| ◤ Computer Science | **COMPSCI 230**<br>**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.

This assignment is marked out of 60 marks, with 50 marks being the equivalent of "full marks" for this assignment. Any excess marks may be used to compensate for loss of marks in other assignments or Coderunner.

## *Due Date*

Due:                        **11:59 pm Sunday 14ᵗʰ April 2019**
Worth:                      **5% 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.
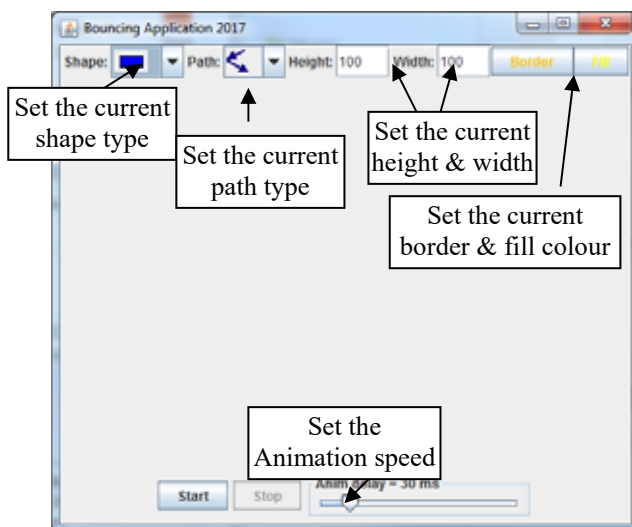


**Actions**
**Shape Creation:**
The user can create a new shape by clicking anywhere within the panel area of the program. The properties of the newly created shape are based on the current values saved in the appropriate UI fields (e.g. height, width border colour, fill colour and path). Once created, 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. |
| 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

Firstly, become familiar with the program supplied. The files included in the program are as follows:
- `A1.java`
- `AnimationPanel.java`
- `MovingShape.java`
- `MovingOval.java`

Download all source files from the assignment course page. You are required to modify the program and add a few more shapes into the program. Your assignment is divided into several stages for ease of completion. Please complete the assignment in order of the stages.

In order to complete the assignment, you will need to create additional classes and make changes to `AnimationPanel.java`, `MovingShape.java` and `A1.java`. You need to be familiar in particular with the purpose of two 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 every shape subclass you create, and ensure that the respective shape gets drawn properly. You should find the `API` documentation of the `Graphics2D` class useful: https://docs.oracle.com/javase/10/docs/api/java/awt/Graphics2D.html
- `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.

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.

You must include your name, UPI and a comment at the beginning of each file you create or modify.

## Stage 1: The `MovingRectangle` Class (5 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`. Some examples are shown as below:
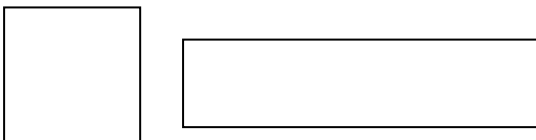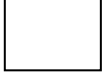
Assessment criteria:
- [1 mark] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice.
- [1 mark] The draw method is overridden correctly
- [1 mark] The contains method is overridden correctly
- [1 mark] Users should be able to create a new rectangle using the current properties in the program.

- [1 mark] Users should be able to change the fill colour, border colour, width, height and bouncing path of selected rectangles.

## Stage 2: The `MovingSquare` Class (5 marks)

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

Assessment criteria:

- [3 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice. (Do you need to override the draw & contains method in the `MovingSquare` class? Do you need to override any existing methods?)
- [1 mark] Users should be able to create a new square using the current properties in the program.
- [1 mark] Users should be able to change the fill colour, border colour, size and bouncing path of selected squares.

