Computer Science

COMPSCI 230
Assignment ONE

Introduction

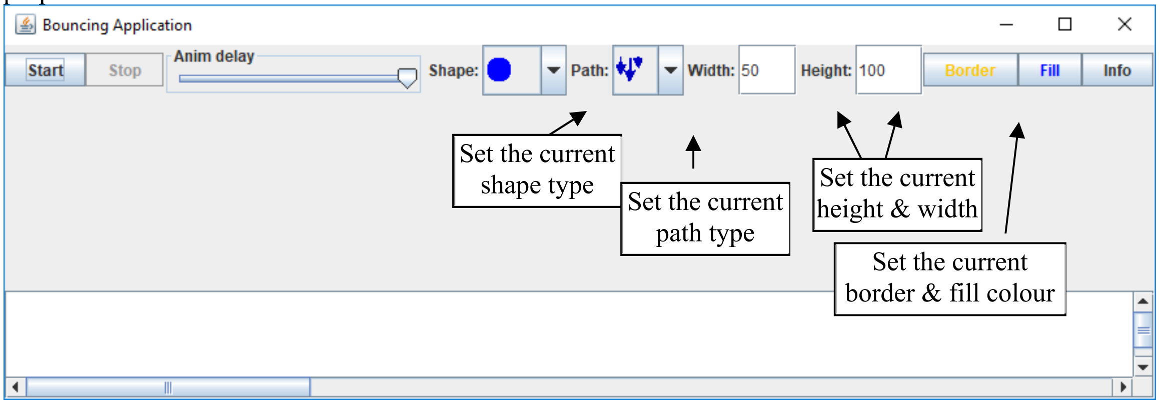
In this programming assignment, you are asked to add extra functions to a skeleton bouncing program that is provided to you. The aim of the assignment is to give you experience with object-oriented programming, principles of inheritance and polymorphism.

Due Date

Due:	**11:59 pm Wednesday 29th April 2020**
Worth:	**10% of the final mark**

Introduction - The Bouncing Program

The application, as given, is a simple bouncing program designed to let different shapes move around along various paths. Most of the code is already provided for you, but you will need to add more shapes. The program is easy to use: The only user actions are to create shapes based on the classes you will write, and to select individual existing shapes on the panel and change their properties.


Bouncing Application
Start
Stop
Anim delay
Shape:
Path:
Width: 50
Height: 100
Border
Fill
Info
Set the current shape type
Set the current path type
Set the current height & width
Set the current border & fill colour

Actions

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 appropriate UI fields (e.g. height, width border colour, fill colour and path). 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 its handles. The user can change the path types/widths/heights/border colours/fill colours for all selected shapes by changing the current values with the help of the tools provided at the top of the application interface. (Note: The shape type itself cannot be modified once a shape has been created.) Clicking on a selected shape will deselect it.

Tools

Shape Combo Box:	The 'Shape' combo box lets you select the shape type for the new shapes that get created when you click on the panel area. In the skeleton application, the combo box is pre-populated with icons for the shape classes you will create. Clicking in the panel area to create a shape as described above will then create a shape of the selected type.
Path Combo Box:	Users may select one of several moving paths for shapes from the 'Path' combo box. Selecting a new path changes the path of all currently selected shapes. The newly selected path also becomes the current path for any new shapes that the user creates. In the skeleton program, two paths are available: a "falling path" that sees shapes move from the top of the panel to the bottom with a little bit of back-and-forth sideways movement, and a "bouncing path" that lets the shape bounce off whichever boundary it hits.
Width TextField:	Users may change the current width of new shapes and currently selected shapes by entering a valid number in the width text field and pressing "ENTER".
Height TextField:	Users may change the current height of new shapes and currently selected shapes by entering a valid number in the height text field and pressing "ENTER".

