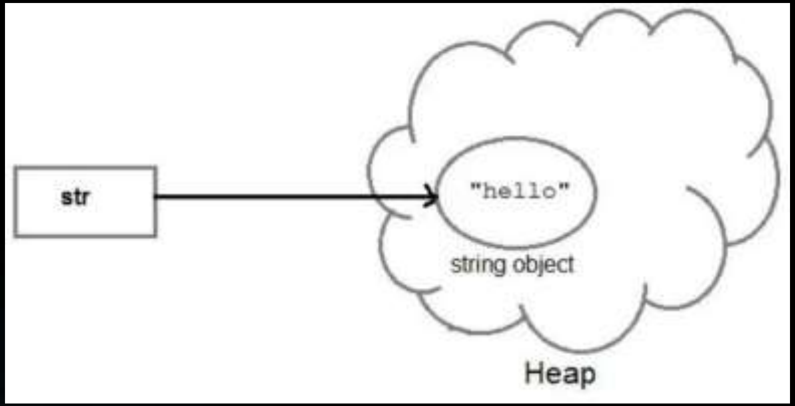
Strings

- String is an object that represents sequence of characters.
- Ex: "hello", "Hari Krishna"
- In Java, String is represented by String class which is available java.lang package
- 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.

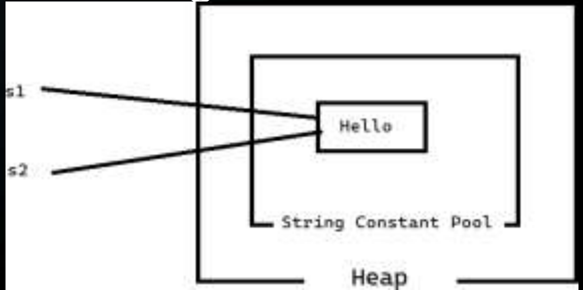
How to create String object in Java?

- -> To handle string data in Java, we need an object of string class. Basically, there are three ways to create a String object in Java.
  - By string literal.
  - By new keyword.
  - By converting character arrays into strings

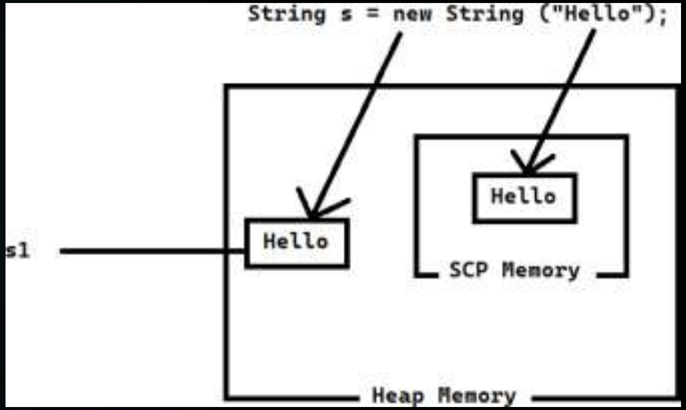
- working with String literal
- -> String literal in Java is created by using double quotes.
- String str = "hello";
- -> The string literal is always created in the string constant pool.



- -> In Java, String constant pool is a special area that is used for storing string objects.
- -> Whenever we create a string literal in Java, JVM checks string constant pool first. If the string already exists in string constant pool, no new string object will be created in the string pool by JVM.
- -> JVM uses the same string object by pointing a reference to it to save memory. But if string does not exist in the string pool, JVM creates a new string object and placed it in the pool.
- For example:
- String s1 = "Hello"; String s2 = "Hello";



- **Creating String Object by using new Keyword**
- -> The second way of creating an object to string class is by using new operator.
- -> It is just like creating an object of any class. It can be declared as follows:
- **String s = new String("Hello");**
- -> Whenever we will create an object of string class using the new operator, JVM will create two objects. First, it will create an object in the heap area and store string “Hello” into the object.
- -> After storing data into memory, JVM will point a reference variable s to that object in the heap. Allocating memory for string objects is shown in the below figure.



- -> Now JVM will create the second object as a copy for literal “Hello” in string constant pool for future purposes. There is no explicit reference variable pointing to the copy object in the pool but internally, JVM maintains the reference variable implicitly.
- -> Remember that the object created in the SCP area is not eligible for garbage collection because implicitly, the reference variable will be maintained by JVM itself.

