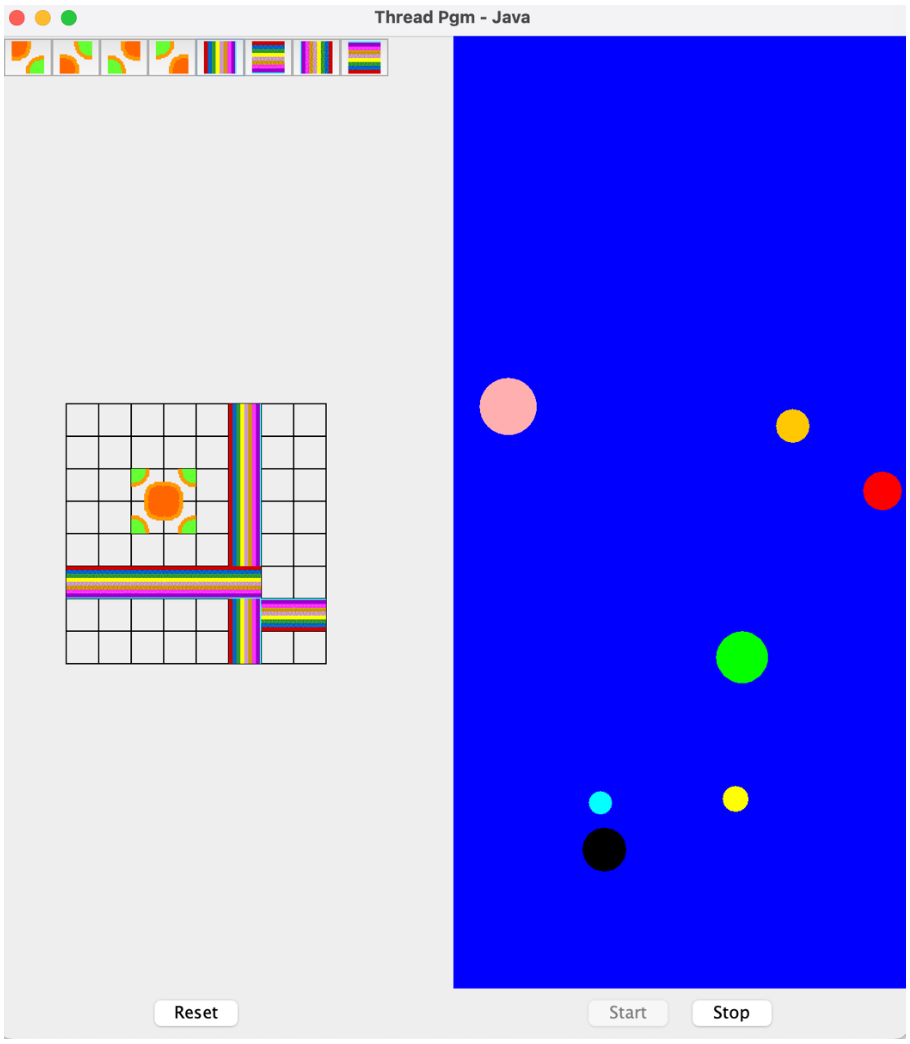
CSCI 470/502 - Assignment 5 - Spring 2023

In this assignment, you will add an animation that runs in a background thread to the GUI application that you wrote for Assignment 4.



Changes to the Assignment 4 Classes

Make the following changes to the classes you wrote for Assignment 4:

1.Add three additional images to the tool bar.
2. Expand the grid size to 8 x 8.
3. Add a `BallAnimation` object in the `EAST` region of the application.

The `BallAnimation` Class

Data Members:

- A pair of buttons to start and stop the animation.
- An `AnimationPanel` to display the animation.

Methods:

Constructor – Sets up the layout and adds listeners for the buttons.

actionPerformed() – If the source of the ActionEvent is the “Start” button”, you should enable the “Stop” button, disable the “Start” button, and call the start() method of the AnimationPanel to start the animation. If the source of the ActionEvent is the “Stop” button, you should enable the “Start” button, disable the “Stop” button, and call the stop() method of the AnimationPanel to start the animation.

The `AnimationPanel` Class

This subclass of `JPanel` will be used to display the animation in a separate background thread. Therefore, it should implement the `Runnable` interface.

