WESTERN UNIVERSITY
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

ES1036B
PROGRAMMING FUNDAMENTALS FOR ENGINEERS
WINTER 2022

# Lab02 – Java Fundamentals (KB 4)

Instructor:
DR. QUAZI RAHMAN

Prepared by:
HIRA NADEEM,
HAREEM NISAR

Acknowledgement:
DR. A. OUDA

Lead TA: Hira Nadeem, hnadeem5@uwo.ca

---

## 1 Goal

Java programs are made up of different parts. In this lab you will learn what these parts are. You will learn about data types, identifiers, variable declarations, constants, comments, and program output.

## 2 Resources

Lecture notes: "Unit 2: Java Fundamentals."
Please review the additional reading "2.6 Common Errors and Coding Conventions" included in OWL under Assignments > Lab 2. It includes very important to consider for this lab and future labs.

## 3 Deliverables

One zip file containing your project folder. Name it *username*_Lab2.zip. where username is the beginning part of your email (e.g. Dr. Rahman's email address is qrahman3@uwo.ca, and his UWO username is: qrahman3). Submission deadline: Submit your code by the end of your lab session. Prepare to demonstrate your understanding during the lab session.

# 4 Good Programming Practices

Below are several programming practices that should be followed to write and maintain quality codes. **Please note that you will be marked on all these components:**

- Include meaningful comments in your program. This will help you remember what each part of your code does, especially after long breaks from your work. Your TA will also appreciate understanding your code by going over your comments.
- Choose meaningful and descriptive names for your variables. There is a balance between descriptive names and code readability but always err on the side of descriptive.
- It is recommended that you follow the naming strategy for variables, methods and class names, as outlined in your course handout (Unit 2). Since the identifiers cannot contain white-space characters (spaces or tabs), words in an identifier should be separated by uppercase letters (myNewFunction(), myNewVariable). For class names, capitalize the first letter of each word in the name (e.g. MyClass, MatrixCalculator). For any constant name, use uppercase letters, and if needed, concatenate two or more words with underscore (e.g., MINIMUM_DRIVING_AGE)
- Initialize variables when declaring them. This means giving them initial values which are easier to track in your program if logical errors are present with your output.
- Indent and properly format your code so that you can read and debug your code easily, and your teaching assistant can read your code with ease while grading your code.
- **Include a commented header in each of your java class files**. The header should include your full name, UWO Student number, date the code was written and a brief description of the program in that file. Format below:

```
/*****************************************
* Add your full name here*
* Add your student number*
* Add Date*
* Give a brief description of the task *
*****************************************/

public class MyClass
{
    public static void main(String args[])
    {
      // Your code here
    }
}
```

# 5 Java Fundamentals

## 5.1 Review comments in JAVA code

Inside a Java program, you can write special text which will be ignored by the java compiler - known as comments. They allow you to exclude code from the compilation process (disable it) or clarify a piece of code to yourself or other developers. The Java programming language supports three kinds of comments.

### 5.1.1 End-of-line comments

The java compiler ignores any text from // to the end of the line:

```
public class Task {
  public static void main(String[] args) {
    // The line below will be ignored
    // System.out.println("Hello, World");
    // It prints the string "Hello, Java"
    System.out.println("Hello, Java"); // You can write a comment here
  }
}
```

3.1.2 Multi-line comments

The compiler ignores any text from /* and the nearest */. It can be used as multiple and single-line comments:

```
public class Task {
  public static void main(String[] args) {
    /* This is a single-line comment */
    /*  This is a example of
        a multi-line comment */
  }
}
```

You can use comments inside other comments:

```
public class Task {
  public static void main(String[] args) {
    /*
    int a = 1; // set a to 1
    int b = 2; // set b to 2
    int c = a + b; // calculate a + b
    */
  }
}
```

### 5.1.2 Java documentation comments

The compiler ignores any text from /** to */ just like it ignores multi-line comments.
This kind of comments can be used to automatically generate documentation about your source code using the javadoc tool. Usually, these comments are placed above declarations of classes, interfaces, methods, and so on. Also, in this case, some special labels such as @param, @return and others are used for controlling the tool.

```
public class Task {
  /**
   * The main method accepts an array of string arguments
   *
   * @param args from the command line
   */
  public static void main(String[] args) {
  // do nothing
  }
}
```

Information until here will help you write Q1 and Q2.

