Java Blog

Part 2: Variables and Scanners

Henry Barthelemy and Evan Wologodzew, July 2020

2.1 - Variables:

In mathematics at school, oftentimes we work with a variable x and perhaps a variable y. Similarly, in programming we also work variables. We can give a variable any name and value we'd like. Later on we can refer back to that variable. Essentially, the variable becomes a placeholder for the value of that variable.

For example, I can tell you that x is equal to five and later on when I say x, you know that means five. Additionally, I could have a variable, lets call it z, that is equal to "Henry" . Although these are both variables, one holds a number and the other variable holds a set of characters. In Java, we refer to the set of characters as a String variable, and numbers as integers.

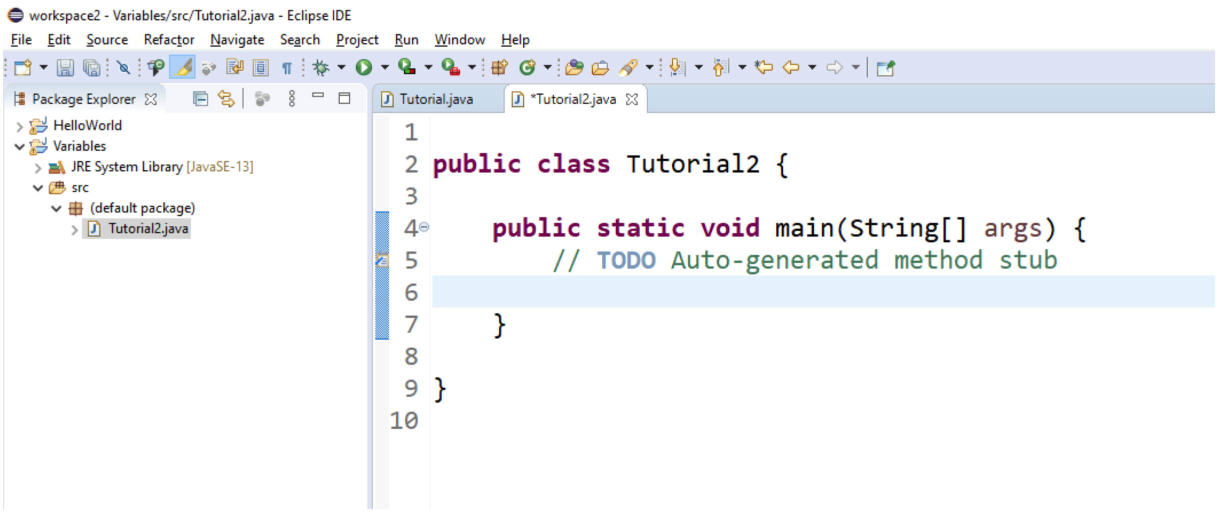
There are many different types of Java variables. Of these types the most important ones to remember are:

Variable Type	**What is holds**	**Example**
boolean	Either a true or false value	true
int	An integer, which is a number without a decimal place that can be positive or negative.	-3
double	Any number, including decimals	3.1415
String	Words, phrases, must be saved in	"Hello there"

	quotes	
char	A single character, held in single quotes	'c'

Let's experiment with these! Open eclipse and create a new Java project. If you are confused on how, read through Part 1 of this blog. I'm going to name mine "Variables" but feel free to name yours whatever you'd like. Afterwards, create a new class in the src folder. Be sure to check the first check mark to auto create the main method. Name the class anything you'd like.

