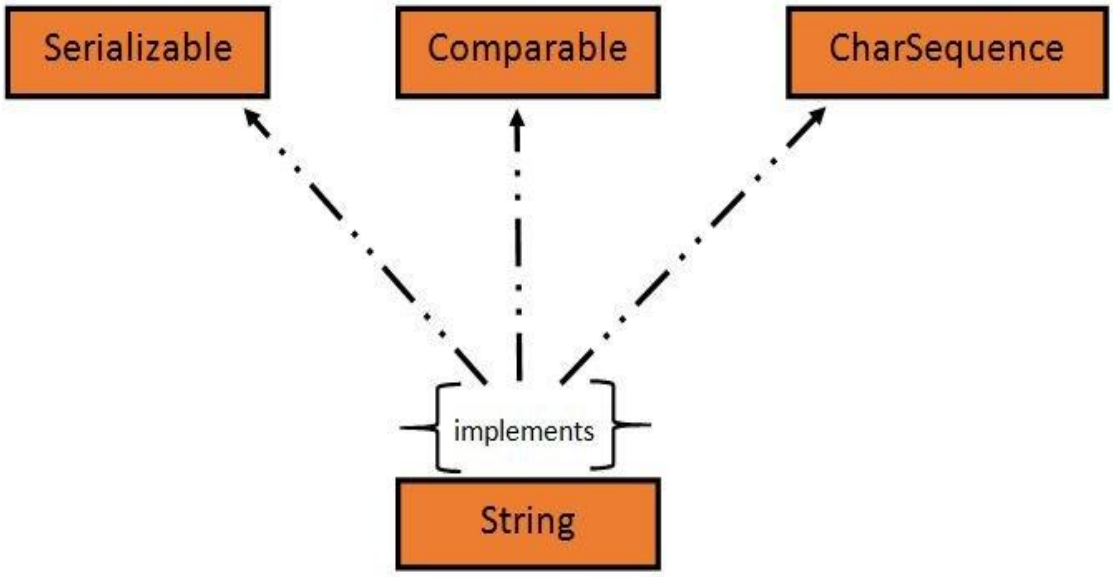
String Handling

- String is an object that represents sequence of characters. In Java, String is represented by String class which is located into java.lang package

- It is probably the most commonly used class in java library. In java, every string that we create is actually an object of type String. One important thing to notice about string object is that string objects are immutable that means once a string object is created it cannot be changed.

- The Java String class implements Serializable, Comparable and CharSequence interface that we have represented using the below image.



In Java, CharSequence Interface is used for representing a sequence of characters. CharSequence interface is implemented by String, StringBuffer and StringBuilder classes. This three classes can be used for creating strings in java.

What is an Immutable object?

An object whose state cannot be changed after it is created is known as an Immutable object. String, Integer, Byte, Short, Float, Double and all other wrapper classes objects are immutable.

Creating a String object

String can be created in number of ways, here are a few ways of creating string object.

1) Using a String literal

String literal is a simple string enclosed in double quotes " ". A string literal is treated as a String object.

```
public class StringDemo1
{
   public static void main(String[] args) {
        String obj = "Java Programming";
        System.out.println(obj);
   }
}
```

2)Using new Keyword

We can create a new string object by using new operator that allocates memory for the object.

```
public class StringDemo2{
   public static void main(String[] args) {
        String obj = new String("Java Programming");
        System.out.println(obj);
   }
}
```