## 5.2 Variables and Types

### 5.2.1 Variables

A variable is a named storage for a value. It has a name (identifier) and a type.
The name identifies a variable in some context. It's possible to read and change the value of a variable by its name. The general form for declaring variables is the following: **`DataType variableName`**. The type (data type) of a variable determines which values can be stored in the variable and the possible operations you can perform on them.
For example:

```
long numberOfStars;
String catName;
int numberOfCats;
```

Here long, String and int are data types. While `numberOfStars`, `catName` and `numberOfCats` are variable Names. Good programming Practice: Always pick up meaningful and descriptive variable names (See Unit 2)

### 5.2.2 Data Types

Java is a strongly typed programming language, which means that every variable and every expression has a type that is known at the compile time.
All data types are separated into two groups:

1. primitive types
2. references types

**A primitive type** variable stores a simple value (such as a number or character). Java has eight primitive data types (See Unit 2 for review). The most used primitive types are int, long, boolean, char and double. The primitive types can be divided into four groups:

1. **integer numbers:** `byte`, `short`, `int`, `long` (for example, 83 is an integer number)
2. **floating-point numbers:** `float`, `double` (for example, 3.1415 is a floating point number)
3. **logical type:** `boolean` (true or false)
4. **characters:** `char` (for example, 'a', '3')

**In Java,** all Class-type variables are reference type variables. **A reference type** variable stores an address in memory where the data is located. The data can be made up as a complex structure that includes other data types as their parts. We will often use the reference type `String`. It represents a sequence of characters like "abc" or "Hello, Java".

### 5.2.3 Variable Declaration

To declare a variable simply create a variable by defining its data data-type and give a valid variable-name. This is called a variable declaration.

Let's declare a variable named *numberOfCats* of the type 'int'. Naturally, when writing your code, try having a suitable variable type. In this case, it does not make sense to have type 'float' – There is no such thing as 2.5 number of cats. (I sincerely hope.)

```
public class CatLady {
  public static void main(String[] args) {

    int numberOfCats; // variable declaration

  }
}
```

Let's declare another variable named *catName1* of type String:

```
public class CatLady {
  public static void main(String[] args) {

    int numberOfCats; // variable declaration of type int
    String catName1;  // variable declaration of type String
  }
}
```

Now what if we want to assign a value to our variables?

### 5.2.4 Variable Assignment and the Assignment operator

To assign a value to a variable we should use the special operator called assignment operator. It is denoted by the equal sign i.e. = . We follow this pattern to assign value to a variable.
**`variableName = value;`** This action or line of code is called variable assignment.

Let's get back to our class *CatLady*. So far, two variables have been declared already – *numberOfCats* and *catName1*. We can assign values to these variables.

```
public class CatLady {
    public static void main(String[] args) {

        int numberOfCats; // variable declaration of type int
        String catName1;  // variable declaration of type String

        numberOfCats = 15;      // variable assignment
        catName1 = "Patches";   //variable assignment
    }
}
```

Can we not combine these two steps? Yes! (see Unit 2 for review)

### 5.2.5 Variable Initialization

It is also possible to declare a variable and assign a value to it in one line:
**`DataType variableName = value;`**
This is called variable initialization. It is a great practice to always initialize your variables with dummy or default values like *0* or "*John Doe*".

In our *CatLady* example so far, we have declared two variables and then assigned values to them individually. Let's now initialize a new variable called *catName2* of type String and assign it a value of "Pumpkin".

```
public class CatLady {
    public static void main(String[] args) {

        int numberOfCats; // variable declaration of type int
        String catName1;  // variable declaration of type String

        numberOfCats = 15;      // variable assignment
        catName1 = "Patches";   //variable assignment

        String catName2 = "Pumpkin"; // variable initialization
    }
}
```

To read the values of a variable we use a kind of print method. There are different methods available for that - print(), println(), and printf(). Now we can use our variables to print a message

```
public class CatLady {
    public static void main(String[] args) {

        int numberOfCats; // variable declaration of type int
        String catName1;  // variable declaration of type String

        numberOfCats = 15;       // variable assignment
        catName1 = "Patches";    //variable assignment

        String catName2 = "Pumpkin"; // variable initialization

        System.out.printf("Hello! I have %d cats. %n The first cat is called %s. " +
                "My favorite is the second cat called %s", numberOfCats, catName1, catName2);
    }
}
```

using the code below.

