LESSON 2
VARIABLES AND ARITHMETIC

INTRODUCTION

In this lesson, we'll look at how to declare and use variables, and perform arithmetic.

2.1 VARIABLES

A **variable** in Java is a storage location in a computer's memory that has a name. The name of a variable is often called its **identifier**.

1. Open Eclipse, and go to **File→Close All** to close the project you were working on before.
2. Create a new project using the instructions in Lesson 1, named **L2_Variables**

   Reminder: File→New→Java Project
3. Add a class source file to the **src** folder in the Package Explorer for your L2_Variables

   Reminder: Right Click src and go to New→Class

   For the **package**, put something useful, such as `com.profjpbaugh.l2variables`.

   For the Name of the class, put **L2_Variables**, just like the project name
4. Make the code of your class match (where appropriate) the following:

Note that the numbers are not part of the code, but are for reference purposes only

```
1  package com.profjpbaugh.l2variables;
2
3  public class L2_Variables {
4
5        public static void main(String[] args)
6        {
7              int someNum;
8              someNum = 15;
9
10             System.out.print("The number is ");
11             System.out.println(someNum);
12       }
13 }
```

Code 2.1-1

Notice that we **declare** a variable named `someNum` on line 7, and that the **data type** of this variable is **int**. The keyword **int** indicates that this variable will contain an **integer**, which is a number without a decimal place. For example, 10, 5, 20, 354, 2000 are all integers whereas 3.14, 6.7, and 12.5 are *not* integers.

Declaring a variable occurs when we state the variable type (in our case in the above example, `int`) followed by the identifier (in our case, `someNum`.) This tells the compiler we want to reserve enough memory to hold an integer.

The **data type**, not surprisingly is the type of data that the variable we declare can hold. There are many different data types available in Java, its libraries, and even custom data types that we can define, which we'll learn about more in later lessons.

We **assign** the variable a value on line 8, in our case, the value 15.

Finally, we make use of two different methods:

- `System.out.print`
- `System.out.println`

There is a subtle distinction between these two methods. We've seen `println` used before in the previous lesson. This method prints out the **argument**(s) you pass it and *then* a newline character, which moves the insertion point to the next line.

The `print` method on the other hand prints out the argument(s) you pass it but *does not* print out a newline character.

Let's observe what happens when you run the application in Code 2.1-1.

```
The number is 15
```

No surprise there! Note that even though in the code we have "The number is " and `someNum` printed on two different lines (in *code*) that what is actually printed is only on one line. This is because we use the `print` method instead of `println` to print the data.

As a note, when we use a value directly, such as the value 15 when assigning this number to the variable someNum, or even when we directly write a string, "The number is ", these are called **literals**, since they are literal values themselves and not variables.

2.1.1 Numeric Data Types

There are many different data types, as mentioned before. In this subsection, we'll take a brief look at the primitive data types for numeric data found in Java, and their sizes.

This information is presented most effectively in a table of data:

Data Type	Size	Range of Data
byte	1 byte	Integers ranging from **-128** to **+127**
short	2 bytes	Integers ranging from **-32,768** to **+32,767**
int	4 bytes	Integers ranging from **-2,147,483,648** to **+2,147,483,647**
long	8 bytes	Integers ranging from **-9,223,372,036,854,775,808** to **+9,223,372,036,854,775,807**
float	4 bytes	Floating point numbers ranging from **+/- 3.4 x 10^{-38}** to **+/- 3.4 x 10^{38}** with an accuracy of 7 digits
double	8 bytes	Floating point numbers ranging from **+/- 1.7 x 10^{-308}** to **+/- 1.7 x 10^{308}** with an accuracy of 15 digits

The integer types byte, short, int, and long probably are fairly clear as to what they represent, since we have seen an integer data type before.

What about the **float** and **double** data types? These are **floating point numbers**, which basically means that they can accommodate decimal points for real numbers. These data types can hold numbers like 3.14159 and 456.890.

2.2 Arithmetic and String Concatenation

In this section we'll discuss and see examples of how to apply **arithmetic operators** (+, -, *, /) to numeric data, and how to apply the **string concatenation operator** (+) to **concatenate**, or **combine**, strings.

2.2.1 Arithmetic and Concatenation Example 1

```
1   package com.profjpbaugh.l2variables;
2
3   public class L2_Variables {
4
5           public static void main(String[] args)
6           {
7                   int num1 = 10;
8                   int num2;
9                   num2 = 15;
10                  int result = num1 + num2;
11
12                  System.out.println("The sum of " + num1 + " and " + num2 + " is " + result);
13          }
14  }
```

Code 2.2-1

Code 2.2-1 is full of examples we can look at to understand arithmetic and concatenation. Notice first, on line 7 that we can do the declaration of and assignment of a value to a variable, `num1`, all on one line. To offer variety and to show contrast, I have included a separation of declaration and assignment on lines 8 and 9 of another variable, `num2`.

Line 10 is one of our examples. Here, we assign the sum of `num1` and `num2` to the variable `result`. This is an example of using the **addition operator**, the plus sign (+) to sum two numbers.

Line 12 is arguably the most interesting because we are not using the + sign to do arithmetic, but rather to **concatenate**, or combine, strings. Also of note is that strings take precedence over numeric data types, so when a

string is combined with an integer such as with "The sum of " + num1, the num1 is automatically converted from an integer to the appropriate string format. This makes programmers' jobs much easier in most cases and is quite useful.

To see the output, run the program, verifying there are no errors and the output seems correct.

2.2.2 Arithmetic Example 2

```
1   package com.profjpbaugh.l2variables;
2
3   public class L2_Variables {
4
5         public static void main(String[] args)
6         {
7               double num1 = 10.75;
8               double num2 = 20.50;
9
10              double diff = num1 - num2;
11              double prod = num1 * num2;
12              double quot = num1 / num2;
13
14
15              System.out.println("Difference: " + diff);
16              System.out.println("Product: " + prod);
17              System.out.println("Quotient: " + quot);
18        }
19  }
```

This example has examples of the other three arithmetic operators in it, with variables to hold the difference (line 10), the product (line 11), and the quotient (line 12.)

The output of this code is:

```
Difference: -9.75
Product: 220.375
Quotient: 0.524390243902439
```

2.3 Constants

In Java, in addition to modifiable variables, we can also use **constants**. A constant cannot be modified once it is declared. An example of declaring a constant is as follows:

```
final double PI = 3.14159;
```

If you try to modify the variable later in the program, you will receive a compiler error.

EXERCISES

1. Write a program to print out your name, followed by a statement saying how old you will be in 10 years. However, do **not** hardcode your age in 10 years, just code your current age and then use the addition operator.

2. Write a program to calculate the area of a rectangle, using three variables, `length`, `width`, and `area`. Recall that the area of a rectangle is found by multiplying the length and the width of the rectangle. Try different values for the length and width.

3. Write a program to calculate the perimeter of a rectangle, using appropriate variable names. Recall that the perimeter of a rectangle is: P = 2 * L + 2 * W, or P = 2 * (L + W), where P is the perimeter, L is the length, and W is the width. Try different values for the length and the width.

4. Write a program to calculate the area of a circle, using three variables:

   - PI, a constant equal to 3.14159
   - R, the radius
   - A, the area

Try different values for the radius to see what happens.