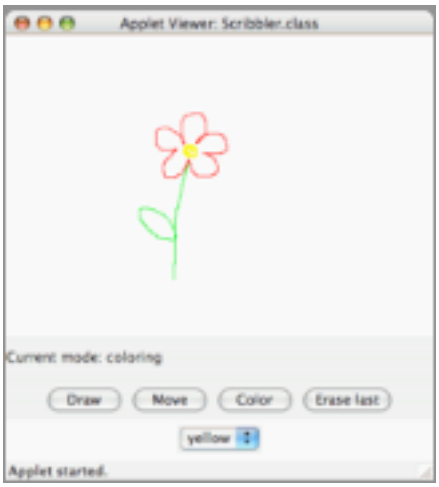
CS 51 Laboratory # 8
Scribbler

Objective: To gain more experience using recursion and recursive data structures.

This week, you will be implementing a program we call "Scribbler." This is a more sophisticated version of the scribble class you've seen in lecture as a demo. While no design is required for this lab, we strongly recommend that you show up with a plan for at least steps 1 and 2 in the "How to Proceed" section.

You are encouraged to work in pairs on this program. You must do *all* of the design and coding together. Only turn in one copy of the program, but make sure both of your names are on the folder and at the top of each class.



The online version has a working version of the program.

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 choice component filled with color names. The modes behave as follows:

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

Move mode: Drag a `Scribble` by pressing the mouse on it and dragging.

Color mode: Click on any `Scribble` to change its color to that selected in the color choice component.

The program also has an "Erase Last" button that will erase the *most recently drawn* `Scribble`. Unlike our simple version in class, if you press the button repeatedly it will continue to erase `Scribble`s in the reverse order of which they ar 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 `Scribble`s 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 `ScribbleIfc` interface, and the `Scribble` and `EmptyScribble` classes. For now, 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. Again, 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 `Scribble`s, you will need to define classes representing a collection of `Scribble`s. The interface for these classes will be called `ScribbleCollectionIfc`, and the classes implementing that interface will be named `EmptyScribbleCollection` and `ScribbleCollection`. This interface and classes will be similar to the ball lists in the `ChainReaction` example we looked at in class.

   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 `Scribble`s will be drawn and stored in our `ScribbleCollection`. Because of this you may **not** use an array to keep track of these. Instead, we're going to do it with recursion.

   Look at the `ScribbleCollectionIfc` interface. Like the `ChainReaction` method, we're going to keep track of our collection of `Scribble`s by keeping a reference to the current `Scribble` and then the rest of the collection.

   In your `Scribbler` class you will need a variable with type `ScribbleCollectionIfc` that will be initialized with an object created from `EmptyScribbleCollection`. You will add new `Scribble`s to it using the constructor from `ScribbleCollection` as they are drawn. Consider carefully what methods your scribble collection needs to be able to support the functionality of the four modes. We have provided the skeleton of a `ScribbleCollectionIfc` interface, and `ScribbleCollection` and `EmptyScribbleCollection` classes.

   - Draw mode needs to add a new `Scribble` to the `ScribbleCollection` when the dragging is done.
   - The "Erase last" button needs to remove the most recently drawn `Scribble` from the canvas and remove it from the `ScribbleCollection`. 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 `Scribble`s left.
   - Move mode and Color mode need to determine if a mouse click takes place on *any* of the `Scribble`s on the canvas, not just the most recently drawn. This requires that your `ScribbleCollection` be able to search through all of its `Scribble`s until either a `Scribble` is located that contains the click point, or it is determined that none of the existing `Scribble`s contain the point (the base case, which should be handled by the `EmptyScribbleColletion` class). 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.

Sketch of classes provided for this program We are providing some classes and interfaces to help you get started. These are almost identical to the Scribbler examples from lecture. Please download the starter to see the classes and interfaces we provide before working on your design. We also include the startup code for the `Scribbler` class and the `ScribbleIfc` interface.

Submitting Your Work The lab is due at 11 PM on Monday. When your work is complete you should drop the folder in the dropbox. Make sure the folder name begins with your name(s) and that it includes the phrase "Lab 11". Also make sure that your name(s) is (are) included in the comment at the top of each Java source file.

Before turning in your work, be sure to double check both its logical organization and your style of presentation. Make your code as clear as possible and include appropriate comments describing major sections of code and declarations.

Table 1: Grading Guidelines

Value	Feature
	Style (5 points total)
2 points	Descriptive comments
1 point	Good names
1 point	Good use of constants
1 point	Appropriate formatting
	Design (6 points total)
1 point	Good use of boolean expressions, loops, conditionals
1 point	Not doing more work than necessary
2 points	Appropriate methods in `ScribbleCollection`
2 points	Appropriate recursive structure in `ScribbleCollection`
	Correctness (9 points total)
1.5 point	Drawing the GUI components correctly
1.5 point	Switching among modes correctly
1.5 point	Draw mode adds correctly to the `ScribbleCollection`
1.5 point	Move mode works correctly
1.5 point	Color mode works correctly
1.5 point	Erase button works correctly

Sketch of classes provided for this program We are providing several classes and interfaces to help you get started. We include here printouts of the main program and the classes and interface for a single scribble. These are very similar to the Scribbler examples from lecture. We have also include the starting `ScribbleCollectionIfc` interface which you will need to modify.

```
public interface ScribbleCollectionIfc
{
   // returns the scribble that contains the point;
   //    if none contain it, returns an empty scribble
   ScribbleIfc scribbleSelected(Location point);
```

```
   // returns the first scribble in the list;
   //   returns null if the list is empty.
   ScribbleIfc getFirst();

   // returns the list of scribbles excluding the first.
   //   returns null if the list is empty.
   ScribbleCollectionIfc getRest();
}
```

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
   private static final int DRAWING = 1;
   private static final int MOVING = 2;
   private static final int COLORING = 3;

   // empty scribble as placeholder when nothing there.
   private static final EmptyScribble empty = new EmptyScribble();

   // the current scribble
   private ScribbleIfc currentScribble;

   // the collection of scribbles
   private ScribbleCollectionIfc 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 Button setDraw, setMove, setErase, setColor;

   // Choice button to select color
   private Choice chooseColor;

   // new color for scribble
   private Color newColor;

   // label indicating current mode
   private Label 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 Button("Draw");
   setMove = new Button("Move");
   setColor = new Button("Color");

   Panel buttonPanel = new Panel();
   buttonPanel.add(setDraw);
   buttonPanel.add(setMove);
   buttonPanel.add(setColor);

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

   setErase = new Button("Erase last");
   Panel choicePanel = new Panel();
   choicePanel.add(setErase);
   choicePanel.add(chooseColor);

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

   add(controlPanel, BorderLayout.SOUTH);

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

   // make the current scribble empty
   currentScribble = empty;

   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 = empty;

      } 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;
   }

   // 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 Scribble(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 button 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 `ScribbleIfc`:

```
public interface ScribbleIfc
{
   // returns whether point is contained in scribble
   boolean contains(Location point);

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

Here is the starting code for `Scribble`:

```
// A class to represent a non-empty scribble
public class Scribble implements ScribbleIfc {
   private Line first; // an edge line of the scribble
   private ScribbleIfc rest; // the rest of the scribble

   public Scribble(Line segment, ScribbleIfc 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));
   }

   // the scribble is moved 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);
   }
}
```

Here is the starting code for `EmptyScribble`:

```
/* Class representing an empty scribble */
public class EmptyScribble implements ScribbleIfc{
   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 dx, double dy) {
   }
}
```

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