Border Button	Users may change the current border colour of new shapes and currently selected shapes by pressing the border button.
Fill Button	Users may change the current fill colour of new shapes and currently selected shapes by pressing the fill button.
Start Button:	Starts the animation.
Stop Button:	Stops the animation.
Animation Slider:	Users may use the animation delay slider to adjust the speed of the animation.
Info Button:	Get a list of information (e.g. position, area, colours etc) of all shapes in the program. The list is sorted by the areas.
Text area	Display a list of information of all shapes in the program.
Popup Menu:	The application has a popup menu, which is activated by clicking the right mouse button anywhere in the panel area (on a windows machine). The popup menu contains a menu item called "Clear All" which allows the user to clear all shapes from the program.

What you are to do

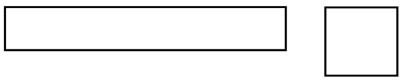
Download all source files from Canvas. The files included in the program are as follows:
- `A1.java`
- `AnimationPanel.java`
- `MovingShape.java`

In order to complete the assignment, you will need to create additional classes and make changes to `AnimationPanel`.java, `MovingShape`.java and `A1`.java. Your assignment 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 `MovingShape`.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 gets 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 the respective shape.

Once you have created a new shape subclass, you will need to add it to `AnimationPanel`. The `createNewShape`() method in AnimationPanel.java is the place to do this.

Stage 1: The `MovingRectangle` Class (6 marks)

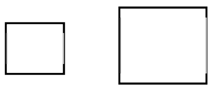
You are required to add a `MovingRectangle` subclass to your program. This class should draw a rectangle/square based on the mouse-point, the current width, the current height, the current border and fill colours and, the current moving path saved in the `AnimationPanel`.



Assessment criteria:
- [2 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice and the constructors are implemented correctly.
- [2 marks] The `contains()` and `getArea`() method are overridden correctly.
- [1 mark] The `draw()` method is overridden correctly.
- [1 mark] Users should be able to change the properties of the selected rectangles.

Stage 2: The `MovingSquare` Class (6 marks)

You are required to add a `MovingSquare` subclass to your program. This class should create a **NEW SQUARE** based on the mouse-point, the size (i.e. the smallest dimensions from the current width and height), border colour, fill colour, and moving path saved in the `AnimationPanel`.



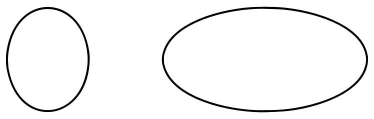
Assessment criteria:
- [2 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice and the constructors are implemented correctly.
- [1 mark] Do you need to override the `draw`(), `contains`() and `getArea`() in the `MovingSquare` class? Do you need to override any other existing methods?)
- [1 mark] A square should not be transformed into a rectangle when increasing its width or height.
- [2 marks] Users should be able to create a **NEW** square using the current properties in the program. (Note: the size of a **NEW** square is the **smallest** dimensions from the current width and height) and be able to change the properties the selected squares.

Stage 3: The `MovingEllipse` Class (6 marks)

You are required to add a `MovingEllipse` subclass to your program. This class should create a **NEW** ellipse based on the mouse-point, the width, height, border colour, fill colour, and moving path saved in the `AnimationPanel`.

Use the following formula to check if the mouse point is in the ellipse or not

```
double dx, dy;
Point EndPt = new Point(x + width, y + height);
dx = (2 * mousePt.x - x - EndPt.x) / (double) width;
dy = (2 * mousePt.y - y - EndPt.y) / (double) height;
return dx * dx + dy * dy < 1.0;
```

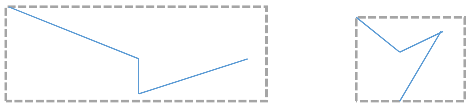


Assessment criteria:

- [2 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice and the constructors are implemented correctly.
- [2 marks] The `contains()` and `getArea()` method are overridden correctly.
- [1 mark] The `draw()` method is overridden correctly.
- [1 mark] Users should be able to change the properties of the selected ellipses.

Stage 4: The `MovingStarsMap` Class (6 marks)

You are required to add a `MovingStarsMap` subclass to your program. This class should create a **NEW** stars- map which based on the mouse-point and the current properties saved in the `AnimationPanel`. A stars-map contains 3 line-segments in the rectangular boundary given by the current width and height. The first point starts at the mouse-point. The other two points are randomly generated during the creation process.

