JETT, Hendrix College, August 6–7, 2004

Staff

Carl Burch	Conway, Arkansas
Cheri Burch	Albuquerque, New Mexico
Dwayne Collins	Conway, Arkansas
Zach Collins	Conway, Arkansas
Gabe Ferrer	Conway, Arkansas

Participants

Gini Cocanower	Conway, Arkansas
Paul Foster	Benton, Arkansas
Neal Gibson	Yellville, Arkansas
Cindy Hanchey	Shawnee, Oklahoma
Dale Hanchey	Shawnee, Oklahoma
Anne Melancon	Baton Rouge, Louisiana
Kathleen Weaver	Dallas, Texas

sponsored by

in affiliation with

Friday schedule

This schedule is likely to compress and stretch as we see what's going on.

11:00	**Welcome**
11:15	**lunch**
12:15	**Introduction to OOP** (Carl) object concept, writing classes, strings *Exercise: Mouse follower*
1:50	break
2:00	**Class design** (Carl) subclasses, protection levels
2:50	break
3:00	**Arrays** (Gabe) arrays (1D and 2D) *Exercise: Image processing*
4:20	break
4:30	**Java collections** (Gabe) java.util classes, interfaces, wrapper classes
5:30	**dinner** (at restaurant)
7:00	*Laboratory: DNA testing*
9:00	retire

(It may well be that we go well ahead of schedule. If this occurs, we'll move the 9:30 session Saturday to before dinner, and we'll insert in its place on Saturday a session about files and exceptions.)

Saturday schedule

8:00	**breakfast** (you can go to house or classroom)
8:30	**Tying up loose ends** (Gabe) abstract classes/methods, class methods/variables, Console I/O, exceptions
9:20	break
9:30	**AP data structures** (Gabe) Stack, Queue, PriorityQueue, ListNode, TreeNode big-O analysis *Exercise: Building PriorityQueue*
10:20	break
10:30	**Swing basics** (Carl) *Exercise: Minesweeper field*
11:35	break
11:45	**Diversity in computer science** (Cheri)
12:30	**lunch** (and evaluation)
1:30	**Drawing program overview** (Carl)
2:00	*Laboratory: Drawing program*
4:00	workshop done

Introduction to OOP

Java development environments

J2SE (formerly called the SDK) is the basic command-line system. It is produced by Sun, and distributed at no charge. Most other development environments are built on top of J2SE, so you'll probably need it even if you don't use the command line in your class. You can download it from

`java.sun.com`

There are several freely available Java development environments designed specifically for introductory programming classes. All are maintained by colleges and are available at no charge. They are very different from each other, and all have strong supporters.

BlueJ	`www.bluej.org`	University of Kent, et. al.
JGrasp	`www.jgrasp.org`	Auburn University
Dr. Java	`drjava.org`	Rice University

If you go beyond AP programming, you may want a more professional IDE. Here are some of the alternatives.

Eclipse	`www.eclipse.org`	open source
JCreator	`www.jcreator.com`	XINOX Software
JEdit	`www.jedit.org`	open source
NetBeans	`www.netbeans.org`	open source

Class documentation

These classes constitute a library that I wrote for introducing programming to students. You will be able to download it, and all other code used in this class, from

`www.cburch.com/proj/jett`

Class Canvas

`Canvas()`
: (Constructor method) Creates a Canvas, 200 pixels wide and 200 pixels tall.

Class Chassis

`Chassis()`
: (Constructor method) Creates a Chassis facing east. This chassis will not appear until it is placed on a canvas via the `place` method.

`void move(double dist)`
: Moves this chassis forward `dist` pixels, in this robot's current direction.

`void place(Canvas c, double x, double y)`
: Places this chassis at the coordinates $(\texttt{x}, \texttt{y})$ of the canvas. Example coordinates include $(0, 0)$ for the top left corner, $(200, 0)$ for the top right corner, and $(100, 100)$ for the canvas's center.

`void turnRight(double degrees)`
: Turns this chassis right (clockwise) by the given angle.

`void turnLeft(double degrees)`
: Turns this chassis left (counterclockwise) by the given angle.

`void remove()`
: Removes this chassis from the current canvas.

Class MouseTracker

`MouseTracker()`
: (Constructor method) Creates a MouseTracker. This must be attached to a Chassis (via the `attach` method) before it can be useful.

`void attach(Chassis chassis)`
: Attaches this tracker to the given chassis.

`void detach()`
: Detaches this tracker from the current chassis.

`double getAngle()`
: Returns the angle (in degrees, clockwise) between the chassis's current direction and the direction from the

chassis to the current mouse location in the chassis's canvas.

`double getDistance()`
: Returns the distance between the chassis and the current mouse location in the chassis's canvas.

Class Pen

`Pen()`
: (Constructor method) Creates a Pen. This must be attached to a Chassis (via the `attach` method) to be effective.

`void attach(Chassis chassis)`
: Attaches this pen to the given chassis. If the chassis already has a pen attached, nothing happens.

`void detach()`
: Detaches this pen from the current chassis.

`void drop()`
: Drops the pen so that it will draw. This is the default when a pen is created.