This code will produce the following output:

```
Hello! I have 15 cats.
 The first cat is called Patches. My favorite is the second cat called Pumpkin
```

This example uses printf() method (Unit 2). Try using println() and print() methods to see how they are different.

### 5.2.6 Variable Names

There are needs and there are wants. Java has some rules for naming variables. It is absolutely necessary that you follow these rules to ensure proper compilation and running of your program. Other than these rules, you should adhere to the conventions of the variable naming. These conventions allow uniformity between codes and increase the readability of your code.

**Variable Naming Rules (Unit 2):**

- names are case-sensitive
- a name can include Unicode letters, digits, and two special characters ($, _ )
- a name can't start with a digit
- a name must not be a keyword (class, static, and int are illegal names)

Examples of some **legal** names of variables: number, $ident, bigValue, _val, abc, k, var
Examples of some **illegal** variable names: @ab, 1c, !ab, class

**Variable Naming Conventions (Unit 2):**

- if a variable name has a single word, it should be in lowercase (for instance: number, val)
- if a variable name includes multiple words, it should be in lowerCamelCase, i.e. the first word should be in lowercase and each word after the first has its first letter written in uppercase (for instance: numberOfCoins)
- variable names should not start with _ or $ characters, even though both are allowed

## 5.3 Input in Java code – via Scanner (Review Unit 2)

Instead of hardcoded values, Java allows for taking in user input. There are several ways to read values from the standard input. The first way is to use **`java.util.Scanner.`**

Let's continue with our CatLady example. Now we read an integer number (to represent newly born kittens) from the standard input (keyboard). We also take another input from the user to assign the value to a String variable named catName3.

```
import java.util.Scanner;

public class CatLady {
    public static void main(String[] args) {

        // always a good idea to initialize variables with default values
        int numberOfNewKittens = 0;

        // define a Scanner object called inputScan.
        // you can use any other appropriate name instead of inputScan as well
        Scanner inputScan = new Scanner(System.in);

        numberOfNewKittens = inputScan.nextInt(); // reads an integer from standard input
        String catName3 = inputScan.next(); // it reads a string from the standard input

        System.out.println("A cat gave birth to " + numberOfNewKittens + " new kittens");
        System.out.print("The third cat is called: " + catName3 );


    }
}
```

Run this code on your own and enter the values from the keyboard (standard input). Use print() statement instead of println() and see the difference. Some programmer prefers print() statement over println() for input-prompt because, in this case, the cursor weights beside the prompting message for the input.

```java
import java.util.Scanner;

public class CatLady {
    public static void main(String[] args) {

        // always a good idea to initialize variables with default values
        int numberOfNewKittens = 0;

        // define a Scanner object called inputScan.
        // you can use any other appropriate name instead of inputScan as well
        Scanner inputScan = new Scanner(System.in);


        System.out.print("\nHow many new kitties are born? "); // user prompt
        numberOfNewKittens = inputScan.nextInt(); // reads an integer from standard input


        System.out.print("What is the name of the third cat? ") ;// user prompt
        String catName3 = inputScan.next(); // it reads a string from the standard input

        System.out.println("A cat gave birth to " + numberOfNewKittens + " new kittens");

        System.out.println("The third cat is called: " + catName3);

    }
}
```

The output for the above code will result in the following :

## 5.4 Output in Java code

There are many ways to display an output in Java. You have already seen the examples of print(), println() and printf() in the codes above. Play around with them and see the differences. For more details, go to Lecture handout' Unit2; Java Fundamentals' and understand them fully.