You should end up with something similar to this.


workspace2 - Variables/src/Tutorial2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
HelloWorld
Variables
JRE System Library [JavaSE-13]
src
(default package)
Tutorial2.java
Tutorial.java
*Tutorial2.java

```

public class Tutorial2 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}

```

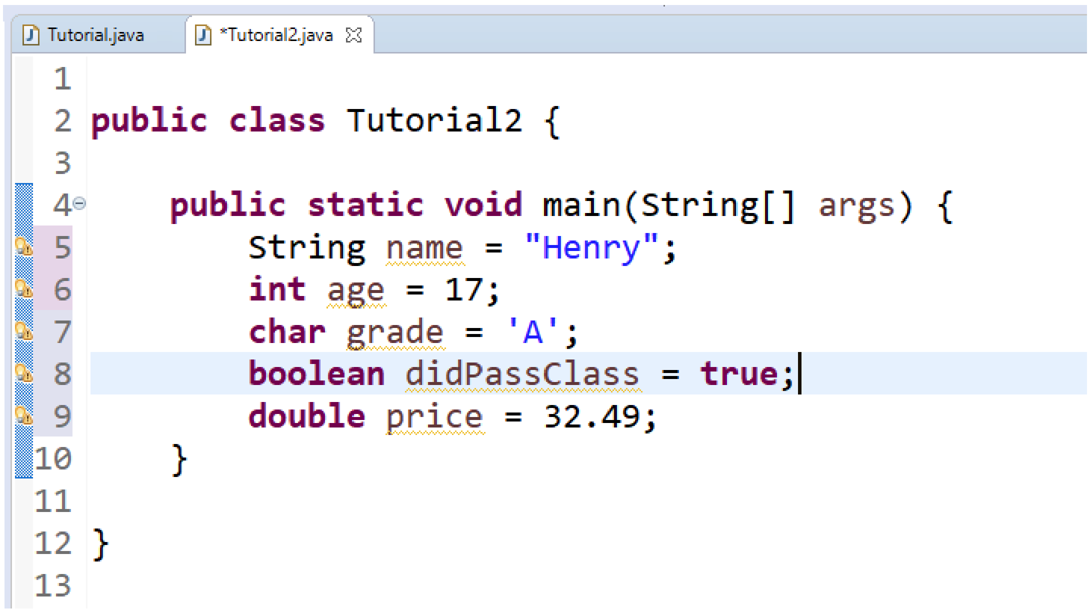
We are going to create a bunch of different types of variables.
The syntax for variables is:

{type of variable} {variable name} = {value of variable};

For example:

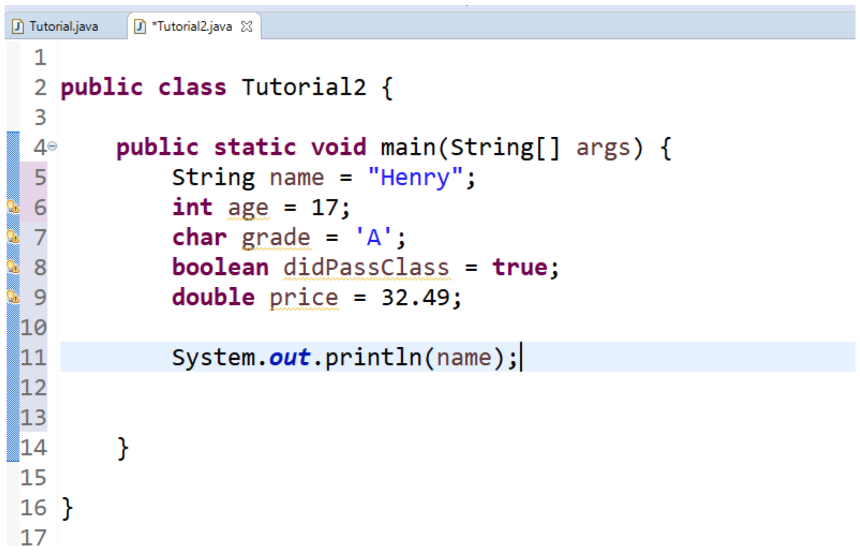
int x = 5;
String name = "Henry";

Lets create a variable of every type.