`void lift()`
: Lifts the pen so that it does not draw when moved.

`void setColor(int red, int green, int blue)`
: Sets the pen's color to the given color components. Each color component should be an integer between 0 and 255.

Class Voice

`Voice()`
: (Constructor method) Creates a Voice.

`void attach(Chassis chassis)`
: Attaches this voice to the given chassis. Being attached to a chassis has two minor effects: The chassis's picture will appear in any dialog boxes displayed, and any dialog boxes will appear centered on the chassis's canvas (if it has been added to a canvas).

`void detach()`
: Detaches this voice from the current chassis.

`void notify(String message)`
: Displays the given message to the user.

`double requestDouble()`
: Returns a double value typed by the user.

`double requestDouble(String prompt)`
: Asks the user for a double value using the given prompt and returns the value typed.

`int requestInt()`
: Returns an integer value typed by the user.

`int requestInt(String prompt)`
: Asks the user for an integer value using the given prompt and returns the value typed.

`String requestString()`
: Returns a string typed by the user.

`String requestString(String prompt)`
: Asks the user for a string using the given prompt and returns the value typed.

Code examples

Class TriangleRobot

```
public class TriangleRobot {
    /**
     * (Constructor) Creates a TriangleRobot.
     */
    public TriangleRobot() { }

    /**
     * Places a robot's chassis into canvas, tells
     * the robot to move in a triangle, and then
     * removes the chassis.
     */
    public void runTriangle(Canvas canv) {
        Chassis wheels;
        wheels = new Chassis();
        wheels.place(canv, 100, 50);
        wheels.turnRight(60);
        wheels.move(100);
        wheels.turnRight(120);
        wheels.move(100);
```

```
        wheels.turnRight(120);
        wheels.move(100);
        wheels.remove();
    }
}
```

Class PolygonRobot

```
public class PolygonRobot {
    private Chassis wheels;
    private Pen pen;
    private Voice voice;

    /**
     * (Constructor) Creates a PolygonRobot.
     */
    public PolygonRobot() {
        wheels = new Chassis();
        pen = new Pen();
        voice = new Voice();
        pen.attach(wheels);
        voice.attach(wheels);
    }

    /**
     * Asks the user for an integer and then uses
     * this robot to trace a polygon with that number
     * of sides. After finishing, the robot removes
     * itself from the canvas.
     */
    public void tracePolygon(Canvas canv) {
        double radius;
        radius = 50.0;
        wheels.place(canv, 100, 100 + radius);
        int sides;
        sides = voice.requestInt();
        double side_length;
        side_length = 2 * 3.1415 * radius / sides;
        double turn_degrees;
        turn_degrees = 360.0 / sides;

        wheels.turnLeft(turn_degrees / 2);
        int sides_drawn;
        sides_drawn = 0;
        while(sides_drawn < sides) {
```



```
            wheels.move(side_length);
            sides_drawn = sides_drawn + 1;
            wheels.turnLeft(turn_degrees);
        }

        wheels.remove();
    }
}
```

Assignment

Using the MouseTracker and Chassis classes, write a MouseFollower robot with a `run` method in which the chassis repeatedly moves toward the mouse cursor. This method will not complete.

Arrays

Image Processing

Image processing can be used in many situations, including applications such as automatically identifying defects on a factory's conveyor belt, targeting missiles, or enhancing TV football broadcasts with a yellow line on the field illustrating where the first down will be.

We have provided a program that loads an image for viewing from a file. Your job is to write code to transform this image into another image. This code will use two classes, whose important methods are described below.

Class ImageBuffer

`Pixel[][] getPixels()`
: Returns a two-dimensional array of pixels, in row-major order; that is, element `[i][j]` is the jth pixel in the ith row of the image. Note that this does not follow the conventional Cartesian ordering: The y-coordinate is first.

Class Pixel

`Pixel(int red, int green, int blue)`
: (Constructor) Creates a Pixel with the given red, green, and blue components. Each component should be an integer between 0 and 255.

`int getRed()`
: Returns this pixel's red component, an integer between 0 and 255.

`int getGreen()`
: Returns this pixel's green component, an integer between 0 and 255.

`int getBlue()`
: Returns this pixel's blue component, an integer between 0 and 255.

Example

Your code will go into the ImageTransform class. We have already provided one transformation, which converts a color image into gray-scale. It does this by taking each pixel, finding the average of the color components, and creating a pixel where all color components equal this average.

```
 1 /** Returns a two-dimensional array of pixels
 2  * representing the image resulting from a
 3  * transformation of the image in the ImageBuffer.
 4  */
 5 public Pixel[][] transform1(ImageBuffer img) {
 6     Pixel[][] pix = img.getPixels();
 7     int height = pix.length;
 8     int width = pix[0].length;
 9     Pixel[][] buf = new Pixel[height][width];

10     for(int y = 0; y < height; y++) {
11         for(int x = 0; x < width; x++) {
12             Pixel p = pix[y][x];
13             int gray = (p.getRed() + p.getGreen()
14                       + p.getBlue()) / 3;
15             buf[y][x] = new Pixel(gray, gray, gray);
16         }
17     }

18     return buf;
19 }
```

Exercise 1

Write the `transform2` method to convert a grayscale image to a black-and-white image. You can do this by applying a "threshold," where you color white any pixel whose brightness is above 127, and any other pixel you color black. You need only look at the red component of the pixel to determine its brightness; since it will be a grayscale image, the other color components will match it.