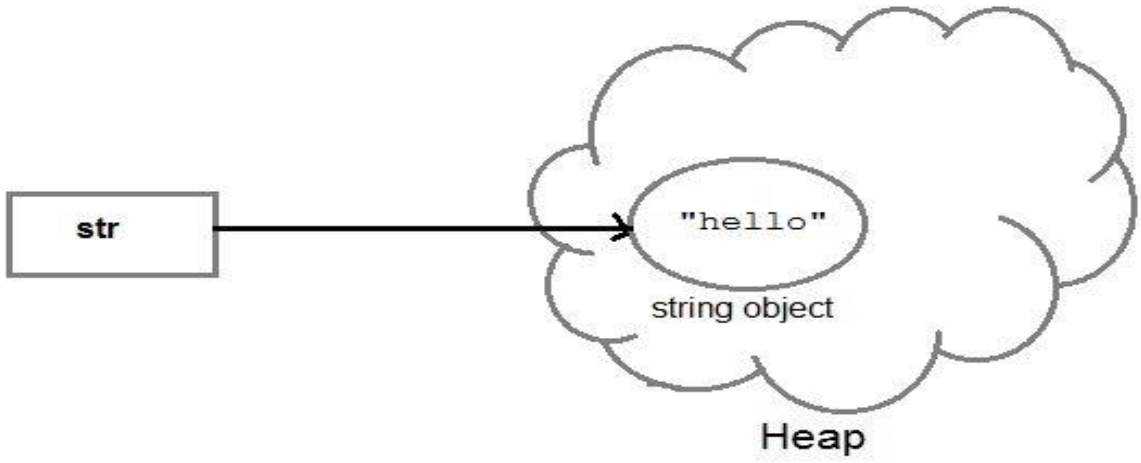
each time we create a String literal, the JVM checks the string pool first. If the string literal already exists in the pool, a reference to the pool instance is returned. If string

does not exist in the pool, a new string object is created, and is placed in the pool. String objects are stored in a special memory area known as string constant pool inside the heap memory.

String object and How they are stored?

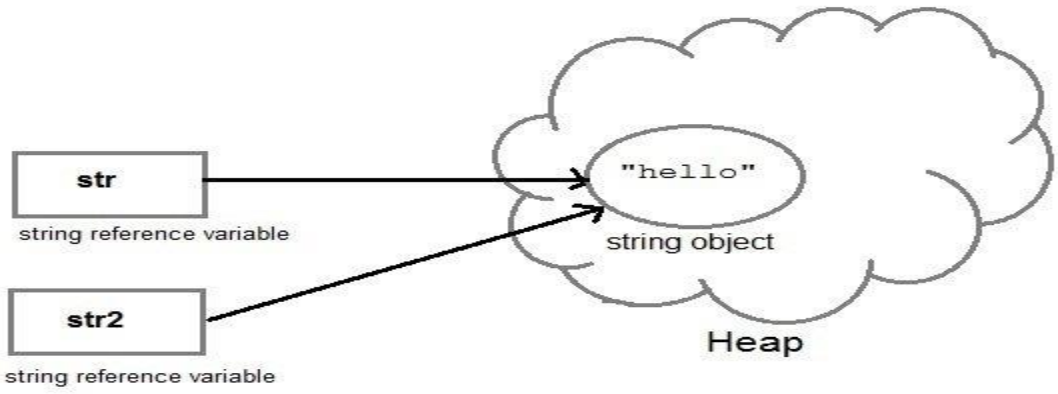
When we create a new string object using string literal, that string literal is added to the string pool, if it is not present there already.

String str= "hello";



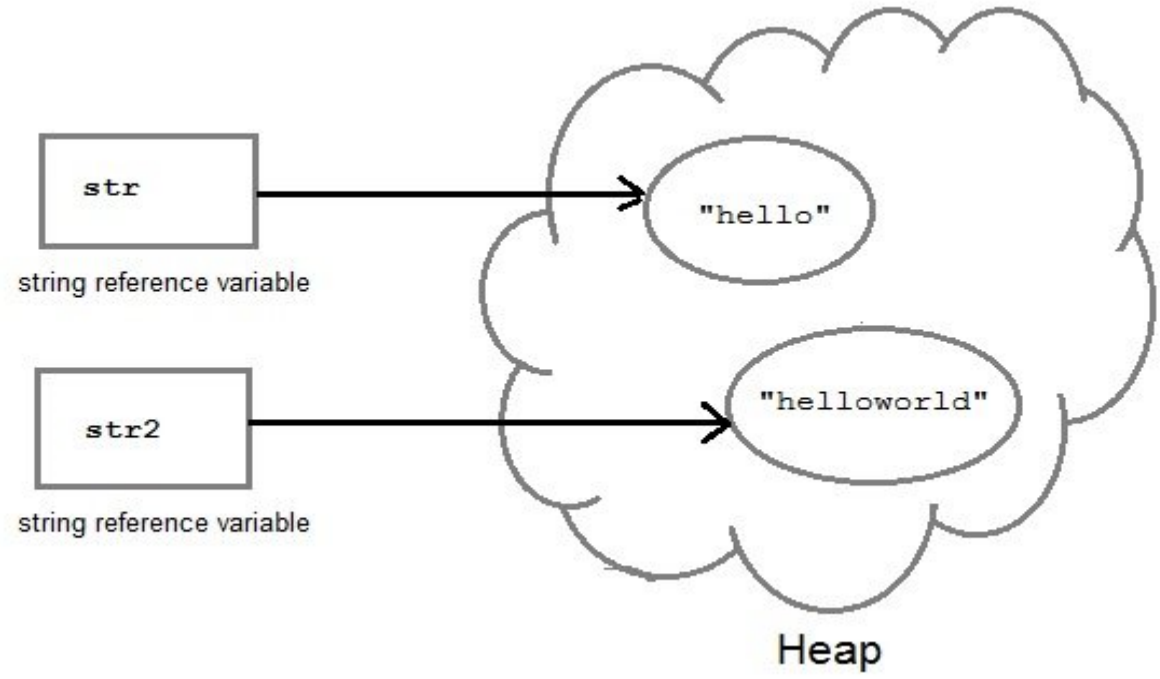
And, when we create another object with same string, then a reference of the string literal already present in string pool is returned.

