OBJECT-ORIENTED PROGRAMMING(CPSC 1811)

Assignment #2

Due Date: October 31, 2022

Mourad Bouguerra
mbouguerra@langara.ca

Part-I
Inheritance: Extending a Class

(50 marks)

✎ *Given the following startup code of a* **Java** *class*:

```
1 public class Point{
2   // The X coordinate of this Point
3   protected double x;
4   // The Y coordinate of this Point
5   protected double y;
6
7 }
```

1. Adding Setter & Getter Methods (8 marks)

(a) *Add a* **getter** *method to access the* `protected` *field* `x` **(2 marks)**

(b) *Add a* **setter** *method to set the* `protected` *field* `x` **(2 marks)**

(c) *Add a* **getter** *method to access the* `protected` *field* `y` **(2 marks)**

(d) *Add a* **setter** *method to set the* `protected` *field* `y` **(2 marks)**

2. Adding Constructors (7 marks)

(a) *Add a* **default constructor** `Point()` *that initializes a point at the origin* $(0, 0)$ **(2 marks)**

(b) *Add a* **constructor** *with two* **parameters** `Point(double x,double y)` *that initializes a point at the specified* (x, y) *coordinates* **(2 marks)**

(c) *Add a* **constructor** *with one* **parameter** `Point(Point p)` *that initializes a point at the* (x, y) *coordinates of the given point* `p` **(3 marks)**

3. Adding Methods **(9 marks)**

(a) *Add a method* `computeDistance(double x,double y)` *that computes the* **Euclidean** *distance between this point and point with* (x, y) *coordinates (use equation 1 on page 3)* **(3 marks)**

✏ *A Euclidean distance between two points* (x_1, y_1) *and* (x_2, y_2) *is given by the formula*

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

(b) *Add a method* `computeDistance(Point p)` *that computes the* **Euclidean** *distance between this point and point* `p` **(3 marks)**

(c) *Add a method* `toString()` *that returns a string representation of this point in the format* (x, y) **(3 marks)**

4. Method Overloading **(2 marks)**

(a) *List the methods of the* `Point` *class that are* **overloaded***?*

5. Extending The `Point` class **(24 marks)**

✎ *Add* `Point3D` *class that* `extends` *the* `Point` *class to represent a point with* (x, y, z) *coordinates*

(a) *Add a* **getter** *method to access the* `protected` *field* `z` **(2 marks)**

(b) *Add a* **setter** *method to set the* `protected` *field* `z` **(2 marks)**

(c) *Add a* **default constructor** `Point3D()` *that initializes a point at the origin* $(0, 0, 0)$ **(2 marks)**

(d) *Add a* **constructor** *with two* **parameter**s `Point3D(double x,double y,double z)` *that initializes a point at the specified* (x, y, z) *coordinates* **(3 marks)**

(e) *Add a* **constructor** *with one* **parameter** `Point3D(Point3D p)` *that initializes a point at the* (x, y, z) *coordinates of the given point* `p` **(3 marks)**

(f) *Add a method* `computeDistance(double x,double y,double z)` *that computes the* **Euclidean** *distance between this point and* 3*D point with* (x, y, z) *coordinates (use equation 2 on page 4)* **(3 marks)**

✏ *A Euclidean distance between two* 3*D points* (x_1, y_1, z_1) *and* (x_2, y_2, z_2) *is given by the formula*

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{2}$$

(g) *Add a method* `computeDistance(Point3D p)` *that computes the* **Euclidean** *distance between this point and* 3*D point* `p` **(3 marks)**

(h) *Add a method* `toString()` *that returns a string representation of this* 3*D point in the format* (x, y, z) **(3 marks)**

(i) *List the methods of the* `Point3D` *class that* **override** *methods of the* `Point` *class?* **(3 marks)**

Part-II
Inheritance: Implementing an Interface

(42 marks)

✎ **Irrational** *number is any* **real** *number that cannot expressed as a ratio of two* **whole** *numbers. The* **decimal expansion** *of an* **irrational** *number never* **repeat***s or* **end***s.*

1. `BigDecimal` Class (12 marks)

✎ *`BigDecimal` class represents* **large floating-point** *numbers with* **high precision**. *Read the* **Java** *documentation of `BigDecimal` (see link BigDecimal).*

