COMP 348: Principles of Programming Languages

Assignment 3 on Ruby and AspectJ

Fall 2020, sections U and DD

November 17, 2020

Contents

1 General Information

Date posted: Tuesday November 17th, 2020.

Date due: Tuesday December 8th, 2020, by 23:59[1].

Weight: 6% of the overall grade.

2 Introduction

This assignment targets two programming paradigms: 1) Multi-Paradigm Programming with Ruby, 2) Aspect Oriented Programming with AspectJ.

3 Ground rules

You are allowed to work on a team of 4 students at most (including yourself). Each team should designate a leader who will submit the assignment electronically. See Submission Notes for the details.

ONLY one copy of the assignment is to be submitted by the team leader. Upon submission, you must book an appointment with the marker team and demo the assignment. All members of the team must be present during the demo to receive the credit. Failure to do so may result in zero credit.

This is an assessment exercise. You may not seek any assistance from others while expecting to receive credit. **You must work strictly within your team).** Failure to do so will result in penalties or no credit.

4 Your Assignment

Your assignment is given in two parts, as follows. 1) Multi-Paradigm Programming with Ruby, 2) Aspect Oriented Programming with AspectJ.

[1] see Submission Notes

4.1 Multi-Paradigm Programming with Ruby

This section consists of three parts that are related, but may be implemented independently.

Q 1. Classes in Ruby

Define the following class hierarchy in Ruby

1. Shape
   (a) : the class constructors receives no parameter.
   (b) : method print(): prints the name of the shape, perimeter, and area of the shape. The name of the shape is the class name (i..e Shape for this class).
   (c) : method perimeter(): default method to be implemented by child classes; returning nil by default.
   (d) : method area(): default method to be implemented by child classes; returning nil by default.
2. Circle
   (a) : the class constructor receives the radius as the only parameter.
   (b) : method perimeter(): overridden, returns the perimeter of the circle.
   (c) : method area(): overridden, returns the area of the circle.
3. Rectangle
   (a) : the class constructor receives the height and width of the rectangle.
   (b) : method perimeter(): overridden, returns the perimeter of the rectangle.
   (c) : method area(): overridden, returns the area of the rectangle.
4. Ellipse
   (a) : the class constructor receives the semi-major and semi-minor axes (a and b).
   (b) : method perimeter(): not to be implemented.

(c) : method area(): overridden, returns the area of the ellipse ($A = \pi ab$).

(d) : method eccentricity(): additional method that returns the eccentricity of the ellipse ($e = \sqrt{a^2 - b^2}$).

Q 2. File / Text Processing

Write a Ruby program that reads text file that contains the shape information (see previous question). Every line in the text file consist of shape name and parameters to construct the shape. The program create every shape and calls the print method and displays the result on the screen. In case of errors (i.e. having negative values for sides, radius, etc.), and error message may be displayed. An example given below:
Input:

```
shape
rectangle 10 20
rectangle 0 10
circle 2
ellipse 2 4
ellipse -1 4
```

Output:

```
Shape, perimeter: undefined, area: undefined
Rectangle, perimeter: 60, area: 200
Rectangle, perimeter: 0, area: 0
Circle, perimeter: 12.5663706144, area: 12.5663706144
Ellipse, perimeter: undefined, area: 25.1327412287
Error: Invalid Ellipse
```

Q 3. Arrays and Hash

Extend the above program to display a statistics of the shapes after the text file is processes. An example is given in the following

Output:

```
Statistics:
    Shape(s): 5
    Rectangle(s): 2
    Circle(s): 1
    Ellipse(s): 1
```

Use hashes to implement the above structure in memory.

Note that rectangles, circles, and ellipses are counted as shapes as well.

4.2 Aspect Oriented Programming with AspectJ

Consider the code base and the driver code in sections 4.2.7 and 4.2.6. The code base provides an implementation for `Circle`, `Rectangle`, and a `Shape` interface. Note that neither `Circle` not `Rectangle` implement the `Shape` interface. Hence, executing the driver code will result in the following output:

```
Error: Rectangle cannot be cast to class Shape
```

The task of this assignment is to make the output look like to the following, as instructed by the following sections.

```
The area of Rectangle(2, 10) is 20.0
The perimeter of Rectangle(-2, 10) is 16.0
The perimeter of Circle(-2) is 0.0
The area of Circle(2) is 12.566370614359172
The last shape ID: 4
```

IMPORTANT: All changes must **strictly** be applied via *aspects*. Neither the code base nor the driver code may be changed[2]. All of the following sections may be implemented within a single aspect.

