University of Missouri-Columbia
College of Engineering
Department of Electrical Engineering and Computer Science

3330 – Object-Oriented Programming

Laboratory 2: Java Fundamentals – Part 1

(No late submissions will be accepted for this Lab)

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, program output, and arithmetic operations.

2 Resources

- Lecture notes "Unit 1"
- Source code: Task1.zip.

3 Directed Lab Work

3.1 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.

3.1.1 End-of-line comments

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

```
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:

```
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:

```
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
    */
  }
}
```

3.1.3 Java documentation comments

The compiler ignores any text from /** to */ just like it ignores multi-line comments.

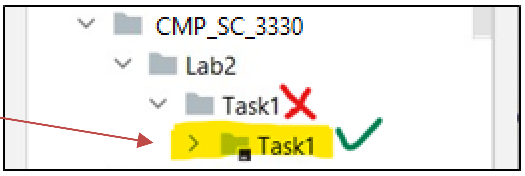
This kind of comments can be used to automatically generate documentation about your source code using the `javadoc` tool. Usually, these comments are placed above declarations of classes, interfaces, methods, and so on. Also, in this case, some special labels such as @param, @return and others are used for controlling the tool.

```
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
  }
}
```

3.1.4 Task 1

- Download the file Task1.zip from Canvas into your course working folder.
- Unzip this downloaded file. **Right-click** Task1.zip then select **Extract All** from the context menu, then Uncheck the 'show extracted files when complete' option box and Click Extract bottom in the pop-up window.

- Open IntelliJ IDE, from the Welcome Screen, choose Open or Import. If the IDE has a project opened already, select File→Open.
- Browse into the unzipped folder "Task1", open the folder and choose the inner folder named "Task1" and click Ok button.



- Examine the provided code, you will find that a programmer wrote this code with a lot of comments, but he forgot to add the required symbol for comments. As a result, the code did not compile. You need to comment some lines in their code to make it compile.
- Note: Please, do not remove any of the comments which are already written!
- Use Cmd + / shortcut to comment the line of your cursor is on. You don't need to move your cursor to the beginning of the line.
- Use Alt + Left Arrow/Right Arrow to navigate through code elements in a line of code.
- Run your code with Ctrl + Shift + R before export the zip file to make sure everything is compiled.

3.2 Variables and Types

3.2.1 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.

3.2.2 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: **primitive types** and **references types**.

A primitive type variable stores a simple value (such as a number or character). Java has eight primitive data types. They can be divided into four groups:

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

The most used primitive types are **int**, **long**, **boolean**, char and **double**.

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".

Let's declare a variable named number of the type int:

```
int number;
```

Here is a variable named text of the type String:

```
String text;
```

To assign a value to a variable we should use the special operator.

3.2.3 The assignment operator

Java has a special operator denoted as **=**. It assigns a value to a variable.

In this case, the variable has to be declared already:

variableName = value;

It is also possible to declare and initialize a variable in one line:

DataType variableName = value;

```
 class Task {
     public static void main(String[] args)
     {
         // declares an integer variable "one" and assigns the value 1 to it
         int one = 1;
         // declares an integer variable "two"
         int two;
         // assign the value 2 to the variable "two"
         two = 2;
         // declares two integer variables and assigns values to them
         int three = 3, four = 4;
     }
}
```

To read the value of a variable you should write its name. For example, let's print the variables:

```
 System.out.println(one);   // prints 1
 System.out.println(two);   // prints 2
 System.out.println(three); // prints 3
 System.out.println(four);  // prints 4
```

In the following example, the variable five is declared and initialized with the value of another variable:

```
// read the value of the variable "four" and assign it to the variable "five"
int five = four;
```

Here are some more variables:

```
// declare a character variable named "ch" and assign 'A' to it
char ch = 'A';

// declare a string variable named "str" and assign text to it
String str = "Hello, Java";

// declare a double variable named "pi" and assign 3.1415 to it
double pi = 3.1415;
```

3.2.4 Rules for naming variables

Java has some rules for naming variables:

- 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)

You cannot break these rules, otherwise, your program will not compile. Here are some legal names of variables:

`number, $ident, bigValue, _val, abc, k, var`

And here are some illegal ones:

`@ab, 1c, !ab, class`

3.2.5 Naming conventions for variables

Also, there are naming conventions for naming variables:

- 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

These conventions are not required, but it is strongly recommended to follow them. They make your code more readable for yourself and other Java programmers.

3.2.6 Task 2

- Use IntelliJ IDE to write a Java program that includes the main method. Your project name must be Task2, with your own choice class name.
- The program should declare the following:
- a String variable named name
- an int variable named age
- a double variable named annualPay
  Store your age, name, and desired annual income as literals in these variables.
- The program should display these values on the screen in a manner similar to the following:

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

- Do not forget to add comments to make your program easy to read and more understandable.
- Run your program and make sure it prints the correct output.

3.2.7 Task 3

- In this task you will be writing the quiz part of this lab. Use the Lab2_Quiz on Canvas to complete and submit answer for this quiz.
- The quiz includes one programming question to evaluate your understanding of this lab material. You will be given **30 minutes** to complete and submit your answer, please follow the following important instructions.
  - You are to complete this quiz independently and alone. Collaborating, discussing, or sharing of information during the quiz is not permitted.
  - Use the IntelliJ Idea IDE to create a new Java project file and named it yourUMId_Lab2_Quiz. **yourUMId is the first part of your UM (University of Missouri) email address before the @ sign**.
  - The quiz link will be inactive on **September 12 at 4:59 PM**. Note that, it is allowed to submit only once, so please review your code carefully before submitting your answer.
  - To start, click Final Assessment link. Once you click "Begin Assessment," you will have 30 minutes to complete this quiz.
  - Read the question and start to write your codes using the opened project file in the IntelliJ IDE.
  - After finishing your program, export your project as yourUMId_Lab2_Quiz.zip file.
  - To submit your answers, click the Browse button, navigate the file explorer to your exported zip file, click Upload button, and then click Submit for Grading button.

4 Hand In

- Create a new folder and name it yourUMId_Lab2, then copy the folders of projects Task1 and Task2 in your new folder yourUMId_Lab2. yourUMId is the first part of your UM (University of Missouri) email address before the @ sign Zip yourUMId_Lab2 folder into yourUMId_Lab2.zip.
- Make sure you followed the instructions for the naming conventions, the types, and the submitted file structure.
- Submit your *yourUMId_* Lab2.zip file into Canvas on **September 12, before 4:59 PM**.
- **No submissions will be accepted after September 12, 4:59 PM**