String Operations:

In general, in all the programming languages we are able to perform String operations like

1. Finding length of a String
2. Concatenation
3. Trim
4. Converting String from lower case to upper case.
5. Converting String from upper case to lower case
6. Sub strings

To perform all the above String operations Java has provided the following predefined library.

1. java.lang.String
2. java.lang.StringBuffer
3. java.lang.StringBuilder
4. java.util.StringTokenizer

Q) What are the differences between String and StringBuffer?

Ans:
----

1. String class objects are immutable objects , where immutable objects are the java objects which are not allowing modifications on their content, if we are trying to perform modifications in immutable objects data then data is allowed for modifications but the resultant modified data will not be stored back in the original object and the resultant modified data will be stored by creating a new Object.

   StringBuffer class objects are mutable objects, where mutable objects are the java objects which are able to allow modifications directly on their content.
2. String class object length is fixed.

   StringBuffer class object length is variable.

Q) What are the differences between StringBuffer and StringBuilder?
—--
Ans:
—----

1. StringBuffer was introduced in the JDK1.0 version.
   StringBuilder was introduced in JDK1.5 version.

2. StringBuffer is a synchronized resource.
   StringBuilder is not a synchronized resource.

3. StringBuffer allows only one thread at a time.
   StringBuilder allows more than one thread at a time.

4. StringBuffer follows sequential execution.
   StringBuilder follows parallel execution.

5. StringBuffer is able to increase application execution time.
   StringBuilder is able to reduce application execution time.

6. StringBuffer is able to provide less performance to the applications.
   StringBuilder is able to provide more performance to the applications.

7. StringBuffer is giving guarantees for Data Consistency.
   StringBuilder is not giving guarantee for Data Consistency.

8. StringBuffer is a threadsafe resource.
   StringBuilder is not a threadsafe resource.

Q) What is String Tokenization and how is it possible to perform String Tokenization in java applications?
—---
Ans:
—---
The process of dividing a string into number tokens is called “String Tokenization”.

To perform StringTokenization in java applications JAVA has provided a predefined class in the form of “java.util.StringTokenizer”.

To create a StringTokenizer class object we have to use the following constructors.

1. Public StringTokenizer(String str):

It is able to create StringTokenizer class objects by performing String tokenization, It is able to perform String Tokenization on the basis of space delimiter.

2. public StringTokenizer(String str, String regExpr)
   It is able to create StringTokenizer class Objects by performing String Tokenization on the basis of the provided regular Expression[Delimiter].

   StringTokenizer st = new StringTokenizer("Durga Software Solutions");

To get the number of tokens from the StringTokenizer object we have to use the following method.

    public int countTokens()

To read elements from the StringTokenizer object we have to use the following steps for each and every token.

1. Check whether more tokens are available or not from the current position of the cursor.

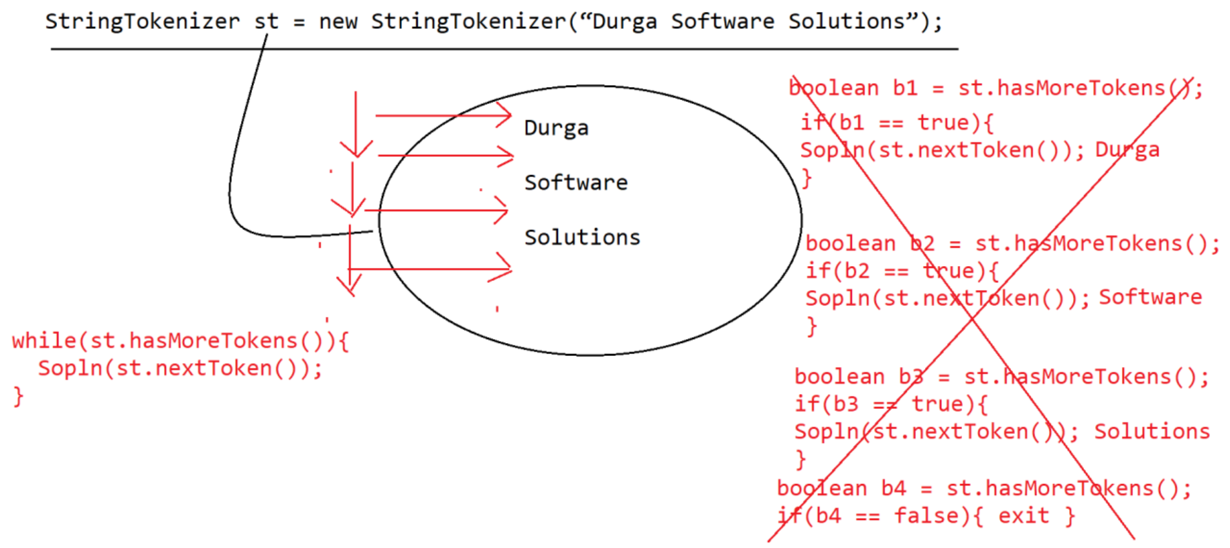
   public boolean hasMoreTokens()
   It will return true value if the next token is available .
   It will return false value if no more tokens are available.

