String class

java. lang. String class:

--> String is a class it is presented under the java. Lang package.

--> collection of characters is called as a string which represents in a double quote.

--> Java String is a powerful concept because everything is treated as a String if you submit any form in window based, web based or mobile application.

--> it is used to manipulate the data.

--> by using the object creation can implement the string.

--> here are two ways to create String object:

1. By string literal
2. By new keyword

*String s=new String("mango");

--> In this case two objects will be created one is on the heap the other one is SCP (String constant pool) and s is always pointing to heap object.

Immutability:

==> Once we create a String object we can't perform any changes in the existing object. If we are trying to perform any changes with those changes a new object will be created. This behavior is called immutability of the String object.

Important methods of String class:

1. public char char at (int index)

--> Returns the character locating at specified index.

Example:

```
class String Demo {
        public static void main (String [] args) {
                String s="Mahita";
                System.out.println(s. char at (3));          //
                System.out.println(s. char At (100));       // RE: StringIndexOutOfBoundsException
        }
}
```

2. public String concat (String str);

Example:

```
class String Demo {
        public static void main (String [] args) {
                String s="Madhan";
                s=s. concat("software");
                //s=s+"software";
                //s+="software";
                System.out.println(s);//Madhan software
        }
}
```

3. public Boolean equals (Object o)

--> For content comparison where case is important.

--> It is the overriding version of Object class. equals () method

4. public Boolean equalsIgnoreCase (String s)

--> For content comparison where case is not important

Example:

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

Note: We can validate username by using. equalsIgnoreCase () method where case is not important and we can validate password by using. equals () method where case is important.

5. public String substring (int begin)

--> Return the substring from begin index to end of the string.

Example:

```
class String Demo {
        public static void main (String [] args) {
                String s="Rajesh soft";
                System.out.println(substring(6));          //soft
        }
}
```

6. public String substring (int begin, int end)

--> Returns the substring from begin index to end-1 index.

Example:

```
class String Demo {
        public static void main (String [] args) {
                String s="Madan soft";
                String s="Rajesh soft";
                System.out.println(substring(6));//soft
                System.out.println(substring(3,7));//eshs
        }
}
```

7. public int length ()

--> Returns the number of characters present in the string.

Example:

```
class String Demo {
        public static void main (String [] args) {
                String s="jobs4times";
                System.out.println(length ());     //10
                //System.out.println(s. length);  //compile time error
        }
}
```

Note: length is the variable applicable for arrays whereas length() method is applicable for String object.

8. public String replace (char old, char new)

--> To replace every old character with a new character.

9. public String to Lowercase ()

--> Converts the all characters of the string to lowercase.

Example:

```
class String Demo {
        public static void main (String [] args) {
                String s="VIVIDHA";
                System.out.println(s. to Lowercase ()) ;//Sadik
        }
}
```

10. public String to Uppercase ()

--> Converts the all characters of the string to uppercase.

Example:

```
class String Demo {
        public static void main (String [] args) {
                String s="Mahitha";
                System.out.println(s. to Uppercase ());    //MAHITHA
        }
}
```

11. public String trim ()

--> We can use this method to remove blank spaces present at beginning and end of the string but not blank spaces present at middle of the String.

12. public int index of (char Ch)

--> It returns index of 1st occurrence of the specified character if the specified character is not available then return -1

Example:

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

```
        String s="omsairam";
        System.out.println(s. index of('a')); // 6
        System.out.println(s. index of('z')); // -1
    }
}
```

13. public int lastIndexOf (Char Ch)

--> It returns index of last occurrence of the specified character if the specified character is not available then return -1.

Example

```
class StringDemo {
    public static void main (String [] args) {
        String s="anurvinya";
        System.out.println(s. lastIndexOf('y')) ;//7
        System.out.println(s. index of('z')) ;//-1
    }
}
```

--> Once we create an object we can't perform any changes in the existing object. If we are trying to perform any changes with those changes a new object will be created.

--> If there is no change in the content then existing object will be reused. This behavior is called immutability

Immutable program:

```
final class Create Immutable {
    private int I;
Create Immutable (int i) {
        this. I=i;
    }

    public Create Immutable modify (int i) {
        if (this. I==i) {
            return this;
```

```
}
        else {
            return (newly Create Immutable(i));
        }

    public static void main (String [] args) {
        Create Immutable c1=newly Create Immutable (10);
        Create Immutable c2=c1.modify(100);
        Create Immutable c3=c1.modify(10);
        System.out.println(c1==c2) ;//false
        System.out.println(c1==c3) ;//true
        Create Immutable c4=c1.modify(100);
        System.out.println(c2==c4) ;//false
    }
}
```

--> Once we create a Create Immutable object we can't perform any changes in the existing object, if we are trying to perform any changes with those changes a new object will be created. If there is no chance in the content then existing object will be reused

--> The class must be declared as final so that child classes can't be created

--> Data members in the class must be declared private so that direct access is not allowed.

