WESTERN UNIVERSITY
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

ES1036B
PROGRAMMING FUNDAMENTALS FOR ENGINEERS
WINTER 2022

Lab04 – Class and Objects – (ET2+KB4)

Instructor:
DR. QUAZI RAHMAN

Prepared by:
HIRA NADEEM

Lead TA: Hira Nadeem, hnadeem5@uwo.ca

---

1 Goal

This lab will explore the fundamentals of object-oriented programming by applying your understanding of classes, objects, constructors and methods in practice. Some questions are similar to previous labs but now, you will use more powerful tools to solve them.

2 Resources

Lecture notes:	Unit 3: Methods in Java
	Unit 4: Class Fundamentals in Coding
Watch:	Unit 4: Pre-recorded Lecture videos/2021 IntelliJ Tutorials
Review:	Unit 2.5: Operators, Frequently used Methods, Casting and printf()

3 Deliverables

One zip file folder containing your project folder. Name it username_Lab3.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.

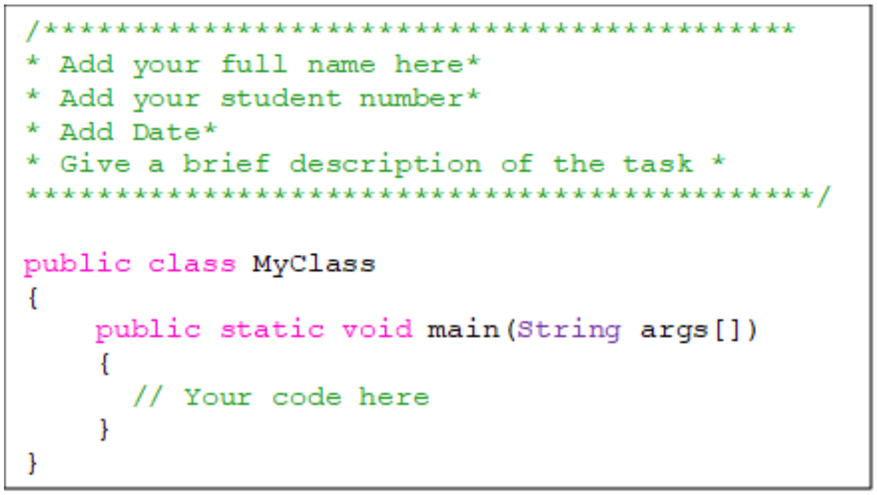
- Choose meaningful and descriptive names for your variables. There is a balance between descriptive names and code readability but always err on the side of descriptive.

- 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. You should write your codes so that your teaching assistant can read and understand your code easily.

- **Include a header in each of your java class files**. The header should include your full name, UWO ID number, date the code was written and a brief description of the program in that file. Use any print statement to print the Header information on the screen based on the 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 Lab Assignment Questions

In each question's driver method, print a header using the **printHeader** method in the beginning, perform the task, and then print an exit message at the end using the **printFooter** method like you did in Lab 3.

The methods:

- **printHeader** -> Outputs a header with your name, lab number and question number. Recycle the **myHeader** method from Lab 3
- **printFooter** -> Outputs an exit message i.e. `*** Goodbye from YourFullName! ***`. Edit the **myFooter** method from lab#3 such that it does not take any argument.

Use IntelliJ IDE to create a project named ***username*_Lab4.**

5.1 Question 1 (10 marks)

For this question, refer to the slides in "Unit 3.1: Method Fundamentals", "Unit 3.2: Method Overloading and the concept of Scope", and the IntelliJ video from 2021 on OWL for "static methods". Take a careful look at the examples to help you structure your code.