2. If the next Token is available then read the next token and move the cursor to the next position in the StringTokenizer object.
   public String nextToken()

EX:
```
import java.util.*;
class Test{
      public static void main(String[] args){
            StringTokenizer st = new StringTokenizer("Durga Software 
Solution");
            System.out.println("No Of Tokens   : "+st.countTokens());
            while(st.hasMoreTokens()){
                  System.out.println(st.nextToken());
            }
      }
}
```

EX:
```
import java.util.*;
```

```
class Test{
      public static void main(String[] args){
            StringTokenizer st = new StringTokenizer("12-10-1996", "-");
            System.out.println("No Of Tokens   : "+st.countTokens());
            while(st.hasMoreTokens()){
                  System.out.println(st.nextToken());
            }
      }
}
```



```
StringTokenizer st = new StringTokenizer("Durga Software Solutions");
```

```
while(st.hasMoreTokens()){
  Sopln(st.nextToken());
}
```

```
boolean b1 = st.hasMoreTokens();
 if(b1 == true){
 Sopln(st.nextToken()); Durga
 }

boolean b2 = st.hasMoreTokens();
if(b2 == true){
Sopln(st.nextToken()); Software
}

boolean b3 = st.hasMoreTokens();
if(b3 == true){
Sopln(st.nextToken()); Solutions
}
boolean b4 = st.hasMoreTokens();
if(b4 == false){ exit }
```

String Class Library:

String is the collection of characters, digits, special symbols,.... Enclosed with “”.

In Java / J2EE applications , only the String data type is able to represent any type of data.

To represent Strings in java, Java has provided a predefined class in the form of java.lang.String .

To create String class objects, Java has provided the following constructors.

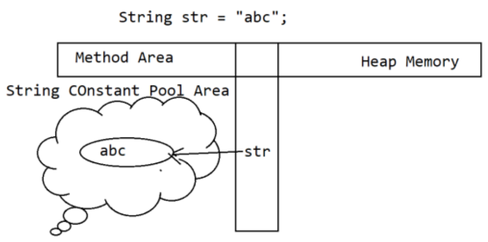
1. public String():
   It can be used to create a String class object without the data.
   EX: String str = new String();

2. public String(String data):
   It can be used to create a String class object with the specified data.
   EX:
```
public class Main {
    public static void main(String[] args) {
        String str = new String("abc");
        System.out.println(str);
    }
}
```
   OP: abc

   When we pass String class object reference variable as parameter to System.out.println() method then JVM will access toString() over the provided String class reference variable. In String class, the Object class provided toString() method was overridden in such a way that to return a string contains the content of the String class object instead of returning String object reference value.

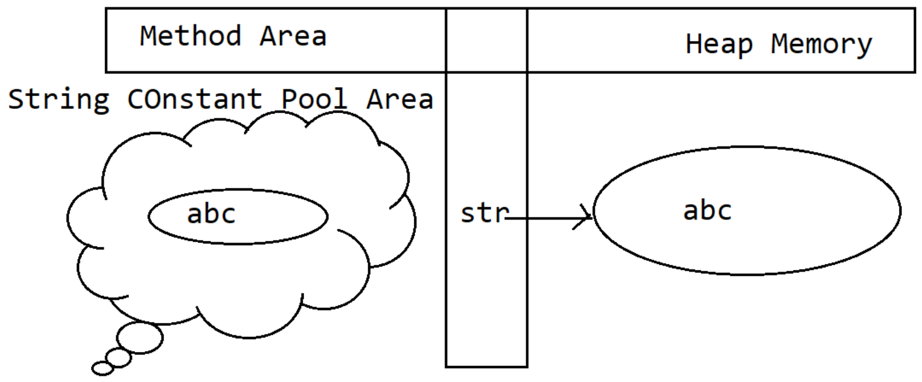
Q)What is the difference between the following two statements?
1. String str = "abc";
2. String str = new String("abc");

Ans:
----
1. To create a String object if we use the first statement then the String class object will be created in "String Constant Pool Area" in Method Area, it doesn't allow Garbage Collector to destroy these String objects, These String objects will be destroyed when java application execution is terminated.