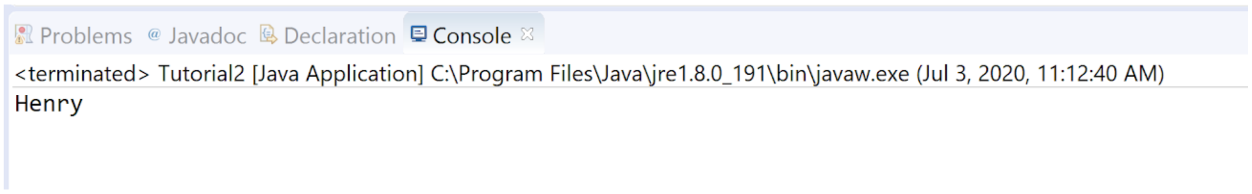

Tutorial.java | *Tutorial2.java

```java

public class Tutorial2 {

    public static void main(String[] args) {
        String name = "Henry";
        int age = 17;
        char grade = 'A';
        boolean didPassClass = true;
        double price = 32.49;
    }

}

```

I create a String, int, char, boolean, and double variable. Once a variable is created, you can use it in the reset of the class/method depending on how you create it. Local variables can only be used within the section it is created in. In our case, we create 5 local variables. We can access these variables anywhere in the main method. By printing out a line with my variable name, it would be equivalent to putting “Henry” in the print call. Name simply becomes a placeholder for my name. If I print out the following code I will just get Henry displayed on the console.


Tutorial.java | *Tutorial2.java

```java

public class Tutorial2 {

    public static void main(String[] args) {
        String name = "Henry";
        int age = 17;
        char grade = 'A';
        boolean didPassClass = true;
        double price = 32.49;

        System.out.println(name);


    }

}

```

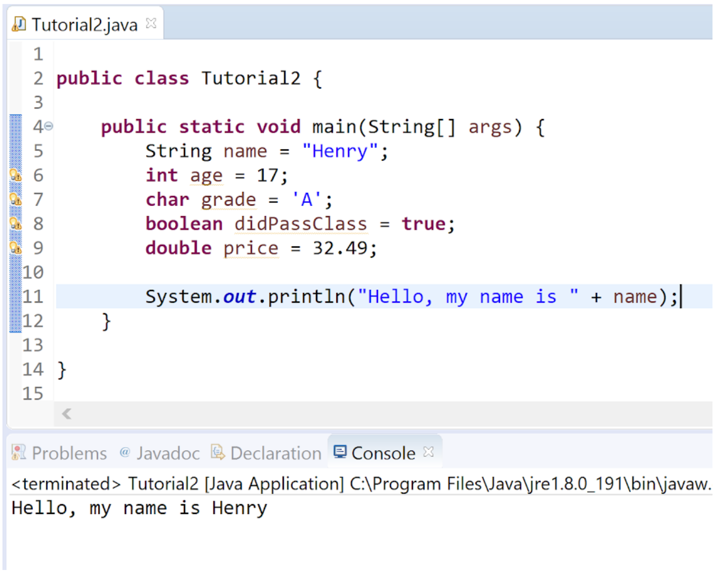


```
<terminated> Tutorial2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.exe (Jul 3, 2020, 11:12:40 AM)
Henry
```

Additionally, you can attach strings together, this is called concatenation. For example if you concatenate the two strings, “snow” and “man” together, you would get “snowman”.

By concatenating “Hello, my name is” and the name variable inside the print statement, it allows us to properly introduce ourselves.

System.out.println("Hello, my name is " + name);