String str2 = str;



But if we change the new string, its reference gets modified.

str2=str2.concat("world");



Concatenating String

There are 2 methods to concatenate two or more string.

- Using concat() method
- Using + operator

Using concat() method: Concat() method is used to add two or more string into a single string object. It is string class method and returns a string object.

```
public class StringDemo3{
        public static void main(String[] args) {
    String str1 = "Advance";
    String str2 = "Java";
    String str3 = str1.concat(str2);
    System.out.println(str3);
  }
}
```

Using + operator: Java uses "+" operator to concatenate two string objects into single one. It can also concatenate numeric value with string object. See the below example.

```
public class StringDemo4{
    public static void main(String[] args) {
        String str1 = "Hello";
        String str2 = "Java";
        String str3 = str1+str2;
        String str4 = "Java"+11;
        System.out.println(str3);
        System.out.println(str4);
    } }
```

String Comparison: To compare string objects, Java provides methods and operators both. So we can compare string in following three ways.

- Using equals() method
- Using == operator
- By CompareTo() method

Using equals() method: equals() method compares two strings for equality. Its general syntax : boolean equals (Object str)

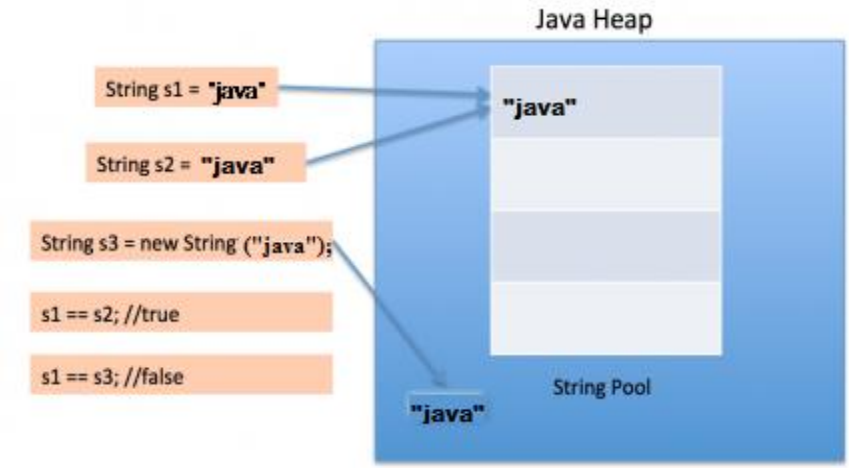
Example: It compares the content of the strings. It will return true if string matches, else returns false.

```
public class StringDemo5{       public static void main(String[] args) {
        String str1 = "Java";
        String str2 = "JAVA";
        String str3 = "JAVA";
        boolean b = str1.equals(str2);    //false
        System.out.println(b);
        b =     str2.equals(str3) ;   //true
        System.out.println(b);      }  }
```