To create a String class object if we use a second statement then String object will be created in both String Constant pool area as per “abc” and in Heap memory as per “new” keyword, where ‘str’ reference variable is able to refer Heap Memory provided String object, here Heap Memory allows Garbage Collector to destroy objects.

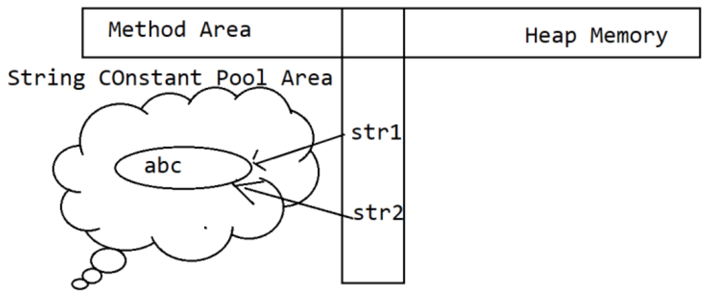
```
String str =  new String("abc");
```



2. String Constant Pool Area objects are permanent.
   Heap Memory provided String objects are temporary.

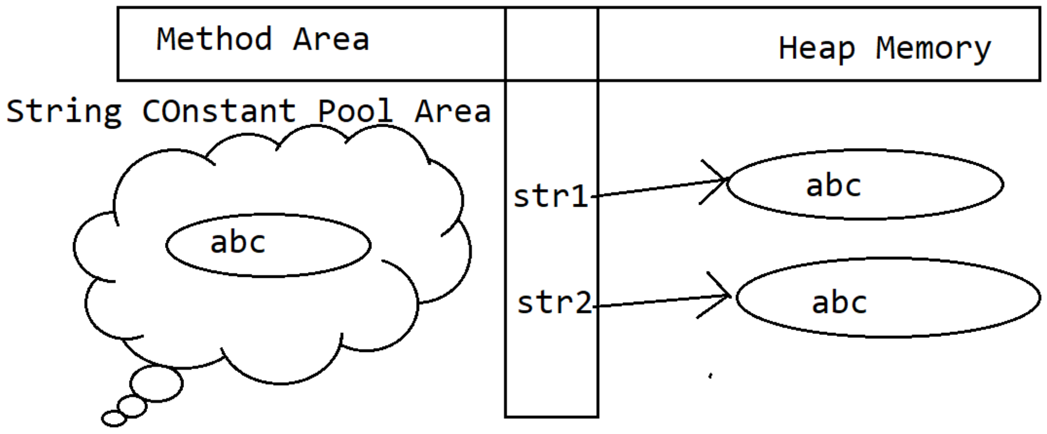
3. String Constant pool area objects are reusable.

```
String str1 = "abc";
String str2 = "abc";
```



Heap memory provided String objects are not reusable.

```
String str1 = new String("abc");
String str2 = new String("abc");
```



```
public class Main {
    public static void main(String[] args) {
        String str1 = "abc";
        String str2 = "abc";
        System.out.println(str1 == str2);// true

        String str3 = new String("abc");
        String str4 = new String("abc");
        System.out.println(str3 == str4);//false
    }
}
```

```
Op:
true
false
```

3. public String(byte[] b):
It is able to convert data from byte[] to the equivalent String object.
EX:

```
public class Main {
    public static void main(String[] args) {
        byte[] b = {65, 66, 67, 68, 69, 70};
        String str = new String(b);
        System.out.println(str);
    }
}
```
OP:

```
ABCDEF
```

4. public String(byte[] b, int startIndex, int numberOfElements):
It can be used to create a String class object with the String equivalent of the specified byte[] which starts from the specified startIndex and up to the specified number of elements.
EX:

```
public class Main {
    public static void main(String[] args) {
        byte[] b = {65, 66, 67, 68, 69, 70, 71, 72};
        String str = new String(b,2, 3);
        System.out.println(str);
    }
}
```
OP:

```
CDE
```

5. public String(char[] ch):
It can be used to convert data from char[] to String.
It can be used to create a String class object with the String equivalent of the specified char[] as data.
EX:

```
public class Main {
    public static void main(String[] args) {
        char[] ch = {'A', 'B', 'C', 'D', 'E', 'F'};
        String str = new String(ch);
        System.out.println(str);
    }
}
```
Op:

```
ABCDEF
```

