When to go for String and When to go for StringBuffer?
   String -> Frequently if there is no change of data then go for String[Immutable]
   StringBuffer -> Frequently if there is a change in the data then go for StringBuffer(mutable)

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

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
==========
  a. public int length()
  b. public int capacity()
  c. public char charAt(int index)
  d. public void setCharAt(int index,char ch)

eg#1.

```
class Test{
      public static void main(String... args) {
            StringBuffer sb = new StringBuffer("sachin");
            System.out.println(sb.length());
            System.out.println(sb.capacity());
            System.out.println(sb.charAt(3));
            System.out.println(sb.charAt(30));
      }
```

```
}

Output
6
22
h
Exception in thread "main" java.lang.StringIndexOutOfBoundsException


eg#2.
class Test{
      public static void main(String... args) {
            StringBuffer sb = new StringBuffer("sachinrameshtendulkar");
            sb.setCharAt(8,'B');
            System.out.println(sb);

      }
}

Output
sachinraBeshtendulkar

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)


eg#1.
class Test{
      public static void main(String... args) {
            StringBuffer sb = new StringBuffer();
            sb.append("The value of PIE is :: ");
            sb.append(3.1414);
            sb.append(" This is exactly" );
            sb.append(true);
            System.out.println(sb);
      }
}
Output
The value of PIE is :: 3.1414 This is exactlytrue


Overloaded method
+++++++++++++++++
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 s)
r.  public StringBuffer insert(int index,Object o)

To insert the String at the specified position we use insert method

eg#1.
```

```
class Test{
      public static void main(String... args) {
            StringBuffer sb = new StringBuffer("abcdefgh");

            sb.insert(2,"xyz");//abxyzdefgh
            sb.insert(11,"9");

            System.out.println(sb);//abxyzdefgh9
      }
}

Output
abxyzcdefgh9


Methods of StringBuffer
=======================
  public StringBuffer delete(int begin,int end)
       It deletes the character from specified index to end-1.

  public StringBuffer deleteCharAt(int index)
       It deletes the character at the specified index.


eg#1.
class Test{
      public static void main(String... args) {
            StringBuffer sb = new StringBuffer("sachintendulkar");
            System.out.println(sb);//sachintendulkar

            sb.delete(6,12);//tendul
            System.out.println(sb);//sachinkar

            sb.deleteCharAt(6);
            System.out.println(sb);//sachinar


      }
}

Output
sachintendulkar
sachinkar
sachinar

public StringBuffer reverse()
      It is used to reverse the given String.


eg#1.
class Test{
      public static void main(String... args) {
            StringBuffer sb = new StringBuffer("pwskills");
            sb.reverse();
            System.out.println(sb);//sllikswp
      }
}
```

public void setLength(int Length)
      It is used to consider only the specified no of characters and remove all the remaining characters.

public void trimToSize()
        This method is used to deallocate the extra allocated free memory such that capacity and size are equal.

public void ensureCapacity(int capacity)
      It is used to increase the capacity dynamically based on our requirement.

eg#1.

```
class Test{
      public static void main(String... args) {

            StringBuffer sb = new StringBuffer("sachinrameshtendulkar");
            sb.setLength(6);
            System.out.println(sb);

            System.out.println();

            StringBuffer sb1 =new StringBuffer(100000);
            System.out.println(sb1.capacity());
            sb1.append("sachin");
            System.out.println(sb1.capacity());
            sb1.trimToSize();
            System.out.println(sb1.capacity());

            System.out.println();

            StringBuffer sb2 = new StringBuffer();
            System.out.println(sb2.capacity());
            sb2.ensureCapacity(10000);

            System.out.println(sb2.capacity());
      }
}
```

Output

```
sachin

100000
100000
6

16
10000
```

EveryMethod present in StringBuffer is synchronized, so at a time only one thread can are allowed to operate on StringBuffer Object, it would increase the waiting time of the threads it would
create performance problems, to overcome this problem we should go for StringBuilder.