<u>By converting Character Arrays into String</u>

- -> The third way to create strings is by converting the character arrays into string. Let’s take a character type array: arr[ ] with some characters as given below:

```
char arr[ ] = {'j','a','v','a'};
```

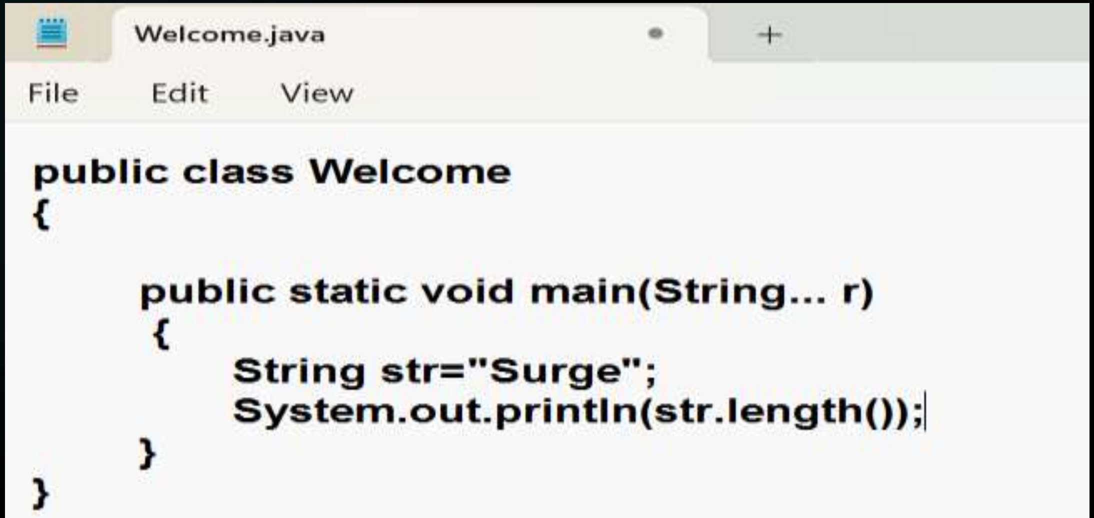
- -> Now create a string object by passing array name to string constructor like this:

```
String s = new String(arr);
```

- -> Now string object ‘s’ contains string “java”. It means that all the characters of the array are copied into string.
- -> If you do not want all the characters of the array into string then you can also mention which character you need, like this:

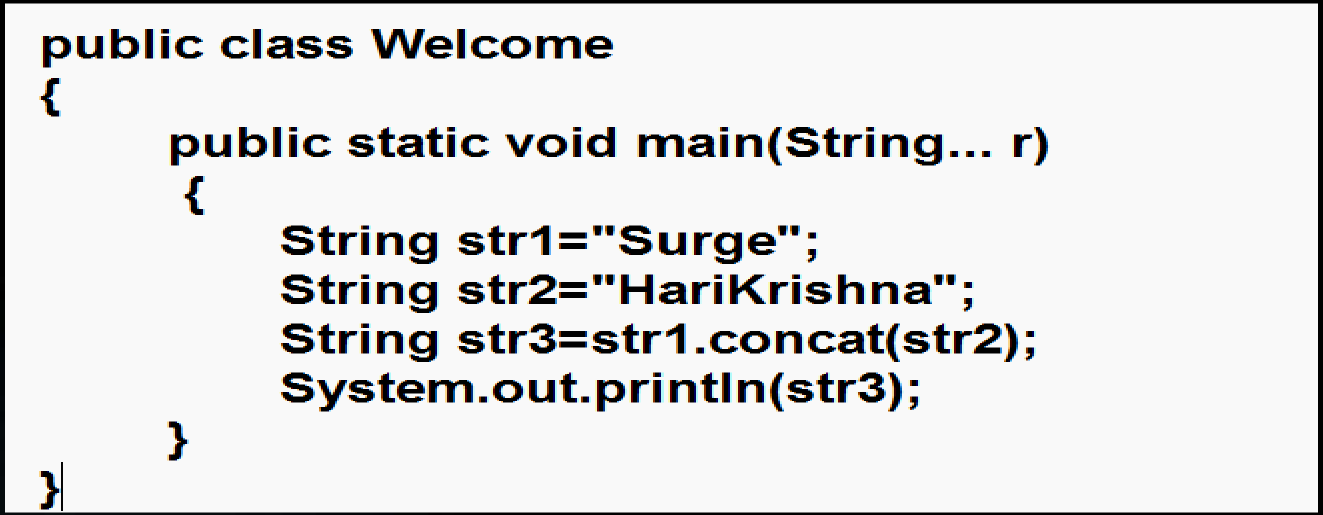
```
String s = new String(arr, 1,3);
```