Exercise 2

Write the `transform3` method to detect lines in a grayscale image. To do this, for each pixel you should do the following:

1. Compute x as the result of applying a mask to the pixel, illustrated as follows:

−1	−2	−1
0	0	0
1	2	1

   This mask represents that you should compute -1 times the brightness of the pixel just above and to the left of the current pixel, plus -2 times the brightness of the pixel just above the current pixel, plus -1 times the brightness of the pixel just above and to the right of the current pixel, plus.... Again, since this is grayscale, each color component of the pixels will be the same, and so you can get the brightness of a pixel by looking at its red component alone.

2. Compute y as the result of applying the following mask to the pixel:

−1	0	1
−2	0	2
−1	0	1

3. Place into the resulting image a gray pixel whose brightness is $|x| + |y|$.

Java collections

Class CollectionExample

```
 1 import java.util.*;

 2 public class CollectionExample {
 3     public static void run() {
 4         // Casting up and down the Object hierarchy
 5         String s = "fred";
 6         Object obj = s;
 7         String t = (String) obj;
 8         System.out.println(t);

 9         // ArrayList
10         System.out.println("ArrayList examples");
11         ArrayList a = new ArrayList();
12         a.add("adam");
13         a.add("brad");
14         a.add("charles");
15         System.out.println("size: " + a.size());
16         a.set(0, "david");

17         // Printing the contents...
18         for(int i = 0; i < a.size(); ++i) {
19             String u = (String) a.get(i);
20             System.out.println(u);
21         }

22         // An equivalent loop with iterators
23         for(Iterator i = a.iterator(); i.hasNext();) {
24             String v = (String) i.next();
25             System.out.println(v);
26         }
27         System.out.println();

28         // Wrapper classes
29         System.out.println("Wrapper class examples");
30         ArrayList b = new ArrayList();
31         for(int i = 0; i < 10; ++i) {
32             b.add(new Integer(i));
33         }
34         // Double each element
35         for(int i = 0; i < b.size(); ++i) {
36             Integer el = (Integer) b.get(i);
37             b.set(i, new Integer(el.intValue() * 2));
38         }

39         // Print them out
40         for(Iterator i = b.iterator(); i.hasNext();) {
41             Integer val = (Integer) i.next();
42             System.out.print(val + " ");
43         }
44         System.out.println();

45         // An alternative:
46         for(Iterator i = b.iterator(); i.hasNext();) {
47             System.out.print(i.next() + " ");
48         }
49         System.out.println();
50         System.out.println();

51         // Other relevant wrapper classes: Double,
52         // Boolean. Use "doubleValue" and
53         // "booleanValue" respectively to access
54         // the primitive type.

55         // LinkedList
56         System.out.println("LinkedList examples");
57         LinkedList c = new LinkedList();
58         c.addFirst("fred");
59         c.addFirst("elmer");
60         c.addLast("george");
61         c.addLast("harold");
62         ListIterator inserter = c.listIterator();
63         inserter.next();
64         inserter.add("falstaff");
65         inserter.next();
66         inserter.set("frederick");
67         for(Iterator i = c.iterator(); i.hasNext();) {
68             System.out.print(i.next() + " ");
69         }
70         System.out.println();
71         c.removeFirst();
72         c.removeLast();
73         System.out.print("Without first and last: ");
74         for(Iterator i = c.iterator(); i.hasNext();) {
```

```
 75            System.out.print(i.next() + " ");
 76        }
 77        System.out.println();
 78        System.out.println();

 79        // Sets
 80        System.out.println("Set examples");
 81        Set d = new HashSet();
 82        d.add(new Cartesian(3.0, -1.2));
 83        d.add(new Cartesian(2.72, -1.2));
 84        d.add(new Cartesian(3.14, 2.0));
 85        d.add(new Cartesian(3.14, -1.2));
 86        d.add(new Cartesian(3.14, 2.0));
 87        System.out.print("HashSet: ");
 88        for(Iterator i = d.iterator(); i.hasNext();) {
 89            System.out.print(i.next() + " ");
 90        }
 91        System.out.println();

 92        Set e = new TreeSet();
 93        e.add(new Cartesian(3.0, -1.2));
 94        e.add(new Cartesian(2.72, -1.2));
 95        e.add(new Cartesian(3.14, 2.0));
 96        e.add(new Cartesian(3.14, -1.2));
 97        e.add(new Cartesian(3.14, 2.0));
 98        System.out.print("TreeSet: ");
 99        for(Iterator i = e.iterator(); i.hasNext();) {
100            System.out.print(i.next() + " ");
101        }
102        System.out.println();
103        System.out.println();

104        // Maps
105        System.out.println("Map examples");
106        Map game = new HashMap();
107                  // TreeMap again works the same way
108        game.put("destroyer", new Cartesian(2, 2));
109        game.put("submarine", new Cartesian(4, 3));
110        game.put("carrier", new Cartesian(7, 2));
111        game.put("cruiser", new Cartesian(8, 3));
112        game.put("battleship", new Cartesian(1, 8));

113        System.out.println("Where's the cruiser?  "
114                   + "It's at " + game.get("cruiser"));
```



```
115        System.out.println("All ships:");
116        Set gameKeys = game.keySet();
117        for(Iterator i = gameKeys.iterator();
118                   i.hasNext();) {
119            Object ship = i.next();
120            System.out.println(ship + " is at "
121                   + game.get(ship));
122        }

123        game.remove("battleship");
124        Object battleship = game.get("battleship");
125        if(battleship == null) {
126            System.out.println(
127                   "You sank my battleship!");
128        }
129    }
130 }
```

