String, StringBuffer and StringBuilder

1. String:

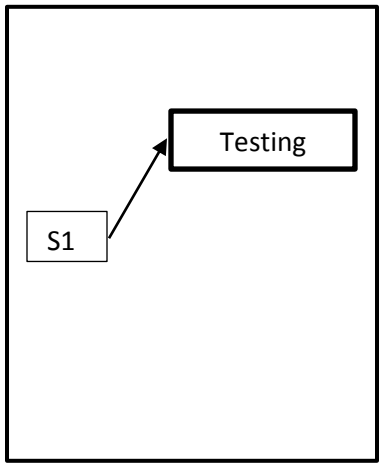
- String is a non-primitive data type in Java
- String is a class in Java
- String class objects are **immutable**.
- Once they are initialized, we cannot change them.
- We can create String objects using two methods.
- **Method #1:** String s=new String("Testing Shastra"); //Created in Heap as well as SCP
- **Method #2:** String s="Testing Shastra"; //Created in String Constant Pool

What is mean by immutable ?

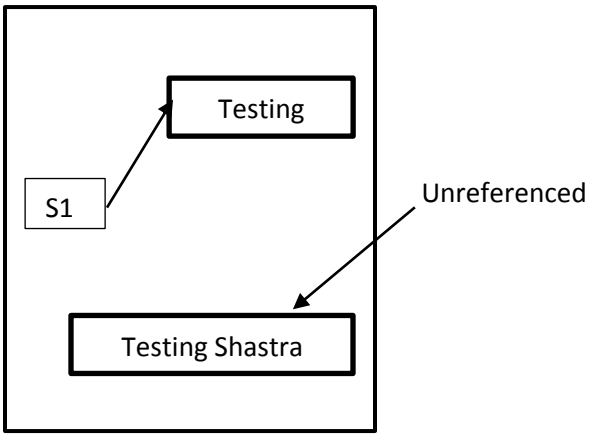
- Immutable objects cannot be deleted or changed once they are created. If we try to change them, a new object with changed value will be generated in memory. This feature is called immutability.

```
public class StringTest{
                Public static void main(String[] args){
                String s1="Testing";
                s1.concat("Shastra");
                System.out.println(s1);
                }
        }
O/P: Testing
```

- In above example, String "Shastra" will not be concatenated to "Testing" directly. Instead it will allocate new memory location to the concatenated String as shown in diagram.



(a)Before Concatenation



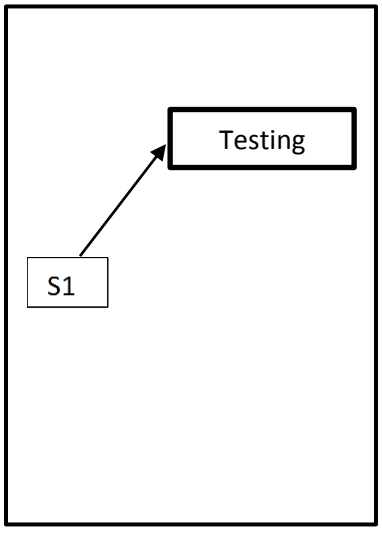
(b) After Concatenation

- We can observe that newly created string object (Testing Shastra) is not printed as output because we have not referenced to the new value yet. It will remain unreferenced till we provide reference to it.
- To get the concatenated string we have to make reference to it. Observe below program.