- ->**<u>How many total objects will be created in memory for following string objects?</u>**
- String s1 = new String("Hari Krishna");
- String] s2 = new String("Hari Krishna");
- String s3= "Hari Krishna";
- String s4 = "Hari Krishna";
- **<u>String Manipulations</u>**
- String class provided several methods to perform operations on Strings
- *#1) Length*
- The length is the number of characters that a given string contains. String class has a length() method that gives the number of characters in a String.



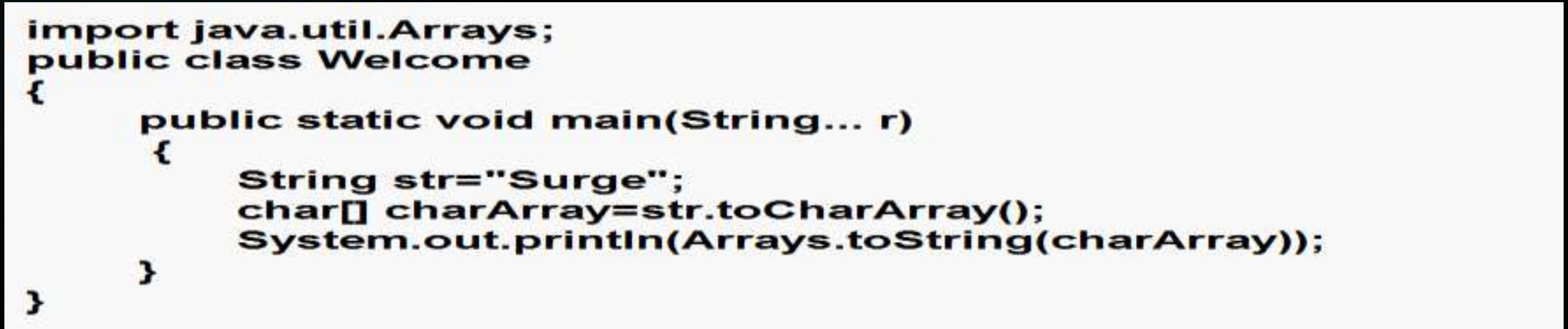
```
public class Welcome
{

        public static void main(String... r)
         {
                String str="Surge";
                System.out.println(str.length());
        }
}
```

- *#2) concatenation*
- Although Java uses a '+' operator for concatenating two or more strings. A concat() is an inbuilt method for String concatenation in Java.



```
public class Welcome
{
        public static void main(String... r)
         {
                String str1="Surge";
                String str2="HariKrishna";
                String str3=str1.concat(str2);
                System.out.println(str3);
        }
}
```

