```
  Methods of String
=================
  1.public char charAt(int index)
  2.public String concat(String str)
  3.public boolean equals(Object o)
  4.public boolean equalsIgnoreCase(String s)
  5.public String substring(int begin)
  6.public String substring(int begin,int end)
  7.public int length()
  8.public String replace(char old,char new)
  9.public String toLowerCase()
  10.public String toUpperCase()
  11.public String trim()
  12.public int indexOf(char ch)
  13.public int lastIndexOf(char ch)



  3.public boolean equals(Object o)
             It is used for Content Comparison,In String class equals() method is
Overriden to check the content of the object

  4.public boolean equalsIgnoreCase(String s)
             It is used for Content Comparison without comparing the case.

eg#1.
public class Test {
      public static void main(String[] args) {
            String s ="java";
            System.out.println(s.equals("JAVA"));//false
            System.out.println(s.equalsIgnoreCase("JAVA"));//true
      }
}

Assignment
==========
credentials(gmail)
username:nitin@ineuron.ai(not case sensitive)
password :************(case sensitive)

5.public String substring(int begin)
        It gives the String from the begin index to end of the String.
             String s="Ineeuron";
           System.out.print(s.substring(2));//searching from 2 to end of the string

  6.public String substring(int begin,int end)
        It gives the String from the begin index to end-1 of the String.
             String s="Ineeuron";
           System.out.print(s.substring(2,6));//searching from 2 to 5 will happen

eg#1.
public class Test {
      public static void main(String[] args) {
            String s ="sachinINDMI";
            System.out.println(s.length());

            System.out.println();

            System.out.println(s.substring(9));
```

```
            System.out.println(s.substring(0,8));
            System.out.println(s.substring(0,9));


      }
}
```

```
8.public String replace(char old,char new)
             String s="abababab";
           System.out.print(s.replace('a','b')); //bbbbbb
```

```
eg#1.
public class Test {
      public static void main(String[] args) {
            String name ="sbchin";
            System.out.println(name.replace('b','a'));//sachin

            String data = "abababab";
            System.out.println(data.replace('a','b'));//bbbbbb
      }
}
```

```
   9.  public String toLowerCase()
  10. public String toUpperCase()
```

```
eg:
public class Test {
      public static void main(String[] args) {
            String name ="sAchIn";//mixedCase
            System.out.println(name.toUpperCase());//SACHIN
            System.out.println(name.toLowerCase());//sachin
      }
}
```

```
11. public String toString();
            Note: When ever we print any reference,by default JVM will call
toString() on the reference
                        eg: System.out.println(name);
                               System.out.pritnln(name.toString());
```

```
eg#1.
//userdefined class
class Student{
      String name = "sachin";
      int id = 10;
}
public class Test {
      public static void main(String[] args) {

            Student std = new Student();
            System.out.println(std);//Student@HexaDecimalValue
            System.out.println(std.toString());////Student@HexaDecimalValue

            System.out.println();

            String name =new String("dhoni");
            System.out.println(name);//dhoni
            System.out.println(name.toString());//dhoni
      }
```

```
}
```

11.public String trim()
To remove the blank spaces present at the begining and end of string but not the
blank spaces present at the middle of the String.

```
eg#1.
public class Test {
      public static void main(String[] args) {
            String name = "Sachin IND";
            System.out.println(name.length());//10
            System.out.println(name.trim());//Sachin IND

            System.out.println();

            String state = "  Karnataka  ";
            System.out.println(state.length());//13
            System.out.println(state.trim());//Karnataka
      }
}
```

12.public int indexOf(char ch)
It returns the index of 1st occurance of the specified character if the specified
character is not available then it returns -1.

```
            String s="sachinramesh";
          System.out.print(s.indexOf('a'));//1
          System.out.print(s.indexOf('z'));//-1
```

13.public int lastIndexOf(char ch)
It returns the index of last occurance of the specified character if the specified
character is not available then it returns -1.

```
            String s="sachinramesh";
          System.out.print(s.lastIndexOf('a'));//7
          System.out.print(s.lastIndexOf('z'));//-1

eg#1.
public class Test {
      public static void main(String[] args) {
            String name = "hyderAbbasbengaluru";

            System.out.println();

            System.out.println(name.indexOf('A'));//5
            System.out.println(name.indexOf('a'));//8

            System.out.println();

            System.out.println(name.indexOf('b'));//6
            System.out.println(name.lastIndexOf('b'));//10

            System.out.println();
            System.out.println(name.lastIndexOf('Z'));//-1
```

```
      }
}
```