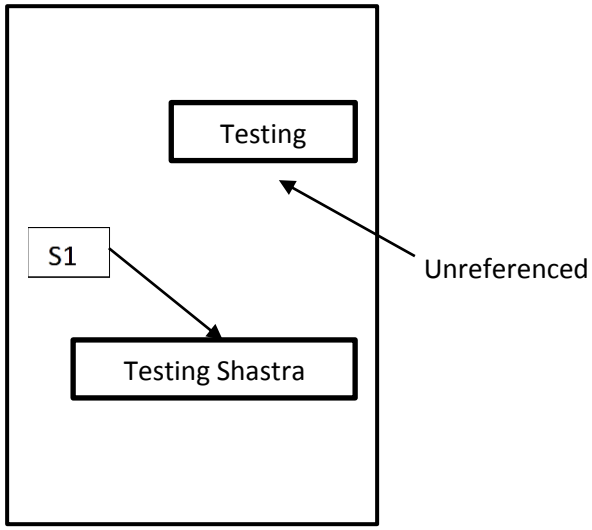
Reference

```
public class StringTest{
                Public static void main(String[] args){
                String s1="Testing";
                s1=s1.concat("Shastra");
                System.out.println(s1);
                }
        }
O/P: Testing Shastra
```

Now we have created reference to newly created string. Old string value will have no reference. But it is still present in memory. This is called **Immutability.**



(a) Before Concatenation



(b) After Concatenation

- All string objects are stored in either **String Constant Pool** or in Heap memory.
- String constant pool is a special type of memory dedicated to store String objects only.
- String Constant pool objects are NOT eligible for garbage collection. They will remain in memory until the life of program.

- Object creation in SCP is optional. If value is already present in SCP then new value will not be created. Instead reference of old value will be given to new object.

Eg.

```
public class StringTest{
         Public static void main(String[] args){
            String s1=”Testing”;
            String s2=”Testing”;
            if(s1==s2){ //Here == operator checks reference
                System.out.println(“s1 and s2 have same address”);
             }
          }
       }
```

- Hence in above example s1 and s2 have same address. SCP doesn't allow duplicate values.
- Whenever a request for new object creation is generated, JVM check whether same value is present in SCP or not. If same value is present then address of that value will be assigned to newly created object, else new object will have new value.

String declaration: literal vs new

- There are two ways through which we can declare and define string objects.
    1. String s=”Testing Shastra” *//Using literal(constant)*
    2. String s=new String(“Testing Shastra”) *//Using new keyword.*
- Let us see the difference between these two methods.

Using Literal	Using new keyword
String s=”Testing Shastra”	String s=new String(“Testing Shastra”)
In this case, only one object will be created in SCP.	In this case two objects will be created. One is SCP and other in Heap.
Here ’s’ will point to value in SCP only	Here‘s’ will always point to value in Heap memory. Whereas SCP value will not be referred by ‘s’
Duplicate objects are not allowed in SCP Hence: String s1=”Testing Shastra”; String s2=”Testing Shastra”; S1 and s2 will point to same value. They have same address.	Objects created in Heap allow duplicate value. Hence String s1=new String(“Testing Shastra”); String s2=new String(“Testing Shastra”); S1 and s2 will have different addresses.

Constructors of String class:

1. String s=new String();
   This will create empty string object.
2. String s=new String("Testing Shastra");
   This will create string object with value "Testing Shastra"
3. String s=new String(StringBuffer sb);
   This will create string object with value of StrinBuffer object.
4. String s=new String(char[] ch)
   This will create string object from character array.

Important Methods of String Class:

1. **public char charAt(int index):**
   This method will return character at specified index.
   E.g String s="Testing Shastra";
   S.o.pl(s.charAt(5)) // 'n' will be printed.
2. **public String concat(String s):**
   This method will concat two strings.

E.g. String s="Testing";
s=s.concat("Shastra");

3. **public int length():**
   This method will return number of characters present in String.
   Eg. String s="Testing Shastra";
   s.o.p(s.length()) //This will print 15. Space will be considered as character
4. **public Boolean equals(String s):**
   This method will return true if content of two strings are exact match including cases.
   E.g. String s="Testing";
   s.o.p(s.equals("Testing")) //This will print true
   s.o.p(s.equals("testing")) //This will print false
5. **Public Boolean equalsIgnoreCase(String s):**
   This method will compare two strings ignoring case.
   E.g. String s="Testing";
   s.o.p(s.equalsIgnoreCase("Testing")) //This will print true
   s.o.p(s.equalsIgnoreCase("testing")) //This will print true
6. **public String trim():**
   This method trims starting and ending spaces if string has any.
   E.g. String s=" Testing Shastra ";
   s.o.p(s.trim()); //This will print "Testing Shastra". It will not trim intermediate spaces.

