Opeyemi Adesina, PhD
Assistant Professor
Computer Information Systems
University of the Fraser Valley
Office: C2435, Abbotsford Campus
Tel: (604) 504-7441 (ext: 4931)
opeyemi.adesina@ufv.ca

Assignment 2 — OO Software Development

COMP 155 : Object-Oriented Programming

(100 points)

When Due : July 10, 2021 – 23:59:00 (PDT) [Submission via Blackboard]

Brief Description

This assignment amounts to **10%** of the entire course grade. In particular, whatever your obtains as a score will be scaled to this value for final grade computation. You are required to work **ALONE**. No late submission will be permitted (see deadline above).

The goal of this assignment is to assess your knowledge and skills on Object-oriented concepts with Java, while developing skills to map requirements (e.g. system model) to program code. You will find a grading scheme at the end of this document – to guide you on instructor's expectations while preparing your submission.

Software Requirements

In this assignment, we will be commencing an implementation of COVID-19 monitoring and alert system. In particular, we shall developing the building blocks for our desired system. The pandemic has hit hard and affected the way we conduct our day-to-day businesses. Besides, the economy is bleeding and needs urgent rescue strategies be deployed to ensure we return to normalcy.

This system will monitor COVID-19 in your neighborhood. It assumes there is a web service designated as a source of information about all persons in Canada (including) their COVID status. Information stored include: *name, age, sex, address* and *a unique identifier* (which could be a Social Insurance Number - SIN). Allowed values for sex include: male, female or unknown (when a person prefers not to specify).

An address is characterized with unit number, street number, street name, postal code, city, province and type. Allowed values for address type are business, home, office. As everyone is at the risk of contracting COVID-19, we allowed the following values for effective monitoring - *positive, negative, symptomatic* (people with some COVID symptoms but are negative), *asymptomatic* (people that are confirmed COVID positive but are not showing related symptoms), and *unknown.*

In addition, we introduce two shapes - rectangle and circle for the purpose of determining the proximity of infection and effectively disseminate orders and warning messages. The dissemination of information is based on an operational radius (which may be based on a policy). Whenever there is an infection, all the persons enclosed in the bounding rectangle of the operational radius are notified. Depending on the cluster of infections, an alert about order or warning is issued by our system.

1. The model in *figure 1* is a conceptual representation of a neighborhood. While red denotes people with confirmed positive COVID cases. In our case, these are people whose status is either positive or asymptomatic. Yellow is designated for people showing symptoms, while green designates people who neither show symptoms nor COVID positive. Our goal is to develop the code corresponding to these domain and implement the system in question.