1. Create a package named **Q1.**
2. Create a java class named **SumCity** which includes the driver method main() and the commented header with your information.
3. Define three overloaded methods with the following:
   - The first will accept two strings as argument and return the added strings
     ```
     public static dataType sumValues(dataType str1, dataType str2)
     ```
   - The second will accept two real numbers as argument and return added result when called
     ```
     public static dataType sumValues(dataType num1, dataType num2)
     ```
   - The third one will accept two characters as argument, and return the resultant character after adding those two, when called.
     ```
     public static dataType sumValues(dataType char1, dataType char2)
     ```
4. Write the driver method according to the following specifications:
   - Print your custom header by calling your **myHeader()** method.
   - Prompt (ask the user) the user to enter two real numbers from the keyboard. Call the sumValues() method and pass these numbers as arguments and then print their sum in main() using printf() method using up to two decimal places.
   - Prompt the user to enter two characters from the keyboard. Call the sumValues() method to add these entered characters. Print their resultant integer value AND Unicode value in main() using the printf() method.
     Cross-check your result with the Unicode character table (https://unicode-table.com/en/#control-character). Consult your instructor or TA if it does not make any sense.

- Prompt the user to input a full name. Call the sumValues() method to add the strings "I am " and the user-entered name. Print the added strings in main() using the printf() method.
- Print a custom footer using the **myFooter()** method.

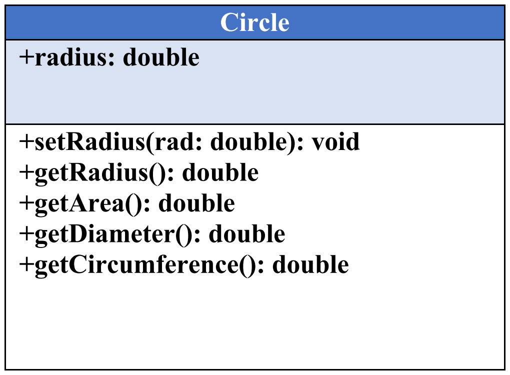
The program output should be something similar to the following Sample Output:

```
****************************************************
YourFullName
Lab 4, Question 1
*****************************************************
Enter a real number: 2.5
Enter a second real number: 13.65
The sum of 2.50 and 13.65 is: 16.15

Enter a character: a
Enter a second character: d
The sum of the characters 'a' and 'd' is: Å, the integer equivalent value is 197.

Enter your full name: yourFullName
I am yourFullName

*** Goodbye from YourFullName! ***
```

5.2 Question 2 (10 marks)

For this question, refer to the slides for "Unit 4: Building a Class and Object". Additional help can be found in the pre-recorded lecture videos from 2021 called "Introducing Class and Objects", "Encapsulation", and "Constructor". The IntelliJ Tutorials for "A Class called Rectangle" and "Parameter Passing" will be helpful as well. The concept of Constructors can sometimes be tricky, so please make sure you review before you start the question.

1. Create a package named **Q2.** In this package, you will create to Java classes.
2. Create a new Java class named **Circle, <u>as shown in the UML diagram</u>.**
   Hint: make sure you know what "+", "-", and other features of a UML diagram to create the class correctly.



Specifications for the Circle Class:

In the circle class, add the following field:

- **radius**: a double type data field with public (+) access modifier

The class should have the following methods:

- **setRadius()**. A mutator/setter method to set the radius field value.
- **getRadius()**. An accessor/getter method to access the radius field.
- **getArea()**. This method will returns the area of the circle, which is calculated as:
  area = PI * radius * radius (You can use Math.PI static field from Math class)

- **getDiameter()**. This method will return the diameter of the circle, which is calculated as: diameter = radius * 2
- **getCircumference()**. Returns the circumference of the circle, which is calculated as: circumference = 2 * PI * radius (You can use Math.PI static field from Math class)

3. Now, you need to write a driver class (that contains the driver method) that demonstrates the Circle class inside the same Java package.
   - Create a class by right-clicking the Q2 package folder and choosing new→Java class.
   - Name this driver class as **CircleInfo*YourFirstName***. For example, if your first name is Quazi, the name of this class should be **CircleInfoQuazi.**
   - **Note:** This class will contain three public-static methods: your driver method, and the header and footer methods you have defined earlier.

4. The driver method (main() method) will be defined with the following specifications that will follow the sequence below:
   - Call the your custom header method **myHeader().**
   - Creates a Circle type reference variable and instantiate the Circle object (e.g., Circle *anyValidName* = new Circle();).
   - Prompt the user to enter the circle's radius.
   - Using the setter method, set the radius of the circle.
   - Print the circle's radius, area, diameter, and circumference using getter methods and printf() method up to 3 decimal places.
   - Print your custom footer method by using your **myFooter()** method.