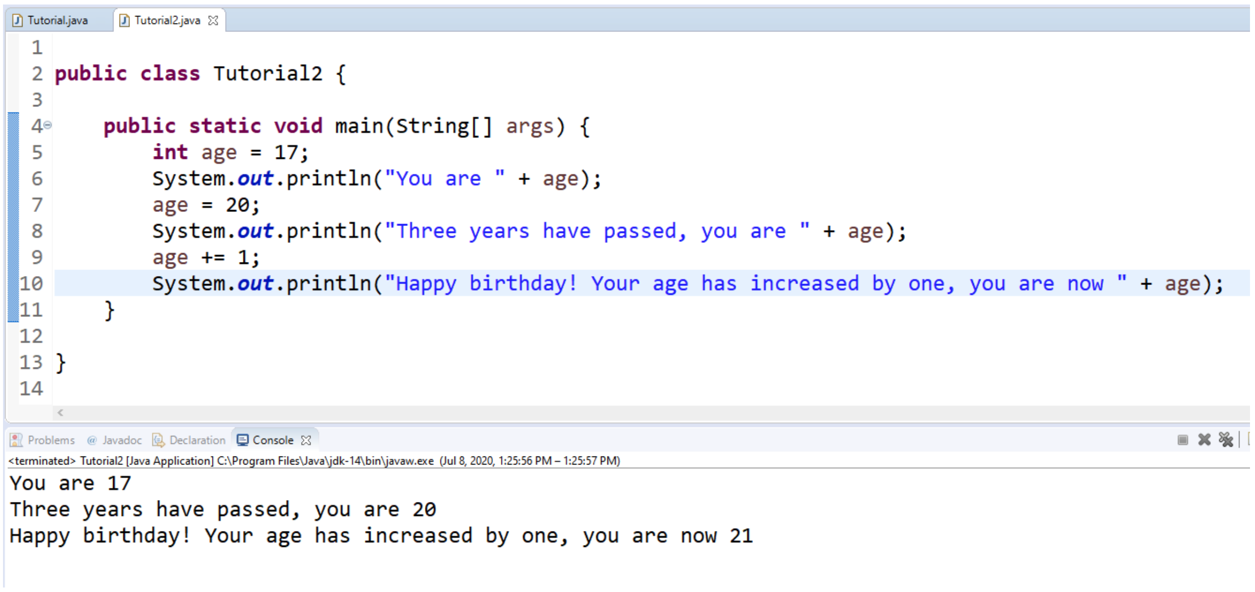
```java

public class Tutorial2 {

    public static void main(String[] args) {
        String name = "Henry";
        int age = 17;
        char grade = 'A';
        boolean didPassClass = true;
        double price = 32.49;

        System.out.println("Hello, my name is " + name);
    }

}

```

```
<terminated> Tutorial2 [Java Application] C:\Program Files\Java\jre1.8.0_191\bin\javaw.
Hello, my name is Henry
```

The output would be “Hello my name is Henry” as seen above. However remember that concatenation will slap one string of words directly on another. If we had: **System.out.println("Hello, my name is" + name);** The output would be “Hello, my name isHenry” with no space between “is” and “Henry”.

Practice creating different types of variables and printing them. Experiment, hit run, and see what the output is.

Variables can also be changed after being initialized.
In this case I initialize an integer called age and set its beginning value to 17. By writing **Age = 20;** The age changes to 20. Afterwards I say **age += 1;** which increases the age by 1.



```java

public class Tutorial2 {

    public static void main(String[] args) {
        int age = 17;
        System.out.println("You are " + age);
        age = 20;
        System.out.println("Three years have passed, you are " + age);
        age += 1;
        System.out.println("Happy birthday! Your age has increased by one, you are now " + age);
    }

}

```

```
<terminated> Tutorial2 [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Jul 8, 2020, 1:25:56 PM – 1:25:57 PM)
You are 17
Three years have passed, you are 20
Happy birthday! Your age has increased by one, you are now 21
```

Play around with changing and printing variables until you are comfortable with it. Try adding variables and concatenating strings together.

2.2 - Scanner:

Another type of variable is create an instance variable. This is an instance of an object, which is saved to a variable. The instance variable allows us to use the methods that the object has written in them. We will get more into objects later on, but for now just think of them like special variables.

Until now we’ve only displayed things, but it's important to also take input from the user. An object we want to work with to accomplish this is the Scanner object. This scanner object will allow us to read what the user types into the console and use it.

The way we create an instance variable is similar to the other variables we have created.

{Name of object class} {instance variable name} = new {Name of object class}(*);

> *: the parentheses will be filled with whatever information the object requires to create an instance of it. What is inside these parentheses is called parameters.

