String in java

Generally, string is a sequence of characters. But in java, string is an object that represents a sequence of characters. String class is used to create string object.

How to create String object?

There are two ways to create String object:

1. By string literal
2. By new keyword

1. **By string literal**

Java String literal is created by using double quotes. For Example:

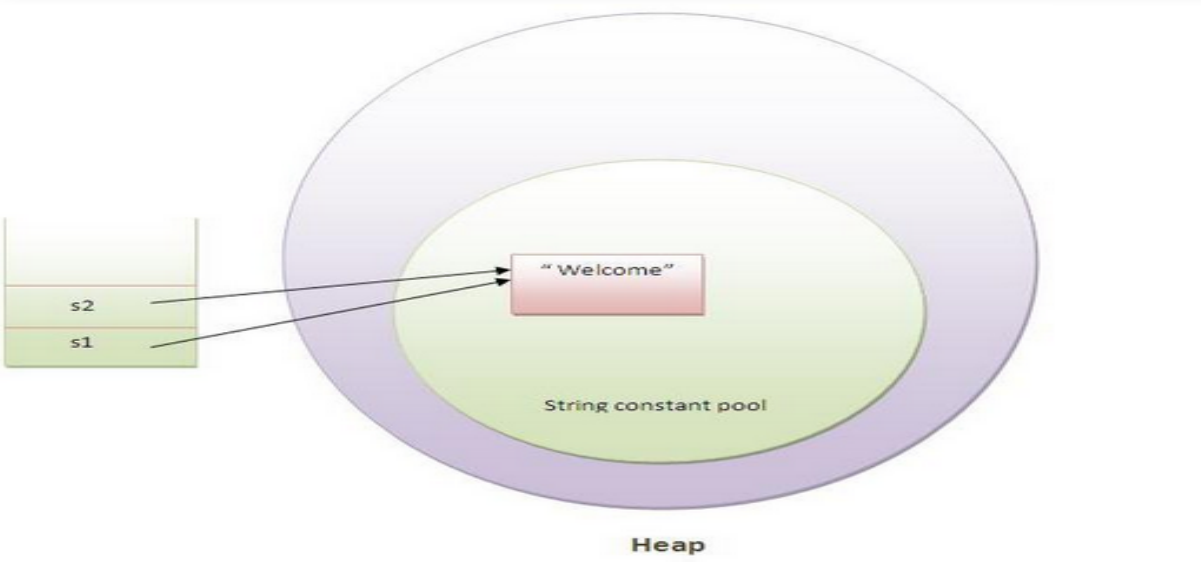
String s="welcome";

Each time you create a string literal, the JVM checks the string constant pool first. If the string already exists in the pool, a reference to the pooled object or instance is returned. If string doesn't exist in the pool, a new string object or instance is created and placed in the String constant pool.

For example:

String s1="Welcome";

String s2="Welcome";//will not create new instance



In the given example only one object will be created. Firstly JVM will not find any string object with the value "Welcome" in string constant pool, so it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create new object but will return the reference to the same instance.

Note: String objects are stored in a special memory area known as string constant pool.

Why java uses concept of string literal?

To make Java more memory efficient (because no new objects are created if it exists already in string constant pool).

2. **By new keyword**

String s=new String("Welcome");

In such case, JVM will create a new string object in normal (non pool) heap memory and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in heap (non pool).

Example:

```
public class StringExample{

public static void main(String args[]){


String s1=new String("example");//creating java string by new keyword

                                //this statement create two object i.e first object is created in heap
                                //memory area and second object is create in String constant pool.


System.out.println(s1);

}}
```