6. public String(char[] ch, int startIndex, int numberOfElements):

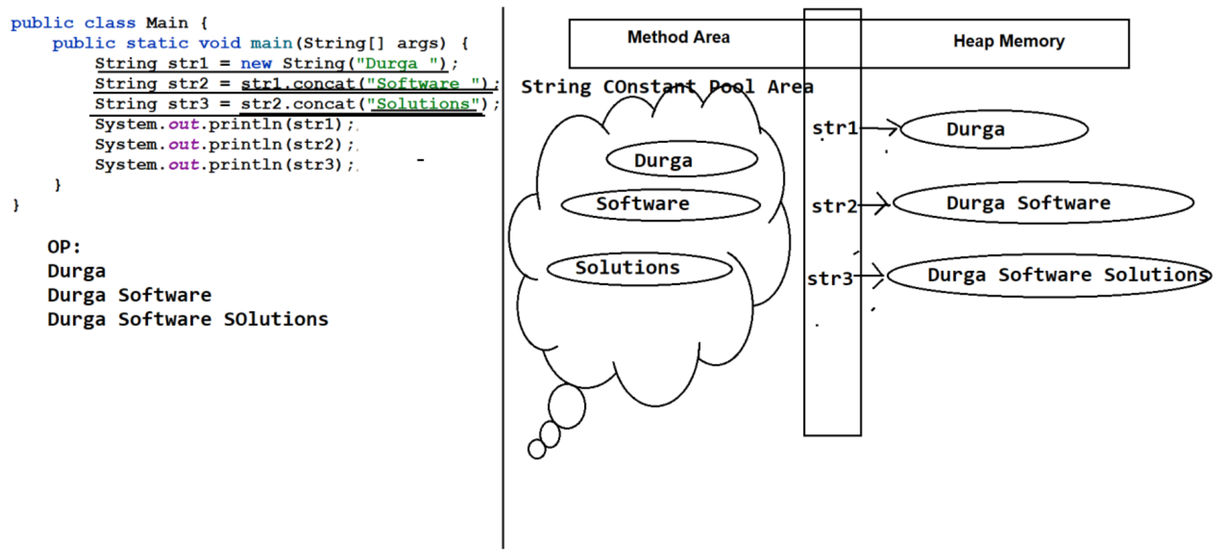
It can be used to create a String class object with the String equivalent of the specified char[] which starts from the specified startIndex and up to the specified number of Elements.
EX:
```
public class Main {
   public static void main(String[] args) {
        char[] ch = {'A', 'B', 'C', 'D', 'E', 'F'};
        String str = new String(ch,3,3);
        System.out.println(str);
    }
}
```
OP:
```
DEF
```

String class Methods:
----
1. public int length():
It will count the number of elements in a String and it will return the count value that is the length of the String.
EX:
```
public class Main {
   public static void main(String[] args) {
       String str1 = new String("Durga Software Solutions");
       System.out.println(str1+"   :     "+str1.length());
       String str2 = new String("   Durga Software Solutions   ");
       System.out.println(str2+"   :    "+str2.length());
       String str3 = new String("   ");
       System.out.println(str3+"   :   "+str3.length());
   }
}
```
OP:
```
Durga Software Solutions   :     24
   Durga Software Solutions      :    30
      :   3
```

2. public String concat(String data):
It can be used to append the specified data to the String object content in an immutable manner.

EX:

```
public class Main {
   public static void main(String[] args) {
        String str1 = new String("Durga ");
        String str2 = str1.concat("Software ");
        String str3 = str2.concat("Solutions");
        System.out.println(str1);// Durga
        System.out.println(str2);// Durga Software
        System.out.println(str3);// Durga Software
Solutions
    }
}
```



```
public class Main {
    public static void main(String[] args) {
        String str1 = new String("Durga ");
        String str2 = str1.concat("Software ");
        String str3 = str2.concat("Solutions");
        System.out.println(str1);
        System.out.println(str2);
        System.out.println(str3);
    }
}
```

```
OP:
Durga
Durga Software
Durga Software SOlutions
```

In Java applications, we are able to perform String concatenation by using ‘+’ operator also.