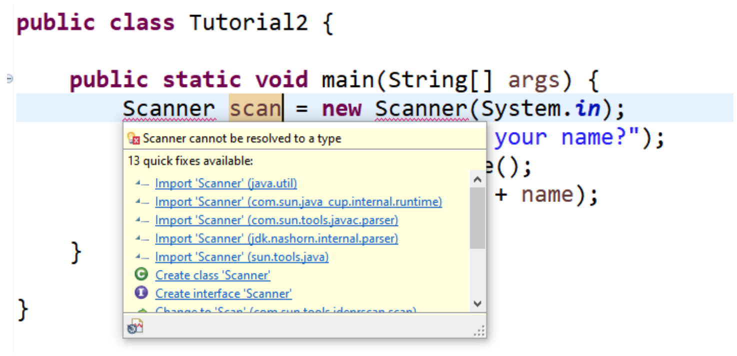
For example to create an instance of the scanner class we would write:

```
Scanner scan = new Scanner(System.in);
```

We could create as many instances of the Scanner object as we wanted, as long as we named it something new. For example if we wanted we could create another scanner by typing.

```
Scanner scan2 = new Scanner(System.in);
```

Both scanners have a parameter of System.in, this is just the way the Scanner object is created, System.in is code that java has prewritten for us to handle user input.



```
public class Tutorial2 {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
                                   your name?");
                                  e();
                                  + name);

    }

}
```

```
Scanner cannot be resolved to a type
13 quick fixes available:
Import 'Scanner' (java.util)
Import 'Scanner' (com.sun.java_cup.internal.runtime)
Import 'Scanner' (com.sun.tools.javac.parser)
Import 'Scanner' (jdk.nashorn.internal.parser)
Import 'Scanner' (sun.tools.java)
Create class 'Scanner'
Create interface 'Scanner'
```

In creating a Scanner variable you may get a red line below Scanner, do not fret, we simply need to import the correct Java code. Hover over the red word and choose Import 'Scanner' (java.util)

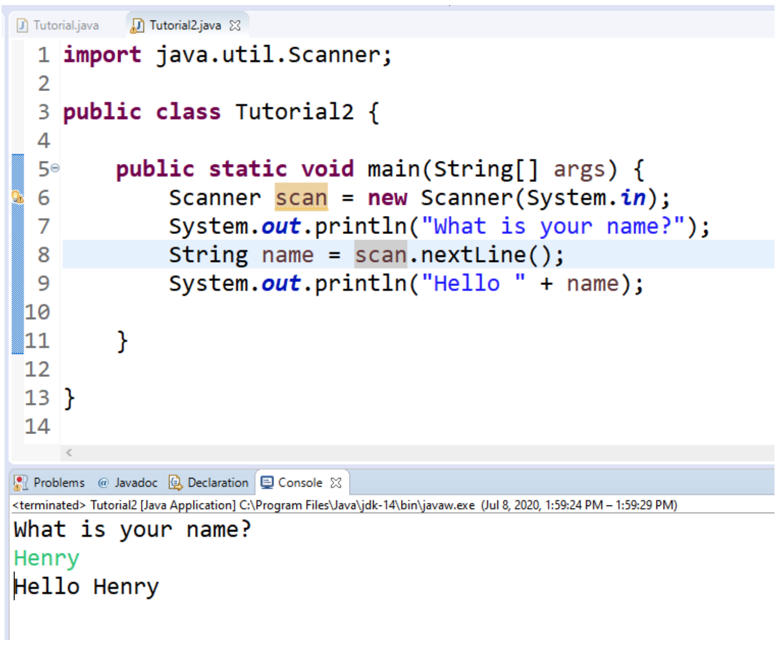
You'll notice an import line at the top of your class now. This simply tells java that we will be using the pre-written Scanner object in our code.

By using **scan.nextLine();** we can grab whatever someone types into the console and save it as a string. We can save it to a string variable by writing

String [any name] = scan.nextLine();

where [any name] is the name of the variable. We can then use the new variable however we'd like.

