COMP1030 Programming Fundamental in Java
Assignment#5 (Final Micro Project) – Beta 3.0

Important Notes:

- **This main and major assignment is considered to be a micro project. Do not share your work with other groups**
- Follow the **exact naming rules** for all your assignment files as explained in the instructions or shown in the diagram
- Although Java code can be written, compiled, and run using different IDEs, based on what we have been using in our class, it's much faster to start the assignment using Microsoft Visual Studio Code
- Submit your assignment files through the Blackboard Assignment Link **(Emails are NOT accepted)**:
- **Follow the submission rules as stated in the assignment**
    - **Zipping/Compressing the folder is NOT allowed**
- **Submit/upload only the following 3 files:**
    1. The java main file to launch the application *(The entry point)* ➔ **Main.java**
    2. The java file for your custom class ➔ **Student.java**
    3. The **PDF file** that contains the screenshots/snapshots as explained below:
        - Use **MS-Word** to capture all the assignment images (screenshots) then put them all in one MS Word Document, then convert it to a PDF file and upload it with your submission.
        - **Please consider the following screenshots (images):**
            - The folder structure of your project inside Visual Studio Code or any other Editor that you are using that also shows the main java file
            - The Output of your code (after running the main Java application file) in the Console Window [VS (Visual Studio) Code Terminal]

Online: Each student must be ready with their Zoom Mic/Cam and Screen share feature to quickly show us the assignment content, Q/A, and run it to see the output (even if it is not working properly).
In-person: I will make a tour to check the running application in the student's computer and asking the technical questions

NOTE: *This presentation/tour will be very quick without detailed correction or feedback in order not to take the full lecture time! The assignment steps, feedback, and solutions will be completely explained to all students (after finishing all the presentations/tour) in the next lecture*

(20 Marks)

Assignment Contents:

This assignment will be a proper Java project demonstration of your skills in working with Object-Oriented Classes (OOP):

- Using classes that have multiple members (attributes and methods/behaviours).
- Creating and instantiating objects.
- Encapsulating of class properties.
- Using Setters and Getters, or class Constructor.
- Polymorphism (Method Overloading).

Assignment Instructions:

1. Using VS Code, you will create a new project named **"Java-Assignment5"** after choosing the folder location (it should be inside your course or program folder)
2. Create a new Java class (new Java file) named **Main.java**, add these ***two*** required sections of code:
   a. The class declaration (name) ➔ the name must be **Main (With capital M)** as the file name and must be public as usual
      - ***NOTE:*** *The class name must be the same as the file name*
   b. Inside the class block, in between the opening "{" and the closing "}" add the **main method** that represents the entry point for any Java project

   Hint:
   - Points (a) and (b) are just the default boilerplate that must be implemented in any Java Application, and it can be easily generated/written through the VS Code IntelliSense or just using the App.java

Lecture Example:
Below is an example of a class named "**EntryPoint**" that has the "**main()**" method:

```
public class EntryPoint {
    public static void main(String[] args) {
            // your code for calling the "Student" class…
     }
}
```

3. Create another Java class (new Java file) named **Student.java**, add this ***one*** required section of code:
   a. The class declaration (name) ➔ the name must be **Student** as the file name with "public" access modifier
   b. Notice that this class doesn't have "main()" method, it's not the entry point to run our application

Lecture Example:
Below is an example about a "Rectangle" class:

```
public class Rectangle {

}
```

(10 Marks)

Assignment Instructions-Part1: Student.Java

In this part, you will write all your code (answer the questions) inside the **"Student.java"** class file

1. Create/Declare the following **6 properties** (variables inside a class). All of them should be private to implement the idea/concept of "Encapsulation" in OOP:
   a. **firstName** of "String" data type ➔ to represent the student's first name
   b. **lastName** of "String" data type ➔ to represent the student's last name
   c. **progName** of "String" data type ➔ to represent the student's program name
   d. **collName** of "String" data type ➔ to represent the student's college name
   e. **value1** of "double" data type ➔ to represent the student's first exam/test/quiz value
   f. **value2** of "double" data type ➔ to represent the student's second exam/test/quiz value

Lecture Example:
Below are two examples for creating two properties "length" and "width" in the "Rectangle" class:

```
// Class members => properties/methods
// Attributes, Properties, Fields:
     private double length; // is set to 0.0 by default
     private double width; // is set to 0.0 by default
```