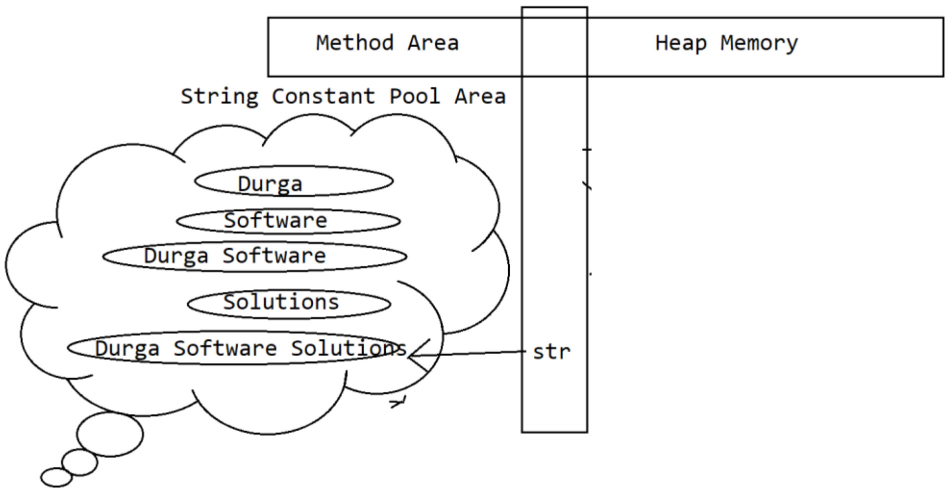
EX:

```
public class Main {
   public static void main(String[] args) {
        String str = "Durga "+"Software "+"Solutions";
        System.out.println(str);
    }
}
```

OP:

```
Durga Software Solutions
```

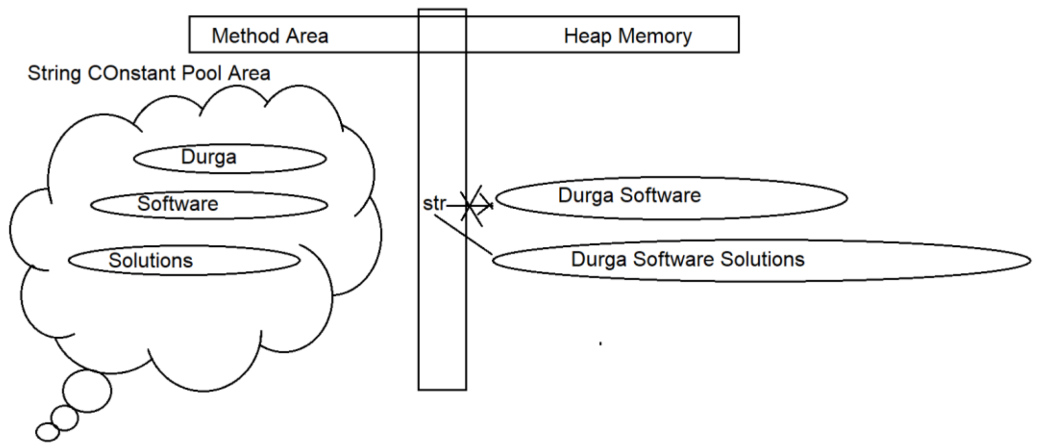
String str = "Durga "+"Software "+"Solutions";



```
public class Main {
   public static void main(String[] args) {
       String str1 = "Durga "+"Software "+"Solutions";
       String str2 = "Durga Software Solutions";
       System.out.println(str1 == str2);
   }
}
```
OP:
true

EX:
```
public class Main {
   public static void main(String[] args) {
       String str = "Durga ".concat("Software 
").concat("Solutions");
       System.out.println(str);
   }
}
```

```
String str = "Durga ".concat("Software ").concat("Solutions");
```



3. public boolean equals(Object obj):

It can be used to compare two String objects whose contents are the same or not.

EX:

```
public class Main {
   public static void main(String[] args) {
       String str1 = new String("abc");
       String str2 = new String("abc");
       System.out.println(str1.equals(str2));

   }
}
```

OP:

```
true
```

Q)What is the difference between == operator and equals() method?

Ans:

Basically, == is a comparison operator, it will check whether the provided two operand values are the same or not, where the provided operands may be normal primitive variables or Object reference variables.

Initially, equals() method was defined in java.lang.Object class, in java.lang.Object class equals() method was defined in such a way that to compare two object reference values instead of two Objects contents.

Object class equals() ----> References Comparison.

As per the String class requirement, java.lang.String class has overridden the Object class provided equals() method in such a way that to compare two String objects contents instead of two String objects reference values.

