Computer Science

COMPSCI 230

Assignment ONE

Due Date

Due: **11:59 pm Saturday 2nd April 2022**
Worth: **3% of the final mark**

Introduction - The Bouncing Program

The aim of the assignment is to give you experience with object-oriented programming, principles of inheritance and polymorphism. The application, as given, is a simple bouncing program designed to let different shapes move around along various paths. Users will be able to create shapes based on the classes you will write, and to select individual existing shapes on the panel and change their properties. This project has 3 iterations and your final program should be able achieve the following:

Shape Creation:
The user can create a new shape by clicking anywhere within the panel area of the program. The properties of the newly created shape are based on the current values saved in the `AnimationViewer` class. Once created, the shape will start moving.

Selecting/deselecting shapes:
A user can select a shape by clicking anywhere on the shape. A selected shape shows all it handles. The user can change the path type / width / height / fill colour for all selected shapes by changing the current values with the help of the tools provided at the top of the application GUI. (Note: The shape type itself cannot be modified once a shape has been created.) Clicking on a selected shape will deselect it.

A1 requires you to complete the first iteration of this project. You will notice that there are some graphical Java elements forming the GUI (Graphical User Interface) of the program. The detailed knowledge about these elements will arrive later in the course. Note that, for this assignment, the code for implementing these elements is already embedded in the program provided to you. So, all you need to do is to complete the specific parts of this larger program. This document contains the necessary instructions for completing these missing parts, that when implemented will activate the relevant features this application is required to produce.

The first iteration is only a TEXT based version of the program. The program simply prints the details of each shape to a `JTextArea`. A1 is divided into two parts for marking purposes. You should complete 17 questions in CodeRunner (10 marks) and complete the entire working program (5 marks). Submit the entire program to the assignment dropbox.

Note: You are strongly recommended to go back and forth between working on the program in your IDE and making CodeRunner task submissions, so that by the end of this process you have a complete "working" Java application ready to be submitted.

Assignment Tasks

Download all basic source files from Canvas. The files included in the program are as follows:

- `ShapeType.java, PathType.java, Shape.java, Rectangle.java,`
- `Painter.java, TextPainter.java, AnimationPanel.java and A1.java`

A1 is divided into several stages for ease of completion. Please complete the assignment in order of the stages. You need to be familiar in particular with the purpose of three methods in `Shape.java`, which you may wish to override in the new shape subclasses you will create:

- `draw()`: This method actually draws the shape in question, using an object that is a subclass of the abstract `Graphics2D` class, which is part of the Java `AWT` graphics package and extends the `Graphics` class in that package. You will need to override this method in the subclass you create, and ensure that the respective shape is drawn properly.
- `contains()`: This method takes a `Point` parameter and is meant to return true if the `Point` is inside the shape and false if it is not. Since you will be creating different shapes, you will need to override this method for each shape that has a new outline, unless it makes sense to simply inherit it from an ancestor class with the same outline.
- `getArea()`: This method returns the area of a particular shape.

Download A1 Help powerpoint slide and video from Canvas. The slide explains the details about the relationship between classes and the steps in creating those new shapes. The video demonstrates the expected behaviour of the final program.

CodeRunner Questions (10 marks)

Stage 1: The `SquareShape` Class

Define a subclass named `SquareShape` which represents a square. This class should create a **NEW** square based on the mouse-point, the minimum value of current width and current height, current fill colour, and current moving path saved in the `AnimationViewer`. What is the superclass of the `SquareShape`? Should you use `Shape` or `RectangleShape` as the superclass? You should complete the following steps:

- Modify the `ShapeType` enum
- Define a subclass named: `SquareShape`

Stage 2: The `OvalShape` Class

Define a subclass named `OvalShape` which represents an oval/ellipse. This class should create a **NEW** oval/ellipse based on the mouse-point, current width, current height, current fill colour, and current moving path saved in the `AnimationViewer`. Use the following formula to check if the mouse point is in the ellipse or not:

```
dx = (2 * mx - x - x1) / w
dy = (2 * my - y - y1) / h
d =  dx * dx + dy * dy
```

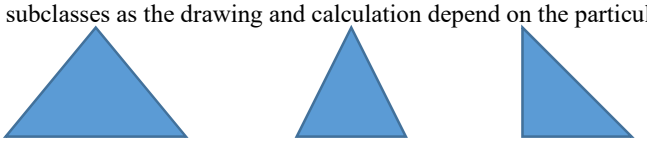
The mouse point `(mx, my)` is within the circle/ellipse if `d` is less than 1 where `(x, y)` is the top left corner, `(x1, y1)` is the bottom right corner, `w` is the width of the embedded rectangle and `h` is the height of the embedded rectangle.

Complete the following steps:

- Modify the `Painter` interface and the `TextPainter` class
- Modify the `ShapeType` enum
- Define a subclass named: `OvalShape`

Stage 3: The `TriangularShape` Class & subclass

Define a subclass (of `Shape`) named `TriangularShape` which represents the superclass of 3 types of triangles, they are: isosceles, equilateral and right angle (45-45-90). The `TriangularShape` contains a private `Polygon(java.awt)` data field named `polygon` that defines the coordinates of a triangle. The `TriangularShape` class contains the `contains()` method which takes a `Point(java.awt)` as a parameter and returns true if the point is located inside the triangle; false otherwise. However, the `TriangularShape` should not contain the `draw`() and `getArea`() methods. The `draw`() and `getArea`() should be defined in the subclasses as the drawing and calculation depend on the particular type of a triangle.



