ECM1410: Object Oriented Programming

Week02: Java Basics Practice I

January, 2020

Note:

- The average duration to conclude this workshop is two hours. Please, note our Monday synchronous session is 1-hour long. Therefore, it is strongly advised to work on these exercises before the session to anticipate questions.
- If you are not able to finish all the exercises during the workshop, you need to spend extra time on them. Make sure you always finish them without problems before the next workshop.
- There will be no solutions offered for this workshop, as you need to do these exercises by yourself. The workshop assistant is always ready for help.

1 `HelloWorldApp.java`

Step1: Create your data directory structure for the workshops

The directory for this week's workshop should be like:
{some-folder}/ECM1410/Workshops/week02/
I would suggest you to use command lines to create the above directory.

- In Windows, navigate Start > All Programs > Accessories > Command Prompt. If you are not familiar with the commands, please search the instructions online of the three commands: `cd`, `dir`, `mkdir`.
- In Linux or Mac, open the terminal. If you are not familiar with the commands, please search the instructions online of the three commands: `cd`, `ls`, `mkdir`.

Step2: Create the source code file `HelloWorldApp.java`

Use any editor to create the source code.

- In Windows, you may use Notepad or Notepad++ to create it. **Notepad++** is a much better choice, as it offers syntax highlighting. Navigate Start > All Programs> Notepad++ > Notepad++. In the file menu, choose the language to Java.
- In Linux/Mac, you may install JEdit, Sublime Text or Atom which offers syntax highlighting for Java. If no such installation, use emacs or gedit is also fine.

  >> `emacs HelloWorldApp.java &` **RETURN**[1]

Type the following codes in the editor and save it.

```
// HelloWorld application
// Your name or student number here
// The date
public class HelloWorldApp {
   public static void main(String[] args) {
      System.out.println("Hello World!");
   }
}
```

NOTE: When you copy codes from a pdf file or online, you may see compile time error (e.g., illegal character). This is because some special characters (such as double quote, single quote) are coded in different way. You may fix it by replacing these special characters.

Step3: Compile the program

In order to run the compiler, please enter the line below to compile the source file.

>> `javac HelloWorldApp.java` **RETURN**

If the compilation is successful, the compiler will generate a file named `HelloWorldApp.class` in the same directory as the Java source file (check using the `dir` command in Windows or the `ls` command in Linux at the command prompt). This class file contains the Java Byte Code we discussed in the first lecture.

If the compilation failed, try to fix it yourself by reading the error message. Please talk to the workshop assistant if you are not able to fix the problem.

Step4: Run the program

Since you have successfully generated the java byte code (`HelloWorldAPP.class`), we may pass it to the **Java virtual Machine (JVM)** to be interpreted and run. To do this at the command line:

>> `java HelloWorldApp` **RETURN**

[1] **RETURN** - press the "return" key. >> indicates the command prompt in a terminal window, this does not need to be typed in.

You should see `"Hello World!"` displayed in the terminal.

Congratulations! You have just created, compiled and executed your first Java program!

Step5: Play it

1. Feel free to try and change the message from "Hello World!" to something else. You'll need to recompile and rerun `HelloWorldApp.java` in order to see the effects of your changes each time.

2. Let's do something cool! How about input your name and age from the terminal and let the program prints them out. Only need to replace the `System.out.println(..);` with:

   ```
   System.out.println("Hello World! I am " + args[0]
     + ". I am " + args[1] + " years old.");
   ```

   Inside the brackets of `System.out.println()` or `System.out.print()`, you can use "+" to concatenate strings with numbers.

   The `args` is the single argument of the `main()` method, which is called as *command line arguments*. We may use `args` to pass some inputs when executing the program. The above statement is to pass two inputs.

   Save it and recompile it as before:

   >> `javac HelloWorldApp.java` **RETURN**

   To run the program, we need **add two inputs** after the HelloWorldApp, as below.

   >> `java HelloWorldApp Alex 20` **RETURN**

   If successful, you would see the output as below:

   ```
   Hello World!  I am Alex.  I am 20 years old.
   ```

   Currently, the program only support a fix number (2) of arguments as input, we will learn how to handle arbitrary number of arguments in the coming week.

3. In the above code, we use string concantenation (with "+") to output the result. Alternatively, you may write a few `System.out.print(..);` statements followed with one `System.out.println(..);` statement to do the same job.

Hints

```
System.out.print("Hello World! I am ");
System.out.print(args[0]);
System.out.print(". I am ");
System.out.print(args[1]);
System.out.println(" years old.");
```

2 `VariblesTest.java` (Literals)

Determine which of the following statements will get a syntax error?