2. Create/Declare the following **class Constructor** that accept **4 parameters** to be assigned to 4 properties:
    a. The parameter#1 for example ***firstName*** or ***any other name*** to be assigned to the class property/field **"firstName"**
    b. The parameter#2 for example ***lastName*** or ***any other name*** to be assigned to the class property/field **"lastName"**
    c. The parameter#3 for example ***progName*** or ***any other name*** to be assigned to the class property/field **"progName"**
    d. The parameter#4 for example ***collName*** or ***any other name*** to be assigned to the class property/field **"collName"**

Note:
- If the parameter name is the same as the field name, don't forget to add **the keyword "this":**
- Examples:

```
this.firstName = firstName;
// OR:
firstName = fName;
// Code below is wrong:
firstName = firstName
```

Hint:
- A class constructor is a special function because:
    - It has the same name as the class name (Starting with Upper Case)
    - Doesn't have a return type
    - Is being called automatically when we create object(s) from the class

Lecture Example:
Below is an example for creating a constructor method for the "Rectangle" class:

```
// Constructor: A special function of class
// Doesn't have a return type
// passing two parameters of double data types: length and width:
public Rectangle(double length, double width) {
       this.length = length;
       this.width = width;
}
```

3. Create/Declare the following **class Constructor** that accept **6 parameters** to be assigned to 6 properties:
    a. The parameter#1 for example ***firstName*** or ***any other name*** to be assigned to the class property/field **"firstName"**
    b. The parameter#2 for example ***lastName*** or ***any other name*** to be assigned to the class property/field **"lastName"**
    c. The parameter#3 for example ***progName*** or ***any other name*** to be assigned to the class property/field **"progName"**
    d. The parameter#4 for example ***collName*** or ***any other name*** to be assigned to the class property/field **"collName"**
    e. The parameter#5 for example ***mark1*** or ***any other name*** to be assigned to the class property/field **"value1"**
    f. The parameter#6 for example ***mark2*** or ***any other name*** to be assigned to the class property/field **"value2"**

4. After creating two custom constructors that accept set of parameters, we need to create/declare again the default class **Constructor** that **doesn't accept any parameters** in case if we need to instantiate an empty object without assigning any values to its properties/attributes:

5. Create **getters** and **seters for all the class fields/properties/attributes**:
    a. Getters ➔ Example for the property value1:
        i. Get the numeric value of the "Student" property/attribute "value1"
        ii. Get the numeric value of the "Student" property/attribute "value2"
    b. Setters ➔ Example for the property value2:
        i. Set the numeric value to the "Student" property/attribute "value1"
        ii. Set the numeric value to the "Student" property/attribute "value2"

Lecture Example:
Below are two examples from the "Rectangle" class. First one "getWidth()" which is a getter method that returns a double data type. The second one "setLength()" which is a setter method that doesn't have a return value:

```
public double getWidth() {
	return width;
}

public void setLength(double length) {
	this.length = length;
}
```

6. Create a **public method** to calculate the **student's average**. ***Use any proper name you prefer for this method***. This method should not accept any parameters as we are going to use the same properties: value1, and value2 and **return the average** of these two values that could be two exams, two assignments, etc.…

Lecture Example:
Below are two examples from the "Rectangle" class. We used the two properties "width" and "length" to calculate the perimeter and the area of the rectangle:

```
    // Two public methods:
    public double calculatePerimeter() {
        double perimeter = 2 * (width + length);
        return perimeter;
    }

    public double calculateArea() {
        return length * width;
    }
```

7. Create another **public method** to just **print the student's information** and doesn't return any value. This method should not accept any parameters as we are going to use the same class 6 properties. **Use any proper name you prefer for this method**. You will need to use your skills with Java concatenating to print these the properties in any readable layout/format you prefer, or you like. This method for:
    a. Printing the student' full name, college name, Program name, and the two marks/grades
    b. Printing the average value and use the following criteria to print the grade:
        1. If average in between 90 and 100 (from 90 to 100), display:
            - **"Your final average is X. Your grade is A+"**
        2. If average in between 80 and 89 (from 80 to 89), display:
            - **"Your final average is X. Your grade is B"**
        3. If average in between 70 and 79 (from 70 to 79), display:
            - **"Your final average is X. Your grade is C"**
        4. If average in between 60 and 69 (from 60 to 69), display:
            - **"Your final average is X. Your grade is D"**
        5. If average is less than 60, display:
            - **"Your final average is X. Your grade is F"**
        6. Else (otherwise), display:
            - **"Sorry, all the marks have to be from minimum 0 to maximum 100!"**

