Computer Science

COMPSCI 230

Assignment THREE

Due Date

Due: **11:59 pm Monday 17th October 2022**
Worth: **4% of the final mark**

Introduction - The Bouncing Program

This is the final instalment of the Bounce project and perhaps the most challenging. The assignment involves working with design patterns and Java's Swing API. You are required to complete and extend the application in a way that makes appropriate use of design principles and patterns. A3 is a model/view application that presents two views of a list of bouncing shapes in the program. Such applications are commonplace and introduce the need for views to be mutually consistent and synchronised with a data structure whose state changes at run-time.

You are advised to start working on the assignment **using this document as a main guide** and switch to/from CR whenever required. Directly working on the CR questions without reading this document will be problematic.

Assessment Criteria

Complete CodeRunner questions 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 a single zip file containing all source files – remember to include your name, UPI and a comment at the beginning of each file. Your program must compile and run to gain any marks.

Description

Task 1: The NestedShape class

Download the template zip file from CodeRunner. Define a new class named `NestedShape` which represents nested shapes. A `NestedShape` contains zero or more inner shapes that bounce around within this shape. The children of a `NestedShape` instance can be either simple shapes, like `RectangleShape` and `OvalShape` objects, or other `NestedShape` instances. Hence, a `NestedShape` object can have an arbitrary containment depth.

The `Shape` and `NestedShape` structure is an application of the **Composite design pattern.** The composite pattern describes a group of objects that are treated the same way as a single instance of the same type of object. The intent of a composite is to "compose" objects into tree structures to represent part-whole hierarchies. Implementing the composite pattern lets clients treat individual objects and compositions uniformly, i.e. users should be able to add one or more rectangle(s)/circle(s) or one or more nested shape(s) into a list of shapes. Each nested shape contains zero or more shapes.

The `NestedShape` should use an array list of shapes to store the shapes within itself. *Each shape inside the `NestedShape` has its own coordinate system.* For example if a rectangle with a location of (10, 20) is contained inside a `NestedShape`, it will be located 10 pixels below and 20 pixels to the right of the top-left corner of that `NestedShape`.

You may want to use graphics translation in order to draw the children of a nested shape. (i.e. adjusting the coordinate system by specifying a new origin (the nested shape's top left corner) that corresponds to a point in the original coordinate system). This can be achieved using the `translate`() method. Once translated, all drawing operations are performed relative to the new origin. Note that any translation should be reversed after painting all inner shapes.

For example, a nested shape has been created at (60, 40), width=80, height=60 and an inner rectangle is at (10, 20), width=40, height=30. Then, your program should do the following:

- Set the pen's color to black
- Draw the nested shape rectangular boundary at x = 60, y = 40, width=80, height=60
- Translate the coordinate system by (60, 40)
    - Set the required pen's color
    - Fill the inner rectangle (10, 20) 40 x 30 (i.e. fill the inner rectangle at the absolute position (70, 60))
- Restore the original origin by translating the coordinate system by (-60, -40)

Note:

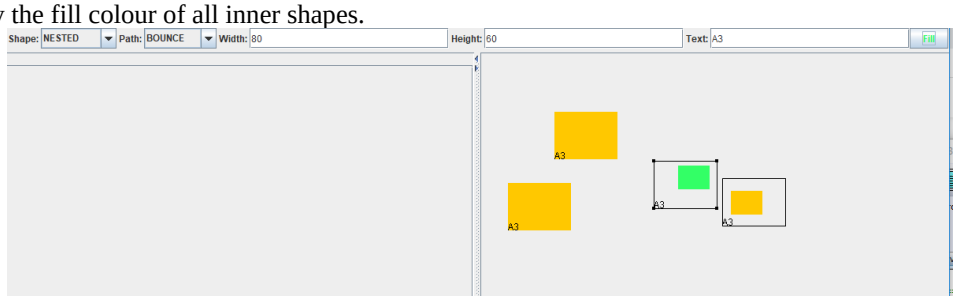
- The `root` of the program is a nested shape without any children. (Note: you should remove the `shapes` array list in the `AnimationViewer` class.) The properties of this nested shape are:
    - the top left corner is (0,0)
    - the width is `DEFAULT_PANEL_WIDTH` and the height is `DEFAULT_PANEL_HEIGHT`
    - the panel width is `DEFAULT_PANEL_WIDTH` and the panel height is `DEFAULT_PANEL_HEIGHT`
    - the fill color is `Color.black`
    - the path type is `PathType.BOUNCE`.
- When the user clicks anywhere on the bouncing area with the "`NESTED`" shape type selected in the `ShapeType` combo box. The `mouseClicked()` method should create a nested shape and add the shape to the root. The nested shape contains one inner rectangle shape by default. The properties of the nested shape are:
    - the top left corner is the mouse point
    - the width/height is the current width/height in the `AnimationViewer` class
    - the panel width/height is the width/height of it's parent (i.e. the root)
    - the fill color is the current fill color in the `AnimationViewer` class
    - the path type is the current path type in the `AnimationViewer` class
    - The properties of the inner rectangle shape are:
        - the top left corner is (0,0)
        - the width/height is the half of the width/height of it's parent
        - the panel width/height is the width/height of it's parent
        - the fill color is the fill color of it's parent
        - the path type is `PathType.BOUNCE`
- Users would be able to select nested shapes and change the fill colour of the selected nested shapes. The method should also modify the fill colour of all inner shapes.