STRING BUILDER

what is string builder?

* StringBuilder in Java is a class used to create a mutable, or in other words,

a modifiable succession of characters.

* Like String Buffer, the StringBuilder class is an alternative to the Java

Strings Class, as the Strings class provides an immutable succession of

characters

* Java StringBuilder class is used to create mutable (modifiable) String.

* The Java StringBuilder class is same as String Buffer class except that it is

non-synchronized. It is available since JDK 1.5.

Hierarchy of the StringBuilder?

why do we need string builder?

* StringBuilder is a class in the Java API that provides a mutable sequence of characters.

* It is used for dynamic string manipulation, such as building strings from many smaller strings or appending new characters to an existing string.

How to implement string builder?

* StringBuilder objects are like String objects, except that they can be modified.

* Internally, these objects are treated like variable-length arrays that contain a sequence of characters.

* At any point, the length and content of the sequence can be changed through method invocations.

Where to use StringBuilder?

* StringBuilder class can be used when you want to modify a string without creating a new object.

* For example, using the StringBuilder class can boost performance when concatenating many strings together in a loop

* What are the Constructors in string builder?

1. StringBuilder ()

2. StringBuilder (String str)

3. StringBuilder (int length)

1 StringBuilder (): -

* It creates an empty string builder with the initial capacity of 16

2 StringBuilder (String Src): -

* It creates a String Builder with the specified string.

3 StringBuilder (int length): -

* It creates an empty String Builder with the specified capacity as length

What are the methods in string builder?

* Numerous methods are available in Java's StringBuilder to execute various actions on the string builder.

* Some of the main StringBuilder class methods are below:

1. String Builder append (String s): -

It is used to append the specified string with this string. The append () method is overloaded like append(char), append (Boolean), append(int), append(float), append(double) etc.

2. String Builder insert (int offset, String s): -

It is used to insert the specified string with this string at the specified position. The insert () method is overloaded like insert (int, char), insert (int, Boolean), insert (int, int), insert (int, float), insert (int, double) etc.

3. String Builder replace (int start, int end, String s): -

It is used to replace the string from specified start Index and end Index.

4. String Builder reverse (): - It is used to reverse the string

5. String Builder delete (int start, int end: -

It is used to delete the string from specified start Index and end Index.

6. Int capacity: - It is used to return the current capacity.

7. Void ensureCapacitty (int min): -

It is used to ensure the capacity at least equal to the given minimum.

8. Char char at (int index): -

It is used to return the character at the specified position.

9. Int length: -

It is used to return the length of the string i.e. total number of characters.

10.String substring (int start, int end): -

It is used to return the substring from the specified begin Index.

11.Int index of (String str): -

It is used to return the substring from the specified begin Index and end

Index

12. Void trim To Size: -

This method attempts to reduce storage used for the character sequence.

Example of the all methods: -

```
package String;
import java. util. Vector;
public class StringBuilderExample {
public static void main (String [] args) {
 /* StringBuilder append () method
StringBuilder sb = new StringBuilder("Kalyan");
sapped(" chinnimgari");
System.out.println(sb);
}*/
//Output: Kalyan Chinnimgari
/*StringBuilder insert () method
StringBuilder si = new StringBuilder("Kalyan");
si. Insert(2, "Krishna ");
System.out.println(si);
}
output: kaKrishna lyan */
/*StringBuilder replace () method
StringBuilder sr = new StringBuilder("Kalyan");
sr. replaces (1, 3, "Krishna ");
System.out.println(sr);
}
output: kKrishna yan*/
/*StringBuilder delete () method
StringBuilder sd = new StringBuilder("Kalyan");
sd. delete (1, 3);
```

```
System.out.println(sd);
}
output: Kyan */
/* StringBuilder reverse () method
StringBuilder sv = new StringBuilder("kalyan");
sv. Reverse();
System.out.println(sv);
}
output: naylak*/
/* StringBuilder capacity () method
StringBuilder Sc = new StringBuilder ();
System.out.println(sc. capacity ());
scrapped("Krishna ");
System.out.println(sc. capacity ());
scrapped("Java is a language");
System.out.println(sc. capacity ());
}
output:16,16,34 */
/* StringBuilder ensure Capacity() method
StringBuilder sec=new StringBuilder ();
System.out.println(sec.capacity());
sec.append("kalyan");
System.out.println(sec.capacity());
sec.append("Java is a language");
System.out.println(sec. capacity());
sec. ensures Capacity (12);
System.out.println(sec. capacity ());
sec. ensures Capacity (35);
System.out.println(sec. capacity ());
}
output: 16,16,34,34,70 */
```

```
/*Char char at (int index)
String ca= "Kalyan, Krishna ";
for (int i=0;i<ca.length();i++) {
char c = ca. charAt(i);
System.out.println("character at index “+i+":"+c);
}
}
output: character at index0: k
character at index1: a
character at index2: l
character at index3: y
character at index4: a
character at index5: n
character at index6:
character at index7: g
character at index8:o
character at index9: p
character at index10: i */
/* Int length ()
String sl="Kalyan";
String sln= "Krishna ";
System.out.println(sl. length ());
System.out.println(sln. length ());
}
output: 6,4
*/
/* String substring (int start, int end)
String s="Kalyan";
String ss="Krishna ";
System.out.println(substring (2));
System.out.println(ss. substring (1,3));
```

```
}
output: lyan, op */
/* Int index of (String str)
String main String = "Kalyan, Krishna AI";
 String substring = "ramuAI";
 int index = mainString.indexOf(substring);
 if (index! = -1) {
 System.out.println("The substring \"" + substring + "\" is found
at index " + index + ".");
 } else {
 System.out.println("The substring \"" + substring + "\" is not
found.");
 }
 }
output: The substring "ramuAI" is not found.
*/
/* Void trim To Size ()
Vector<Integer> vec = new Vector<>(8);
 vec. Add (1);
 vec. Add (2);
 vec. Add (3);
 vec. Add (4);
 vec. Add (5);
 vec. trimetozine();
 System.out.println("The elements of a vector are:");
 for (Integer number: vec) {
 System.out.println("Number = " + number);
 }
 }
output: The elements of a vector are:
Number = 1
```

