Supplementary notes on String objects

There are two ways to create a string:

- String s1 = “Welcome”;   // String literal will go into StringPool
- String s3 = new String (“Welcome”);   //String object

For s3, the new String(String) initializes a newly created String object.

If now, you write String s2 = “Welcome”;
This time it check if "Welcome" literal is already available in the StringPool or not. As now it exists, so s2 will refer to the same literal referred to by s1.

Take a look at the example below:

```
public class StringEquals {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
	 String s1 = "Welcome";
	 String s2 = "Welcome";

	 String s3 = new String ("Welcome");
	 String s4 = new String ("Welcome");

	 System.out.println(s1 == s2);  // true
	 System.out.println(s3 == s4);  //false
	 System.out.println(s1.equals(s2));   // true
	 System.out.println(s3.equals(s4));   // true

	 String e = "JDK";
	 String f = new String("JDK");
	 System.out.println(e == f); // False
	}

}
```

What is difference between String literal and String object in Java?

String literals are returned from string pool and Java put them in pool if not stored already.

We use == operator for reference comparison (**address comparison**) and .equals() method for **content comparison**. That is, == checks if both objects point to the same memory location whereas .equals() compares the values.

It is suggested to use equals() method to compare two String objects and never compare them using == operator, because you never know which one is coming from the pool and which one is created using new() operator.