Predict the output
==========

```
Q>
String s1="sachin";                    // s1,s3 -> sachin (scp)
String s2=s1.toUpperCase(); //  s2->SACHIN(heap area)
String s3=s1.toLowerCase();
System.out.print(s1==s2);//false
System.out.print(s1==s3);//true


Q>
String s1="sachin";           // s1,s2-> sachin (SCP)
String s2=s1.toString();
System.out.print(s1==s2);//true


Q>
String s1=new String("sachin");
String s2=s1.toString();
String s3=s1.toUpperCase();
String s4=s1.toLowerCase();
String s5=s1.toUpperCase();
String s6=s1.toLowerCase();
System.out.print(s1==s6);//true
System.out.print(s3==s5);//false
```

final vs Immutability
==========

=> final is a modifer applicable for variables, where as immutability is applicable only for Objects.
=> If reference variable is declared as final,it means we cannot perform reAssignment for the reference variable,
      it doesnot mean we cannot perform any change in that object.
=> By declaring a reference variable as final, we wont get immutablity nature.
=> final and Immutablity is different concept.

```
eg:: final StringBuilder sb=new StringBuilder("sachin");
         sb.append("tendulkar");
            System.out.println(sb);
         sb=new StringBuilder("dhoni"); //CE::Cannot assign a value to final variable
```

Note::
final variable(valid),
final object(invalid),
immutable variable(invalid)
immutable object(valid)

StringBuilder,StringBuffer are by default mutable.
All Wrapper classes(Byte,Short,Long,Integer,Float,Double,Boolean,Character) are by Default Immutable.

Mutable -> can be changed
Immutable => can't be changed

StringBuffer
==========
1. If the content will change frequently then it is not recomonded to go for String object becoz for every new
change a new Object will be created.
2. To handle this type of requirement, we have StringBuffer/StringBuilder concept

Constructors of StringBuffer
=====================
1.StringBuffer sb=new StringBuffer();
creates a empty StringBuffer object with default intital capacity of "16".
Once StringBuffer reaches its maximum capacity a new StringBuffer Object will be created
new capacity = (currentcapacity+1) * 2;

```
eg1::StringBuffer sb = new StringBuffer();
     System.out.println(sb.capacity());//16
     sb.append("abcdefghijklmnop");
     System.out.println(sb.capacity());//16
     sb.append('q');
     System.out.println(sb.capacity());//34
```

2. StringBuffer sb=new StringBuffer(initalCapacity);
It creates an Empty String with the specified inital capacity.

```
eg1::StringBuffer sb = new StringBuffer(19);
     System.out.println(sb.capacity());//19
```

3. StringBuffer sb=new StringBuffer(String s);
It creates a StringBuffer object for the given String with the capacity = s.length() + 16;

```
eg1::StringBuffer sb = new StringBuffer("sachin");
     System.out.println(sb.capacity());//22
```

Important methods of StringBuffer/StringBuilder
==========
a. public int length()
b. public int capacity()
c. public char charAt(int index)
d. public void setCharAt(int index,char ch)
==========
e. public StringBuffer append(String s)
f. public StringBuffer append(int i)
g. public StringBuffer append(long l)
h. public StringBuffer append(boolean b)
i. public StringBuffer append(double d)
j. public StringBuffer append(float f)
k. public StringBuffer append(int index,Object o)
==========
l. public StringBuffer insert(int index,String s)
m. public StringBuffer insert(int index,int i)
n. public StringBuffer insert(int index,long l)
o. public StringBuffer insert(int index,double d)
p. public StringBuffer insert(int index,boolean b)
q. public StringBuffer insert(int index,float f)
r. public StringBuffer insert(int index,Object o)

```
================================================
public StringBuffer delete(int begin,int end)
public StringBuffer deleteCharAt(int index)
public StringBuffer reverse()
public void setLength(int Length)
public void trimToSize()
public void ensureCapacity(int capacity)

eg::
   StringBuilder sb = new StringBuilder("sachinrameshtendulkar");
   System.out.println(sb.length());//21
   System.out.println(sb.capacity());//21 + 16 = 37
   System.out.println(sb.charAt(20));//'r'
   System.out.println(sb.charAt(100));//StringIndexOutOfBoundsException

eg::
  StringBuffer sb1 =new StringBuffer("kohlianushka");
   sb1.setCharAt(5,'A');
   System.out.println(sb1);//kohliAnushka
```