Number = 2

Number = 3

Number = 4

Number = 5

*/

}

}\

StringBuffer Class

→ String buffer is a one of the classes in char sequence interface.

→ Java string buffer class is used to create mutable string objects.

→ The string buffer class in java is the same as string class except mutable i.e. it can be changed.

→ String buffer and string builder classes are used for creating mutable strings.

→ Mutable string means a string that can be modifies or changed that known as mutable string.

→ String buffer allows to modify the content of the string without creating a new object each time.

→ Java string buffer class is a thread-safe that is multiple threads cannot accept it simultaneously so, it is safe and will result same in an order.

→ The main need of the string buffer is to efficient manipulations modifications and create mutable strings.

→ At a time only, single thread is to allow to operate.

→ String buffer can be used when need to perform a lot of string manipulations operations and insertion, deletions and modifications.

→ In java string buffer class can contain some important constructors those are like

- StringBuffer ()
- StringBuffer (String str)
- StringBuffer (int capacity)

→ In string buffer class can contain some methods. Those are

- Append ()
- Insert ()

- Delete ()
- Replace ()
- Reverse ()
- Capacity ()
- Length ()
- Char At ()

StringBuffer class Append () method

Append method is one of the method in string buffer class.

Append () method concatenates the given argument with this string.

```
1.Class StringBufferExample {
2.Public static void main (String args [])
```

StringBuffer Class

```
3.StringBuffer sb=new StringBuffer("hello");
4.Sb. append("java");
5.System.out.println(sb);
6.}
7.}
```

Output: hello java

StringBuffer class insert() method

The insert() method inserts the given string with this string at the given position

```
1.Class StringBufferExample {
2.Public static void main(String args[])
3.StringBuffer sb=new StringBuffer("Lakshmi");
4.sb.insert(1,"Prasanna");
5.System.out.println(sb);
6.}
7.}
```

Output: Lprasannaakshmi

StringBuffer replace() method:

The replace method replaces the given string from the specified begin index and end index.

```
1.Class StringBufferExample {
2.Public static void main(String args[])
3.StringBuffer sb=new StringBuffer(“hello”);
4.sb.replace(1,3,”java”);
5.System.out.println(sb);
6.}
7.}
```

Output: hjavalo

StringBuffer delete() method:

The delete() method of the string buffer class deletes the string from the specified

begin index to end index

```
1.Class StringBufferExample {
2.Public static void main(String args[])
3.StringBuffer sb=new StringBuffer(“madam”);
4.sb. delete (1,3,);
```

StringBuffer Class

```
5.System.out.println(sb);
6.}
7.}
```

Output: mdm

StringBuffer Reverse () method:

The reverse () method of the string builder class reverses the current string.

```
1.Class StringBufferExample {
2.Public static void main (String args[])
3.StringBuffer sb=new StringBuffer(“Lakshmi”);
4.sb. reverse ();
5.System.out.println(sb);
6.}
7.}
```

Output: imhskaL

String Buffer Capacity () method:

The default capacity of the buffer is 16.

The capacity () method of the string buffer class returns the capacity of the buffer.

For example, current capacity is 16, it will be (16*2)+2=34.

```
1.Class StringBufferExample {
2.Public static void main (String args [])
3.StringBuffer sb=new String Buffer ();
4.Sytem.out.println(sb. capacity ());
5.Sb. append(“hi”)
6.System.out.println(sb. capacity);
7.sb. append (“java is my favourite language”);
8.System.out.println(sb. capacity ());
9.}
10.}
```

Output:

16

16

34

String Buffer Class