Crash Course Java
Shapes and Tools

louisV, mareinK, koenD

2015

In this assignment you will once again extend the code that you built for the previous assignment. Where before, you clicked a button to place random shapes in random positions, at the end of this assignment you will be able to choose which shape to place where. Additionally, you will be able to choose the size of shapes and delete shapes.
We will implement this functionality using the concept of different 'modes' or 'tools' (rectangle tool, delete tool...).

1 Buttons and MouseListener

Change the existing buttons so that there is one button for each shape (at least Rectangle, Ellipse and Line) and one additional button for the delete tool.
For handling clicks on the screen, we will be using two new interfaces, similar to the way we handled button presses. Explore what methods the `MouseListener` and `MouseMotionListener` interfaces add to your class and keep them in mind while reading the rest of the assignment .Choose or create a class to use as mouse listener for the `RectPanel` and add it as a listener to `RectPanel`. (you might want to refactor this class' name to `DrawingPanel` or something else more suitable).

MouseListener Events

mousePressed Fired when you press a mouse button (and don't release it immediately).

mouseReleased Fired when you release a mouse button.

mouseClicked Fired when you press and release a mouse button in quick succession.

mouseEntered Fired when the cursor enters the screen of your application.

mouseExited Fired when the cursor exits the screen of your application.

MouseMotionListener Events

mouseDragged Fired in rapid succession whenever moving the mouse with a button down.

mouseMoved Fired in rapid succession whenever moving the mouse without a button down.

2 Modes

We will use the word 'mode' or 'tool' to describe a type of interaction that is currently active within the program. For example, 'placing rectangles', 'deleting shapes' or 'resizing shapes'. You should think about where and how to record the current mode and how to give different parts of the program access

to it. You might want to create an `enum` to define which modes there are in the program. Another option you might consider is creating a `Tool` abstract class and extending that into `RectangleTool`, `DeleteTool`, and so on, similar to the way `MyShape` and the shape classes are defined. How you approach this is for you to decide!

3 Placing shapes

When a shape tool is active, clicking any place on the canvas should create a new shape of the selected kind at that location. For now, the created shapes should all be of a fixed size.

4 Deleting shapes

When the delete tool is active, clicking on any shape on the canvas should remove that shape from the list of shapes. When a mouse click event is fired in your listener, how do you find out which shape was clicked? One way you could do this is by creating a method in the shapes such as `boolean contains(int x, int y)` which determines whether a certain location is within the shape. Since calculating whether a shape contains a certain point is different for different kinds of shapes, you might add this as an abstract method to `MyShape` and implement the body of the method for each of the subclasses.

5 Dragging shapes

At this point shapes are all added with a fixed size. Of course, we want to be able to choose the size of shapes that we add. We will do this by clicking and dragging from one point on the canvas to another point on the canvas, with a shape tool active. This should create a shape that fits within the area that we dragged over (for example, the start and end points of our drag should define the two opposite corners of a rectangle). Make sure that we can already see the shape while we are clicking and dragging.

6 Going further...

We would like to be able to change the color of the rectangle – its 'stroke' (the outline) and its 'fill' (the inside) – as well as the thickness of the stroke. In order to do this, you will create a slider to set the thickness of the stroke, and a collection of buttons to select a color.
For the slider, you should use a `JSlider`. Just like with buttons, you will need to set a listener to react to events from the slider. In the case of `JSlider`, the listener is called `ChangeListener`.
You will probably need to add new attributes (variables) to the `MyShape` class in which you record the colors and thickness of a shape object.
To change the stroke thickness and color, use the method `g2d.setStroke`, where `g2d` is a `Graphics` object. The arguments supplied to this method determine the visual look of all following shapes painted using `g2d.draw`.
To create a fill color for your shapes, use the method `g2d.fill` in addition to `g2d.draw`. This will not create an outline, but a filled shape.
To select a color for future drawing, use `g2d.setColor`.

7 Tips

Use your favorite search engine (no, not Bing) to find out what information you can acquire from MouseEvent objects. For all other problems, the student-assistants and Google (seriously, not Bing)

are your friends as well!
Make sure you keep a maintainable code structure whilst developing. You're allowed to change everything you've made so far, but try to keep in mind that we designed these assignments in such a way that you can build on them for the final exercise!