## Stage 3: The `MovingQuadPie` Class (10 marks)

You are required to add a `MovingQuadPie` subclass to your program. This class should draw a special shape based on the mouse-point, the size (i.e. the smallest dimensions from the current width and the current height), the current border colour (e.g. orange), the current fill colour (e.g. blue), and the current moving path saved in the AnimationPanel. Some examples are shown as below.
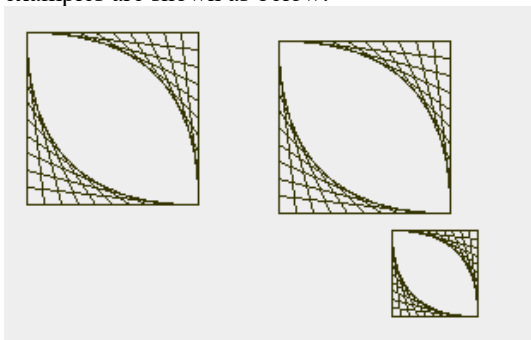
Assessment criteria:

- [3 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice. Do you need to override the `draw` and/or `contains`?
- [4 marks] Users should be able to create a special shape using the current properties in the program.
- [3 marks] Users should be able to change the fill colour, border colour, size and bouncing path of selected shapes.

## Stage 4: The `MovingSquarePattern` Class (10 marks)

You are required to add a `MovingSquarePattern` subclass to your program. This class should draw a special pattern based on the mouse-point, the size (i.e. the smallest dimensions from the current width and the current height), the current **FILL** colour (e.g. black), and the current moving path saved in the `AnimationPanel`. Some examples are shown as below:

Assessment criteria:

- [3 marks] The class hierarchy should be developed sensibly and in accordance with good object-oriented programming practice. Do you need to override the `draw` and/or `contains`?
- [4 marks] Users should be able to create a new shape pattern using the current **FILL** colour, size and path in the program.

- [3 marks] Users should be able to change the **FILL** colour, size and bouncing path of selected patterns.

## Stage 5: Adding a New Path (10 marks)

In this part, you are required to add your own designed path to the bouncing program. The `MovingPath` is an abstract inner class which contains an abstract method. You are required to add a new subclass which extends 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.
- [3 marks] Users should be able to add a new shape that moves in your own designed path.
- [2 marks] Users should be able to change the bouncing path of selected shapes to your own designed path.
- [3 marks] In the marker's opinion, your work shows great preparation, creativity or effort.

## Stage 6: Using an ArrayList (10 marks)

The maximum number of shapes that can be stored in your program is 10 at this stage. However, you are required to modify the `AnimationPanel` class which enable users to store more than 10 shapes using an `ArrayList`. Replace the array in `AnimationPanel` class with an `ArrayList`. You will need to modify 10 different methods in `AnimationPanel`: `mouseClicked`, `createNewShape`, `setCurrentPathType`, `setCurrentWidth`, `setCurrentHeight`, `setCurrentBorderColor`, `setCurrentFillColor`, `resetMarginSize`, `clearAllShapes`, and, `paintComponent`.

Assessment criteria
- [2 marks] Shapes are stored in an ArrayList.
- [3 marks] Users should be able to create a new shape and add it to the ArrayList.
- [2 marks] Users should be able to remove all shapes from the ArrayList
- [3 marks] Users should be able to modify the fill and border colours, path, height, width and margin sizes of selected shapes in the ArrayList.

## Bonus (10 marks)

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.

Assessment criteria:
- [1 mark] The class hierarchy must be developed sensibly and in accordance with good object-oriented programming practice. Do you need to override the draw and/or contains?
- [2 marks] Users must be able to create a new shape pattern using the current properties.
- [2 marks] Users must be able to change the fill colour and/or border colour, width, height and bouncing path of selected shapes.
- [5 marks] In the marker's opinion, your work shows great preparation, creativity or effort.

---

### *Submission*

Submit your assignment online via the assignment dropbox (https://adb.auckland.ac.nz/) at any time from the first submission date up until the final 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

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.**

**What to include inside the A1.txt file**

You must include a text file named A1.txt in your submission. This text file must contain the following information:

- Your name, login name (such as jlim123) and (7 or 9 digit) ID number
- How much time did the assignment take you overall?
- Which areas of the assignment did you find easy and which ones were difficult?
- Which topics in the course did the assignment most help you understand?
- Any other comments you would like to make.

## 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 acceptable:

- Lecture notes, tutorial notes, skeleton code, and advice given by us in person or online, with the exception of sample solutions from past semesters.
- The textbook.
- The official Java documentation and other online sources, as long as these describe the general use of the methods and techniques required, and do not describe solutions specifically for this assignment.
- Piazza posts by, or endorsed by an instructor.
- Fellow students pointing out the cause of errors in your code, without providing an explicit solution.

The following sources of help are not acceptable:

- Getting another student, friend, 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.