Computer Science Virtual Learning

# HS Computer Science A

April 21st, 2020

Lesson: String Equality

**Objective/Learning Target:**

Understanding why and when to use an equality operator when coding in Java

## String Equality

When the operator `==` is used with object variables it returns true when the two variables *refer to the same object*. With strings this happens when one string variable is set to another or when strings are set to the same string literal.

# String Equality

The code to the right will print `Bye` since s3 has been assigned to a copy of the value in s2 which is an object reference to the String object that has the characters “Bye” in it. In addition, `s2 == s3` will be true since the two variables refer to the same object. Also, `s2.equals(s3)` will also be true, again since the two variables refer to the same object, of course the characters will be the same.

Use `equals` to test if two strings have the same characters in the same order. Only use `==` to test if two strings refer to the same object. Most of the time you will want to use `equals` and not `==` with strings.

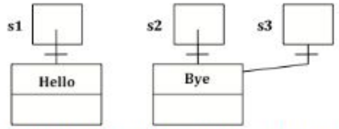
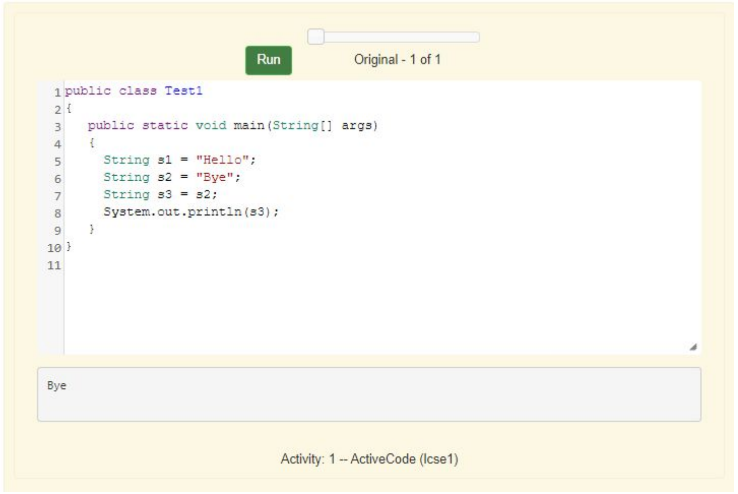


Figure 1: Several String variables with references to objects of the String class.



```
public class Test1
{
   public static void main(String[] args)
   {
     String s1 = "Hello";
     String s2 = "Bye";
     String s3 = s2;
     System.out.println(s3);
   }
}
```

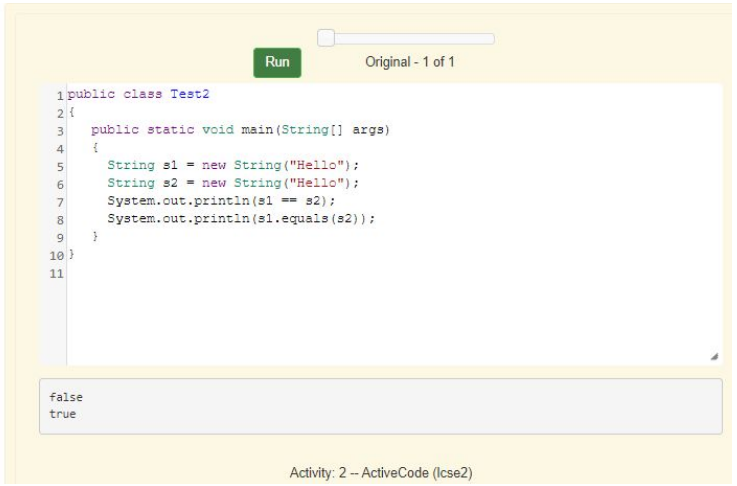
```
Bye
```

Activity: 1 -- ActiveCode (lcse1)

## Using the New Keyword

If you use the `new` keyword to create a string it will create a new string object. So, even if we create two string objects with the same characters using the new operator they will not refer to the same object.