1. `float area = 20.3F, radius;`
2. `char a = '\u5468', b = '\\';`
3. `byte nCount = 30*21;`
4. `int nHours = 022;`
5. `double ratio = 7/5;`

Then write a Java program (called `VariblesTest.java`) to verify your judgement. Find a solution to the bug. Print out all the variables' values, and check if they are what you expect. (Pay attention to the values of `radius`, `a`, `nHours` and `ratio`)

3 `VariblesTest.java` (Rounding errors)

It is fairly rare for floating point arithmetic to ever be 'exact'. Floating point operations may include some *rounding errors*. To verify it, let's do a simple test by adding a few statements in the above `VariblesTest.java` file.

1. First declare a `double` variable `d` and initialize it with value 29.0.

   ```
   double d = 29.0;
   ```

2. Then multiply it by 0.01. You may consider use the assignment operator `*=`:

   ```
   d *= 0.01; // equivalent to d = d*0.01;
   ```

3. Divide `d` by 0.01 (also use the assignment operator `/=`).

   ```
   d /= 0.01; // equivalent to d = d/0.01;
   ```

For each step, print out the value of `d`. In theory, the `d`'s value after multiplying and dividing 0.01 must be unchanged, i.e., `d=29.0`. However, in practice, you must have seen `d`'s value is not exactly 29.0. This is because of *rounding errors* of floating-point operations.

Due to the existence of *rounding errors*, we never use "==" to compare two floating point variables as what we do for the integer variables. Instead, the correct way to test floats for "equality" is:

```
if (Math.abs(dOld - dNew) < epsilon) { ... }
```

where `epsilon` is a very small number like 1e-10, depending on the desired precision. `Math` class, offered by Java library, contains methods for performing basic numeric operations. `abs()` method is used to calculate the absolute value of an input number.

4 `MyBirthday.java`: What day is my birthday?

Given today's date and day as follows.

```
int currentDate = 14;
int currentDay = 1; // Sunday is 0, Monday is 1, ...
```

- Write a program `MyBirthday.java` to compute what day my birthday (e.g., 29 in the same month) is. You need declare two integer variables (`myBirthdayDate` and `myBirthdayDay`) and one constant to hold the unchangable value 7 (the number of days a week). Follow the naming convension for both variables and constants.

Hints

```
final int DAYS_A_WEEK = 7; // final is used for declaring
    constants
int myBirthdayDate = 29;
int myBirthdayDay =
    (myBirthdayDate-currentDate)%DAYS_A_WEEK+currentDay;
myBirthdayDay = myBirthdayDay%DAYS_A_WEEK;// % is the modulus
    operator
```

Compile and run it. Change the `myBirthdayDate` for any numbers which is larger than 14 and smaller than 31 to test your program. Make sure the program is run correctly before moving to the next question.

- Since `myBirthdayDay` won't be bigger than 6, let's use a shorter integer date type (for example, `byte` type) to hold its value to save storage space (but don't change the `int` type of the `currentDate` and `currentDay` for this test).

```
byte myBirthdayDay = ...;
```

See what consequense it will cause.

Hints

You must have a syntax error by saying "incompatible types: possible lossy conversion from int to byte". To debug this problem, you need explicitly convert the date type (aka *narrowing casting*) as follows.

```
byte myBirthdayDay = (byte)...;
```

- If you change the type of `myBirthdayDay` to a longer data type (for example, `long`, `float`, or `double`, though not necessary), you won't get the error, because this is called *widening casting* which is done implicitly. Try it yourself.

5 RectangleComputation.java

Follow the example (`CircleComputation.java`) in the lecture, write a program called `RectangleComputation.java`.

1. Print out the area and perimeter of a rectangle, given its height and width (in `double`s).
2. Print out the ratio of width to height. Declare a boolean variable `isSquare` and assign a value to it depending on the ratio. If the ratio equals to 1, it is a square, otherwise it is not.

   Note: use the correct way of testing floats for equality, as mentioned above. The syntax for `if-else` statement in Java is as follows.

   ```
   if (condition) {
     ...
   } else {
     ...
   }
   ```

6 More exercises

If you finish all the above exercises, download examples from the lectures, do some changes and run it. Make sure you understand how they work.

There are also many examples and exercises available online. I have listed a few links on the ELE. For example, this link from NTU (Nanyang Technological University, Singapore) contains many exercises for practicing Java basics: `https://www.ntu.edu.sg/home/ehchua/programming/java/J2a_BasicsExercises.html`