Computer Science Virtual Learning

HS Computer Science A

April 20th, 2020

Lesson: What is a String?

Objective/Learning Target:

Understanding what a String is and the how to apply it using Java

Bell Ringer Activity

Watch the following video: Click Here

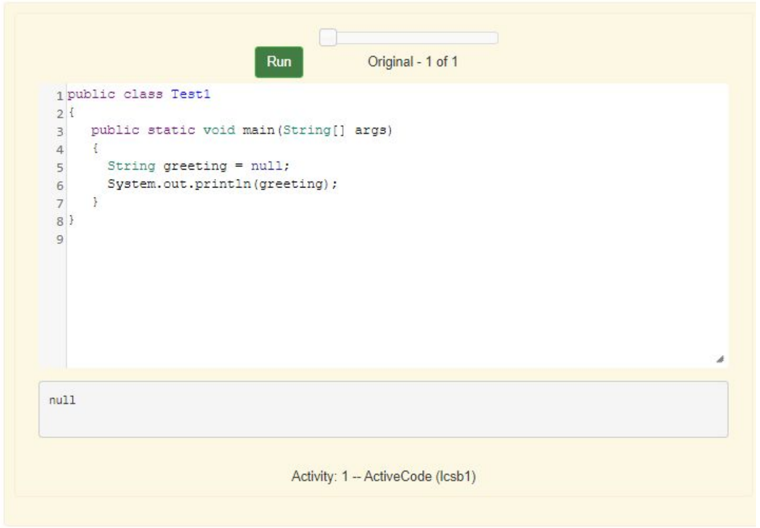
What is a String?

Strings in Java are objects of the `String` class that hold sequences of characters (a, b, c, $, etc). Remember that a class (or classification) in Java defines the data that all objects of the class have (the fields) and the behaviors, the things that objects know how to do (the methods).

Declaring and Creating Strings

You can declare a variable to be of type `String`.

Class names in Java, like String, begin with a capital letter. All primitive types: int, double, and boolean, begin with a lowercase letter. This is one easy way to tell the difference between primitive types and class types.



```java
public class Test1
{
   public static void main(String[] args)
   {
     String greeting = null;
     System.out.println(greeting);
   }
}
```

```
null
```

Activity: 1 -- ActiveCode (lcsb1)

Declaring and Creating Strings

The code to the right declares an object variable named `greeting` and sets the value of greeting to the Java keyword `null` to show that it doesn't refer to any object yet. So `System.out.println(greeting);` will print `null`. Object variables **refer** to objects in memory. A reference is a way to find the actual object, like adding a contact to your phone lets you reach someone without knowing exactly where they are. The value of greeting is null since the string object has not been created yet.

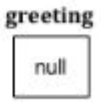
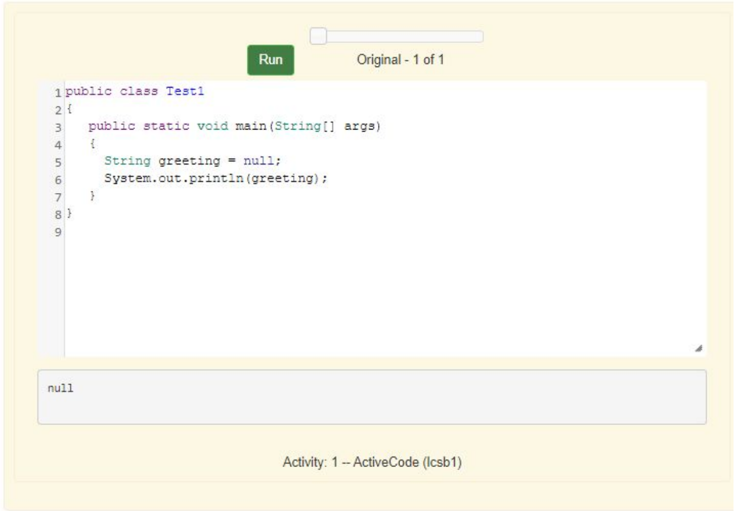


Figure 1: Initial value for an object reference



```
public class Test1
{
  public static void main(String[] args)
  {
    String greeting = null;
    System.out.println(greeting);
  }
}
```

```
null
```

Activity: 1 -- ActiveCode (lcsb1)

Declaring and Creating Strings

In Java there are two ways to create an object of the `String` class. You can use the `new` keyword followed by a space and then the class name and then in parentheses you can include values used to initialize the fields of the object. This is the standard way to create a new object of a class in Java.

```
greeting = new String("Hello");
```