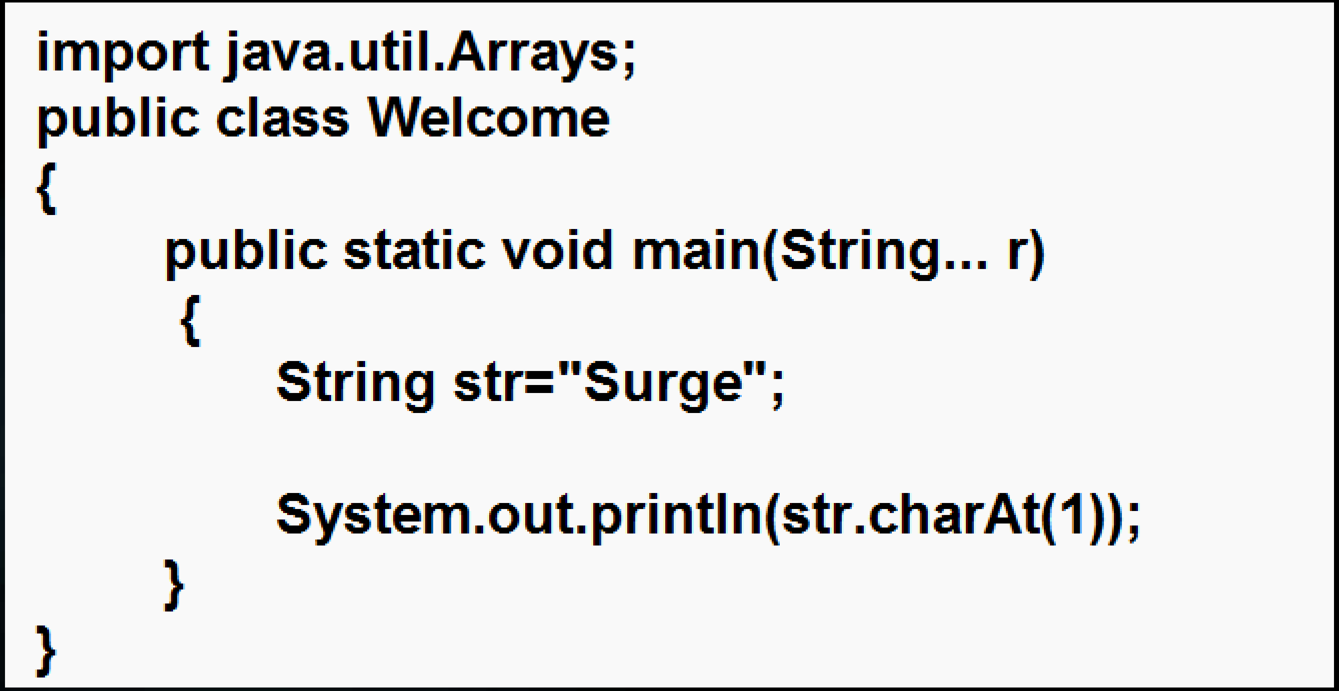
- *#3) String to CharArray()*
- This method is used to convert all the characters of a string into a Character Array. This is widely used in the String manipulation programs.



```
import java.util.Arrays;
public class Welcome
{
        public static void main(String... r)
         {
                String str="Surge";
                char[] charArray=str.toCharArray();
                System.out.println(Arrays.toString(charArray));
        }
}
```

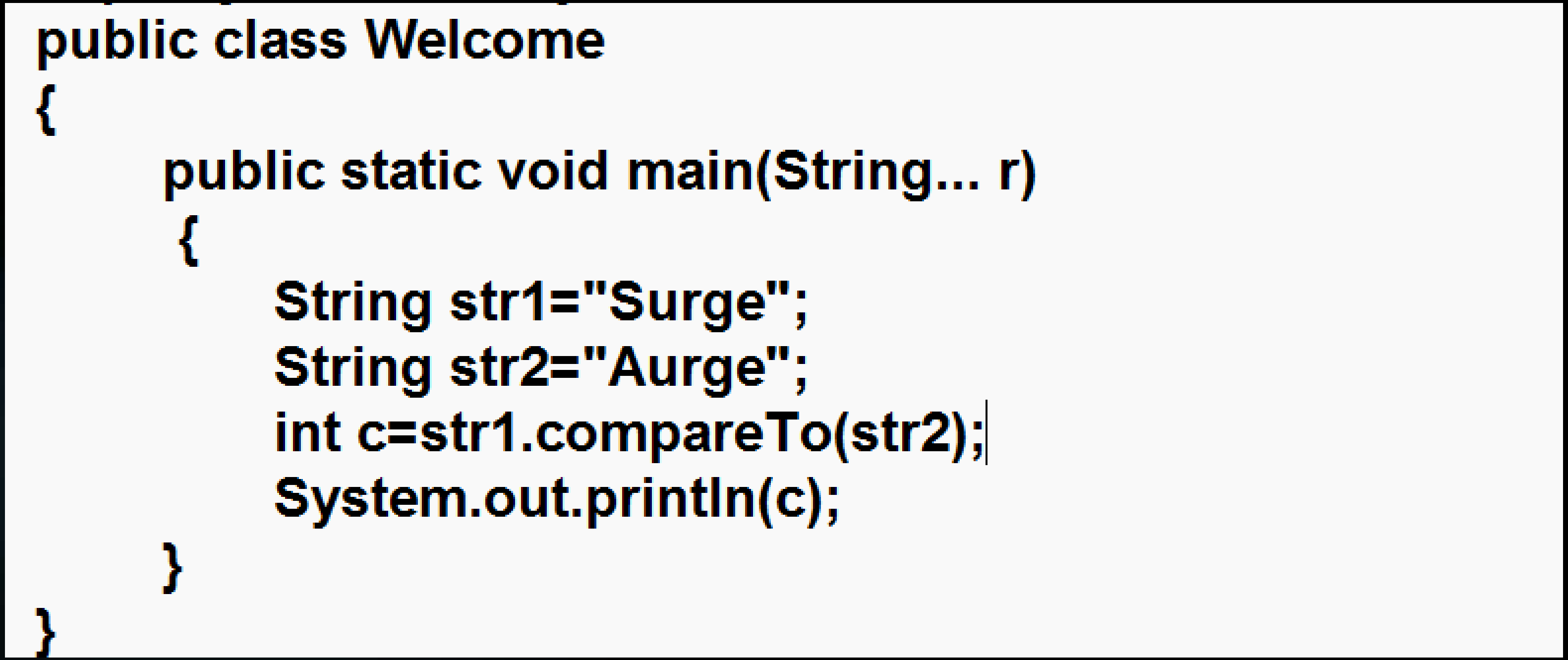
- *#4) String charAt()*
- *This method is used to retrieve a single character from a given String. The syntax is given as:*