Complete the CodeRunner questions

- Question 1: Complete the `ShapeType` class
- Question 2: Modify the `Shape` class
- Question 3 - 5: Complete the basic structure of the `NestedShape` class
- Question 6 - 7: Complete the `AnimationViewer` class

You should make changes in the template files. Your program should support the following:

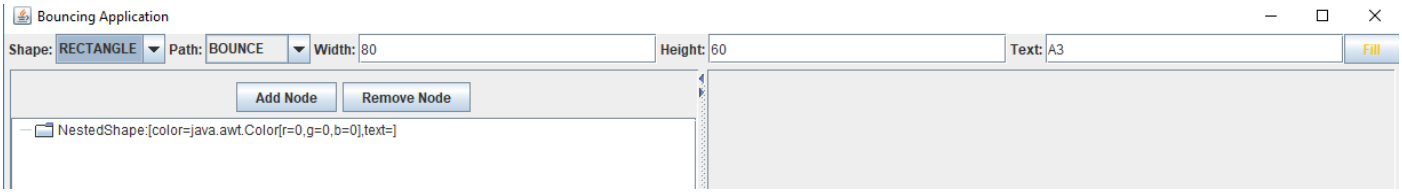
- Users would be able to create new nested shapes by clicking anywhere within the panel area of the program.
    - The properties of the newly nested shape are based on the current values saved in the appropriate UI fields.
    - The inner shape of the nested shape is a *rectangle*, the width/height is *half* of it's parent. The path is *bouncing*.
- Users would be able to select nested shapes and change the fill colour of the selected nested shapes. The method should also modify the fill colour of all inner shapes.



Task 2: The Tree Model

This task requires you to lay the foundation for a `JTree`. The `JTree` displays a hierarchical view of the composition structure of bouncing shapes. The `AnimationViewer` class implements the `TreeModel` interface. A JTree object does not actually contain your data; it simply provides a view of the data. Like any non-trivial Swing component, the tree gets data by querying its data model. `TreeModel` specifies methods for getting a particular node of the tree, getting the number of children of a particular node, determining whether a node is a leaf, notifying the model of a change in the tree, and adding and removing tree model listeners. Each node in the program contains a `Shape` object. A leaf node is just a simple shape but a composite node is a nested shape.

In order to provide the above functionalities, the structure of the `AnimationViewer` class has been changed. We **do not** use an `ArrayList` to store a list of shapes now. The `AnimationViewer` defines a root of type `NestedShape` for holding the data to be presented by a `JTree`.



Note:

- When the user selects the root and clicks the "Add Node" button, a new shape will be added to the root and a new node will be added to the root as well. The properties of the newly created shape are:
    - the top left corner is (0,0)
    - the width/height is the current width/height in the `AnimationViewer` class
    - the fill color is the current fill color in the `AnimationViewer` class
    - the shape/path type is the current shape/path type in the `AnimationViewer` class
    - the panel width/height is the width/height of it's parent (i.e. the root)
- When the user selects a nested node and clicks the "Add Node" button, a new shape will be added to the selected nested shape and a new node will be added to the selected node as well. The properties of the newly created shape are:
    - the top left corner is (0,0)
    - the width/height is the half of current width/height in the `AnimationViewer` class
    - the fill color is the current fill color in the `AnimationViewer` class
    - the shape/path type is the current shape/path type in the `AnimationViewer` class
    - the panel width/height is the width/height of it's parent

Complete the following CodeRunner questions:

- Q08: Implement the basic tree model functionality
- Q09-10: Implement methods for adding a new node
- Q11: Update the `mouseClicked` method in the `AnimationViewer` class
- Q12 Implement the `AddShapeNode` method for adding a node in the `AnimationViewer` class
- Q13 Implement an inner class for adding a node
- Q14 - 15: Implement methods for removing the selected node
- Q16: Implement an inner class for removing the selected node in the `AnimationViewer` class

You should also make changes in the template files and complete the following:

- Modify the `A3` class
    - Use the `AnimationViewer` object to create a `JTree`.
- Modify the `AnimationViewer` class such that the `AnimationViewer` implements the `TreeModel` interface

Your program should support the following:

- Create a new node (i.e. a shape) by clicking a point in the bouncing area.
- Create a new shape by clicking the "Add Node" button. The new node will be added to the selected node. The selected node must be an instance of `NestedShape`. Your program should display an error message if the node is not a nested shape using a message dialog box.

- Remove the selected node by clicking the "Remove Node" button. The selected node must NOT be the root of the `JTree`. Your program should display an error message if the root is selected using a message dialog box.

MARKING SCHEME

The total marks of this assignment is 24.

1 mark	Comments at the top of each class (containing your name, upi, and a brief description of the class)
1 mark	The code is self-documenting (good and meaningful variable names, etc).
Task1	
6 marks	CodeRunner questions
1 mark	User would be able to create a new nested shape using the given properties. The nested shape is bouncing around the panel area of the program correctly.
1 mark	The inner shape of the nested shape is a rectangle, the width/height is half of it's parent. The path is bouncing.
1 mark	User would be able to change the color of the nested shape and all inner shape objects.
Task2	
9 marks	CodeRunner questions

1 mark	User should be able to add a new shape by clicking anywhere within the bouncing area and the tree panel is updated accordingly.
1 mark	When the root is selected, users should be able to add a new shape by pressing the "Add Node" FA but an ERROR message should be shown when pressing the "Remove Node" button.
1 mark	When a nested node is selected, users should be able to add a new shape by pressing the "Add Node" button and should be able to remove the selected shape when pressing the "Remove Node" button.
1 mark	When a non-nested node is selected, should be able to remove the selected shape when pressing the "Remove Node" button but an ERROR message should be shown when pressing the "Add Node" button.

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.