8 ਸਟ੍ਰਿੰਗਸ ਅਤੇ ਰੈਪਰ ਕਲਾਸਾਂ (Strings & Wrapper Classes)

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)

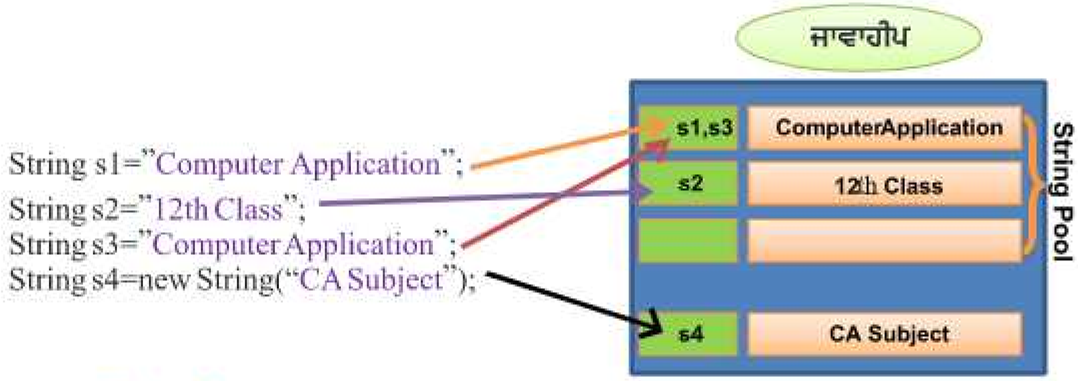
ਸਟ੍ਰਿੰਗਸ ਹਰ ਖੇਤਰ ਦਾ ਇੱਕ ਜ਼ਰੂਰੀ ਹਿੱਸਾ ਹੁੰਦੇ ਹਨ ਕਿਉਂਕਿ ਅਸੀਂ ਆਪਣੇ ਡੇਟਾ ਨੂੰ ਡਿਜ਼ੀਟਲ ਤੌਰ ਤੇ ਪ੍ਰੋਸੈੱਸ ਕਰਨ ਲਈ ਉਸਨੂੰ ਸਟ੍ਰਿੰਗ ਦੇ ਰੂਪ ਵਿੱਚ ਸਟੋਰ ਕਰ ਸਕਦੇ ਹਾਂ। ਇਸ ਲਈ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ, ਸਟ੍ਰਿੰਗ ਇੱਕ ਆਮ ਤੌਰ ਤੇ ਵਰਤੀ ਜਾਣ ਵਾਲੀ ਡਾਟਾ ਕਿਸਮ ਹੈ ਜੋ ਸਾਨੂੰ ਕਿਸੇ ਵੀ ਵਸਤੂ ਨਾਲ ਸੰਬੰਧਤ ਕੋਈ ਡਾਟਾ ਜਾਂ ਹਰ ਉਸ ਮੁੱਲ ਨੂੰ ਸਟੋਰ ਕਰਨ ਦੀ ਇਜਾਜ਼ਤ 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 ਆਊਟਪੁੱਟ 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

```
D:\JavaProg>javac CAProg9.java

D:\JavaProg>java CAProg9
Computer Application
Punjab School Education Board
SAS Nagar
Mohali

D:\JavaProg>
```

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

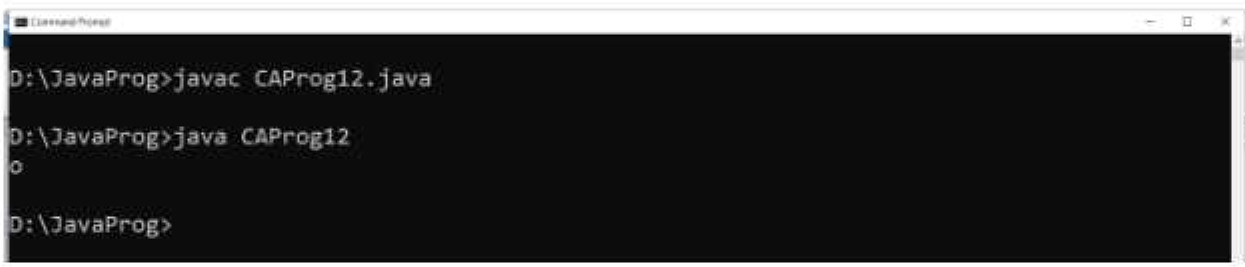
ਆਉਟਪੁੱਟ (Output) :

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



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 ॥ ਅਸੀਂ ਕਹਿ ਸਕਦੇ ਹਾਂ, ਇਹ ਮੈਥਡ ਕਿਸੇ ਵੀ ਮੌਜੂਦਾ ਸਟ੍ਰਿੰਗ ਨਾਲ ਇੱਕ ਦਿੱਤੇ ਗਏ ਸਟ੍ਰਿੰਗ ਨੂੰ ਜੋੜਦਾ ਹੈ। । ਇਹ ਮੇਥਡ 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 ਮੌਜੂਦਾ 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 ॥ 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 ਨੇ 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 :

ComputerApplicationPSEB

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 ਐਗਜ਼ੀਕਿਊਟ 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

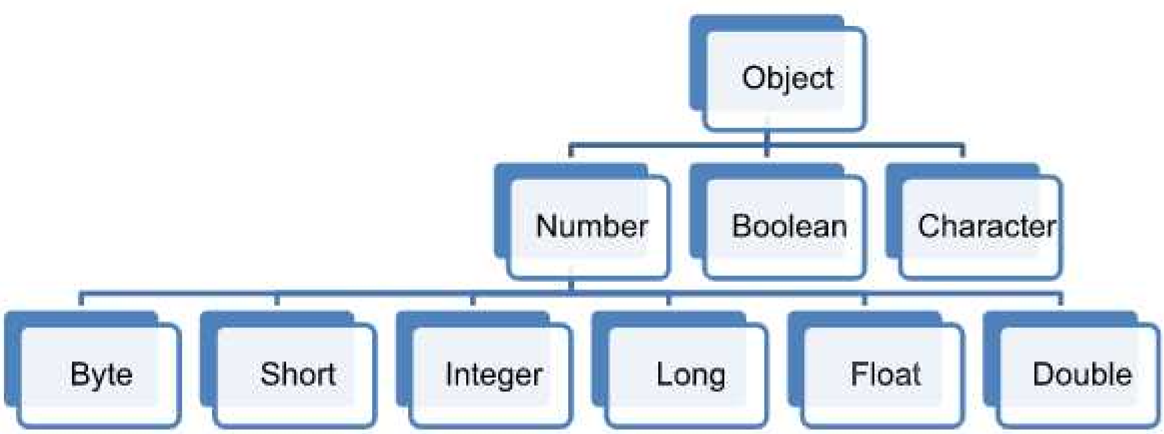
8.3 ਜਾਵਾ ਰੈਪਰ ਕਲਾਸਾਂ (JAVA WRAPPER CLASSES)

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 ਰੂਪਾਂਤਰਣ (automatic conversion) ਨੂੰ ਆਟੋਬਾਕਸਿੰਗ (autoboxing) ਅਤੇ ਇਸ ਤੋਂ ਉਲਟ ਰੂਪਾਂਤਰਣ 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 ਸਿਰਫ 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॥