Data Members:

- An `ArrayList` of `Ball` objects.
- A reference to a `Dimension` object that is initially set to `null`.
- A reference to a `Thread` object that is initially set to `null`.

Methods:
`public void start()` – If the `Thread` reference is `null`, create a new `Thread` object and call its `start()` method to make it runnable.

`public void stop()` – Set the `Thread` reference to `null`. This will cause the loop in the `run()` method to exit. Once the loop is finished, the method is finished as well, so the thread will terminate.

`protected void paintComponent(Graphics g)` – This method should be overridden to do the following:

- If the `Dimension` object reference is `null`, create a set of `Ball` objects and add them to the `ArrayList`, then get the dimensions of the panel.
- Draw the blue background.
- Call the `move()` method for each `Ball` object in the `ArrayList`, passing the dimensions of the panel to the method.
- Call the `draw()` method for each `Ball` object in the `ArrayList`, passing the `Graphics` context to the method.

`public void run()` – This is effectively the "main" method that will run in the separate background thread. It will have a loop that continues while the current thread is equal to the Thread reference data member, e.g.:

```
      while (Thread.currentThread == animationThread)
```

The loop body should put the thread to sleep for a short amount of time (100 milliseconds is about right), and then call `repaint()` (which will eventually result in `paintComponent()` being executed).

The `Ball` Class

This class represents a single bouncing ball in the animation.

Data Members:

- The `Color` of the ball.
- The `radius` of the ball (integer).
- The `x` and `y` coordinates of the ball's center point (integers).
- `int dx` – the amount of change in the ball's `x` coordinate each time the ball moves. A negative value means the ball is currently moving to the left; a positive value means the ball is currently moving to the right.
- `int dy` – the amount of change in the ball's `y` coordinate each time the ball moves. A negative value means the ball is currently moving up; a positive value means the ball is currently moving down.

Methods:

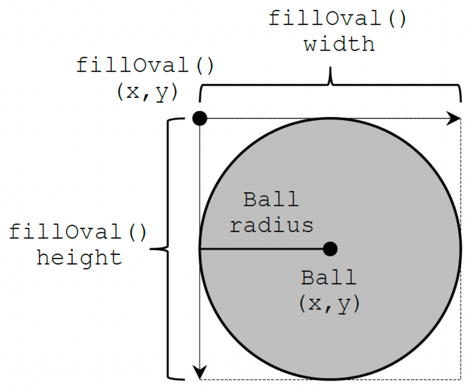
Constructor – Has six parameters corresponding to the data members described above, allowing you to initialize the ball.

`public void move(Dimension d)` – Has one parameter, a `Dimension` object that contains the width and height of the panel in which the ball is moving. Does the following:

- If the x coordinate of the ball is less than or equal its radius OR greater than or equal the width of the panel minus the radius of the ball, reverse the horizontal direction of the ball (i.e., change the sign for `dx`).
- If the y coordinate of the ball is less than or equal its radius OR greater than or equal the height of the panel minus the radius of the ball, reverse the vertical direction of the ball (i.e., change the sign for `dy`).
- Add `dx` to x.
- Add `dy` to y.

`public void draw(Graphics g)` – Has one parameter, the `Graphics` object that will be used to draw the ball on an off-screen image. Use the `Graphics` object to set the drawing color to the `Color` of the ball, and then use it to call `fillOval()` to draw the ball.

The `fillOval()` method takes four arguments: the `x` coordinate of the upper left corner of the oval to be filled, the `y` coordinate of the upper left corner of the oval to be filled, the `width` of the oval to be filled, and the `height` of the oval to be filled. The diagram below shows the relationship of these parameters to the `x` and `y` coordinates of the ball's center point and its `radius`.



Programming Notes

- Your AnimationPanel should create several `Ball` objects with different colors, starting x and y coordinates, and dx / dy values. The radius for each ball may be the same, or they may be different. For example:

```
ballList.add(new Ball(Color.GREEN, (d.width * 2 / 3), (d.height - 28),
-2, -4, 20));
```

- Due to the way the collision detection code works, starting a ball too close to the edge of the panel may cause it to "thrash", "dribbling" along the edge of the panel.
- You may want to override the getSize() method for the AnimationPanel; the panel in the screenshot above is 350 x 350.

Submission

You should zip up all the source files (.java files) and supporting images (or other text files), for submission. Name the zip file as required.