## 5.5 Arithmetic Operators

Java allows to perform simple arithmetic equations and functions. The rules of mathematics are followed in Java as well. Review the operators, and operator-precedence from Unit 2.

| Operator | Meaning |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulus |

Let's go back to our previous example of CatLady class and combine all our codes. We will add the numberOfCats variable and the numberOfNewKittens to get the updated numberOfCats and display it using println() method.

```
How many new kitties are born?5
What is the name of the third cat? Penny
A cat gave birth to 5 new kittens
The third cat is called: Penny
```

```java
import java.util.Scanner;

public class CatLady {
    public static void main(String[] args) {

        // **********   previous code omitted here to save space ****

        int numberOfCats; // variable declaration of type int
        String catName1;  // variable declaration of type String

        numberOfCats = 15;       // variable assignment
        catName1 = "Patches";    //variable assignment

        String catName2 = "Pumpkin"; // variable initialization

        System.out.printf("Hello! I have %d cats. %n The first cat is called %s. " +
                "My favorite is the second cat called %s", numberOfCats, catName1, catName2);

        // always a good idea to initialize variables with default values
        int numberOfNewKittens = 0;

        // define a Scanner object called inputScan.
        // you can use any other appropriate name instead of inputScan as well
        Scanner inputScan = new Scanner(System.in);

        System.out.print("\nHow many new kitties are born?"); // user prompt
        numberOfNewKittens = inputScan.nextInt(); //reads an integer from standard input

        System.out.print("What is the name of the third cat?") ;// user prompt
        String catName3 = inputScan.next(); // it reads a string  from the standard input

        System.out.println("A cat gave birth to " + numberOfNewKittens + " new kittens");
        System.out.print("The third cat is called: " + catName3 );

        // updating the number of cats by addition.
        numberOfCats =  numberOfCats + numberOfNewKittens;
        // the variable numberOfCats just changed its value

        //printing out our updated number of cats
        System.out.println("\nHazaaa! We now have " + numberOfCats + " cats in total!" );

    }
}
```

The output for the above code will result in the following:

The output for the above code will result in the following:

```
Hello! I have 15 cats.
The first cat is called Patches. My favourite is the second cat called Pumpkin.
How many new kitties are born? 5
What is the name of the third cat? Penny
A cat gave birth to 5 new kittens
The third cat is called: Penny
Hazaa! We now have 20 cats in total.
```

# 6 Lab Assignment Questions

**Begin:** Use IntelliJ IDE to create a project named ***username*_Lab2** where username is your student ID.

## 6.1 Question 1 (10 marks)

- Create a package named **Q1.**
- Create a java class named **MyMoney** which includes the commented header with your information.
- No data-entry from the standard input is required in this code, and no data-validation is required.
- The program should declare an appropriate variable which will be initialized with your Full Name (e.g., Hira Nadeem).
- Now, declare two more appropriate variables to store age and salary amounts and initialize those variables with arbitrary age and expected salary values.
- The program should display these values on the screen in a manner like the following:

```
My name is Joe Mahoney, my age is 26 and
I hope to earn $100000.0 per year.
```

- Run your program and make sure it prints the correct output with your name and other initialized values.

## 6.2 Question 2 (15 marks)

- Create a package named **Q2.**
- Create a java class named **MathBugs** which includes the commented header with your information.
- Use the Scanner class to take a user input. The program should prompt the user to enter a two-digit integer number between 11 and 99 inclusive without any zero (no numbers like 20, 30, etc). No data-validation is required. You can assume that the user will enter a 2-digit integer number without any zeros. Store this number in a variable called *num*. Use arithmetic operators (no mark will be awarded if arithmetic operators are not use) to determine the first and second digit of the number the user entered. Assign them to

variables called *tensDigit* and *onesDigit.* For example, if the user entered 25, then the *tensDigit* equal to 2 and *onesDigit* will be 5.