String class equals() ----> Content comparison

EX:
```java
class A{

}
public class Main {
   public static void main(String[] args) {
       int i = 10;
       int j = 20;

       A a1 = new A();
       A a2 = new A();

       String str1 = new String("abc");
       String str2 = new String("abc");

       System.out.println(i == j);// false
       System.out.println(a1 == a2);// false
       System.out.println(str1 == str2);// false
       System.out.println();

       System.out.println(a1.equals(a2));// false
       System.out.println(str1.equals(str2));//  true
```

```
    }
}
```

```
OP:
false
false
false

false
true
```

4. public boolean equalsIgnoreCase(String str):

   In String class, equals() method is able to perform case sensitive comparison between two String objects contents.

   In the String class, equalsIgnoreCase() is able to perform case INsensitive comparison between two String objects contents.

   EX:

```java
public class Main {
   public static void main(String[] args)  {

          String str1 = new String("abc");
          String str2 = new String("ABC");

          System.out.println(str1.equals(str2));// false

System.out.println(str1.equalsIgnoreCase(str2));//
true



    }
}
```

```
OP:
false
true
```

5. public int compareTo(Object obj):
   It is able to compare two String objects' contents as per dictionary order.

   str1.compareTo(str2);
   If str1 comes first when compared with str2 in dictionary order then compareTo() method will return -ve value.

   If str2 comes first when compared with str1 in dictionary order then compareTo() method will return +ve value.

   If str1 and str2 are at the same position in dictionary order then compareTo() method will return 0 value.

   EX:

```
public class Main {
   public static void main(String[] args) {

        String str1 = new String("abc");
        String str2 = new String("def");
        String str3 = new String("abc");

        System.out.println(str1.compareTo(str2));// -3
        System.out.println(str2.compareTo(str3));// +3
        System.out.println(str3.compareTo(str1));// 0



    }
}
```

   OP:

```
-3
3
0
```

6. public boolean startsWith(String str):
   It will check whether the String starts with the specified string or not.

7. public boolean endsWith(String str):
   It will check whether the string ends with the specified String or not.

8. public boolean contains(String str):
   It will check whether the String contains the specified String or not.

EX:

```java
public class Main {
   public static void main(String[] args) {

       String str = new String("Durga Software Solutions");
       System.out.println(str.startsWith("Durga"));// true
       System.out.println(str.startsWith("Solutions"));// false
       System.out.println(str.endsWith("Solutions"));// true
       System.out.println(str.endsWith("Durga"));// false
       System.out.println(str.contains("Software"));// true
       System.out.println(str.contains("Durga"));// true
       System.out.println(str.contains("Solutions"));// true
       System.out.println(str.contains("Durgasoft"));// false


   }
}
```

OP:

```
true
false
true
false
true
true
true
false
```

9. public String replace(char oldChar, char newChar):
It can be used to replace the specified old character with the specified new character.

EX:

```
public class Main {
   public static void main(String[] args) {
         String str = new String("Durga Software
Solutions");
         System.out.println(str);
         System.out.println(str.replace('S', 's'));

   }
}
```

OP:

```
Durga Software Solutions
Durga software solutions
```

10. public char charAt(int index):

It can be used to return an character available at the specified index value, where if the specified index value is not in the range of the String index values then JVM will raise an exception like java.lang.StringIndexOutOfBoundsException.

EX:

```
public class Main {
   public static void main(String[] args) {
         String str = new String("Durga Software
Solutions");
         System.out.println(str);
         System.out.println(str.charAt(6));
         //System.out.println(str.charAt(50)); --->
java.lang.StringIndexcOutOfBoundsException
   }
}
```

OP:

```
S
```

11. public int indexOf(String str):

It can be used to return an index where the provided String exists at its first occurrence, if the provided String does not exist in the String then indexOf() method will return -1 value.

EX:
```
public class Main {
   public static void main(String[] args) {
        String str = new String("Durga Software
Solutions");
        System.out.println(str);
        System.out.println(str.indexOf("So"));
        System.out.println(str.indexOf("ZZZ"));
   }
}
```
OP:
```
6
-1
```

12. public int lastIndexOf(String str):
It can be used to return an index value where the provided String exists at its last occurrence, if the specified String does not exist in the String object content then lastindexOf() method will return -1.

EX:
```
public class Main {
   public static void main(String[] args) {
        String str = new String("Durga Software
Solutions");
        System.out.println(str);
        System.out.println(str.lastIndexOf("So"));
        System.out.println(str.lastIndexOf("ZZZ"));
   }
}
```
OP:
```
15
-1
```

13. public String substring(int index):
It can be used to generate a substring which starts from the specified start index value.