5. The program output should be something like the following Sample Output:

```
*****************************************************
YourFullName
Lab 4, Question 2
******************************************************

Enter the radius of a circle: 5

The circle's radius is 5.000
The circle's area is 78.540
The circle's diameter is 10.000
The circle's circumference is 31.416

*** Goodbye from YourFullName! ***
```

5.3 Question 3 (10 marks)

For this question, refer to the slides for "Unit 4: Building a Class and Object". Additional help can be found in the pre-recorded lecture videos from 2021 called "Introducing Class and Objects", "Encapsulation", and "Constructor". The IntelliJ Tutorials for "A Simple Class called Student", and "Constructors in Student Class" will be helpful as well. The concept of Constructors can sometimes be tricky, so please make sure you review before you start the question.

1. Create a package named **Q3.** In this package, you will create two Java classes.
2. Create a new Java class called **Student, <u>as shown in the UML diagram</u>.**
   Hint: make sure you know what "+", "-", and other features of a UML diagram to create the class correctly.

Specifications for the Student Class:

- Define the fields as shown in the UML diagram.
- The class should have the following methods:

Student
-studentNumber: int **-firstName: String** **-lastName: String** **-emailAddress: String** **-yearOfBirth: int**
+Student() **+Student(sn: int, fName: String, lName: String)** **+getStudentNumber(): int** **+getName(): String** **+getEmailAddress(): String** **+getAge(): int**

- Default **Constructor**: Accepts no arguments. Will assign all values to the fields studentNumber, firstName, lastName, and emailAddress based on information about yourself. Use the year 1998 (arbitrarily chosen) for the yearOfBirth field.
- **Constructor with parameters:** Accepts a student number, first name and last name as arguments, then assigns these values to the fields studentNumber, firstName and lastName. Also, it should assign an empty string to the emailAddress field, and the number 0 to the yearOfBirth field.
- **Accessor/Getter** methods for all the fields:
  - **getName()** returns the student's full name (the first name followed by a space followed by the last name).
  - **getNumber()** returns the student number.
  - **getEmail()** returns the student's email address.
  - **getAge()** calculates and returns the student's age based on the current year value, which is 2022. (e.g., if the student's year of birth is 2020, this method should return 2)

3. In the same Java package, create a new driver class with the name **StudentDemoClass*YourFirstName*.**
   - This class will contain three public-static methods: your driver method, and the header and footer methods you have defined earlier.

   The driver method (main() method) will be defined with the following specifications that will follow the sequence below:

- Print a custom header using the **myHeader** method.
- Print your information:
  - Declare a **Student** type reference variable and instantiate it to create an object with the help of the constructor with no arguments.
  - Using the **printf()** method, print the information of the above student object including the age (see the sample output).
- Now, using the data from the Sample Output shown in the table, declare three more Student type reference variables and instantiate the corresponding reference objects. For each object:
  - The Constructor with arguments will initialize the values for the fields studentNumber, firstName, and lastName. You can hard code these values without prompting the user.
  - The mutator/accessor methods will set/get the values for the rest of the instance fields. You can hard code these values without prompting the user.
- With the **printf()** method, the program will display the students' information. The output should follow the format shown in the sample output (see Unit 2 section 2.5 for formatting with field width in printf()).
- Print a custom footer using the **myFooter** method.

4. The program output should be something similar to the following Sample Output:

```
****************************************************
YourFullName
Lab 4, Question 3
*****************************************************

Here is my information:
=======================
I am yourFirstName yourLastName
Student Number: yourStudentNumber
Email address: yourEmailID@uwo.ca
Age: theAge

Here is the info on other students:
=============================
Number    Name
========= ===================
250221375 Addie Slowgrave
250309716 Talia Hanscom
250136525 Valeria McCloughen
==============================

*** Goodbye from YourFullName! ***
```

6 Submission

Zip the project file, name it ***username_Lab4.zip*** and 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