[2]Except uncommenting the two lines of code corresponding to the section 4.2.5

4.2.1 Introducing Shape Parent

Q 4. Using aspects, introduce Shape as a parent interface for both `Circle` and `Rectangle`.

To implement `getName()` method, the `Circle` object returns "`Circle`" and the `Rectangle` object returns "`Rectangle`".

4.2.2 Overriding toString()

Q 5. Using aspects, override the `toString()` method in both `Circle` and `Rectangle` classes.

The `toString()` methods call the `getName()` in 4.2.1 and generates a string, representing the object.

Examples:

- A `Circle` with a radius of 2 is represented as the string "`Circle(2)`".
- A `Rectangle` with a width of 2 and a height of 10 is represented as "`Rectangle(2, 10)`".

4.2.3 Implementing Circle.getArea()

Q 6. The logic for calculating the area of the circle has not been implemented (as it currently throws a `RuntimeException`). Using aspects, implement it.

4.2.4 Handling Negative Values

Q 7. Using aspects, make sure passing a negative value to a `radius` of a `Circle` or to either `width` or `height` of a `Rectangle` results in 0 value for the value of the area and the perimeter.

4.2.5 Introducing Shape IDs

We want to generate a unique ID for every shape that is created. The shape IDs start with 1 and is automatically incremented for every newly created shape. To do so, a private static counter may be implemented in an aspect that is incremented automatically upon calling the constructors for both `Circle` and `Rectangle`.

Q 8. Using aspects, implement the `getId()` method for both shapes.

Note that you need to uncomment the corresponding sections in both the code base and the driver code.

4.2.6 Driver

```
public class Main {
    public static void main(String[] args) {
        try {
            Shape s;
            s = (Shape) new Rectangle(2, 10);
            System.out.println("The area of " + s + " is " + s.getArea());
            s = (Shape) new Rectangle(-2, 10);
            System.out.println("The perimeter of " + s + " is " + s.getPerimeter());
            s = (Shape) new Circle(-2);
            System.out.println("The perimeter of " + s + " is " + s.getPerimeter());
            s = (Shape) new Circle(2);
            System.out.println("The area of " + s + " is " + s.getArea());
            // Uncomment the following (required by 4.3.5)
            // System.out.println("The last shape ID: " + s.getId());
        }
        catch(Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

4.2.7 Code Base

```
public interface Shape {
    // Uncomment the following (required by 4.3.5)
    // int getId();
    String getName();
    double getPerimeter();
    double getArea();
}
public class Rectangle {
    private double width, height;
    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }
    public double getPerimeter() {
        return 2 * (this.width + this.height);
    }
    public double getArea() {
        return this.width * this.height;
    }
}
public class Circle {
    private double radius;
    public Circle(double radius) {
        this.radius = radius;
    }
    public double getPerimeter() {
        return 2 * Math.PI * this.radius;
    }
    public double getArea() {
        throw new RuntimeException("Oops, I don't know how to calculate this :(");
    }
}
```

5 What to Submit

The whole assignment is submitted by the due date under the corresponding assignment box. Your instructor will provide you with more details. It has to be completed by ALL members of the team in one submission file.

Submission Notes

Clearly include the names and student IDs of all members of the team in the submission. Indicate the team leader.

IMPORTANT: You are allowed to work on a team of 4 students at most (including yourself). Any teams of 5 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted. You must make sure that you upload the assignment to the correct assignment box on Moodle. No email submissions are accepted. Assignments uploaded to the wrong system, wrong folder, or submitted via email will be discarded and no resubmission will be allowed. Make sure you can access Moodle prior to the submission deadline. The deadline will not be extended.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention. The zip file should be called a#_studids, where # is the number of the assignment, and studids is the list of student ids of all team members, separated by (_). For example, for the first assignment, student `12345678` would submit a zip file named `a1_12345678.zip`. If you work on a team of two and your IDs are `12345678` and `34567890`, you would submit a zip file named `a1_12345678_34567890.zip`.
Submit your assignment electronically on Moodle based on the instruction given by your instructor as indicated above: `https://moodle.concordia.ca`
Please see course outline for submission rules and format, as well as for the required demo of the assignment. A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. You must keep

a record of your submission confirmation. This is your proof of submission, which you may need should a submission problem arises.

6 Grading Scheme

Q1	20 marks
Q2	20 marks
Q3	15 marks
Q4	10 marks
Q5	5 marks
Q6	5 marks
Q7	10 marks
Q8	15 marks

Total: 100 marks.

References

1. Ruby Programming Language: `https://www.ruby-lang.org/en/`
2. AspectJ:

   `https://www.ibm.com/developerworks/library/j-ajdt/index.html`