Republic of the Philippines
Department of Education
REGION III
SCHOOLS DIVISION OFFICE OF NUEVA ECIJA

SPECIAL PROGRAM IN ICT 8
BASIC PROGRAMMING
Second Quarter, Week 8

Java Variables

BACKGROUND INFORMATION FOR LEARNERS

Declaring Variables in Java

Computers store all values using **bits** (binary digits). A **bit** can represent two values and we usually say that the value of a bit is either 0 or 1.

To create a variable, you must tell Java its type and name. Creating a variable is also called **declaring a variable**. When you create a **primitive variable** Java will set aside enough bits in memory for that primitive type and associate that memory location with the name that you used. You have to tell Java the type of the variable because Java needs to know how many bits to use and how to represent the value.

To **declare** (create) a variable, you will specify the type, leave at least one space, then the name for the variable and end the line with a semicolon (;). Java uses the keyword **int** for integer, **double** for a floating point number (a double precision number), and **boolean** for a Boolean value (true or false).

type name;

Figure 2: How to Declare a Variable

Here is an example declaration of a variable called score.

```
int score;
```

The value of score can be set later as shown below.

```
public class Test1
{
  public static void main(String[] args)
  {
    int score;
    score = 0;
    System.out.println(score);

 double price;
    price = 2.55;
    System.out.println(price);
    boolean won;
    won = false;
    System.out.println(won);
  }
}
```

When the above code was compiled and executed, it produces the following result −

```
0
2.55
false
```

Initializing variables with initializers in Java

Java also allows you to initialize a variable on the same statement that declares the variable. To do that, you use an *initializer*, which has the following general form:

```
type name = expression;
```

In effect, the initializer lets you combine a declaration and an assignment statement into one concise statement. Here are some examples:

```
int x = 0;
String lastName = "Lowe";
double radius = 15.4;
```

In each case, the variable is both declared and initialized in a single statement.
When you declare more than one variable in a single statement, each variable can have its own initializer.
The following code declares variables named x and y, and initializes x to 5 and y to 10:

```
int x = 5, y = 10;
```

Display Variables

The println() method is often used to display variables.
To combine both text and a variable, use the + character:

```
String name = "John";
System.out.println("Hello " + name);
```

Example

```
public class Main {
  public static void main(String[] args) {
    String name = "John";
    System.out.println("Hello " + name);
  }
}
```

When the above code was compiled and executed, it produces the following result −

```
Hello John
```

You can also use the + character to add a variable to another variable:

```
String firstName = "John ";
String lastName = "Doe";
String fullName = firstName + lastName;
System.out.println(fullName);
```

Example

```
public class Main {
  public static void main(String[] args) {
    String firstName = "John ";
    String lastName = "Doe";
    String fullName = firstName + lastName;
    System.out.println(fullName);
  }
}
```

When the above code was compiled and executed, it produces the following result −

```
John Doe
```

For numeric values, the + character works as a mathematical operator (notice that we use int (integer) variables here):

```
int x = 5;
int y = 6;
System.out.println(x + y); // Print the value of x + y
```

Example

```
public class Main {
  public static void main(String[] args) {
    int x = 5;
    int y = 6;
    System.out.println(x + y); // Print the value of x + y
  }
}
```

When the above code was compiled and executed, it produces the following result −

```
11
```

From the example above, you can expect:

- x stores the value 5
- y stores the value 6
- Then we use the println() method to display the value of x + y, which is **11**

Java Constant

As the name suggests, a **constant** is an entity in programming that is immutable. In other words, the value that cannot be changed. In this section, we will learn about **Java constant** and **how to declare a constant in Java**.

What is constant?

Constant is a value that cannot be changed after assigning it. Java does not directly support the constants. There is an alternative way to define the constants in Java by using the non-access modifiers static and final.

How to declare constant in Java?

In Java, to declare any variable as constant, we use static and final modifiers. It is also known as **non-access** modifiers. According to the Java naming convention the identifier name must be in **capital letters**.

Static and Final Modifiers

- The purpose to use the static modifier is to manage the memory.
- It also allows the variable to be available without loading any instance of the class in which it is defined.
- The final modifier represents that the value of the variable cannot be changed. It also makes the primitive data type immutable or unchangeable.

The syntax to declare a constant is as follows:

```
static final datatype identifier_name=value;
```

For example, **price** is a variable that we want to make constant.

```
static final double PRICE=432.78;
```

Where static and final are the non-access modifiers. The double is the data type and PRICE is the identifier name in which the value 432.78 is assigned.
In the above statement, the **static** modifier causes the variable to be available without an instance of its defining class being loaded and the **final** modifier makes the variable fixed.
Here a question arises that **why we use both static and final modifiers to declare a constant?**
If we declare a variable as **static**, all the objects of the class (in which constant is defined) will be able to access the variable and can be changed its value. To overcome this problem, we use the **final** modifier with a static modifier.
When the variable defined as **final**, the multiple instances of the same constant value will be created for every different object which is not desirable.
When we use **static** and **final** modifiers together, the variable remains static and can be initialized once. Therefore, to declare a variable as constant, we use both static and final modifiers. It shares a common memory location for all objects of its containing class.

LEARNING COMPETENCY

Use variables in a Java program

ACTIVITIES

ACTIVITY 1:

- Answer the attached Google Form in our Google Classroom

ACTIVITY 2:

1. What is the output of this code? Write your answer in the Private Comment section of our Google Classroom

```
public class Main {
  public static void main(String[] args) {
    int x = 12;
    int y = 8;
    int z = 3;
    System.out.println(x + y * z); // Print the value of x + y * z
  }
}
```

REFERENCES

https://runestone.academy/runestone/books/published/apcsareview/VariableBasics/declareVars.html
https://www.dummies.com/programming/java/initialize-variables-java/
https://www.w3schools.com/java/java_variables.asp
https://www.javatpoint.com/java-constant

Prepared by:

Maximo A. Luna Jr
Name of Writer

Noted by:

Laberne A. Ladignon Jr
Division ICT Coordinator/ OIC EPS