਸ ਐਬਸਟਰੈਕਸ਼ਨ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ ਇੱਕ ਵਿਧੀ ਹੁੰਦੀ ਹੈ।
15. Super ਕੀਅ ਵਰਡ ਸਾਨੂੰ ਸੁਪਰ ਕਲਾਸ ਦੇ ਮੈਂਬਰਾਂ ਤੱਕ ਪਹੁੰਚ ਕਰਨ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ।

16. Super ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਪੇਰੈਂਟ ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਨੂੰ ਕਾਲ ਕਰਨ ਲਈ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।
17. ਐਬਸਟਰੈਸਕਸ਼ਨ ਕਿਸੇ ਵੀ ਪ੍ਰੋਗਰਾਮ ਦੀ ਅੰਦਰੂਨੀ ਕਾਰਜਸ਼ੀਲਤਾ ਨੂੰ ਛੁਪਾਉਣਾ ਅਤੇ ਯੂਜ਼ਰ ਨੂੰ ਵਰਤੋਂ ਲਈ ਸਿਰਫ਼ ਲੋੜੀਂਦੇ ਪਹਿਲੂ ਦਿਖਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਹੁੰਦੀ ਹੈ।
18. ਜੇਕਰ ਅਸੀਂ ਮੈਥਡ ਨੂੰ final ਕੀਅ-ਵਰਡ ਨਾਲ ਘੋਸ਼ਿਤ (declare) ਕਰਦੇ ਹਾਂ ਤਾਂ ਇਸ ਨੂੰ ਕਿਸੇ ਵੀ ਸਬ-ਕਲਾਸ ਦੁਆਰਾ ਓਵਰਰਾਈਡ ਨਹੀਂ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।
19. ਜੇਕਰ ਅਸੀਂ ਕਲਾਸ ਨੂੰ final ਕਲਾਸ ਦੇ ਤੌਰ ਤੇ ਡਿਕਲੇਅਰ ਕਰਦੇ ਹਾਂ, ਤਾਂ ਅਸੀਂ ਕਲਾਸ ਨੂੰ ਇਨਹੈਰੇਂਟ ਹੋਣ ਤੋਂ ਰੋਕ ਲਗਾ ਸਕਦੇ ਹਾਂ।
20. ਜੇਕਰ ਅਸੀਂ final ਕੀਅ-ਵਰਡ ਨਾਲ ਵੇਰੀਏਬਲ ਨੂੰ ਡਿਕਲੇਅਰ ਕਰਦੇ ਹਾਂ ਤਾਂ ਅਸੀਂ ਇਸਦੇ ਮੁੱਲ ਨੂੰ ਬਦਲਣ ਤੋਂ ਰੋਕ ਸਕਦੇ ਹਾਂ।

ਅਭਿਆਸ

ਪ੍ਰਸ਼ਨ 1. ਬਹੁਪਸੰਦੀ ਪ੍ਰਸ਼ਨ :

i. ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਕਿਹੜੀ ਕਿਸਮ ਦੋ ਜਾਂ ਦੋ ਤੋਂ ਵੱਧ ਇਨਹੈਰੀਟੈਂਸ ਕਿਸਮਾਂ ਦਾ ਸੁਮੇਲ ਹੁੰਦੀ ਹੈ।
(ੳ) ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ (Single Inheritance)
(ਅ) ਮਲਟੀਲੇਵਲ ਇਨਹੈਰੀਟੈਂਸ (Multi-level Inheritance)
(ੲ) ਹਾਇਰਾਰਕੀਕਲ ਇਨਹੈਰੀਟੈਂਸ (Hierarchical Inheritance)
(ਸ) ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ (Hybrid Inheritance)

ii. ਜਾਵਾ ਵਿੱਚ ਕਲਾਸਾਂ ਉੱਪਰ ਕਿਸ ਕਿਸਮ ਦੀ ਇਨਹੈਰੀਟੈਂਸ ਦੀ ਇਜ਼ਾਜਤ ਨਹੀਂ ਹੁੰਦੀ।
(ੳ) ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ (Multiple Inheritance) (ਅ) ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ (Multi-level Inheritance)
(ੲ) ਸਿੰਗਲ ਇਨਹੈਰੀਟੈਂਸ (Single Inheritance) (ਸ) ਹਾਈਬ੍ਰਿਡ ਇਨਹੈਰੀਟੈਂਸ (Hybrid Inheritance)

iii. ਬੇਸ ਕਲਾਸ ਦੇ ਕੰਸਟਰਕਟਰ ਨੂੰ ਚਲਾਉਣ ਲਈ ਕਿਹੜੇ ਮੈਥਡ ਦੀ ਵਰਤੋਂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।
(ੳ) ਫਾਇਨਲ (final) (ਅ) ਸਟੈਟਿਕ (static)
(ੲ) ਸੁਪਰ (super) (ਸ) ਉਪਰੋਕਤ ਵਿੱਚੋਂ ਕੋਈ ਨਹੀਂ