  char charAt(int i);



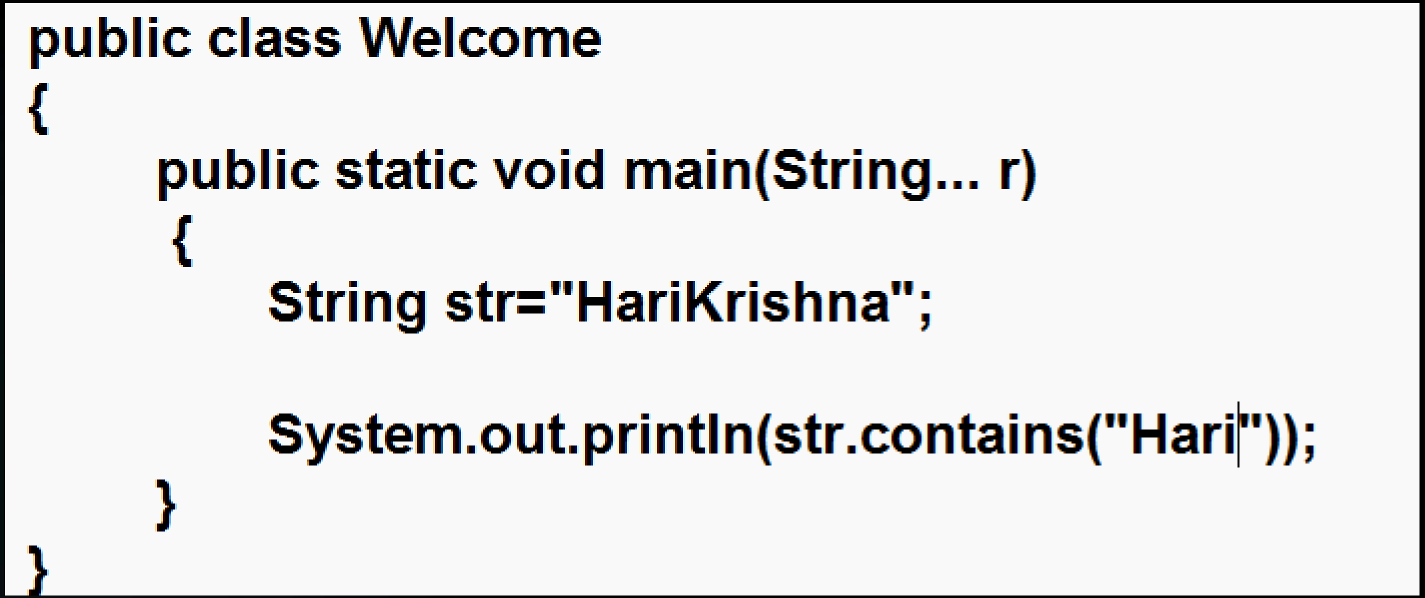
```
import java.util.Arrays;
public class Welcome
{
        public static void main(String... r)
         {
                String str="Surge";

                System.out.println(str.charAt(1));
        }
}
```

