Unit 6

String

Date 12/13
Page

Introduction To String

↳ A string is an object that represents a sequence of characters.

↳ Java.lang.String class is used to create a string object.

↳ Double quotes (" ") is used to represent string constant.

↳ For example: "Ram" is a string containing a sequence of character 'R', 'a' and 'm'.

String Declaration:

String can be declared by two ways.
① By string literal
② By new keyword

① By string literal:

↳ It is direct method to create string using

Syntax:

```
String Stringname = "string";
```

Example:

```
String s1 = "Ram";
String s2 = "welcome";
```

// Java program to create string by string literal public class StringTest

```
public class StringTest
{
    public static void main(String [] args)
    {
        String s1 = "Ram"; // creating string
        String s2 = "Hello Java";
        System.out.println("First string is = "+s1);
        System.out.println("Second string is = "+s2);
    }
}
```

String declaration

1. By string literal
2. By new keyword

↳ The new keyword is used to create objects including String

↳ It is less common and typically done when we want to create string explicity.

Syntax:

```
String str = new String ("String");
```

Example:-

```
String str = new String ("Hello Java");
```

Here,

a new string object str is created with the value "HelloJava"
The new keyword is followed by the String constructor which takes a string literal "HelloJava" as its argument.

```
// Program to create String using new keyword

public class StringTest
{
  public static void main(String [] args)
{
  String s1 = "Java"; // creating string by
                        string literal
  String s2 = new String ("Programming");
                // creating string using new keyword
  System.out.println ("First String : "+s1);
  System.out.println ("Second String : "+s2);
}
}
```

String manipulation methods / functions

↳ Java.lang.String class includes many methods for manipulating on String.

Some methods are:

① length() method:

↳ Used to find length of given string
↳ It returns no of characters included within string.

Syntax:

```
String_variable.length();
```

//Example to demonstrate length() method:

```
import java.lang.String;
class Stringlength
{
  public static void main(String[]args)
  {
    String str="welcome"; //creating String
    System.out.println("String is="+str);
    int len = str.length(); //method calling
    System.out.println("The length of String ="+len);
  }
}
```

② concat() method:

↳ This method join or concat two string and form a single string.

Syntax:

```
Str.Var1.concat(Str_Var2);
```

// Example to demonstrate length() method.

```
import java.lang.String;
public class StringLength
{
    public static void main(String [] args)
    {
        String s1 = "Ram" ; // creating string
        String s2 = "Hari";
        System.out.println("First String:"+s1);
        System.out.println("Second string:"+s2);
        String s3 = s1.concat(s2);
        System.out.println("Concatenated string:"+s3);
    }
}
```

③ equals()

↳ This method is used to make comparision between two strings to check whether strings are equal or not.

↳ It returns logical value either true or false.

Syntax:

```
str1.equals(str2);
```

Example:

```
import java.~~util~~lang.String;
public class
{
  public static void main(String [ ] args)
{
    String s1 = "Java";
    String s2 = "Java";
    String s3 = "Programming";
    boolean result = s1.equals(s2);          Strings
    System.out.println("First and second are
                        equal:"+result);
    boolean result = s1.equals(3);
    System.out.println("First and third strings
                        are equal:"+result);
  }
}
```

StringBuffer class:

↳ StringBuffer is a class included within java.io Package.

↳ It is used to create mutable (changable) string objects.

↳ It is same as ~~String~~ class except it is mutable i.e can be modified the content of object after it has been created.

↳ It provides an alternative to the immutable String class.

Some methods of StringBuffer class are:-

① append()
② insert()
③ replace()
④ delete()
⑤ reverse()

① append() method

↳ Used to concatenate (join) the given argument with string.

Syntax:

```
obj.append("String");
```

Example

```
import java.io.*;
public class StringTest
{
  public static void main(String[] Jargs)
  {
    StringBuffer sb=new StringBuffer("Hello");
    sb.append("Java");
    System.out.println("New string ="+sb);
  }
}
```

② insert()

↳ This method inserts the given string with this string at the given position.

Syntax:

```
sb.insert(position, "new string");
```

Here, sb → String object that holds original string.

position → integer number that represents Position of character in string.

Example:

```
import java.io.*;
class program
{
public static void main(String args[])
{
StringBuffer sb = new StringBuffer("Hello");
sb.insert(1, "java"); // original string is changed
System.out.println("New string :"+sb);
}
}
```

o/p :- HjavaeIlo

③ Replace()

↳ This method replaces the given string from the specified begin index and endIndex - 1.

Syntax:

```
replace(beginIndex, endIndex-1, "string");
```

Example:

```
import java.io.*;
class programs
{
public static void main(String[]args)
{
StringBuffer sb = new StringBuffer("Hello");
sb.replace(1, 3, "Java");
System.out.println(sb);
}
}
```

o/p HJavaO

4. delete() method

↳ This method delete the string from specified beginIndex to endIndex - 1.

Syntax:

```
delete(beginIndex, endIndex - 1)
```

Example:

```
import java.io.*;
class Program
{
public static void main(String args[])
{
StringBuffer sb = new StringBuffer("Hello");
sb.delete(1, 3);
System.out.println(sb);
}
}
```

5. reverse() method

↳ This method reverse the current String.

Syntax:

```
sb.reverse();
```

where, sb is stringBuffer string object

Example:

```
import java.io.*;
class Reverse
{
public static void main(String args[])
{
StringBuffer sb = new StringBuffer("Hello");
sb.reverse()
System.out.println(sb);
}
}
```

StringBuilder class

↳ It is ~~mutt~~ mutable class located in java.lang package used to manipulate string.

↳ It is alternative to string class.

↳ Similar to StringBuffer class but differs on the basic of synchronization.

Consist of following methods:

① append() method
② insert() method
③ delete() method
④ replace() method
⑤ reverse() method

① append() method

↳ Used to concatenate (Join) the given argument with string.

Syntax:

```
obj.append("String");
```

Example:

```
import java.lang
public class Append
{
public static void main(String args[])
```

```
{
    StringBuilder sb = new StringBuilder
                                ("Hello");
        sb.append ("Java");
    System.out.println(sb);
}
}
```

O/P

② insert() method:

↳ This method insert the given string with this String at the given position.

Syntax:

```
sb.insert(position, "new string");
```

Here, sb → String object that holds original string.

position → integer number that represents position of character in string.

Example:

```
import java.lang
class Insert
{
public static void main(String args[])
{
StringBuilder sb=new StringBuilder("Hello");
sb.insert(1,"Java");
System.out.println("New string :"+sb);
}
}
```

O/P

③ delete() method

↳ This method deletes the string from specified beginIndex to endIndex.

Syntax:

```
delete(beingIndex, endIndex-1)
```

Example:

```
import java.lang
class Delete
{
public static void main(String args[])
{
StringBuilder sb = new StringBuilder("Hello");
sb.delete(1, 3);
System.out.println(sb);
}
}
```

O/P :-

④ Replace() method

↳ This method replaces the given string from the specified beginIndex and endIndex.

Syntax:

```
replace(beingIndex, endIndex, "String");
```

Example:

```
import java.lang
class Replace
{
```

```
public static void main(String args[])
{
StringBuilder sb = new StringBuilder("Hello");
    sb.replace(1, 3, "Java");
    System.out.println(sb);
}
}
```

o/p

⑤ reverse() method

↳ This method reverse the current string.

Syntax:

```
sb.reverse();
```

Example:

```
import java.lang
class Reverse
{
public static void main(String[] args)
{
StringBuilder sb = new StringBuilder("Hello");
```

```
sb.reverse()
System.out.println(sb);
}
}
```

O/P