This is the full code for a program that utilizes this. The first line imports the scanner which allows us to use it in our class. On line #6 we create a Scanner variable and name it scan. Afterward we ask the user what is your name?. Whatever you type into the console (hit return after you finish typing) will be saved as a String variable called "name". On the following line we say Hello followed by this name variable.



```
 1 import java.util.Scanner;
 2 
 3 public class Tutorial2 {
 4 
 5     public static void main(String[] args) {
 6         Scanner scan = new Scanner(System.in);
 7         System.out.println("What is your name?");
 8         String name = scan.nextLine();
 9         System.out.println("Hello " + name);
10 
11     }
12 
13 }
14 
```

```
<terminated> Tutorial2 [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Jul 8, 2020, 1:59:24 PM – 1:59:29 PM)
What is your name?
Henry
Hello Henry
```

Challenge:

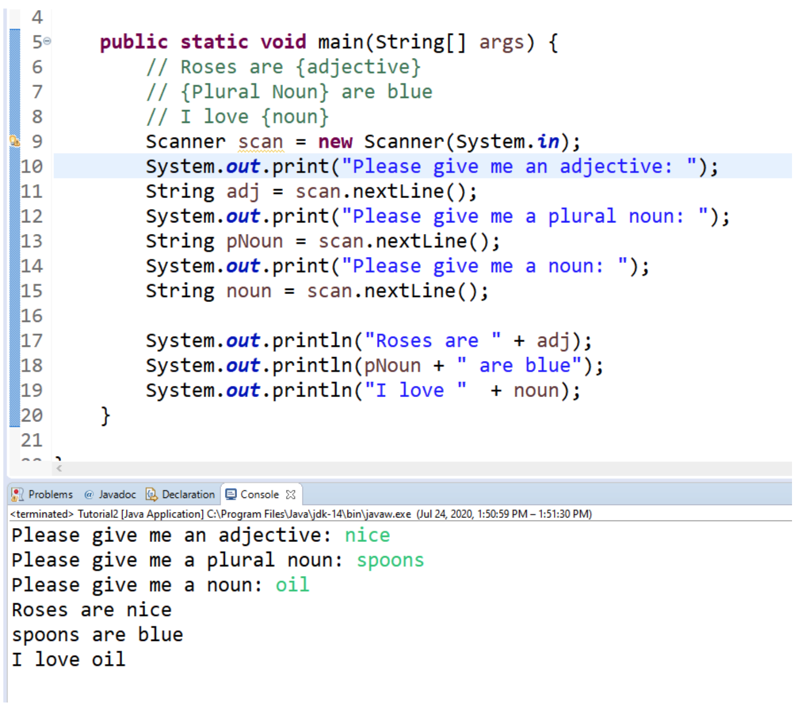
Using what you learned by reading this tutorial, create your own mad libs game.

Create a poem missing some words and prompt the user for a noun/verb/adjective to fill it in.

SPOILER: ANSWER

This will be my madlibs poem, feel free to use it or write your own story and/or poem.

Roses are {adjective}
{Plural Noun} are blue
I love {noun}



```
    public static void main(String[] args) {
        // Roses are {adjective}
        // {Plural Noun} are blue
        // I love {noun}
        Scanner scan = new Scanner(System.in);
        System.out.print("Please give me an adjective: ");
        String adj = scan.nextLine();
        System.out.print("Please give me a plural noun: ");
        String pNoun = scan.nextLine();
        System.out.print("Please give me a noun: ");
        String noun = scan.nextLine();

        System.out.println("Roses are " + adj);
        System.out.println(pNoun + " are blue");
        System.out.println("I love "  + noun);
    }
```

```
<terminated> Tutorial2 [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Jul 24, 2020, 1:50:59 PM – 1:51:30 PM)
Please give me an adjective: nice
Please give me a plural noun: spoons
Please give me a noun: oil
Roses are nice
spoons are blue
I love oil
```

Here is my completed code for the mad libs challenge.

The double "/" (found doing shift + "?" key) allow you to add comments to your code.

A second thing to note is: System.out.print instead of System.out.println. Println will start a new line at the end, versus print will stay on the same line.