Output when executed:

```
fred
ArrayList examples
size: 3
david
brad
charles
david
brad
charles

Wrapper class examples
8 10 12 14 16 18
8 10 12 14 16 18

LinkedList examples
elmer falstaff frederick george harold
Without first and last: falstaff frederick george

Set examples
HashSet: (2.72, -1.2) (3.14, -1.2) (3.14, 2.0) (3.0, -1.2)
TreeSet: (2.72, -1.2) (3.0, -1.2) (3.14, -1.2) (3.14, 2.0)

Map examples
Where's the cruiser?  It's at (8.0, 3.0)
All ships:
```

```
destroyer is at (2.0, 2.0)
submarine is at (4.0, 3.0)
battleship is at (1.0, 8.0)
carrier is at (7.0, 2.0)
cruiser is at (8.0, 3.0)
You sank my battleship!
```

Class Cartesian

```
 1 public class Cartesian implements Comparable {
 2     private double x;
 3     private double y;

 4     public Cartesian() {
 5         x = 0.0;
 6         y = 0.0;
 7     }

 8     public Cartesian(double x, double y) {
 9         this.x = x;
10         this.y = y;
11     }

12     // toString(), hashCode(), and equals() are methods
13     // common to all Objects
14     public String toString() {
15         return "(" + x + ", " + y + ")";
16     }

17     // Key guarantee:
18     // For objects A and B, A.equals(B) should imply
19     // A.hashCode() == B.hashCode()
20     public int hashCode() {
21         return toString().hashCode();
22     }

23     public boolean equals(Object other) {
24         Comparable oth = (Comparable) other;
25         return compareTo(oth) == 0;
26     }

27     // Having a comparison is required if this object is
28     // to be stored in a tree-based collection
29     public int compareTo(Object other) {
30         Cartesian oth = (Cartesian) other;
```



```
31         if(x == oth.x) {
32             if(y > oth.y) {
33                 return 1;
34             } else if(y < oth.y) {
35                 return -1;
36             } else {
37                 return 0;
38             }
39         } else if(x > oth.x) {
40             return 1;
41         } else {
42             return -1;
43         }
44     }
45 }
```

Laboratory: DNA testing

Assignment

Criminal investigations often hinge on DNA testing, in which a program compares DNA found at the crime scene to a number of suspects. In this assignment, you will implement a simple DNA tester.

A DNA sample is a long string of nucleotides. In humans, there are four possible nucleotides, abbreviated A, C, T, and G. (In the program, we will represent a DNA sample as a String object of these four characters.) For example, suppose we have two people with the following DNA sequences.

Groucho	`ACGCCAGCAATTTCAACTCC`
Harpo	`GACCTTTTGACAATCATGCT`

To see how close Groucho and Harpo's DNA sequences are, we can simply count how many nucleotides they have in common at the same place. For these sequences, there are 4 matches (a common C at the fourth position, a common A near the middle, another common A at the end, and a common C in the next-to-last spot). Since there are 20 characters in these DNA sequences, we would say that there is a $4/20 = 20\%$ match.*

The handout code includes several files. The only one you need to modify is the `computeTopMatches` method in the DnaComputation class.

`DnaResult[] computeResults(DnaSequence query, Set sequences)`

Computes matching results, arranged in decreasing order by score. The `sequences` parameter is a set of DnaSequence objects, which will not include `query`. All of these sequences, including that of `query`, will have the same length.

The method computes the closeness of each sequence in `sequences` with the `query` sequence. The score will be the fraction, between 0.0 and 1.0, of nucleotides matching between the pair. Remember that the results in the returned array should be arranged starting with the closest match.

*In the data file handed out, DNA sequences actually have 100 characters (though your program should not depend on this), and the similarity between Groucho and Harpo is not the same.

To verify that your solution works, you should compare the results of your tester with your classmates. If they disagree, then both of you can work together to try to determine which is incorrect.

Class documentation

(All of the classes are designed to be accessible to people familiar with the AP subset and with Swing, but only the following two are important to completing this assignment.)

Class DnaSequence

`DnaSequence(String myName, String mySequence)`

(Constructor) Creates a DnaSequence associated a person's name with that person's DNA sequence.

`String getName()`

Returns the name of the person with this sequence.

`String getSequence()`

Returns the string of nucleotides found in the sequence.

