Lab 6

━━━ Scribbler ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

Objective   To gain experience using recursion and recursive data structures.

Note   You may work with a partner on this lab. If you do, please turn in only one copy of your completed program.

The Scenario   This week, you will implement a program we call "Scribbler." You have seen a simple version of this program in class as a demo. In lab, we would like you to extend the program from class to include the ability to move or color *any* of the scribbles that have been drawn rather than just the last one. We encourage you to come to lab with a design for steps 1 and 2 in the "How to Proceed" section, but it is not required. See the handouts web page to obtain a copy of the starter folder.
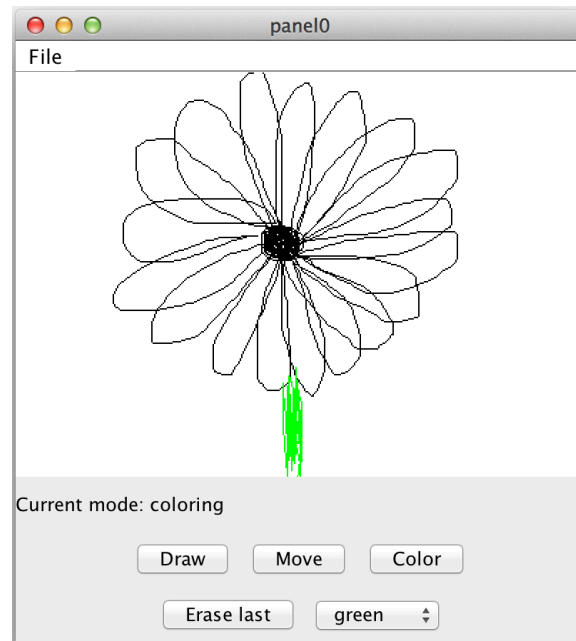
There is also a working version of the Scribbler on the handouts page. You may want to visit the web page after you read the handout so that you can experiment with the program to clarify any details that are not clear to you.

The Scribbler program has three modes: Draw mode, Move mode, and Color mode. The program starts in Draw mode. Draw mode, Move mode, and Color mode are selected by pressing buttons, and the color used by Color mode is selected by choosing a color from a `JComboBox` filled with color names. The modes behave as follows:

Draw mode: Drag to draw a new scribble on the canvas.

Move mode: Reposition a scribble by pressing the mouse on it and dragging.

Color mode: Click on any scribble to change its color to the color selected in the color menu.

The program also has an "Erase Last" button that will erase the most recently drawn scribble. Pressing the button repeatedly will erase scribbles in the reverse order of which they were drawn.

How to Proceed

The starter for this lab is a working Scribbler, but it has only the Draw mode and a simplified Move mode that is capable of moving only the most recently drawn scribble. You will need to add code that will manage your scribbles to allow the various modes and the "Erase last" button to work correctly.

There are a number of step-by-step approaches you could take to solve this problem, but it is important that you have a plan, and that you add functionality one step at a time. Here is one possible ordering of the tasks. We recommend that you develop and test your program incrementally—make sure you have a working implementation at each step before moving on to the next.

1. Implement a simplified Color mode. This is similar to Move mode, except that you set the color of the scribble that contains the mouse point instead of moving it. To do this, you will have to add a `setColor` method to the `ScribbleInterface` interface, and the `NonEmptyScribble` and `EmptyScribble` classes. Initially, you will only be able to set the color of the most recently drawn scribble.

2. Implement a simplified Erase mode. Here, you respond to the "Erase last" button's `actionPerformed` event by deleting the most recently drawn scribble from the canvas. For now, you will only be able to erase the most recently drawn scribble. A second button press will do nothing.

3. Since you need to keep track of a number of scribbles, you will need to define classes representing a collection of scribbles. The interface for these classes will be called `ScribbleCollectionInterface`, and the classes implementing that interface will be named `EmptyScribbleCollection` and `NonEmptyScribbleCollection`. The interface and classes will be similar to the stacks used in WoodStack and ball lists in the ChainReaction.