iv. final ਕੀਅ-ਵਰਡ .... ਨਾਲ ਵਰਤਿਆ ਜਾ ਸਕਦਾ ਹੈ।
(ੳ) ਵੇਰੀਏਬਲ (Variables) (ਅ) ਕਲਾਸ (Classes)
(ੲ) ਮੈਥਡ (Methods) (ਸ) ਉਪਰੋਕਤ ਸਾਰੇ

v. ਯੂਜ਼ਰ ਤੋਂ ਕਿਸੇ ਵੀ ਐਲੀਮੈਂਟ ਦੀ ਅੰਦਰੂਨੀ ਬਣਤਰ/ਜਟੀਲਤਾ ਨੂੰ ਛੁਪਾਉਣ ਦੀ ਪ੍ਰਕਿਰਿਆ ਹੈ।
(ੳ) ਇਨਹੈਰੀਟੈਂਸ (Inheritance) (ਅ) ਓਵਰਰਾਈਡਿੰਗ (Overriding)
(ੲ) ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) (ਸ) ਇੰਟਰਫੇਸ (Interface)

2. ਸਹੀ ਜਾਂ ਗਲਤ ਦੱਸੋ :

i. final ਕਲਾਸ ਨੂੰ ਉਸਦੀ ਸਬਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਕਰਨਾ ਲਾਜ਼ਮੀ ਹੁੰਦਾ ਹੈ।

ii. ਐਬਸਟਰੈਕਟ ਕਲਾਸ ਦਾ ਆਬਜੈਕਟ ਸਿੱਧੇ ਤੌਰ ਤੇ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

iii. ਬੇਸ ਕਲਾਸ ਦੇ ਮੈਥਡਜ਼ ਨੂੰ ਡਰਾਇਵਡ ਕਲਾਸ ਵਿੱਚ ਓਵਰਰਾਈਡ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

iv. ਫੀਲਡਜ਼ ਅਤੇ ਮੈਥਡਜ਼ ਨੂੰ ਡਰਾਇਵਡ ਕਲਾਸ ਵਿੱਚ ਇਨਹੈਰੇਂਟ ਕੀਤਾ ਜਾ ਸਕਦਾ ਹੈ।

v. ਜਾਵਾ ਵਿੱਚ ਸਿਰਫ਼ interfaces ਦੀ ਮਦਦ ਨਾਲ ਹੀ ਮਲਟੀਪਲ ਇਨਹੈਰੀਟੈਂਸ ਲਾਗੂ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

ਪ੍ਰਸ਼ਨ 3. ਛੋਟੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਇਨਹੈਰੀਟੈਂਸ (inheritance) ਦੀ ਪਰਿਭਾਸ਼ਾ ਦਿਓ।

ii. ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ?

iii. super ਕੀਅ-ਵਰਡ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।

iv. ਮੈਥਡ ਓਵਰਰਾਈਡਿੰਗ (Method overriding) ਕੀ ਹੁੰਦੀ ਹੈ ?

v. final ਵੇਰੀਏਬਲ ਤੇ ਇੱਕ ਛੋਟਾ ਨੋਟ ਲਿਖੋ।

ਪ੍ਰਸ਼ਨ 4. ਵੱਡੇ ਉੱਤਰਾਂ ਵਾਲੇ ਪ੍ਰਸ਼ਨ :

i. ਇਨਹੈਰੀਟੈਂਸ (inheritance) ਕੀ ਹੁੰਦੀ ਹੈ ? ਇਨਹੈਰੀਟੈਂਸ ਦੀਆਂ ਕੋਈ ਤਿੰਨ ਕਿਸਮਾਂ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।

ii. ਐਬਸਟਰੈਕਸ਼ਨ (Abstraction) ਤੋਂ ਤੁਹਾਡਾ ਕੀ ਭਾਵ ਹੈ ? ਜਾਵਾ ਵਿੱਚ ਐਬਸਟਰੈਕਟ (Abstract) ਐਲੀਮੈਂਟਸ ਦੀ ਵਿਆਖਿਆ ਕਰੋ।

iii. final ਕੀਅ-ਵਰਡ ਦਾ ਕੀ ਪ੍ਰਯੋਗ ਹੈ ? ਉਦਾਹਰਨ ਦੇ ਨਾਲ ਫਾਈਨਲ ਕਲਾਸ ਅਤੇ ਮੈਥਡ ਸਮਝਾਓ।

iv. ਢੁਕਵੀਂ ਉਦਾਹਰਣ ਦੇ ਨਾਲ ਮਲਟੀਪਲ-ਇਨਹੈਰੀਟੈਂਸ (Multiple Inheritance) ਦੀ ਵਿਆਖਿਆ ਕਰੋ ?