Class DnaResult

`DnaResult(double myScore, DnaSequence mySequence)`

(Constructor) Creates a DnaResult representing the score computed for a particular sequence.

`double getScore()`
: Returns the score associated with this result.

`DnaSequence getDnaSequence()`
: Returns the sequence for which this result applies.

Tying up loose ends

Class LineReader

```
 1 import java.io.*;

 2 public class LineReader {

 3     boolean error = false;

 4     public String readLine() {
 5         try {
 6             BufferedReader console =
 7                 new BufferedReader
 8                 (new InputStreamReader(System.in));
 9             return console.readLine();
10         } catch (IOException e) {
11             error = true;
12             return null;
13         }
14     }

15     public boolean ioError() {
16         boolean err = error;
17         error = false;
18         return err;
19     }

20     public int readInteger() {
21         return Integer.parseInt(readLine());
22     }

23     public double readDouble() {
24         return Double.parseDouble(readLine());
25     }
26 }
```

Class IOTest

```
 1 public class IOTest {
 2     public static void run() {
 3         LineReader reader = new LineReader();
```



```
 4         System.out.print("Enter a string: ");
 5         String echo = reader.readLine();
 6         System.out.println("Echo: " + echo);

 7         System.out.print("Enter an integer: ");
 8         int i = reader.readInteger();
 9         System.out.println(i + " + 2 = " + (i + 2));

10         System.out.print("Enter a double: ");
11         double d = reader.readDouble();
12         System.out.println(d + " / 2 = " + (d / 2));
13     }
14 }
```

Class FileManager

```
 1 import java.io.*;

 2 public class FileManager {
 3     private String filename;
 4     private BufferedReader input;
 5     private PrintWriter output;

 6     public FileManager(String name) {
 7         filename = name;
 8         input = null;
 9         output = null;
10     }

11     // Returns false if an error occurs
12     public boolean openForRead() {
13         try {
14             input = new BufferedReader
15                 (new FileReader(filename));
16             return true;
17         } catch (FileNotFoundException e) {
18             return false;
19         }
20     }

21     // Returns false if an error occurs
22     public boolean openForWrite() {
23         try {
24             output = new PrintWriter
```

```
25                 (new FileWriter(filename));
26             return true;
27         } catch (IOException e) {
28             return false;
29         }
30     }

31     public void close() {
32         try {
33             if(input != null) {
34                 input.close();
35             }
36             if(output != null) {
37                 output.close();
38             }
39         } catch (IOException e) {
40             // Exceptions are ignored
41             // Just close everything up
42         } finally {
43             input = null;
44             output = null;
45         }
46     }

47     // Returns null if an error occurs or if the
48     // end of the file has been reached
49     public String readLine() {
50         try {
51             return input.readLine();
52         } catch (IOException e) {
53             return null;
54         }
55     }

56     // Returns false if an error occurs
57     public void println(String s) {
58         output.println(s);
59     }
60 }
```

Class TextDumper

```
 1 public class TextDumper {
 2     public static void run() {
 3         boolean input = false;
```

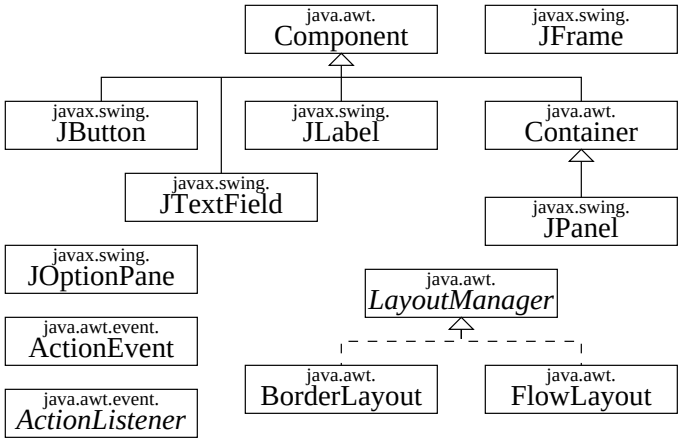


```
 4         boolean output = false;
 5         LineReader user = new LineReader();
 6         System.out.print("Do what (display/create)? ");
 7         String selection = user.readLine();

 8         if(selection.equals("display")) {
 9             input = true;
10         } else if(selection.equals("create")) {
11             output = true;
12         } else {
13             System.out.println("Bad choice. Aborted.");
14             System.exit(1);
15         }

16         System.out.print("Name of file: ");
17         String filename = user.readLine();

18         FileManager file = new FileManager(filename);
19         if(input) {
20             file.openForRead();
21             for(String line = file.readLine();
22                  line != null;
23                  line = file.readLine()) {
24                 System.out.println(line);
25             }
26         } else {
27             file.openForWrite();
28             System.out.println
29                 ("Begin entering text. " +
30                  "Type x by itself when done.");
31             LineReader reader = new LineReader();
32             for(String line = reader.readLine();
33                  !line.equals("x");
34                  line = reader.readLine()) {
35                 file.println(line);
36             }
37         }

38         file.close();
39     }
40 }
```