Immutable String in Java

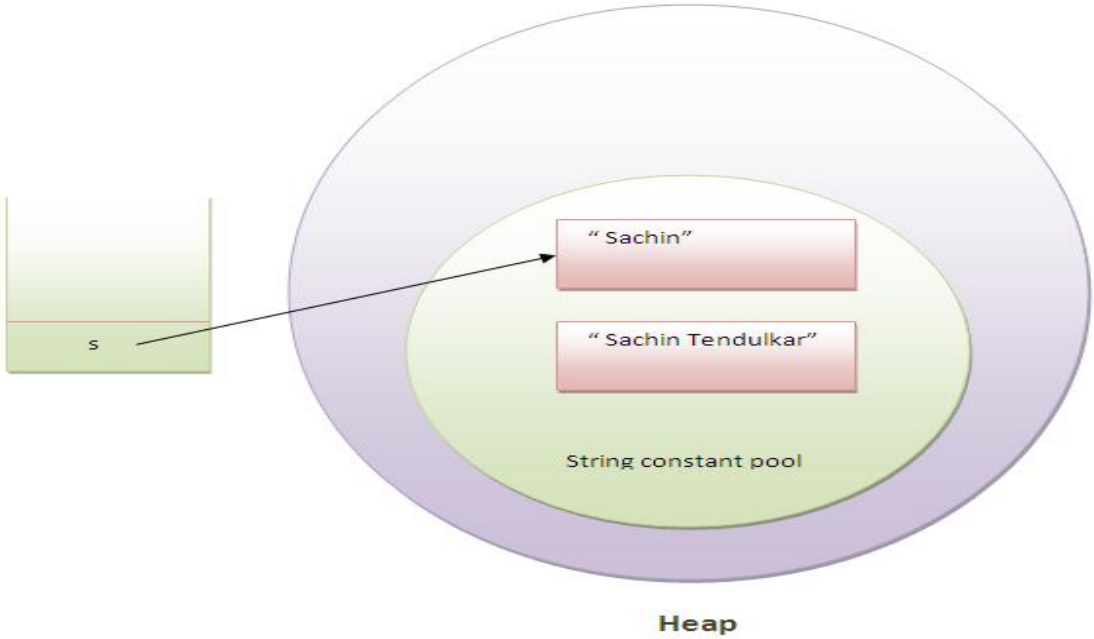
In java, string objects are immutable. Immutable simply means unmodifiable or unchangeable.

Once string object is created its data or state can't be changed but a new string object is created.

```
class Demo{
 public static void main(String args[]){
String s="Sachin";
s.concat(" Tendulkar");//concat() method appends the string at the end
System.out.println(s);//will print Sachin because strings are immutable objects
 }
}
```

o/p: Sachin

Now it can be understood by the diagram given below. Here Sachin is not changed but a new object is created with sachintendulkar. That is why string is known as immutable.



As you can see in the given figure that two objects are created but s reference variable still refers to "Sachin" not to "Sachin Tendulkar".

But if we explicitly assign it to the reference variable, it will refer to "Sachin Tendulkar" object.
For example:

```
class Demo{

 public static void main(String args[]){

String s="Sachin";

s=s.concat(" Tendulkar");

System.out.println(s);

 }

}
```

Why string objects are immutable in java?

Because java uses the concept of string literal. Suppose there are 5 reference variables, all referes to one object "sachin". If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java.

String class Methods/ Functions

1. **char charAt(int index) –** return the character value at the given index number.

   The index number starts from 0 and goes to n-1, where n is length of the string. It returns **StringIndexOutOfBoundsException** if given index number is greater than or equal to this string length or a negative number.

   Eg:

   ```
   class Demo{

    public static void main(String args[]){

   String str="Ram";

   System.out.println(str.charAt(0));  // R }}
   ```

2. **int length() –** returns the total number of character in the string.

   Eg:

```
class Demo{
 public static void main(String args[]){
String str="Ram Singh";
System.out.println(str.length());  // 9
}
}
```

3. **String concat(String str) –** combines specified string at the end of this string. It returns combined string. It is like appending another string.

   Eg:

```
class Demo{
 public static void main(String args[]){
String str="Manjeet";
 System.out.println(""+str.concat(" Kumar"));}
}
```

4. **boolean equals(String str) –** return true if this String contains the same character as str(including case) and false otherwise.

   Eg:

```
class Demo{
 public static void main(String args[]){
String str="Manjeet";
      System.out.println(""+str.equals("Manjeet"));
```

}} output: true

5. **int compareTo(String str) –** return an integer indication if this string is lexically before(a negative return value), equal to(a zero return value) or lexically after(a positive return value).

```
class Demo{
public static void main(String args[]){
String s1="hello";
String s2="hello";
String s3="meklo";
String s4="hemlo";
String s5="flag";
System.out.println(s1.compareTo(s2));//0 because both are equal
System.out.println(s1.compareTo(s3));//-5 because "h" is 5 times lower than "m"
System.out.println(s1.compareTo(s4));//-1 because "l" is 1 times lower than "m"
System.out.println(s1.compareTo(s5));//2 because "h" is 2 times greater than "f"
}}
```

6. **String replace(char oldchar, char newchar) –** return a new String that is identical with this String except that every occurrence of old-char is replaced by new-char.

```
class Demo{

 public static void main(String args[]){

String str="Manjeet";

System.out.println(""+str.replace('M','R'));}}
```
o/p: Ranjeet

7. **String toLowerCase() –** return a new string identical to this string. All uppercase letters are converted to their lowercase equivalent.

```
class Demo{

 public static void main(String args[]){

String str="MANJEET";

System.out.println(""+str.toLowerCase());}

}
```

o/p: manjeet

8. **String toUpperCase() –** return a new string identical to this string. All lowercase letters are converted to their uppercase equivalent.

```
class Demo{

 public static void main(String args[]){

String str="manjeet";

System.out.println(""+str.toUpperCase());

}

}
```

o/p: MANJEET

9. **substring**

A part of string is called substring. There are two signature of substring() method in Java.