Lecture Example:
Below is just an example from the "Person" class that contains two properties only: "age" and "name" and just print them both. *Please be advised that this is just an example for printing two class properties, so don't use the same syntax "name is " or "Age is "!!! Use any English phrase or labels you prefer. Also, you can use one, two, three, println() statement it's up to you. Some students used another method named printf() you can use it if you like to*

```
public void getInfo() {
        System.out.println("Name is " + name);
        System.out.println("Age is " + age);
}
```

(40 Marks)

Assignment Instructions-Part2: Main.Java

In this part, you will write all your code (answer the questions) inside the "Main.java" class file which represents the "Entry-Point" to run/launch your application because it has **the "main()" method**.

1. Create **3 instances (objects) variables** of type "Student" with any proper name/title you prefer. For example: student1, student2, … or st1, st2, … or stud1, stud2, or any other 3 names/title you prefer, just add the number 1,2,3 at the end of each name to refer to object1, object2, object3 respectively. Based on the class constructors:
    a. you can pass these 4 values (arguments) for one student object when you create/instantiate it:
        i. The student's first name
        ii. The student's last name
        iii. The student's program
        iv. The student's college
    b. you can pass these 5 values (arguments) for the other student object when you create/instantiate it:
        i. The student's first name
        ii. The student's last name
        iii. The student's program
        iv. The student's college
        v. Mark1
        vi. Mark2
    c. Don't pass any value (argument) for the last student object when you create/instantiate it. Use the setters' methods to assign values to this object.

Lecture Example:

Below are just two different examples from our lecture about instantiating/creating two different instances/objects from different classes: "Person" and "Car" inside the "main()" method:

```
/*
 * An object/instance named "person1" of "Person class" data type
 */
Person person1 = new Person("Alex Chow", 48);
person.getInfo();
/*
 * Name => Alex Chow
 * Age => 48
 */
```

```
/*
 * An object/instance named "myCar" of "Car class" data type
 */
Car myCar = new Car("Dodge", 2010, "SUV", "CXT");
```

2. By using the 3 objects that you created, access their public method for printing the information for each student. So, you call the method to print the information for each student.

Lecture Example:
Below are just two examples about using the same instance/object "myCar" from the class "Car" to access their two public methods "getInfo()" and "moveBackward()". Notice that in your assignment you have the opposite as you will use three different objects (for the three students that you have created) to access the same method for printing their info:

```
myCar.getInfo();
myCar.moveBackward();
```

3. By using the setters, set the two numeric values for each object (student). These two numeric values (value1 and value2) could be used for exam1, exam2 or quize1, quize2 or any other course related term, so the numeric values should be between 0 and 100.

Lecture Example:
Below are just two examples about using an instance/object variable named"myRec1" from the class "Rectangle" to access its public setter methods "setLength()" and "setWidth":

```
/*
 * Using the setter() method
 * set the values for the properties:
 * - length
 * - width
 */
myRec1.setLength(7);
myRec1.setWidth(9);
```

4. By using the 3 objects that you created for each student, access their public method for returning the average for each one. Notice that this method doesn't have the printing functionality (doesn't have println()), it just returns a value. You can either place the entire method inside println() method by concatenating it with a simple and clear message, or you can assign the returned value of this method into a variable then print this variable with a simple message using println(). It's up to you how you want to display the result of the 3 averages for the 3 individual students.

Lecture Example:
Below is just an example about using the same instance/object "myRec1" from the class "Rectangle" to access its public method named "calculateArea()" to calculate the area of this rectangle based on the values of the length and width that we set previously using the setter() method:

```
/*
 * Calling the public method "calculateArea()" for the object "myRec2"
 * by concatenating it with a simple text message "The area of rec2 is: "
 * so we can have a clear idea about the result in the terminal window:
 */
System.out.println("The area of rec2 is: " + myRec2.calculateArea());
```

Yes, it would much better and required to place your text message the student's name, example:
The average for Alex Chow is 89.78

5. You will create the fourth object, for example student4, stud4, or any name you prefer according to the user input:
    a. Using your skills in reading the user input from the terminal window, you will ask the user to enter all the class properties with simple English sentences
    b. Assign each value to a variable
    c. Instantiate the final object with all the values from the variables (passing them to the constructor)
    d. Using the public method to print all the student's info including the final average and the grade

6. Finally, encapsulate your code with Try/Catch block as demonstration of using exception. Please refer to our class recordings/code for more help and tips

(30 Marks)

Happy Coding 😊