Graphics Silver Dollar Game

Due Sunday, September 10, 2017

Objectives

For this assignment, you will:

- Refresh your memory of the Java programming language
- Gain practice using the `ArrayList` class
- Gain exposure to inner classes
- Gain exposure to Java Graphics
- Gain exposure to Java event handling

Description

In this assignment, you will create a graphical version of Bailey's Silver Dollar Game. Read Section 3.10 of the textbook *Java Structures* for a description of the game. In this version of the game, the user will use the mouse to move the coins instead of typing commands on the keyboard.

The correctness of the assignments in this class will be automatically verified. For this reason, you must follow all naming conventions specified in this assignment.

Classes

Coin

The `Coin` class represents a single coin. This class is already implemented for you.

CoinSquare

The `CoinSquare` class represents a square. This class is already implemented for you.

GraphicsCoinStrip

The `GraphicsCoinStrip` class uses the `Coin` and `CoinSquare` class to implement the Silver Dollar Game. This class is partially implemented. There are comments suggesting what you need to add. The `contains` methods that `Coin` and `CoinSquare` inherit from `Eclipse2D` and `Rectangle2D` may be helpful.

Notice that there are no `play` or `move` methods in the `GraphicsCoinStrip` class because the mouse is in control of the game. Much of what drives the game is the mouse event handling which can be found in the inner class `CoinMouseListener` inside the `GraphicsCoinStrip` class. The purpose of the inner class `CoinMouseListener` is to encapsulate all of the methods that deal with the mouse.

You can add whatever methods you think would be useful to the `GraphicsCoinStrip` or `CoinMouseListener` classes, **but do not change the names of any of the existing methods or variables**.

After you have a working copy of the game, write a method in this class that checks to see if the game is over and, if so, signal this to the user in some fashion. Possible examples are to print out a message to the console, or better, change the color of all of the coins. You can also make sure that the coins no longer move

once the game is completed (although this is not required).

Getting Started

1. Read through the CS 062 style guide linked on the course web page under Documentation and Handouts. You must follow these guidelines for all of your assignments.

2. Create a new Java project in Eclipse named `Assignment01`. Add the `BAILEY` variable to the project. If you've forgotten how to do this, see the instructions on the Documentation page of the course website.

3. Using Finder, copy the starter files for this assignment into the `src` directory of your newly created project. The starter files can be found at `/common/cs/cs062/assignments/assignment01` inside of the `silverdollar` directory. Copy the entire `silverdollar` directory over into your `src` directory ("silver dollar" is the package name), and also grab the asg01.json file, which youll need to submit (you can just put it on your Desktop for now).

4. Refresh your project in Eclipse.

5. The `Coin` and `CoinSquare` classes are complete. You do not need to modify them. However, take a look at them to see what methods are available.

6. You are now ready to get started! This assignment asks you to fill in the constructor and add the appropriate methods in the `GraphicsCoinStrip` class to play the game. As much as possible, *try and develop incrementally. That is, get one small piece working and then move on to another piece.*

7. You will notice that the game board flickers annoyingly when a coin is dragged. This is because the screen is sometimes redrawn before the game board has been completely updated. Luckily it is easy to fix this by rendering the image completely before drawing it on the screen.

   The way to do this is to create a `BufferedImage`. A `BufferedImage` can supply a `Graphics` context that can be drawn on just like you directly write on a `JFrame`. At the top level of the `GraphicsCoinStrip` class, create a `BufferedImage` instance variable as follows:

   ```
   BufferedImage bf = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
   ```

   where the parameters `width` and `height` are replaced by the actual dimensions of the `JFrame` you are drawing on.

   Now modify the `paint` method as follows. Replace the first two lines

   ```
   public void paint(Graphics g) {
      Graphics2D g2 = (Graphics2D) g;
   ```

   by

   ```
   public void paint(Graphics g) {
      Graphics g1 = bf.getGraphics();
      Graphics2D g2 = (Graphics2D) g1;
   ```

   If the rest of your `paint` method draws with `g1`, then rather than drawing directly on the screen, you will be drawing to the (hidden) buffered image `bf`. Finally at the end of your paint method, add the line

```
g.drawImage(bf,0,0,null);
```

This line will copy the contents of the buffered image to the `JFrame` you are drawing on. Pretty simple, eh!

8. When you are done, read the Submitting Your Work section below. You should submit this assignment as asg01.

Grading

You will be graded based on the following criteria:

criterion	points
game starts with random coin positions	1
coins can be dragged	2
coins can be dragged multiple squares	1
dropped coins end up centered in correct location	1
illegal moves are not allowed	3
game over is indicated correctly	2
general correctness	2
code quality*	3
appropriate comments (including JavaDoc)	2
style and formatting	2
submitted correctly	1
Total	20

*Code quality refers to the correct use of Java constructs including booleans, loop constructs, etc. Think of it as good writing style for programs.

NOTE: Code that does not compile will not be accepted! Make sure that your code compiles before submitting it.

Submitting Your Work

1. Before you submit, you must comment your code. We will be using the JavaDoc commenting style. To be compliant with JavaDoc, you *must* have the following:

   - Each comment on a method or class should start with /** and end with **/. Every line in between should start with a * and be appropriately indented. (Comments on variables and constants do NOT have to use this style unless they are public.)
   - A comment describing the class right before the class declaration (i.e. after the `import` statements). This comment should include the `@author` tag after the class description, and the `@version` tag after the author tag.
   - A comment for each method describing what the method does. This comment should describe the *what* but not the *how*.
   - `@param`, `@return` and `@throws` tags for each method (when appropriate)
   - pre- and post- conditions as appropriate

2. Export your project from Eclipse and submit it using the instructions at `http://www.cs.pomona.edu/classes/cs062/handouts/eclipse_submission.pdf`. Please follow these very carefully. If you do not, your program will NOT be submitted correctly and you will not receive credit for your assignment (and you and we will all be very sad!).