I. **public String substring(int startIndex) –**

This method returns new String object containing the substring of the given string from specified startIndex.

```
class Demo{

 public static void main(String args[]){

String str="manjeet";

System.out.println(""+str.substring(3));

}

}
```

o/p: jeet

II. **public String substring(int startIndex,int endIndex) –**

This method returns new String object containing the substring of the given string from specified startIndex to endIndex – 1.

```
class Demo{

 public static void main(String args[]){

String str="manjeet";

System.out.println(""+str.substring(3,6));

}

}
```

o/p: jee

10. **String trim() –** This method omitted/eliminates leading and trailing space of the given string.
11. **split()**

The split() method splits the string against given regular expression and returns a array of Sting.

There are two signature of substring() method in Java.

A. public String[] split(String regex);

B. public String[] split(String regex, int limit);

Parameter

regex: regular expression to be applied on String.

limit: limit for the number of strings in array. If it is zero, it will returns all the Strings matching in regex.

1 .Eg: split the string on the basis of single while space character (\\s) or for multiple while space (\\s+)

```
class Demo{

 public static void main(String args[]){

String str="I am 5th semester B.Tech student";

    String s[]=str.split("\\s");
```

```
        int i;
        for( i=0;i<s.length;i++ )
        System.out.println(""+s[i]);
         }
    }
```

Output:

I

am

5th

semester

B.Tech

student

2.Eg: split the string on the basis of “/” character

```
    class Demo{
     public static void main(String args[]){
    String str="10/01/2020";
        String s[]=str.split("/");
        int i;
        for( i=0;i<s.length;i++ )
        System.out.println(""+s[i]);
         }
    }
```

Output:

10

01

2020

3.Eg: split the string on the basis of “,” and “?” character.(i.e for multiple regex)

```
class Demo{
 public static void main(String args[]){
String str="java . string , split method ? by Manjeet kumar";
        String s[]=str.split("[\\,\\?]");
        int i;
        for( i=0;i<s.length;i++ )
        System.out.println(""+s[i]);
         }
}
```

Output:

java . string

split method

by Manjeet kumar

4. eg

```
class Demo{
 public static void main(String args[]){
String str="10/01/2020";
        String s[]=str.split("/", 1 );
        int i;
        for( i=0;i<s.length;i++ )
        System.out.println(""+s[i]);
```

```
        }
    }
```

Output:

10/01/2020

5.eg

```
class Demo{
 public static void main(String args[]){
String str="10/01/2020";
    String s[]=str.split("/", 2 );
    int i;
    for( i=0;i<s.length;i++ )
    System.out.println(""+s[i]);
     }
}
```

Output:

10

01/2020

6.eg

```
class Demo{
 public static void main(String args[]){
String str="10/01/2020";
    String s[]=str.split("/", 3);
    int i;
    for( i=0;i<s.length;i++ )
    System.out.println(""+s[i]);
```

```
        }
    }
```

Output:

```
10
01
2020
```

12. **public char[] toCharArray()**

this method converts the String into character array, its length is similar to the string.

Eg:

```
class Demo{
 public static void main(String args[]){
String str="Manjeet";
      char ch[]=str.toCharArray();
      int i;
      for( i=0;i<ch.length;i++ )
      System.out.print(""+ch[i]);        }
}
```

Output:

Manjeet

Q:. Write a program to reverse String in java using toCharArray() method.

```
import java.io.*;
import java.util.*;
```

```
public class reverseString {
   public static void main(String[] args) {
      String input="";
      System.out.println("Enter the input string");
      try
      {
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         input = br.readLine();
         char[] try1= input.toCharArray();
         for (int i=try1.length-1;i>=0;i--)
         System.out.print(try1[i]);
      }
      catch (IOException e) {
         e.printStackTrace();
      }
}
}
```

Note: **length()** is a method of a String class; which returns the number of characters in the String whereas **.length** is a property of an array; will give the number of elements stored in the array.