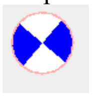


Assessment criteria:

- [2 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice and the constructors are implemented correctly.
- [1 mark] The stars-map shape contains three lines.
- [1 mark] Should you override the `contains()` and `getArea()` methods?
- [2 marks] Users should be able to create a **NEW** stars-map using the current properties in the program and be able to change the properties of the selected stars-map shapes.

Stage 5: The `MovingSpinningCircle` Class (6 marks)

You are required to add a `MovingSpinningCircle` subclass to your program. This class should create a **NEW** spinning circle which based on the mouse-point, the size, border colour, fill colour, and moving path saved in the `AnimationPanel`.

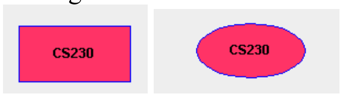


Assessment criteria:

- [2 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice and the constructors are implemented correctly.
- [1 mark] Do you need to override the `draw()`, `contains()` and `getArea()` in this class? Do you need to override any other existing methods?)
- [1 mark] A circle should NOT be transformed into an ellipse when increasing its width or height.
- [1 mark] Users should be able to create a **NEW** spinning circle using the current properties in the program. (Note: the size of the circle is the **smallest** dimensions from the current width and height) be able to change the properties of the selected circles.

Stage 6: The `Message` property (8 marks)

You are required to add a message property to the `BouncingShape` program. You are required to modify the `AnimationPanal` class, `MovingShape` class and all subclasses of `MovingShape`. The textbox and GUI event have been completed for you. The default message is "CS230". The message should be drawn at the centre of each shape.



Assessment criteria:

- [2 marks] Modify the `MovingShape` class correctly.
- [2 marks] Modify the `AnimationPanal` class correctly.
- [1 mark] Constructors in the `MovingShape` and all subclasses are modified correctly.
- [1 mark] The `drawMessage()` method is implemented correctly
- [2 marks] The `draw()` method in the `MovingShape` and all subclasses are modified correctly.

Stage 7: The `BoundaryPath` class (5 marks)

In this part, you are required to add a special simple boundary path to the bouncing program. The `MovingPath` is an abstract inner class which contains an abstract method. The new subclass should extend the `MovingPath` and implement the `move()` method.
Assessment criteria

- [2 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice and the constructor is implemented correctly.
- [3 marks] Users should be able to add a new shape that moves around the boundary of the `JPanel`.

Stage 8: Sorting (5 marks)

In this part, you are required to complete the `getSortedInfo()` method to display a list of sorted `MovingShape` objects. The `movingShape` class should implement the `Comparable` interface and implement the `compareTo()` method. The list should be sorted in ascending order by area.
Assessment criteria

- [2 marks] The `movingShape` class should implement the Comparable interface and implement the `compareTo()` method.
- [3 marks] Information of shapes are displayed in ascending order by area.

Stage 9: Complete the Program (2 marks)

Assessment criteria

- [2 marks] Include your name, UPI and a comment at the beginning of **ALL YOUR FILES**.
- Users should be able to draw a new rectangle/square/ellipse/stars-map/spinning circle and modify the properties.

Bonus

You are now required to get creative and add your own special shape(s) that will make the bouncing program more interesting! You may want to add Text, Images, 3D shapes, Sound etc into the bouncing program.

Submission

Complete CodeRunner A1 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 following files:

1. All source files (i.e. new, changed, and unchanged) – remember to include your name, UPI and a comment at the beginning of each file you create or modify.
2. All gif files (used as icons in the program)
3. A1.txt. It must contain the following information:
   - Your name, login name (such as jlim123) and (7 or 9 digit) ID number
   - The JDK Version that you used in this program
   - How much time did the assignment take you overall?
   - Which areas of the assignment did you find easy and which ones were difficult?

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 and A1.txt (see below) 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.**

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.