- *#5) Java String compareTo()*
- *This method is used to compare two Strings. The comparison is based on alphabetical order. In general terms, a String is less than the other if it comes before the other in the dictionary.*



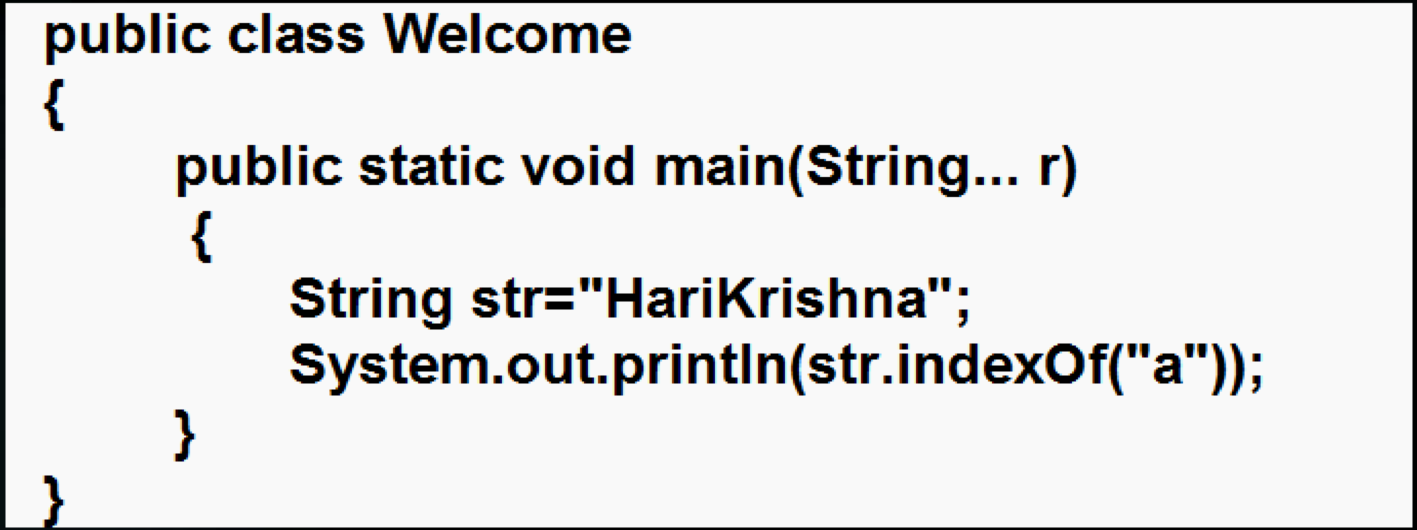
```
public class Welcome
{
        public static void main(String... r)
         {
                String str1="Surge";
                String str2="Aurge";
                int c=str1.compareTo(str2);
                System.out.println(c);
        }
}
```

- *#6) String contains()*
- *This method is used to determine whether a substring is a part of the main String or not. The return type is Boolean.*
- *For E.g. In the below program, we will check whether “Krishna” is a part of “Hari” or*
- *not andwe will also check whether “blog” is a part of “Hari Krishna”.*
-