- Now find the root-mean-squared(RMS) value of these two digits. The RMS of two numbers x and y is given by the formula:

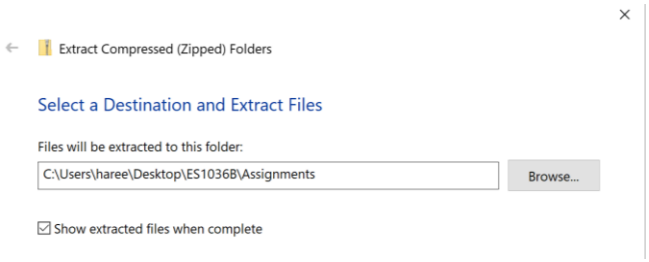
$$RMS = \sqrt{\frac{x^2 + y^2}{2}}$$

- Print your final output on the screen using the printf() method and make sure to include up to 2 decimal places only. For example, if user enters 25, the RMS will be 3.81

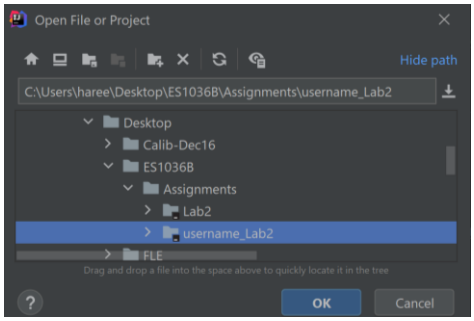
- NOTE: Use appropriate type of variable (int, double, float etc.). In this question you must NOT use Strings and treat everything as numbers/ numerical data. You will be marked on this.

## 6.3 Question 3 (5 marks)

- Unzip the username_Lab2.zip file. To unzip, select the file, right-click, select 'Extract All' and then select the path to be YOUR assignments folder.

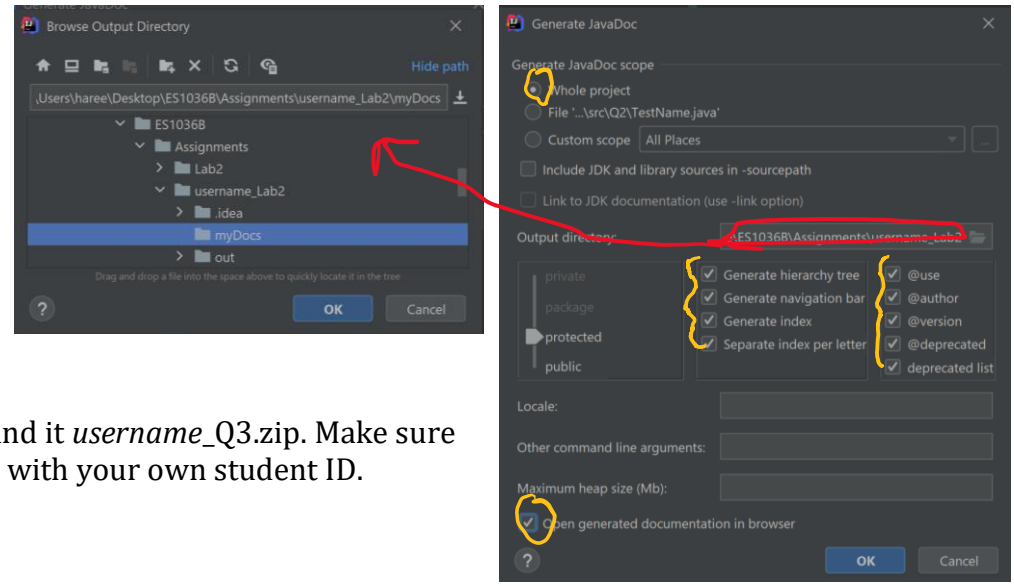



- Open IntelliJ IDE. (Close all the existing projects if you can.) Now 'Open' a new project. Select the location where you extracted the file. Select the project folder with the tiny black symbol with its name.




- In the Package Q0, open the CatLady class. Above the main method signature, type **/**** and then press **Enter** and this will create Javadoc stubs for you.

- From the main menu, choose Tools > Generate javaDoc. In the pop-up window choose your output directory (folder) name. Choose the folder already created for you called *myDocs*. Examine the generated JavaDoc in your Internet browser.




- Now zip the project file again and it *username*_Q3.zip. Make sure to replace the words *username* with your own student ID.

# 7 Lab Assignment Questions

SELECT TWO FOLDERS: Your project folder for Lab 2 AND the zip file from Q3. Place them in a folder and zip the folder. Name it *username_Lab2.zip.* Submit in OWL by the end of your lab session.

**If you have any questions about the Lab Handout, please contact the Lead TA, Hira Nadeem at hnadeem5@uwo.ca**