String Inbuilt methods

substring()

public String substring(int startIndex): This method returns new String object containing the substring of the given string from specified startIndex (inclusive).

public String substring(int startIndex, int endIndex): This method returns new String object containing the substring of the given string from specified startIndex to endIndex.

substring() - Example

```
public class TestSubstring
{
    public static void main(String args[]){
        String s = "Hardik Pandya";
        System.out.println(s.substring(7));
        System.out.println(s.substring(0,7));
    }
}
```

Output:

```
Pandya
Hardik
```

replace()

The **java string replace()** method returns a string replacing all the old char or CharSequence to new char or CharSequence.

```
public class ReplaceExample1
{
   public static void main(String args[]){
      String s1 = "Negative thoughts";
      String replaceString = s1.replace('e','a');
      System.out.println(replaceString);
   }
}
```

Output:

```
Nagativa thoughts
```

replace()

```
public class ReplaceExample1
{
   public static void main(String args[]){
       String s1 = "Negative thoughts";
  String replaceString = s1.replace("Negative","Positive");
       System.out.println(replaceString);
   }
}
```

Output:

```
Positive thoughts
```

contains()

The **java string contains()** method searches the sequence of characters in this string. It returns *true* if sequence of char values are found in this string otherwise returns *false*.

contains() - Example

```
public class ContainsExampleMain1 {
   public static void main(String args[]){
        String name = "what do you know about me";
            System.out.println(name.contains("do you know"));
        System.out.println(name.contains("about"));
        System.out.println(name.contains("hello"));
   }
}
```

Output:

```
true
true
false
```

contains() - Example

```
public class ContainsExampleMain2 {
    public static void main(String[] args) {
        String str = "Hello T#E#C#H#N#O#S#E#R#V#E@2.0 readers";
        boolean isContains = str.contains(“T#E#C#H#N#O#S#E#R#V#E");
        System.out.println(isContains);
        System.out.println(str.contains(“TECHNOSERVE"));
    }
}
```

Output:

```
true
false
```

contains() - Example

```
public class ContainsExampleMain3 {
    public static void main(String[] args) {
        String str = "To learn Java visit abc.in";
        if(str.contains(“abc.in.com"))
        {
                    System.out.println("This string contains abc.in");
        }
        else
            System.out.println("Result not found");
    }
}
```

Output:

```
Result not found
```

equalsIgnoreCase()

The **String equalsIgnoreCase()** method compares the two given strings on the basis of content of the string irrespective of case of the string.

It is like equals() method but doesn't check case.

equalsIgnoreCase() - Example

```
public class EqualsIgnoreCaseExample{
  public static void main(String args[]){
     String s1 = "The walking Dead";
     String s2 = "The walking Dead";
     String s3 = "THE WALKING DEAD";
     String s4 = "The WEST WEEDS";
     System.out.println(s1.equalsIgnoreCase(s2));
     System.out.println(s1.equalsIgnoreCase(s3));
     System.out.println(s1.equalsIgnoreCase(s4));
  }
}
```

Output:

```
true
true
false
```

indexOf()

The **java string indexOf()** method returns index of given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

int indexOf(int ch) - returns index position for the given char value

int indexOf(int ch, int fromIndex) - returns index position for the given char value and from index

int indexOf(String substring) - returns index position for the given substring

int indexOf(String substring, int fromIndex) - returns index position for the given substring and from index

indexOf() -Example

```
public class IndexOfExample{
    public static void main(String args[]){
        String s1 = "This is the world";
        int index4 = s1.indexOf('s');
        System.out.println(index4);
        int index1 = s1.indexOf("is");
        int index2 = s1.indexOf("world");
        System.out.println(index1);
        System.out.println(index2);
        int index3 = s1.indexOf("is",4);
        System.out.println(index3);
    }
}
```

Output:

```
3
2
12
5
```

LastIndexOf()

The **java string lastIndexOf()** method returns last index of the given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

1. **int lastIndexOf(int ch)** - returns last index position for the given char value
2. **int lastIndexOf(int ch, int fromIndex)** - returns last index position for the given char value and from index
3. **int lastIndexOf(String substring)** - returns last index position for the given substring
4. **int lastIndexOf(String substring, int fromIndex)** - returns last index position for the given substring and from index

LastIndexOf() - Example

```
public class LastIndexOfExample{
    public static void main(String args[]){
        String s1 = "this is the world";
        int index1 = s1.lastIndexOf('s');
        System.out.println(index1);
    }
}
```

```
Output:
6
```

length()