Complete the following steps:

- Modify the `Painter` interface and the `TextPainter` class
- Modify the `ShapeType` enum
- Define the `TriangularShape`, `IsoscelesTriangle`, `EquilateralTriangle` and `Right454590Triangle` subclasses

Stage 4: Creating new shape - The AnimationViewer class

You are required to modify the `createNewShape(int x, int y)` method in the `AnimationViewer` class to create a new shape. The method should create a new rectangle, oval or triangle shape and add the new shape to the `shapes` array list. The method should use the following values in the `AnimationViewer` class to create a new shape:

- `x` and `y` method parameters define the point representing the top left corner of a shape.
- `currentWidth` and `currentHeight` fields are declared inside `AnimationViewer` class and define the dimensions of a shape.
- `marginWidth` and `marginHeight` fields are declared inside `AnimationViewer` class and define the boundaries of the bouncing area.
- `currentColor` field is declared inside `AnimationViewer` class and defines the fill colour of a shape.
- `currentPathType` field is declared inside `AnimationViewer` class and defines the moving path of a shape.
- `currentShapeType` field is declared inside `AnimationViewer` class and defines the shape type of a shape. If the `ShapeType` is `'RECTANGLE'`, the method should add a rectangle to the array list. If it is "`OVAL`", the method should add an oval to the array list. If it is "`SQUARE`", the method should add a square to the array list and so on.

Stage 5: The `BoundaryPath` class

The `abstract MovingPath` class contains an abstract move method which moves all shapes in the array list. The `MovingPath` class is an inner class which is defined inside the `Shape` class. The `BouncingPath` is a concrete class and implements the `move()` method to move shapes in a bouncing motion. Define a concrete class named `BoundaryPath` which contains the `move`() method such that shapes are moving around the **boundary** of the bouncing area. The `BoundaryPath` should extend the `MovingPath` and implement the `move`() method. You should complete the following steps:

- Modify the `PathType` enum

- Define an inner member subclass: `BoundaryPath`

Stage 6: Creating new Path - The `Shape` class

You are required to modify the `setPath()` method in the `Shape` class such that shapes are moving either in a bouncing path or boundary path based on the following:

- The method should set the current path to a `BouncingPath` path if the PathType is "`BOUNCE`". The method should create a bouncing path with `delaX` =1 and `delaY` = 2.
- The method should set the current path to a `BoundaryPath` path if the PathType is "`BOUNDARY`". The method should create a boundary path with `delaX` = 5 and `delaY` = 5.

The Final Program (5 marks)

Complete the program. *When the user clicks the "`Add`" button for the first time, the program should create a **rectangle** which moves in the "**`BOUNCE`**" path (but not visible yet). When the user clicks the "`Add`" button for the second time , the program should create a **square** which moves in the "**`BOUNDARY`**" path. When the user clicks the "`Add`" button next, the program should create an **oval** which moves in the "**`BOUNCE`**" path. When the user clicks the "`Add`" button again, the program should create an **isosceles** triangle which moves in the "**`BOUNDARY`**" path and so on.*

When the user clicks the "Show" button, the program should display the details of each shape in the shapes array list.

Assessment Criteria

Complete CodeRunner A1 question submissions **AND** submit the entire program via the **assignment dropbox (https://adb.auckland.ac.nz/)** at any time from the first submission date up until the final due date. You will receive an electronic receipt. Submit ONE A1.zip file containing all the source files. **Remember to include your name, UPI and a comment at the beginning of each file you create or modify.**

You may make more than one submission, but note that every submission that you make replaces your previous submission. Submit ALL your files in every submission. Only your very latest submission will be marked. Please double check that you have included all the files required to run your program in the zip file before you submit it. Your program must compile and run to gain any marks. We recommend that you check this on the lab machines before you submit.

Marking Criteria

CR	10 marks	Complete 17 questions in CR
Q1	1	Include your name, UPI and a comment at the beginning of ALL YOUR FILES.
Q2	1	Users should be able to create new shapes using the Add button
Q3	1	Users should be able to show the details of each shape using the Show button
Q4	1	Shapes are created in the correct order
Q5	1	Shapes are moving in the correct order: bouncing and boundary alternately.

ACADEMIC INTEGRITY

The purpose of this assignment is to help you develop a working understanding of some of the concepts you are taught in the lectures. We expect that you will want to use this opportunity to be able to answer the corresponding questions in the tests and exam. We expect that the work done on this assignment will be your own work. We expect that you will think carefully about any problems you come across, and try to solve them yourself before you ask anyone for help. The following sources of help are not acceptable:

- Getting another student, or other third party to instruct you on how to design classes or have them write code for you.
- Taking or obtaining an electronic copy of someone else's work, or part thereof.
- Give a copy of your work, or part thereof, to someone else.
- Using code from past sample solutions or from online sources dedicated to this assignment.

The Computer Science department uses copy detection tools on all submissions. Submissions found to share code with those of other people will be detected and disciplinary action will be taken. To ensure that you are not unfairly accused of cheating:

- Always do individual assignments by yourself.
- Never give any other person your code or sample solutions in your possession.
- Never put your code in a public place (e.g., Piazza, forum, your web site).
- Never leave your computer unattended. You are responsible for the security of your account.
- Ensure you always remove your USB flash drive from the computer before you log off, and keep it safe.