   Just as we do not know how many line segments will make up a scribble when we start to draw one, we will not know how many scribbles will be drawn and stored in a scribble collection. In your `Scribbler` class you will need a variable with type `ScribbleCollectionInterface` that will be initialized with an object created from `EmptyScribbleCollection`. You will add new scribbles to it using the constructor from `NonEmptyScribbleCollection` as they are drawn. Consider carefully what methods your scribble collection needs to support the functionality of the four modes. We have provided the skeleton of a `ScribbleCollectionInterface` interface, and `NonEmptyScribbleCollection` and `EmptyScribbleCollection` classes.

   - Draw mode needs to add a new scribble to the scribble collection after the dragging is done.
   - The "Erase last" button needs to remove the most recently drawn scribble from the canvas and remove it from the scribble collection. This means there is now a new "most recently drawn" scribble, and a second press would remove that one, and so on. Be careful in the case when there are no scribbles left.
   - Move mode and Color mode need to determine if a mouse click takes place on *any* of the scribbles on the canvas, not just the most recently drawn. This requires that your scribble collection classes be able to search through all of the scribbles until either a scribble is located that contains the click point, or it is determined that none of the existing scribbles contain the point. Once you have determined which scribble contains the click point, you can move or color that scribble as you did in the simplified versions of these modes.

Submitting Your Work

Once you have saved your work in BlueJ, please perform the following steps to submit your assignment:

- First, return to the Finder. You can do this by clicking on the smiling Macintosh icon in your dock.

- From the "Go" menu at the top of the screen, select "Connect to Server...".

- For the server address, type in "afp://Guest@fuji" and click "Connect".

- A selection box should appear. Select "Courses" and click "Ok".

- You should now see a Finder window with a "cs134" folder. Open this folder.

- You should now see the drop-off folders for the three lab sections. Drag your "Lab6Recursion" folder into the appropriate lab section folder. When you do this, the Mac will warn you that you will not be able to look at this folder. That is fine. Just click "OK".

- Log off of the computer before you leave.

You can submit your work up to 11 p.m. on Wednesday if you're in the Monday night lab; up to 6 p.m. on Thursday if you're in the Tuesday morning lab; and up to 11 p.m. on Thursday if you're in the Tuesday afternoon lab. If you submit and later discover that your submission was flawed, you can submit again. We will grade the latest submission made before your lab deadline. The Mac will not let you submit again unless you change the name of your folder slightly. It does this to prevent another student from accidentally overwriting one of your submissions. Just add something to the folder name (like the word "revised") and the re-submission will work fine.

Grading Guidelines

As always, we will evaluate your program for both style and correctness. Here are some specific items to keep in mind and focus on while writing your program:

Style
Descriptive comments
Good names
Good use of constants
Appropriate formatting

Design
Good use of boolean expressions and conditionals
Not doing more work than necessary
Appropriate methods in `NonEmptyScribbleCollection`
Appropriate recursive structure in `NonEmptyScribbleCollection`

Correctness
Switching among modes correctly
Draw mode adds correctly to the scribble collection
Move mode works correctly
Color mode works correctly
Erase button works correctly

We are providing several classes and interfaces to help you get started. We include here printouts of the main program and the classes and the interface for a single scribble. These are very similar to the Scribbler examples from lecture. Here is the code from the basic `Scribbler` class. You may need to add additional methods to it:

```
/*
 * A very simple drawing program.
 */
public class Scribbler extends WindowController implements ActionListener {

  // User modes for what operation is selected.
  // We are using ints rather than boolean to allow for extension to
  // other modes
  private static final int DRAWING = 1;
  private static final int MOVING = 2;
  private static final int COLORING = 3;

  // the current scribble
  private ScribbleInterface currentScribble;

  // the collection of scribbles
  private ScribbleCollectionInterface scribbles;

  // stores last point for drawing or dragging
  private Location lastPoint;

  // whether the most recent scribble has been selected for moving
  private boolean draggingScribble;

  // buttons that allow user to select modes
  private JButton setDraw, setMove, setErase, setColor;

  // Choice menu to select color
  private JComboBox chooseColor;

  // new color for scribble
  private Color newColor;

  // label indicating current mode
  private JLabel modeLabel;

  // the current mode -- drawing mode by default
  private int chosenAction = DRAWING;

  // create and hook up the user interface components
  public void begin() {

    setDraw = new JButton("Draw");
    setMove = new JButton("Move");
    setColor = new JButton("Color");

    JPanel buttonPanel = new JPanel();
```