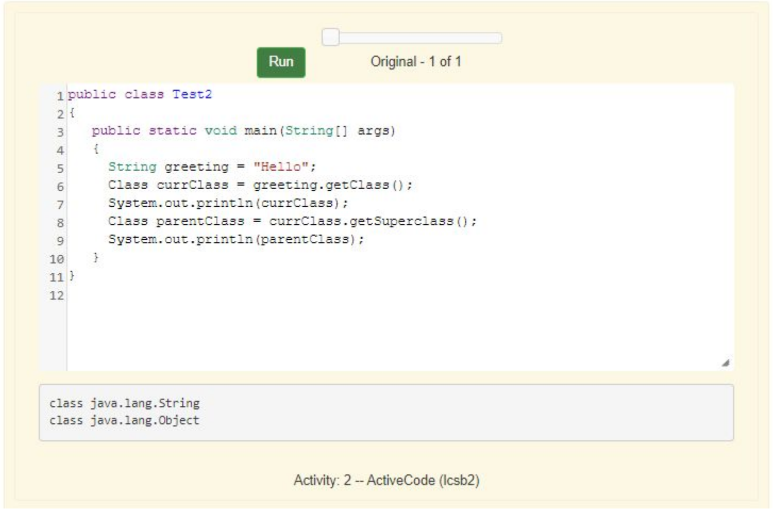
In Java you can also use just a **string literal**, which is a set of characters enclosed in double quotes (`"`), to create a `String` object.

```
greeting = "Hello";
```

In both cases an object of the `String` class will be created in memory and the value of the variable greeting will be set to an object reference, a way to find that object. Now that greeting refers to an actual object we can ask the object what class created it.

Declaring and Creating Strings

The code to the right first prints `class java.lang.String` since `greeting` was created by the `String` class. The full name for the `String` class is `java.lang.String`. The `java.lang` part is the **package** name. Every class in the Java language is in a package and the standard classes like `String` are in the `java.lang` package. Every object in Java contains a reference to the class that created it. Also, every class contains a reference to its **parent** class. Yes, a class can have a parent class, just as you have parents. But, in Java a class can only have one parent. A class can `inherit` object fields and methods from a parent class, just like you might inherit musical ability from a parent. The fourth line will print `class java.lang.Object` because the parent class (**superclass**) of the String class is the Object class. All classes in Java inherit from the Object class at some point in their ancestry.



```
public class Test2
{
  public static void main(String[] args)
  {
    String greeting = "Hello";
    Class currClass = greeting.getClass();
    System.out.println(currClass);
    Class parentClass = currClass.getSuperclass();
    System.out.println(parentClass);
  }
}
```

```
class java.lang.String
class java.lang.Object
```

Activity: 2 -- ActiveCode (lcsb2)

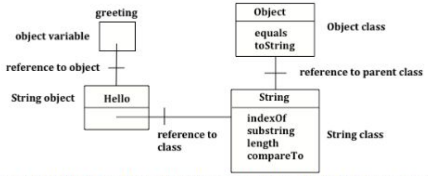


Figure 2: Object variable of type String with a reference to a String object which has a reference to the String class which has a reference to the Object class.

Sequence in Java

A string holds characters in a sequence. Each character is at a position or **index** which starts with 0 as shown below. An **index** is a number associated with a position in a string. The length of a string is the number of characters in it including any spaces or special characters. The string below has a length of 14.

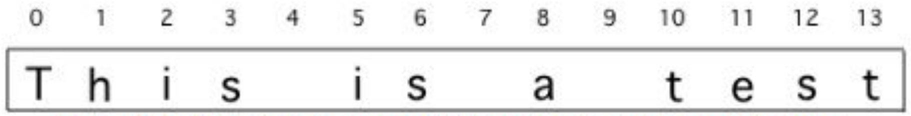


Figure 1: A string with the position (index) shown above each character

The first character in a string is at index 0 and the last characters is at the length - 1.

Check Your Understanding

1. What is the value of pos after the following code executes?

```
String s1 = "abccba";
int pos = s1.indexOf("b");
```

```
a.    2
b.    1
c.    4
d.   -1
```

2. What is the value of len after the following code executes?

```
String s1 = "baby";
int len = s1.length();
```

```
a.    2
b.    3
c.    4
d.    -1
```

3. What is the value of str2 after the following code executes?

```
String s1 = "baby";
String s2 = s1.substring(0,3);
```

```
a.    baby
b.    b
c.    ba
d.    bab
```

2. What is the value of s3 after the following code executes?

```
String s1 = "Hi";
String s2 = "Bye";
int answer = s1.compareTo(s2);
```

```
a.    positive(>0)
b.    0
c.    negative(<0)
```

For More Resources and to Check Answers

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

https://runestone.academy/runestone/books/published/apcsareview/Strings/sMethods.html