- The java String length() method returns the length of the string
- The length of the string will be an integer

```
class Main{
   public static void main(String args[]){
      String str = "Programming";
      System.out.println(str.length());
      String s = "Let's continue";
      System.out.println(s.length());
   }
}
```

```
11
14
```

charAt()

- The java String charAt() method returns a char value at the given index number
- The index number starts from 0 and goes up to (n-1), where n is length of the string.

```
class Main{
    public static void main(String args[]){
        String str = "Java";
        char ch = str.charAt(2);
        System.out.println(ch);
        System.out.println(str.charAt(3));
        System.out.println(str.charAt(4));
        System.out.println(str.charAt(-1));
    }
}
```

```
v
a
Exception
```

toUpperCase()

- The java String toUpperCase() method converts all the lowercase characters of a string into uppercase
- It will not alter the already existing uppercase characters in the string.

```
class Main{
   public static void main(String args[]){
      String str = "Java is love";
      System.out.println(str.toUpperCase());
}
}
```

```
Output: JAVA IS LOVE
```

toLowerCase()

- The java String toLowerCase() method converts all the uppercase characters of a string into lowercase
- It will not alter the already existing lowercase characters in the string.

```
class Main{
   public static void main(String args[]){
      String str = "JAVA IS LOVE";
   System.out.println(str.toLowerCase());
}
}
```

Output: java is love

concat()

- The Java string concat() method allows you to join two strings.
- This method returns a string with the value of the string passed into the method is appended to the end of the string.

```
class Main{
  public static void main(String args[]){
        String s1 = "Nice”;
        String s2 = "Day“;
        System.out.println(s1.concat(s2));
}
}
```

Output: Nice Day

+ operator

+ operator is also used to concatenate strings

```
class Main{
    public static void main(String[] args) {
        String s = "Are", t = "you", u = "ready";
        System.out.println(s + t + u);
        System.out.println(s.concat(t));
    }
}
```

Output:

```
Areyouready
Areyou
```

Difference between + operator & concat() method

Number of arguments the concat() method and + operator takes:

concat() method takes only one argument of string and concatenate it with other string.

+ operator takes any number of arguments and concatenates all the strings.

trim()

The Java string ***trim*() method** is used to eliminate leading and trailing blank spaces.
The Unicode value of 'space' character is 20. This method checks the Unicode value before and after the string. If it exists, it removes the spaces and returns the remaining string.

```
public class StringTrimExample{
    public static void main(String args[]){
        String s1 = "  Game of  ";
        System.out.println(s1 + "thrones");
        System.out.println(s1.trim() + "thrones");
    }
}
```

Output:

```
  Game of  thrones
Game ofthrones
```

toCharArray()

The **java string toCharArray()** method converts this string into character array. It returns a newly created character array, its length is similar to this string and its contents are initialized with the characters of this string.

```
public class StringToCharArrayExample{
    public static void main(String args[]){
        String s1= "Twilight Saga";
        char[] ch = s1.toCharArray();
        for(int i = 0; i < ch.length; i++){
            System.out.print(ch[i]);
        }
    }
}
```

Output:

```
Twilight Saga
```

startsWith()

```
public boolean startsWith(String prefix)
public boolean startsWith(String prefix, int offset)

public class StartsWithExample{
   public static void main(String args[]){
         String s1 = "You must be the change you wish to see in the world";
         System.out.println(s1.startsWith("Y"));
         System.out.println(s1.startsWith("You must"));
         System.out.println(s1.startsWith("a"));
         System.out.println(s1.startsWith("o", 1));
   }
}
```

Output:

```
true
true
false
true
```

endsWith()

The **java string endsWith()** method checks if this string ends with given suffix. It returns true if this string ends with given suffix else returns false.

```
public class EndsWithExample{
    public static void main(String args[]){
        String s1 = "Beauty is in the eye of the beholder";
        System.out.println(s1.endsWith("r"));
        System.out.println(s1.endsWith("holder"));
        System.out.println(s1.endsWith("eye"));
    }
}
```

Output:

```
true
true
false
```

format()

```
public class FormatExample{
    public static void main(String args[]){
        String name = "CSK";
        String sf1 = String.format("%s",name);
        String sf2 = String.format("%f",32.33434);
        String sf3 = String.format("%16.12f",32.33434);
        System.out.println(sf1);
        System.out.println(sf2);
        System.out.println(sf3);
    }
}
```

Output:

```
CSK
32.334340
 32.334340000000 //returns 12 char fractional part filling with 0. The
output should contain 16 charcaters, here it is 15, one space is printed
```

isEmpty()