Swing basics

Class hierarchy

Here is a diagram of the Swing classes that we'll use in the following two examples.*



Class SqrtCalculator

```
 1 import java.awt.Container;
 2 import java.awt.FlowLayout;
 3 import java.awt.event.ActionListener;
 4 import java.awt.event.ActionEvent;
 5 import javax.swing.JFrame;
 6 import javax.swing.JButton;
 7 import javax.swing.JOptionPane;

 8 public class SqrtCalculator extends JFrame
 9         implements ActionListener {
10     // instance variables
11     private JButton compute;
12     private JButton quit;
```

*This diagram represents the inheritance structure as you ought to think of it, not as it actually is. The reality is more complex, largely because of the decision to build Swing underneath the existing AWT library.


```
13     // constructor methods
14     public SqrtCalculator() {
15         setTitle("Square Root Calculator");

16         Container contents = getContentPane();
17         contents.setLayout(new FlowLayout());

18         compute = new JButton("Compute...");
19         compute.addActionListener(this);
20         contents.add(compute);

21         quit = new JButton("Quit");
22         quit.addActionListener(this);
23         contents.add(quit);

24         pack();
25     }

26     // instance methods
27     public void actionPerformed(ActionEvent event) {
28         Object src = event.getSource();
29         if(src == compute) {
30             String what = JOptionPane.showInputDialog(
31                 this, "Find the square root of what?");
32             if(what != null) {
33                 double val = Double.parseDouble(what);
34                 JOptionPane.showMessageDialog(this,
35                     "The square root of " + what
36                     + " is " + Math.sqrt(val));
37             }
38         } else if(src == quit) {
39             System.exit(0); // terminate program
40         }
41     }

42     // class methods
43     public static void run() {
44         SqrtCalculator win = new SqrtCalculator();
45         win.show();
46     }
47 }
```

Class SqrtCalculator2

```
 1 import java.awt.Container;
 2 import java.awt.BorderLayout;
 3 import java.awt.event.ActionListener;
 4 import java.awt.event.ActionEvent;
 5 import javax.swing.JFrame;
 6 import javax.swing.JButton;
 7 import javax.swing.JTextField;
 8 import javax.swing.JLabel;
 9 import javax.swing.JPanel;

10 public class SqrtCalculator2 extends JFrame
11         implements ActionListener {
12     // instance variables
13     private JTextField input;
14     private JLabel output;
15     private JButton compute;
16     private JButton quit;

17     // constructor methods
18     public SqrtCalculator2() {
19         setTitle("Square Root Calculator");

20         input = new JTextField(10);
21         output = new JLabel("Nothing computed");

22         JPanel buttons = new JPanel();
23         compute = new JButton("Compute");
24         compute.addActionListener(this);
25         buttons.add(compute);
26         quit = new JButton("Quit");
27         quit.addActionListener(this);
28         buttons.add(quit);

29         Container contents = getContentPane();
30         contents.add(input, BorderLayout.NORTH);
31         contents.add(output, BorderLayout.CENTER);
32         contents.add(buttons, BorderLayout.SOUTH);

33         pack();
34     }

35     // instance methods
36     public void actionPerformed(ActionEvent event) {
37         Object src = event.getSource();
```

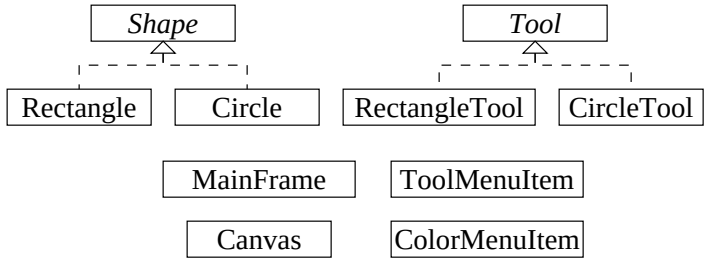


```
38         if(src == compute) {
39             String what = input.getText();
40             double val = Double.parseDouble(what);
41             output.setText("sqrt(" + what + ")"
42                 + " = " + Math.sqrt(val));
43         } else if(src == quit) {
44             System.exit(0); // terminate program
45         }
46     }

47     // class methods
48     public static void run() {
49         SqrtCalculator2 win = new SqrtCalculator2();
50         win.show();
51     }
52 }
```

Laboratory: Drawing program

Overview

This program is a fairly large multi-class program that combines the usage of polymorphism and Swing into a fairly realistic program for drawing shapes on a canvas. It includes eight classes and two interfaces.