EX:
```
public class Main {
   public static void main(String[] args) {
        String str = new String("Durga Software 
Solutions");
        System.out.println(str);
        System.out.println(str.substring(6));
    }
}
```
OP:
```
Durga Software Solutions
Software Solutions
```

14. public String substring(int startIndex, int endIndex)
It will generate a substring which starts from the specified start index value and up to the specified endIndex value.

EX:
```
public class Main {
   public static void main(String[] args) {
        String str = new String("Durga Software 
Solutions");
        System.out.println(str);
        System.out.println(str.substring(6, 14));
    }
}
```

OP:
```
Durga Software Solutions
Software
```

15. public byte[] getBytes():
It can be used to convert String data to the equivalent byte[] with ASCII values.

EX:
```
public class Main {
   public static void main(String[] args) {
       String str = new String("Durga Software
Solutions");
       System.out.println(str);
       byte[] b = str.getBytes();
       for(int index = 0; index < b.length; index++){
           System.out.println(b[index]+"----
>"+(char)b[index]);
       }
   }
}
```

```
OP:
Durga Software Solutions
68---->D
117---->u
114---->r
103---->g
97---->a
32---->
83---->S
111---->o
102---->f
116---->t
119---->w
97---->a
114---->r
101---->e
32---->
83---->S
111---->o
108---->l
117---->u
116---->t
105---->i
111---->o
110---->n
115---->s
```

16. public char[] toCharArray():
It can be used to convert the String data into the equivalent char[].
EX:
```
public class Main {
   public static void main(String[] args) {
       String str = new String("Durga Software 
Solutions");
       System.out.println(str);
       char[] ch = str.toCharArray();
       for(int index = 0; index < ch.length; 
index++){
           System.out.print(ch[index]+"  ");
       }
   }
}
```
OP:
```
Durga Software Solutions
D  u  r  g  a     S  o  f  t  w  a  r  e     S  o  l  u  t  i  o  n  s
```

17. public String split(String regEx):
It is able to split the provided String into the number of pieces in the form of String[] on the basis of the provided regular Expression.
EX:
```
public class Main {
   public static void main(String[] args) {
       String str = new String("Durga Software 
Solutions");
       System.out.println(str);
       String[] items = str.split(" ");
       for(int index = 0; index < items.length; 
index++){
           System.out.println(items[index]);
       }
   }
}
```

```
OP:
Durga Software Solutions
Durga
Software
Solutions
```

EX:

```java
public class Main {
   public static void main(String[] args) {
        String mobileNo = "91-998877665";
        System.out.println("Mobile Number :
"+mobileNo);
        String[] items = mobileNo.split("-");
        if(items[0].equals("91") && items[1].length()
== 10){
            System.out.println("Status   : Valid
Mobile Number");
        }else{
            System.out.println("Status   : Invalid
Mobile Number");
        }
   }
}
```

```
OP:
Mobile Number : 91-998877665
Status   : Invalid Mobile Number
```

18. public String toLowerCase():
It can be used to convert data from Upper case letters to lower case letters.

19. public String toUppercase():
It can be used to convert data from Lower case letters to the Upper case letters.

EX:

```java
public class Main {
   public static void main(String[] args) {
       String str = "Durga Software Solutions";
       System.out.println(str);
       System.out.println(str.toLowerCase());
       System.out.println(str.toUpperCase());
   }
}
```

OP:

```
Durga Software Solutions
durga software solutions
DURGA SOFTWARE SOLUTIONS
```

20. public String trim():
It can be used to remove spaces before the String and after the string.

EX:

```
public class Main {
    public static void main(String[] args) {
        String str = "      Durga Software Solutions
";
        System.out.println(str+"    :
"+str.length());
        String str1 = str.trim();
        System.out.println(str1+"   :
"+str1.length());
    }
}
```

OP:

```
      Durga Software Solutions          :   37
Durga Software Solutions  :   24
```

Note: If trim() method does not change the data in the given String then trim() method will not create a new string object, it will create new String object when new data is generated from trim() method.
EX:

```
public class Main {
    public static void main(String[] args) {
        String str1 = new String("abc");
        String str2 = str1.trim();
        System.out.println(str1 == str2);
        System.out.println(str1.equals(str2));
    }
}
```

OP:
```
true
true
```

EX:

```
public class Main {
    public static void main(String[] args) {
        String str1 = new String("Durgasoft");
        String str2 = str1.concat("");
        String str3 = str1.concat(" ");

        System.out.println(str1 == str2);// true
        System.out.println(str1 == str3);// false

        System.out.println(str1.equals(str2));// true
        System.out.println(str1.equals(str3));// false
    }
}
```

OP:
```
true
false
true
false
```

StringBuffer class Library:

StringBuffer class is a mutable class whose objects are able to allow modifications directly.