- The **java string isEmpty()** method checks if this string is empty or not.
- True is returned if string is empty otherwise it returns false.

```
public class IsEmptyExample{
    public static void main(String args[]){
        String s1 = "";
        String s2 = "java";
        System.out.println(s1.isEmpty());
        System.out.println(s2.isEmpty());
    }
}
```

Output:

```
true
false
```

join()

```
public class StringJoinExample{
  public static void main(String args[]){
    String joinString1=String.join("","welcome","to","jurassic","world");
    System.out.println(joinString1);
  }
}
```

Output:

```
welcometojurassicworld
```

valueOf()

The java string **valueOf() method** converts different types of values into string.
By the help of string valueOf() method, you can convert primitive type to string and object to string.

```
public class StringValueOfExample{
    public static void main(String args[]){
        int value = 30;
        String s1 = String.valueOf(value);
        System.out.println(s1 + 10);
    }
}
```

Output:

```
3010
```

Predict the output

```
public class IsEmptyExample2 {
    public static void main(String[] args) {
        String s1 = "";
        String s2 = "Wonderla";
        if(s1.length()==0 || s1.isEmpty())
            System.out.println("s1 is empty");
        else System.out.println("s1");
        if(s2.length() == 0 || s2.isEmpty())
            System.out.println("s2 is empty");
        else System.out.println(s2);
    }
}
```

Output:

```
s1 is empty
Wonderla
```

Predict the output

```
class Main{
  public static void main(String args[]){
    String s = 50 + 50 + "error" + 50;
    System.out.println(s);
  }
}
```

After a string literal, all the + will be treated as string concatenation operator.

Output:

```
100error50
```

Predict the output

```
class Main{
    public static void main(String args[]){
        String s = "Apple";
        int a = 10;
        System.out.println(s + a);
        System.out.println(s.concat(a));
    }
}
```

Output:

```
Error
prog.java:6: error: incompatible types: int cannot be converted
to String
System.out.println(s.concat(a));
```

Predict the output

```
class Main{
   public static void main(String args[]){
      String s = "Apple";
      String r = null;
      System.out.println(s + r);
      System.out.println(s.concat(r));
   }
}
```

Output:

```
Applenull
Exception in thread "main" java.lang.NullPointerException at
java.lang.String.
concat(String.java:2027) at Main.main(File.java:6)
```

Predict the output

```
class Main{
   public static void main(String args[]){
     String s = "Great", t = "H";
     String u = s.concat(t);
     if(u == s){
         System.out.println("Same");  }
     else{
         System.out.println("Not same");   }
     String e = s + t;
     if (e == s){
         System.out.println("Same");
     }
     else{
         System.out.println("Not same"); }  }  }
```

Output:

```
Not Same
Not Same
```

Predict the output

```
public class IndexOfExample2 {
    public static void main(String[] args) {
        String s1 = "This is the example";
        int index = s1.indexOf("example", 10);
        System.out.println(index);
        index = s1.indexOf("example", 20);
        System.out.println(index);
    }
}
```

Output:

```
12
-1
```

Predict the output

```
public class IndexOfExample3 {
    public static void main(String[] args) {
        String s1 = "This is indexOf method";
        int index = s1.indexOf('O', 12);
        System.out.println(index);
    }
}
```

Output:

```
13
```

Predict the output

```
public class StringValueOfExample{
   public static void main(String[] args) {
      float f  = 10.05645f;
      double d = 10.02;
      String s1 = String.valueOf(f);
      String s2 = String.valueOf(d);
      System.out.println(s1);
      System.out.println(s2);
   }
}
```

Output:

```
10.05645
10.02
```

Predict the output

```
public class LastIndexOfExample{
    public static void main(String args[]){
        String s1 = "this is the world";
        int index1 = s1.lastIndexOf('s', 5);
        System.out.println(index1);
    }
}
```

Output:

```
3
```

Predict the output

```
public class LastIndexOfExample{
    public static void main(String[] args) {
        String str = "This is last index of example";
        int index = str.lastIndexOf("last");
        System.out.println(index);
        index = str.lastIndexOf("of", 25);
        System.out.println(index);
        index = str.lastIndexOf("of", 10);
        System.out.println(index);
    }
}
```

Output:

```
8
19
-1
```

Predict the output

```
public class StringToCharArrayExample2 {
    public static void main(String[] args) {
        String s1 = "Welcome to Jumanji";
        char[] ch = s1.toCharArray();
        int len = ch.length;
        System.out.println(len);
        for (int i = 0; i < len; i++) {
            System.out.print(ch[i]);
        }
    }
}
```

Output:

```
18
Welcome to Jumanji
```