StringBuilder(1.5v)
  StringBuilder is same as StringBuffer(1.0V) with few differences

StringBuilder

```
   No methods are synchronized
   At at time more than one thread can operate so it is not ThreadSafe.
   Threads are not requried to wait so performance is high.
   Introduced in jdk1.5 version

StringBuilder vs StringBuffer
++++++++++++++++++++++++++++
StringBuffer
      -> JDK1.0 version
      -> Every method present in StringBuffers is synchronized
      -> Only one thread is allowed to operate on StringBuffer object.
      -> ThreadSafety.
      -> increases the waiting time so performance is low.

StringBuilder
      -> JDK1.5 version
      -> Every method present in StringBuilder is not synchronized.
      -> Multiple thread are allowed to operate on StringBuffer object.
      -> Not ThreadSafety.
      -> no waiting so performance is high.


When to go for String,StringBuffer,StringBuilder

String vs StringBuffer vs StringBuilder
======================================
 String        => we opt if the content is fixed and it wont change frequently
 StringBuffer  => we opt if the content changes frequently but ThreadSafety is
required
 StringBuilder => we opt if the content changes frequently but ThreadSafety is not
required


MethodChaining
=============
  Most of the methods in String,StringBuilder,StringBuffer return the same type
only, hence after applying method on the result
  we can call another method which forms method chaining.

eg::
StringBuffer sb = new StringBuffer();
sb.append("sachin").insert(6, "tendulkar").reverse().append("IND").delete(0,
4).reverse();
System.out.println(sb);


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 differnt 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 is Mutable.
       All Wrapper classes(Integer,Float,Double,Byte,Short,Character,Boolean)  and
String are by default Immutable.

Question::
   1. Difference b/w String and StringBuilder?
   2. Difference b/w String and StringBuffer?
   3. Other than Immutablity and Mutablity what is the difference b/w String and
StringBuffer?
   4. What is SCP?
   5. What is the advantage of SCP?
   6. What is the disadvantage of SCP?
   7. Why SCP is applicable only for String and not for StringBuilder?
   8. Is their any Object which is Immutable just like String?
   9. What is interning?
   10.Difference b/w final and Immutablity?


Questions

1.
String s1 = "null"+null+1;
System.out.println(s1);

Output : nullnull1


2.
String s1 = 1+null+"null";
System.out.println(s1);

Output: CE: bad operand type '+'


String str = "sachin ramesh tendulkar";
System.out.println(str.indexOf('a') + str.indexOf("dulkar"));//18


String s1="sachinrameshtendulkar";
System.out.println(s1.replace('a', 'A').indexOf('a'));//-1


String str = "pwskills Private Limited";
System.out.println(str.indexOf('i', 5));//5


String str = "ineurontechnologyprivatelimited";
System.out.println(str.charAt(str.length()));//SIOBE


StringBuilder sb = new StringBuilder(-32);
sb.append("ABC");
```

```
System.out.println(sb);//NegativeArraySizeException
```

Q>
```
StringBuilder sb = new StringBuilder("0123456789");
System.out.println(sb.delete(3, 6).deleteCharAt(4).deleteCharAt(5));// 01268
```

Q>
```
String str1 = "123321123";
System.out.println(str1.replaceFirst("123", "321").replaceAll("12",
"21").substring(3, 6));//321321213 => 321
```

Q>
```
String str1 = "OnE tWo ThReE fOuR";
String str2 = "oNeTwOtHrEeFoUr";
System.out.println(str1.trim().equalsIgnoreCase(str2));//false
```

Q>
```
StringBuffer sb = new StringBuffer("11111");
System.out.println(sb.insert(3, false).insert(5, 3.3).insert(7,
"One"));//111fa3.One3lse11
```

Q>
```
String str1 = "Java J2EE Spring Hibernate SQL";
String str2 = "Python Java Scala C C++";
System.out.println(str1.contains("HTML") == str2.contains("HTML"));//true
```

Q>
```
StringBuffer sb = new StringBuffer("One Two Three Four Five");
System.out.println(sb.reverse().indexOf("Two"));//-1
```