Constructors:

1.public StringBuffer():
It can be used to create an empty StringBuffer class object with the initial capacity value of 16 elements.
EX:
```
public class Main {
    public static void main(String[] args) {
        StringBuffer stringBuffer = new StringBuffer();
         System.out.println(stringBuffer);
         System.out.println(stringBuffer.capacity());
    }
}
```
OP:
```

16
```

2. public StringBuffer(int capacity):
It can be used to create an empty StringBuffer object with the specified capacity value.
EX:
```
public class Main {
    public static void main(String[] args) {
        StringBuffer stringBuffer = new
StringBuffer(10);
         System.out.println(stringBuffer.capacity());
    }
}
```
OP:
```
10
```

3. public StringBuffer(String data):
It can be used to create a StringBuffer class object with the specified data.
EX:

```
public class Main {
   public static void main(String[] args) {
      StringBuffer stringBuffer = new
StringBuffer("abc");
       System.out.println(stringBuffer);
       System.out.println(stringBuffer.capacity());
   }
}
```

OP:

```
abc
19
```

Methods:

1. public StringBuffer append(String data):

It can be used to append the specified String data to the StringBuffer object content in a mutable manner.

EX:

```
public class Main {
   public static void main(String[] args) {
      StringBuffer sb1 = new StringBuffer("Durga ");
      StringBuffer sb2 = sb1.append("Software ");
      StringBuffer sb3 = sb2.append("Solutions");

       System.out.println(sb1);
       System.out.println(sb2);
       System.out.println(sb3);

       System.out.println(sb1 == sb2);
       System.out.println(sb2 == sb3);
       System.out.println(sb3 == sb1);

   }
}
```

OP:

```
Durga Software Solutions
Durga Software Solutions
Durga Software Solutions
```

```
true
true
true
```

2. public int capacity():
It can be used to get the capacity value of the StringBuffer object.
EX:

```
public class Main {
   public static void main(String[] args) {
      StringBuffer sb = new
StringBuffer("Durgasoft");
        System.out.println(sb.capacity());


   }
}
```

NewCapacity = InitialCapacity+lengthOfData

3. public void ensureCapacity(int capacity):
It can be used to provide capacity value to the StringBuffer object explicitly.

If the Capacity < 16 then capacity value is 16.
If the capacity > 16 and capacity < 34 then the capacity value is 34.
If the capacity > 34 then the capacity value is the specified value.

EX:

```
public class Main {
   public static void main(String[] args) {
        StringBuffer sb = new StringBuffer();
        sb.ensureCapacity(60);
        System.out.println(sb.capacity());
   }
}
```

OP:

```
60
```

4. public StringBuffer reverse():
It can be used to reverse the data in the StringBuffer object.
EX:

```
public class Main {
   public static void main(String[] args) {
        StringBuffer stringBuffer = new
StringBuffer("Durga Software Solutions");
        System.out.println(stringBuffer);
        stringBuffer.reverse();
        System.out.println(stringBuffer);
    }
}
```

OP:

```
Durga Software Solutions
snoituloS erawtfoS agruD
```

EX:

```
public class Main {
   public static void main(String[] args) {
        StringBuffer stringBuffer1 = new
StringBuffer("LEVEL");
        StringBuffer stringBuffer2 =
stringBuffer1.reverse();
        if(stringBuffer1.equals(stringBuffer2)){
            System.out.println("LEVEL is Palindrome
String");
        }else{
            System.out.println("LEVEL is not
Palindrome String");
        }
    }
}
```

OP:

```
LEVEL is Palindrome String
```

5. public StringBuffer insert(int index, String data)
It can be used to insert the specified data at the specified index value in the StringBuffer object content.
EX:

```
public class Main {
   public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Durga
Solutions");
        System.out.println(sb);
        sb.insert(6,"Software");
        System.out.println(sb);
    }
}
```

OP:

```
Durga  Solutions
Durga Software Solutions
```

6. public StringBuffer delete(int startIndex, int endIndex):
It can be used to delete the String that existed in between the specified startIndex and the specified endIndex.
EX:

```
public class Main {
   public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Durga
Software Solutions");
        System.out.println(sb);
        sb.delete(6,14);
        System.out.println(sb);
    }
}
```

OP:

```
Durga Software Solutions
```
