Tutorial: Java Data Types

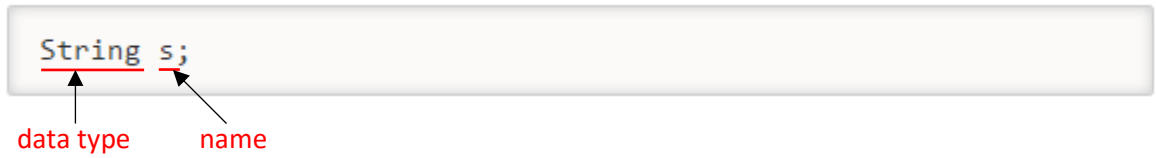
Goals:

- Learn to differentiate Java Data Types
- Understand String immutability

Description:

By now, you should be used to seeing a statement similar to the one below.

Variable Definition



In Java, all variables are defined with a **name** and a **data type**. In this tutorial, you will understand the significance of the data types and how they differ from each other.

Contents:

Section I: Data Type

Data types are necessary in the definition of a variable because they determine:

1. The **value** a variable may contain.
2. The **operations** that may be performed on it.

(***Source***: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html)

To illustrate the points above, take a look at the following code snippet.

Variable Value & Operations

```
String str1 = "hello"; // This value is allowed.
String str2 = hello; // This value is NOT allowed.
String str3 = "hello" + 2; // This operation is allowed. It yields "hello2".
String str4 = "hello" / 2; // This operation is NOT allowed.
```

In the case that you try to perform lines 2 and 4 in your program, you will receive a **compiler error**.

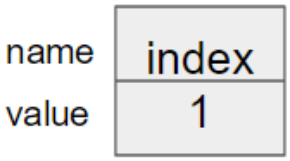
There are 2 data types in Java – **primitive** and **reference**. Refer to the table below for a list of data types supported by Java and the categories that they fall under.

Primitive Store *actual value*		**Reference** Store *address* of the object referred	
Data Type	**Example**	**Data Type**	**Example**
byte	byte b = 0;	Objects	String s = "hello"; Scanner sc = new Scanner(System.in);
short	short s = 0;	Arrays	int arr [] = new int[10];
Int	int i = 0;		
long	long l = 0;		
float	float f = 0.0;		
double	double d = 0.0;		
boolean	boolean t = true; boolean f = false;		
char	char c = 'a';		

The **primitive** and **reference** data types differ in the **values that they store**. Let us take a look at the illustration provided below to understand what this means.

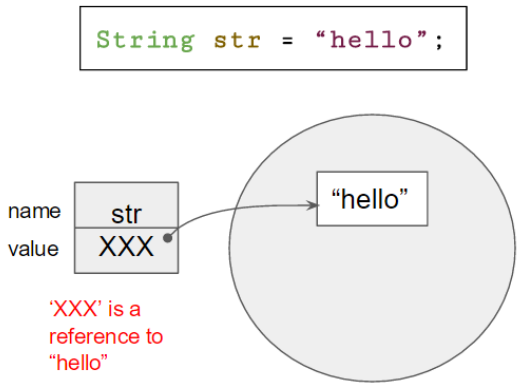
Primitive Data Type

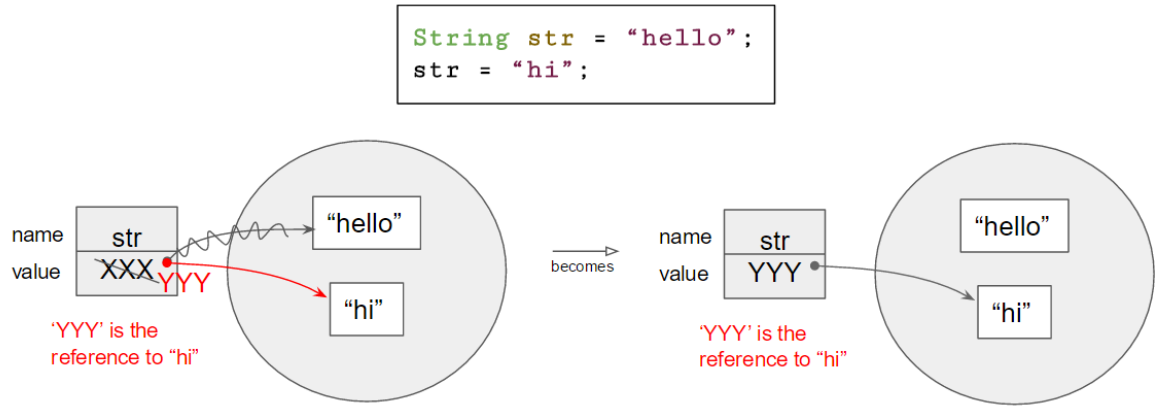
```
int index = 1;
```




```
int index = 1;
index = 2;
```

Reference Data Type





In other words, a primitive data type stores **the value it is assigned**, whereas a reference data type stores **the reference to the object it is assigned**. In this case, we say "*str points to the String object 'hi'*".

Section II: String Immutability

In order to use strings effectively in Java, you have to first understand the properties of the String object:

- **Data type**: A **String** variable is an **Object**, which is a **Reference** data type.
- **Immutability**: The values of a String **cannot be changed**.

Immutability

A String object is **immutable (not mutable)**. This means that once a String object is created, its state (data) cannot be changed. Now you may have seen examples where 2 String objects have been concatenated using the '+' operator:

String Concatenation

```
public class HelloWorld {
    public static void main(String[] args) {
        String hello = "Hello";
        hello = hello + " world!";
        JOptionPane.showMessageDialog(null, hello);
    }
}
```

and you think,

> *Hey, didn't the code above change the value of the **String hello** from "Hello" to "Hello World!"?*

In effect, what that piece of code did was:

- Create a **new String object** "Hello world!"
- Assign the **variable hello** the **reference** to "Hello world!"

Now the **variable hello** points to the String object "Hello world!", as illustrated below.