List of Concepts Involved:

- Mutable String
- String Buffer vs String Builder
- Inbuilt Methods

Mutable String

Once if we create a String, on that String if we try to perform any operation and if those changes get reflected in the same object then such strings are called “Mutable String”.

Example: StringBuffer, StringBuilder

How to create a mutable string?

We can use the StringBuffer and StringBuilder classes to generate mutable strings. Which class we should use entirely depends on the scenario. Both classes generate a mutable object of string.
If the string needs to be thread-safe and you wish to operate in a multithreading environment, you should use the StringBuffer class. On the other hand, StringBuilder is not necessary if you don't want a multithreading environment.
However, if speed is your top priority, StringBuilder outperforms StringBuffer in terms of speed.

StringBuffer

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

1.StringBuffer sb=new StringBuffer();

- Creates an empty StringBuffer object with default initial capacity of "16".
- Once StringBuffer reaches its maximum capacity a new StringBuffer Object will be created.
  new capacity = (current capacity+1) * 2;

Example

```
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(initialCapacity);

It creates an Empty String with the specified initial capacity.

Example

```
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;

Example

```
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)
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)

Example

```
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
```

Example

```
StringBuilder sb = new StringBuilder("sachin");
sb.setCharAt(2, 'C');
System.out.println(sb);
```

Append method is an overloaded method, methodName is the same but changes in the argument type.

Example

```
StringBuffer sb = new StringBuffer();
sb.append("PI value is :: ");
sb.append(3.1414);
sb.append(" This is exactly ");
sb.append(true);
System.out.println(sb);// PI value is ::3.1414 This is exactly true
```

Overloaded methods

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
Example

```
StringBuffer sb = new StringBuffer("sacin");
sb.insert(3, 'h');
System.out.println(sb);//sachin
sb.insert(6, "IND");
System.out.println(sb);//sachinIND
```

Methods of StringBuffer

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

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

Example:

```
StringBuffer sb=new StringBuffer("sachinrameshtendulkar");
sb.delete(6,12);
System.out.println(sb);//sachintendulkar
sb.deleteCharAt(13);
System.out.println(sb);//sachintndulkar
```

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

Example

```
StringBuffer sb=new StringBuffer("sachin");
sb.reverse();
System.out.println(sb);//nihcas
```

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

Example

```
StringBuffer sb=new StringBuffer("sachinramesh");
sb.setLength(6);
System.out.println(sb);//sachin
```

public void trimToSize()

This method is used to deallocate the extra allocated free memory such that capacity and size are equal.

Example

```
StringBuffer sb = new StringBuffer(1000);
System.out.println(sb.capacity());//1000

sb.append("sachin");
System.out.println(sb.capacity());//1000

sb.trimToSize();

System.out.println(sb);//sachin
System.out.println(sb.capacity());//6
```

public void ensureCapacity(int capacity)

It is used to increase the capacity dynamically based on our requirements.

Example

```
StringBuffer sb = new StringBuffer();
System.out.println(sb.capacity());//16
sb.ensureCapacity(1000);
System.out.println(sb.capacity());//1000
```

Note

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 a time more than one thread can operate so it is not ThreadSafe.
- Threads are not required to wait so performance is high.
- Introduced in jdk1.5 version

StringBuilder is a mutable sequence of characters in Java and provides several inbuilt methods to modify its contents. Here is a list of some commonly used methods in StringBuilder:

- **append(Object obj):** Adds the string representation of an Object to the end of the StringBuilder
- **append(String str):** Adds a String to the end of the StringBuilder
- **insert(int offset, Object obj):** Inserts the string representation of an Object into the StringBuilder at a specified position

- **insert(int offset, String str):** Inserts a String into the StringBuilder at a specified position
- **delete(int start, int end):** Removes the characters in the StringBuilder between the specified start and end positions
- **deleteCharAt(int index):** Removes the character at a specified position in the StringBuilder
- **replace(int start, int end, String str):** Replaces the characters in the StringBuilder between the specified start and end positions with the specified String
- **reverse():** Reverses the order of characters in the StringBuilder
- **toString():** Returns the String representation of the contents of the 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.

Example

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