4

```java
  buttonPanel.add(setDraw);
  buttonPanel.add(setMove);
  buttonPanel.add(setColor);

  chooseColor = new JComboBox();
  chooseColor.addItem("red");
  chooseColor.addItem("blue");
  chooseColor.addItem("green");
  chooseColor.addItem("yellow");

  setErase = new JButton("Erase last");
  JPanel choicePanel = new JPanel();
  choicePanel.add(setErase);
  choicePanel.add(chooseColor);

  JPanel controlPanel = new JPanel(new GridLayout(3,1));
  modeLabel = new JLabel("Current mode: drawing");
  controlPanel.add(modeLabel);
  controlPanel.add(buttonPanel);
  controlPanel.add(choicePanel);

  Container contentPane = this.getContentPane();
  contentPane.add(controlPanel, BorderLayout.SOUTH);

  // add listeners
  setDraw.addActionListener(this);
  setMove.addActionListener(this);
  setErase.addActionListener(this);
  setColor.addActionListener(this);

  // make the current scribble empty
  currentScribble = new EmptyScribble();

  this.validate();
}

/*
 * If in drawing mode then start with empty scribble.
 * If in moving mode then prepare to move.
 */
public void onMousePress(Location point) {
  if (chosenAction == DRAWING) {
    // start with an empty scribble for drawing
    currentScribble = new EmptyScribble();

  } else if (chosenAction == MOVING) {
    // check if user clicked on current scribble
    draggingScribble = currentScribble.contains(point);
  }

  // remember point of press for drawing or moving
  lastPoint = point;
}
```

```java
  /*
   * If in drawing mode, add a new segment to scribble.
   * If in moving mode then move it.
   */
  public void onMouseDrag(Location point) {
    if (chosenAction == DRAWING) {
      // add new line segment to current scribble
      Line newSegment = new Line(lastPoint, point, canvas);

      currentScribble =
      new NonEmptyScribble(newSegment, currentScribble);
    } else if (chosenAction == MOVING) {
      if (draggingScribble) {
        // move current scribble
        currentScribble.move(point.getX() - lastPoint.getX(),
          point.getY() - lastPoint.getY());
      }
    }
    // update for next move or draw
    lastPoint = point;
  }

  public void onMouseRelease(Location point) {
    // Add code when have collection of scribbles
  }

  /*
   * Set mode according to JButton pressed.
   */
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == setDraw) {
      chosenAction = DRAWING;
      modeLabel.setText("Current mode: drawing");
    } else if (e.getSource() == setMove) {
      chosenAction = MOVING;
      modeLabel.setText("Current mode: moving");
    }
  }
}
```

Here is the starting code for ScribbleInterface:

```java
public interface ScribbleInterface {

  // returns whether point is contained in scribble
  public boolean contains(Location point);

  // move scribble by dx in x-direction and dy in y-direction
  public void move(double xOffset, double yOffset);
}
```

Here is the starting code for EmptyScribble:

```java
/*
 * Class representing an empty scribble
 */
public class EmptyScribble implements ScribbleInterface {

  public EmptyScribble() {
  }

  // point is never in an empty scribble!
  public boolean contains(Location point) {
    return false;
  }

  // nothing to move, so do nothing!
  public void move(double xOffset, double yOffset) {
  }
}
```

Here is the starting code for `NonEmptyScribble`:

```java
public class NonEmptyScribble implements ScribbleInterface {

  // an edge line of the scribble
  private Line first;

  // the rest of the scribble
  private ScribbleInterface rest;

  public NonEmptyScribble(Line segment, ScribbleInterface theRest) {
    first = segment;
    rest = theRest;
  }

  /*
   * Returns true iff the scribble contains the point.
   */
  public boolean contains(Location point) {
    return (first.contains(point) || rest.contains(point));
  }

  /*
   * Move the scribble by xOffset in the x direction
   *    and yOffset in the y direction
   */
  public void move(double xOffset, double yOffset) {
    first.move(xOffset, yOffset);
    rest.move(xOffset, yOffset);
  }
}
```

The starter code for the collection interface and classes is omitted here.