In the code to the right, since we used the `new` keyword two different String objects will be created that each have the characters `Hello` in them. So `s1 == s2` will be false since they don't refer to the same object, but `s1.equals(s2)` is true since the two different object contain the same characters in the same order.



```java
public class Test2
{
  public static void main(String[] args)
  {
    String s1 = new String("Hello");
    String s2 = new String("Hello");
    System.out.println(s1 == s2);
    System.out.println(s1.equals(s2));
  }
}
```

```
false
true
```

Activity: 2 -- ActiveCode (lcse2)

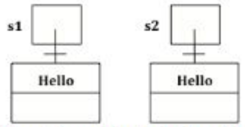


Figure 2: Two string variables and two string objects that contain the same characters in the same order.

## Using String Literals

In the code to the right, since we used string literals this time rather than the `new` keyword, the Java run-time will check if that string literal already exists as an object in memory, and if so reuse it. So `s1` and `s2` will refer to the same string object. That means that both `==` and `equals` will be true.

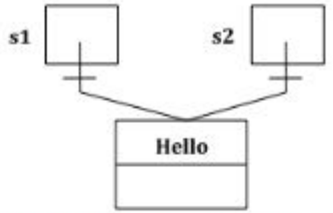
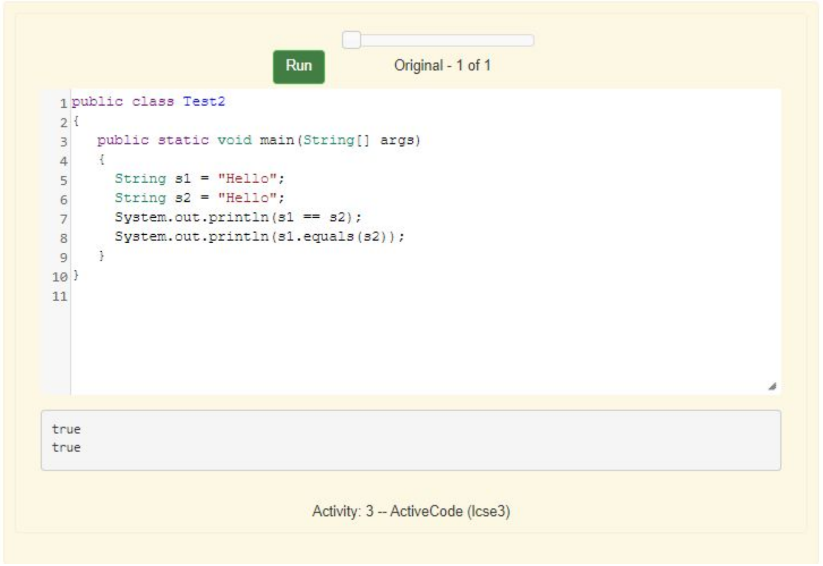


Figure 3: Two string variables that refer to the same string literal.



```java
public class Test2
{
  public static void main(String[] args)
  {
    String s1 = "Hello";
    String s2 = "Hello";
    System.out.println(s1 == s2);
    System.out.println(s1.equals(s2));
  }
}
```

```
true
true
```

Activity: 3 -- ActiveCode (lcse3)

## Check Your Understanding

1. Which of the following is true after the code executes?

```
String s1 = new String("hi");
String s2 = "bye";
String s3 = "hi";
s2 = s1;
```

a. s1 == s2 && s1 == s3
b. s1 == s2 && s1.equals(s3)
c. s1 != s2 && s1.equals(s3)
a.

2. Which of the following is true after the code executes?

```
String s1 = "hi";
String s2 = "bye";
String s3 = new String("hi");
```

a. s1 == s3 && s1.equals(s3)
b. s2.equals(s3) && s1.equals(s3)
c. !(s1 == s2) && !(s1 == s3)

## For More Resources and to Check Answers

Go to: https://runestone.academy/runestone/books/published/apcsareview/Strings/sEquality.html