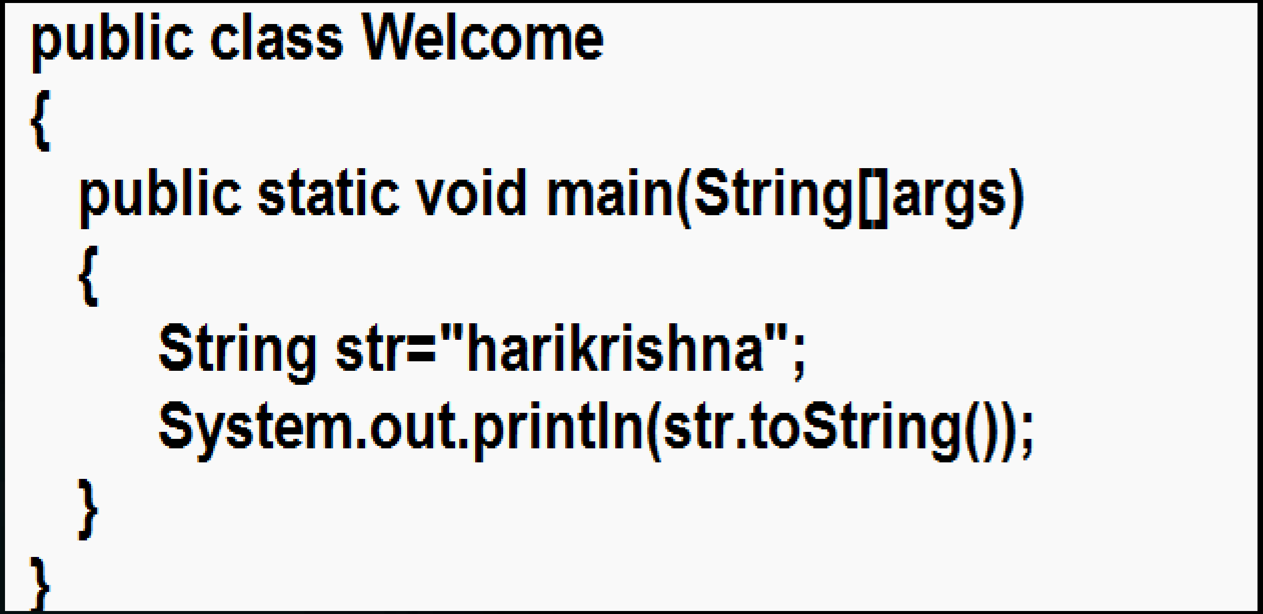
```
public class Welcome
{
        public static void main(String... r)
         {
              String str="HariKrishna";

              System.out.println(str.contains("Hari"));
        }
}
```

- *#7) Java String indexOf()*
- *This method is used to perform a search operation for a specific character or a substring on the main String. There is one more method known as lastIndexOf() which is also commonly used.*
- *indexOf() is used to search for the first occurrence of the character.*

  lastIndexOf() is used to search for the last occurrence of the character.
- 

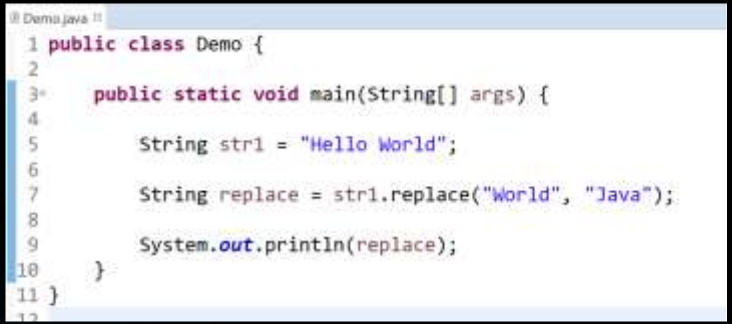
```
public class Welcome
{
          public static void main(String... r)
           {
                    String str="HariKrishna";
                    System.out.println(str.indexOf("a"));
          }
}
```

- *#8) Java String toString()*
- *This method returns the String equivalent of the object that invokes it. This method does not have any parameters. Given below is the program where we will try to get the String representation of the object.*



```
public class Welcome
{
   public static void main(String[]args)
   {
       String str="harikrishna";
       System.out.println(str.toString());
   }
}
```

- *#9) String replace ()*
- The replace() method is used to replace the character with the new characters in a String.



```
public class Demo {

    public static void main(String[] args) {

        String str1 = "Hello World";

        String replace = str1.replace("World", "Java");

        System.out.println(replace);
    }
}
```

- *StringBuffer Class*
- *-> StringBuffer class is used to create a mutable string object. It means, it can be changed after it is created.*
- *-> It is similar to String class in Java both are used to create string, but stringbuffer object can be changed.*
- *-> StringBuffer class is used when we have to make lot of modifications to our string.*
- *-> It is also thread safe i.e multiple threads cannot access it simultaneously.*

- **Java StringBuilder class**
- -> StringBuilder is identical to StringBuffer except for one important difference that it is not synchronized, which means it is not thread safe.
- -> StringBuilder also used for creating string object that is mutable and non synchronized.
- -> The StringBuilder class provides no guarantee of synchronization.
- -> StringBuffer and StringBuilder both are mutable but if synchronization is not required then it is recommended to use StringBuilder class.