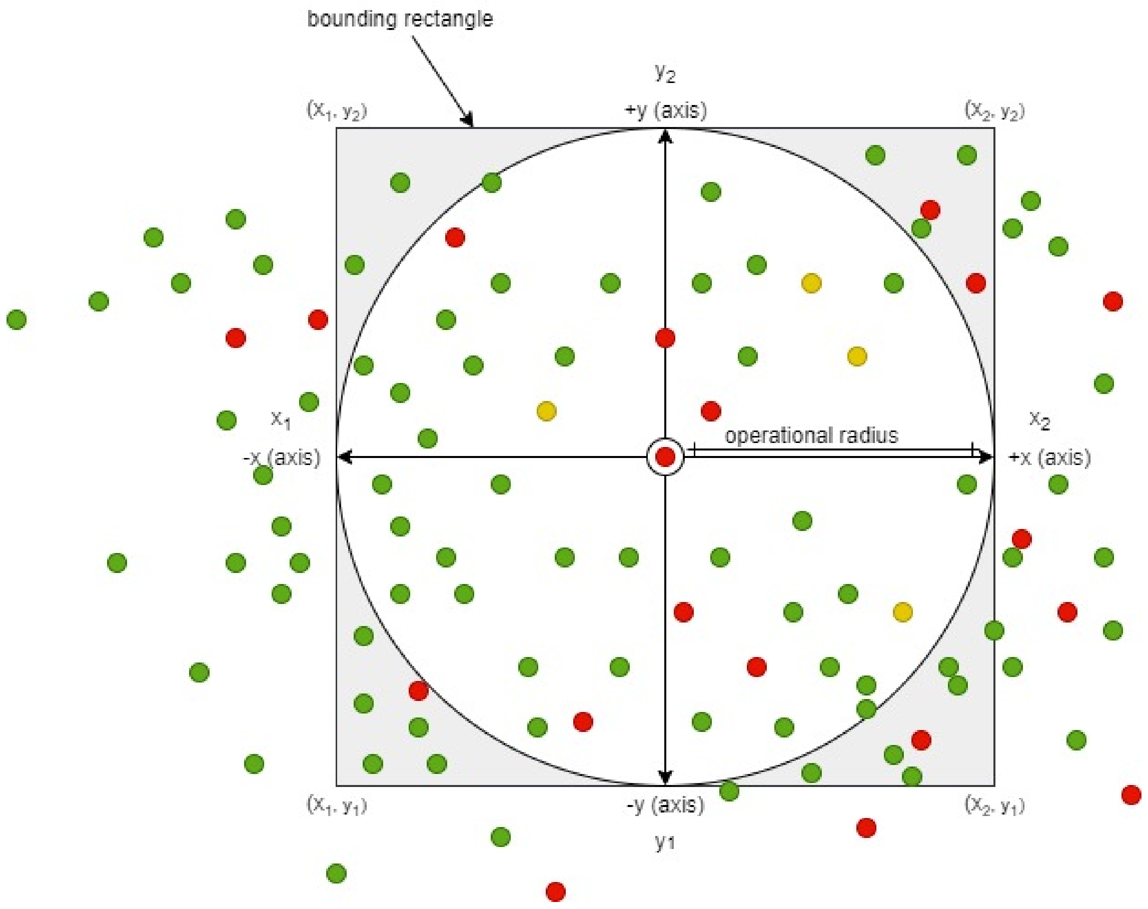


Figure 1: Conceptual model of a neighborhood – showing infections

2. It figure 2, we present a domain model which represents the internal structure of system. It is represented in a unified modelling notation with the goal of helping you understand the underlying structure and to help you navigate the code. **You are required to provide implementations for the following:**

13 points Location Class

- constructors
- accessor methods (getters and setters – for each instance variable)
- an overload of *equal*(. . .) method
- an overload of *toString*(. . .) method (see the output for acceptable format)

15 points Circle Class

- constructors
- accessor methods (getters and setters – for each instance variable)
- an overload of *equal*(. . .) method

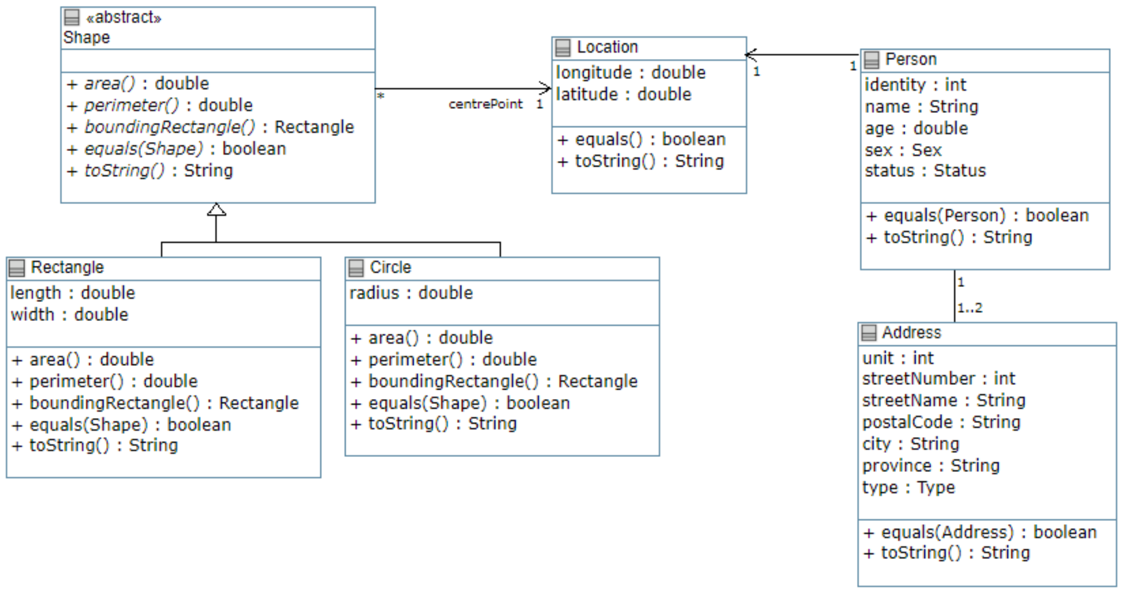


Figure 2: COVID-19 Alert and Monitoring Domain Model

- an overload of $toString(\dots)$ method (see the output for acceptable format)
- specialized implementations for the following:
  - double $perimeter(\dots)$ – $2\pi r$
  - double $area(\dots)$ – πr^2
  - Rectangle $boundingRectangle(\dots)$ – returns rectangle whose $length = width = 2r$.

15 points Rectangle Class

- constructors
- accessor methods (getters and setters – for each instance variable)
- an overload of $equal(\dots)$ method
- an overload of $toString(\dots)$ method (see the output for acceptable format)
- specialized implementations for the following:
  - double $perimeter(\dots)$ which is $2(l + b)$
  - double $area(\dots)$ which is $l * b$
  - Rectangle $boundingRectangle(\dots)$ – returns itself.