Interface Shape

```
1 import java.awt.Graphics;

2 public interface Shape {
3     /** Draws this shape using the Graphics object. */
4     public void draw(Graphics g);
5 }
```

Class Rectangle

```
1 import java.awt.Color;
2 import java.awt.Graphics;

3 public class Rectangle implements Shape {
4     private Color color; // color of rectangle
5     private int x;       // x-coordinate of top left
6     private int y;       // y-coordinate of top left
7     private int width;   // width of rectangle in pixels
8     private int height;  // height of rectangle in pixels
```



```
 9     /** (Constructor) Creates a rectangle whose top left
10      * corner is at (myX, myY), and whose width and
11      * height are as given. */
12     public Rectangle(Color myColor, int myX, int myY,
13             int myWidth, int myHeight) {
14         color = myColor;
15         x = myX;
16         y = myY;
17         width = myWidth;
18         height = myHeight;
19     }

20     /** Draws this rectangle using the Graphics
21      * object. */
22     public void draw(Graphics g) {
23         g.setColor(color);
24         g.fillRect(x, y, width, height);
25     }
26 }
```

Class Circle

```
 1 import java.awt.Color;
 2 import java.awt.Graphics;

 3 public class Circle implements Shape {
 4     private Color color; // color of circle
 5     private int x;    // x-coordinate of circle's center
 6     private int y;    // y-coordinate of circle's center
 7     private int r;    // radius of circle

 8     /** (Constructor) Creates a circle of the given color
 9      * whose center is at (myX, myY), and whose radius
10      * is myR. */
11     public Circle(Color myColor, int myX, int myY,
12                   int myR) {
13         color = myColor;
14         x = myX;
15         y = myY;
16         r = myR;
17     }

18     /** Draws this circle using the Graphics object. */
19     public void draw(Graphics g) {
20         g.setColor(color);
```

```
21          g.fillOval(x - r, y - r, 2 * r, 2 * r);
22      }
23  }
```

Class Canvas

```
 1  import java.awt.Color;
 2  import java.awt.Dimension;
 3  import java.awt.Graphics;
 4  import java.awt.event.MouseListener;
 5  import java.awt.event.MouseMotionListener;
 6  import java.awt.event.MouseEvent;
 7  import javax.swing.JPanel;
 8  import java.util.ArrayList;
 9  import java.util.List;
10  import java.util.Iterator;

11  public class Canvas extends JPanel
12          implements MouseListener, MouseMotionListener {
13      private Tool cur_tool;
14      private Color cur_color;
15      private ArrayList shapes;
16      private boolean drag_started;
17      private int drag_start_x;
18      private int drag_start_y;
19      private int drag_cur_x;
20      private int drag_cur_y;

21      //
22      // public methods intended for other classes' use
23      //
24      public Canvas() {
25          cur_tool = null;
26          cur_color = Color.black;
27          shapes = new ArrayList();
28          drag_started = false;

29          setPreferredSize(new Dimension(200, 200));
30          setBackground(Color.white);
31          addMouseListener(this);
32          addMouseMotionListener(this);
33      }

34      /** Returns the current tool used for this canvas. */
35      public Tool getTool() {
```



```
36          return cur_tool;
37      }

38      /** Sets the current tool used for this canvas. */
39      public void setTool(Tool tool) {
40          cur_tool = tool;
41      }

42      /** Gets the currently associated color. */
43      public Color getColor() {
44          return cur_color;
45      }

46      /** Sets the currently associated color. */
47      public void setColor(Color color) {
48          cur_color = color;
49      }

50      /** Adds a shape to this canvas. */
51      public void addShape(Shape to_add) {
52          shapes.add(to_add);
53          repaint();
54      }

55      /** Removes a shape from this canvas. */
56      public void removeShape(Shape to_remove) {
57          shapes.remove(to_remove);
58          repaint();
59      }

60      /** Return a list of Shapes for this canvas,
61       * beginning with the ''deepest'' shape. If any
62       * shapes overlap, the shape appearing later in the
63       * list will be drawn on top.
64       */
65      public List getShapes() {
66          return shapes;
67      }

68      //
69      // methods to do other things (and which happen to
70      // need to be public)
71      //
72      /** Draws the shape on the canvas. This overrides the
73       * JPanel's paintComponent method. */
```

```
 74     public void paintComponent(Graphics g) {
 75         super.paintComponent(g);

 76         for(int i = 0; i < shapes.size(); i++) {
 77             Shape sh = (Shape) shapes.get(i);
 78             sh.draw(g);
 79         }

 80         if(drag_started) {
 81             cur_tool.drawDragInProgress(g,
 82                 drag_start_x, drag_start_y,
 83                 drag_cur_x, drag_cur_y);
 84         }
 85     }

 86     // MouseListener methods
 87     public void mouseClicked(MouseEvent e) { }
 88     public void mouseEntered(MouseEvent e) { }
 89     public void mouseExited(MouseEvent e) { }
 90     public void mousePressed(MouseEvent e) {
 91         if(cur_tool != null) {
 92             drag_started = true;
 93             drag_start_x = e.getX();
 94             drag_start_y = e.getY();
 95             drag_cur_x = drag_start_x;
 96             drag_cur_y = drag_start_y;
 97             repaint();
 98         }
 99     }
100     public void mouseReleased(MouseEvent e) {
101         if(drag_started) {
102             drag_started = false;
103             drag_cur_x = e.getX();
104             drag_cur_y = e.getY();
105             cur_tool.mouseDragged(this,
106                     drag_start_x, drag_start_y,
107                     drag_cur_x, drag_cur_y);
108         }
109     }

110     // MouseMotionListener methods
111     public void mouseMoved(MouseEvent e) { }
112     public void mouseDragged(MouseEvent e) {
113         if(drag_started) {
114             drag_cur_x = e.getX();
```



```
115             drag_cur_y = e.getY();
116             repaint();
117         }
118     }
119 }
```