Disadvantage of String Class:

- String class is immutable hence it will create new value in memory after each and every operation.
- This will lead to memory wastage at some extent.
- To overcome this advantage Java have introduced StringBuffer and StringBuilder classes.

2. StringBuffer:

- If you are modifying your string frequently then it is not recommended to use String objects as they are immutable.
- When we have frequent operation as modification of string then we should compulsorily go for StringBuffer.
- Objects of StringBuffer class are mutable. They can be changed once they are initialized.
- Modifications are performed in existing objects only. No new object is created.

Constructors of StringBuffer:

1. **StringBuffer sb=new StringBuffer();**
   - This will create empty StringBuffer object with default initial capacity as 16.
   - Once StringBuffer object reaches its initial capacity, a new StrinBuffer object with new capacity is created using formula:
     New Capacity=(Current Capacity+1)*2
2. **StringBuffer sb=new StrinBuffer(int initialCapacity);**
   - This will create an empty StringBuffer object with specified initial capacity.
3. **StringBuffer sb=new StrinBuffer(String s);**
   - This will create an StringBuffer object for the string with capacity= 16+s.length();

Important methods of StringBuffer:

i. public int length() //It gives number of characters in StringBuffer
ii. public int capacity() //It gives current capacity of StringBuffer
iii. public void setCharAt(int index, char c); //It replaces character at specified index with specified character.
iv. public StringBuffer append() //This method is overloaded and it appends two strings or values.
E.g public StringBuffer append(String s);
public StringBuffer append(int i);
public StringBuffer append(double d);
public StringBuffer append(Object o); etc.

3. StringBuilder:

- StringBuilder and StringBuffer are same except, all methods of StringBuffer are synchronized.
- That means only one thread can operate on StringBffer object at a time. This will lead to slow performance.
- If you open the StringBuffer class by pressing Ctrl+click, you can see that 'synchronized' keyword is attached in the declaration of each and every method, like: `public synchronized void trimToSize()`
- For example if a single thread takes 1 sec to operate on StringBuffer object and there are 10 threads in system which are requesting access to same StringBuffer object, then total execution time of program will be raised by 10 seconds. Because each and every object have to wait for completion of previously operating thread.
- To overcome this issue, Java has introduced StringBuilder.
- None of the methods of StringBuilder are synchronized.

Difference between String, StringBuffer and StringBuilder:

Sr. No	String	StringBuffer	StringBuilder
1	String class is immutable in Java	StringBuffer is mutable	StringBuilder is mutabl
2	None of the method of String class is synchronized	Each method of StringBuffer is synchronized	None of the method of StringBuilder class is synchronized
3	String is not thread safe	StringBuffer is thread safe	StringBuilder is not thread safe
4	It may lead to memory wastage	It may not lead to memory wastage	It will not lead to memory wastage
5	Not recommended where frequent operations are modification	Recommended where frequent operations are modifications	Recommended where frequent operations are modifications

Interview Questions

1. What is String class in Java?
2. What is the difference between StringBuffer and StringBuilder?
3. What is mean by immutable.
4. Can we create our own immutable class? (Yes. By writing public final class Test)
5. Find number of digits in a given String.
6. Convert first character of each word to capital (I/p= “testing shastra” O/p=”Testing Shastra”)
7. Find only consecutive duplicate characters from string.
8. Validate the password on basis of below mentioned rules
   - Password length should be min. 6 and max 12 characters long.
   - Password should contain atleast one digit.
   - Password should contain atleast one special character.

   Note: For this assignment you need to study regular expression. Because for every string question interviewer expects you to write regular expression

9. Reverse the string without using any predefined methods.
10. Replace special characters to ‘X’ in string.
    E.g Input= “Te$!ng $h@str@”
    Output=”TeXXng XhXstrX”;