Using == operator: The double equal (==) operator compares two object references to check whether they refer to same instance. This also, will return true on successful match else returns false.

```
public class StringDemo6{
   public static void main(String[] args) {
        String str1 = "Java";
        String str2 = "Java";
        String str3 = new String ("Java");
        boolean b = (s1 == s2);     //true
        System.out.println(b);
        b =     (s1 == s3);      //false
        System.out.println(b);
   }
}
```

Explanation: We are creating a new object using new operator, and thus it gets created in a non-pool memory area of the heap. s1 is pointing to the String in string pool while s3 is pointing to the String in heap and hence, when we compare s1 and s3, the answer is false.



By compareTo() method: String compareTo() method compares values and returns an integer value which tells if the string compared is less than, equal to or greater than the other string. It compares the String based on natural ordering i.e alphabetically.

G. Syntax: int compareTo(String str)

```
public class StringDemo7{
   public static void main(String[] args) {
        String s1 = "NERIST";
        String s2 = "NIRJULI";
        String s3 = "ABCXYZ";
        int a = s1.compareTo(s2);     //return -ve no because s1 < s2
        System.out.println(a);
        a = s1.compareTo(s3);     //return 0 because s1 == s3
        System.out.println(a);
        a = s2.compareTo(s1);     //return +ve no  because s2 > s1
        System.out.println(a);
   }
}
```

Java String class functions

- **charAt() method:** String charAt() function returns the character located at the specified index.

```
public class StringDemo8 {
   public static void main(String[] args) {
      String str = "MultiThreaded";
      System.out.println(str.charAt(0));
   }
}
```

Output: M

- **equalsIgnoreCase() method:** String equalsIgnoreCase() determines the equality of two Strings, ignoring their case (upper or lower case doesn't matter with this method).

```
public class StringDemo9 {
    public static void main(String[] args) {
        String str = "java";
        System.out.println(str.equalsIgnoreCase("JAVA"));   //true
    }
}
```

format() Method: String format() is a string method. It is used to the format of the given string.

Following are the format specifier and their datatype:

Format Specifier	Data Type
%a	floating point
%b	Any type
%c	character
%d	integer
%e	floating point
%f	floating point
%g	floating point
%h	any type
%n	none
%o	integer
%s	any type
%t	Date/Time
%x	integer

```
public class StringDemo10 {
    public static void main(String[] args) {
        String a1 = String.format("%d", 125);
        String a2 = String.format("%s", "JAVA");
        String a3 = String.format("%f", 125.00);
        String a4 = String.format("%x", 125);
        String a5 = String.format("%c", 'a');
        System.out.println("Integer Value: "+a1);
        System.out.println("String Value: "+a2);
        System.out.println("Float Value: "+a3);
        System.out.println("Hexadecimal Value: "+a4);
        System.out.println("Char Value: "+a5);
    }
}
```

Methods of Java String class

The java.lang.String class provides many useful methods to perform operations on sequence of char values.

No.	Method	Description
1	char charAt(int index)	It returns char value for the particular index
2	int length()	It returns string length
3	String substring(int beginIndex)	It returns substring for given begin index.
4	String substring(int beginIndex, int endIndex)	It returns substring for given begin index and end index.
5	boolean contains(CharSequence s)	It returns true or false after matching the sequence of char value.
6	boolean equals(Object another)	It checks the equality of string with the given object.

7	boolean isEmpty()	It checks if string is empty.
8	String concat(String str)	It concatenates the specified string.
9	String replace(char old, char new)	It replaces all occurrences of the specified char value.
10	String replace(CharSequence old, CharSequence new)	It replaces all occurrences of the specified CharSequence.
11	static String equalsIgnoreCase(String another)	It compares another string. It doesn't check case.
12	int indexOf(int ch)	It returns the specified char value index.
13	int indexOf(int ch, int fromIndex)	It returns the specified char value index starting with given index.
14	int indexOf(String substring)	It returns the specified substring index.
15	int indexOf(String substring, int fromIndex)	It returns the specified substring index starting with given index.
16	String toLowerCase()	It returns a string in lowercase.
17	String toUpperCase()	It returns a string in uppercase.
18	String trim()	It removes beginning and ending spaces of this string.