✎ *`BigDecimal` class can be used in `Question` ② of this part.*

(a) *What is the* **superclass** *of `BigDecimal` class?* **(2 marks)**

(b) *What is the* **package** *of `BigDecimal` class?* **(2 marks)**

(c) *Which `interface` the `BigDecimal` class implements?* **(2 marks)**

(d) *How many* **instance** *variables the `BigDecimal` has?* **(2 marks)**

(e) *How many* **class** *variables the `BigDecimal` has?* **(2 marks)**

(f) *How many* **constructor***s the `BigDecimal` has?* **(2 marks)**

2. Implementing Java Interface **(30 marks)**

✎ *Given the following* **Java** *`interface`*:

```
1 public interface Irrational{
2   public double computePI(int precision);
3   public double computeEulerConstant(int precision);
4   public double computeSquareRootOfTwo(int precision);
5 }
```

✎ *Add* ***`IrrationalApproximation`*** **Java** *class that `implements` the interface* ***`Irrational`****?*

(a) *To implement `public double computePI(int precision) abstract` method use* **Leibniz** *formula for* Π *(use equation 3 on page 6)* **(10 marks)**

$$\Pi = 4 \times \sum_{i=0}^{\infty} \frac{(-1)^i}{2 * i + 1} = 4 \times (1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots) \tag{3}$$

(b) *To implement `public double computeEulerConstant(int precision) abstract` method use the formula for* e *given by equation 4 on page 6:* **(10 marks)**

$$e = \sum_{i=0}^{\infty} \frac{1}{i!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} \cdots = 1 + 1 + \frac{1}{6} + \frac{1}{12} + \dots \tag{4}$$

✏ ***`Factorial`*** *of non-negative integer* N *is denoted by* $N!$ *is given by the the following formulae (see equations 5 & 6)*

$$N! = 1 \times 2 \times 3 \times 4 \times \cdots \times (N-1) \times N \tag{5}$$

$$0! = 1 \tag{6}$$

(c) *To implement `public double computeSquareRootOfTwo(int precision) abstract` method use the* **Babylonian** *method for approximating* $\sqrt{2}$ *(see equation 7 on page 6)* **(10 marks)**

① `Start with an initial approximation` X_0
② `Compute a new approximation` X_1 `using` $X_1 = \frac{1}{2}(X_0 + \frac{2}{X_0})$
③ `Repeat step ② until you get an approximation with the specified precision`

$$X_{n+1} = \frac{1}{2}(X_n + \frac{2}{X_n}) \tag{7}$$

Submission

✎ *You have to submit a* ***`YourName-ID.jar`*** *file that is the archive of the following folder, subfolders, and files (see figure 1 on page 7)*

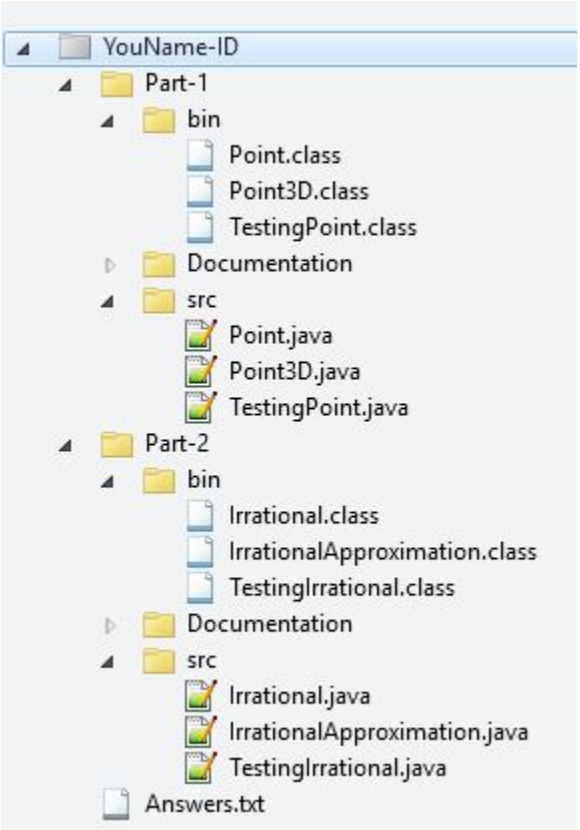

Figure 1:

✏ ***`Answers.txt`*** *file should include your answers to the questions of* ***`Part I`*** *and* ***`Part II`***

Marking Scheme

Task	Marks
Part I	50
Part II	42
Coding Style	4
Using javadoc to generate documentation	2
Using jar to archive all assignment files	2
Total	**100**