20 points Address Class

- constructors
- accessor methods (getters and setters – for each instance variable)
- an overload of $equal(\dots)$ method
- an overload of $toString(\dots)$ method (see the output for acceptable format)

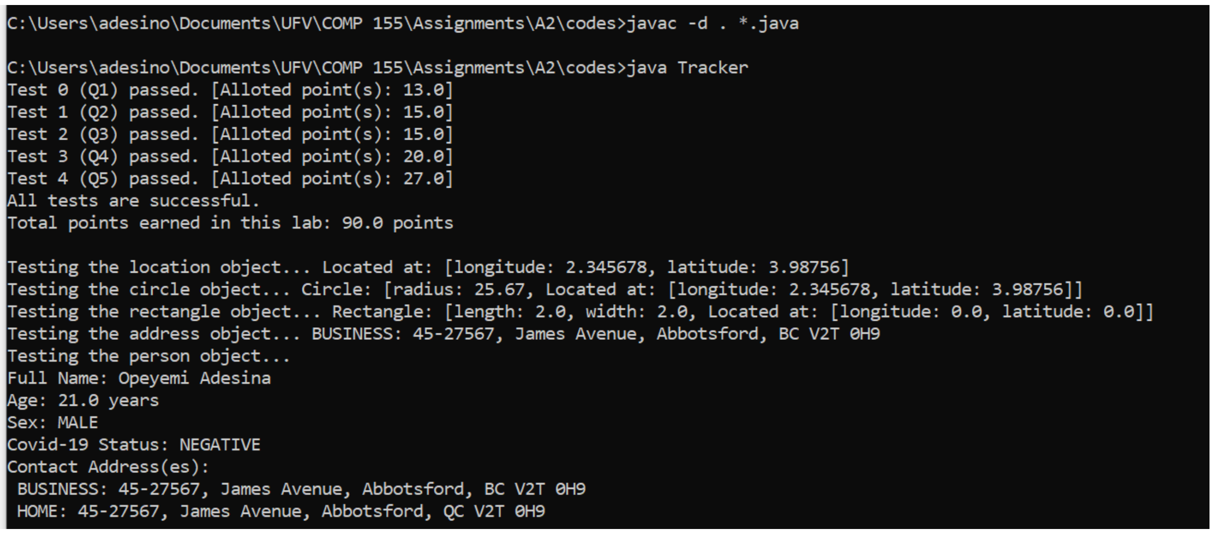
27 points Person Class

- constructors
- accessor methods (getters and setters – for each instance variable)
- an overload of $equal(\dots)$ method
- an overload of $toString(\dots)$ method (see the output for acceptable format)

For implementation purposes you will find statements of the form: ”INSERT YOUR CODE”. Apart from these places, do not modify any other part of the program unless you have consulted with the instructor. I have included a test cases for each of the classes you are to create and their associated methods in the *QualityAssurance.java* file. An example output for a run is given in *figure 3*. You can as well obtain hint to implement the *toString*(...) from the output (of example run) in the stub directory.

To compile and execute your code, please follow these chain of commands in the given sequence:

- javac -d . *.java (from the directory container the files - hard compile, compiling classes into their respective packages)
- java Tracker (program execution from the same directory)



```
C:\Users\adesino\Documents\UFV\COMP 155\Assignments\A2\codes>javac -d . *.java

C:\Users\adesino\Documents\UFV\COMP 155\Assignments\A2\codes>java Tracker
Test 0 (Q1) passed. [Alloted point(s): 13.0]
Test 1 (Q2) passed. [Alloted point(s): 15.0]
Test 2 (Q3) passed. [Alloted point(s): 15.0]
Test 3 (Q4) passed. [Alloted point(s): 20.0]
Test 4 (Q5) passed. [Alloted point(s): 27.0]
All tests are successful.
Total points earned in this lab: 90.0 points

Testing the location object... Located at: [longitude: 2.345678, latitude: 3.98756]
Testing the circle object... Circle: [radius: 25.67, Located at: [longitude: 2.345678, latitude: 3.98756]]
Testing the rectangle object... Rectangle: [length: 2.0, width: 2.0, Located at: [longitude: 0.0, latitude: 0.0]]
Testing the address object... BUSINESS: 45-27567, James Avenue, Abbotsford, BC V2T 0H9
Testing the person object...
Full Name: Opeyemi Adesina
Age: 21.0 years
Sex: MALE
Covid-19 Status: NEGATIVE
Contact Address(es):
 BUSINESS: 45-27567, James Avenue, Abbotsford, BC V2T 0H9
 HOME: 45-27567, James Avenue, Abbotsford, QC V2T 0H9
```

Figure 3: Expected Output

Grading Scheme

The following scheme will be used to grade your submission. Therefore, you may also use it as a guide in preparing your deliverable.

Grade Item	Weight
A syntactically and semantically correct program, passing all the test cases included. You will be able to monitor your progress using the test cases as a guide.	**90**
A program with detailed program documentation and uses sensible variable names. **Your program's file name and other files should be zipped and named in the following format - [firstName_lastName_studentID]**	**10**
Total	**100**