Interface Tool

```
 1 import java.awt.Graphics;

 2 public interface Tool {
 3     /** Returns the name associated with this tool (as
 4      * it should appear in the Tool menu. */
 5     public String getName();

 6     /** Draws any information the user should see
 7      * while the mouse is being dragged. */
 8     public void drawDragInProgress(Graphics g,
 9             int draw_start_x, int drag_start_y,
10             int drag_cur_x, int drag_cur_y);

11     /** Performs an action when a mouse drag is done. */
12     public void mouseDragged(Canvas c,
13             int drag_start_x, int drag_start_y,
14             int drag_end_x, int drag_end_y);
15 }
```

Class RectangleTool

```
 1 import java.awt.Color;
 2 import java.awt.Graphics;

 3 public class RectangleTool implements Tool {
 4     /** (Constructor) Creates a RectangleTool. */
 5     public RectangleTool() { }

 6     /** Returns the name of this tool. */
 7     public String getName() {
 8         return "Rectangle Tool";
 9     }

10     /** Draws a ghost showing the extent of the rectangle
11      * as it would appear on the canvas. */
12     public void drawDragInProgress(Graphics g,
13             int drag_start_x, int drag_start_y,
```

```
14              int drag_cur_x, int drag_cur_y) {
15          int x = Math.min(drag_start_x, drag_cur_x);
16          int y = Math.min(drag_start_y, drag_cur_y);
17          int width = Math.abs(drag_start_x - drag_cur_x);
18          int height= Math.abs(drag_start_y - drag_cur_y);
19          g.setColor(Color.gray);
20          g.drawRect(x, y, width - 1, height - 1);
21      }

22      /** Adds a rectangle to the canvas. */
23      public void mouseDragged(Canvas c,
24              int drag_start_x, int drag_start_y,
25              int drag_end_x, int drag_end_y) {
26          int x = Math.min(drag_start_x, drag_end_x);
27          int y = Math.min(drag_start_y, drag_end_y);
28          int width = Math.abs(drag_start_x - drag_end_x);
29          int height= Math.abs(drag_start_y - drag_end_y);
30          Rectangle rect = new Rectangle(c.getColor(),
31                                         x, y, width, height);
32          c.addShape(rect);
33      }
34  }
```

Class CircleTool

```
 1  import java.awt.Color;
 2  import java.awt.Graphics;

 3  public class CircleTool implements Tool {
 4      /** (Constructor) Creates a CircleTool. */
 5      public CircleTool() { }

 6      /** Returns the name of this tool. */
 7      public String getName() {
 8          return "Circle Tool";
 9      }

10      /** Draws a ghost showing the extent of a circle as
11       * it would appear on the canvas. */
12      public void drawDragInProgress(Graphics g,
13              int drag_start_x, int drag_start_y,
14              int drag_cur_x, int drag_cur_y) {
15          int x = drag_start_x;
16          int y = drag_start_y;
17          double rf = Math.sqrt(Math.pow(x - drag_cur_x, 2)
```



```
18                          + Math.pow(y - drag_cur_y, 2));
19          int r = (int) Math.round(rf);
20          g.setColor(Color.gray);
21          g.drawOval(x - r, y - r, 2 * r - 1, 2 * r - 1);
22      }

23      /** Adds a circle to the canvas. */
24      public void mouseDragged(Canvas c,
25              int drag_start_x, int drag_start_y,
26              int drag_end_x, int drag_end_y) {
27          int x = drag_start_x;
28          int y = drag_start_y;
29          double rf = Math.sqrt(Math.pow(x - drag_end_x, 2)
30                          + Math.pow(y - drag_end_y, 2));
31          int r = (int) Math.round(rf);
32          Circle circ = new Circle(c.getColor(), x, y, r);
33          c.addShape(circ);
34      }
35  }
```

Class MainFrame

```
 1  import java.awt.Color;
 2  import java.awt.Container;
 3  import java.awt.event.ActionListener;
 4  import java.awt.event.ActionEvent;
 5  import java.awt.event.WindowListener;
 6  import java.awt.event.WindowEvent;
 7  import javax.swing.ButtonGroup;
 8  import javax.swing.JFrame;
 9  import javax.swing.JMenuBar;
10  import javax.swing.JMenu;
11  import javax.swing.JMenuItem;
12  import java.util.ArrayList;

13  public class MainFrame extends JFrame
14          implements ActionListener, WindowListener {
15      /** (Constructor) Creates a frame with a canvas. */
16      public MainFrame() {
17          super("Draw");
18          addWindowListener(this);
19          setResizable(false);

20          Container contents = getContentPane();
21          Canvas canv = new Canvas();
```

```
22          contents.add(canv);

23          JMenuBar menubar = new JMenuBar();
24          menubar.add(createFileMenu());
25          menubar.add(createToolMenu(canv));
26          menubar.add(createColorMenu(canv));
27          setJMenuBar(menubar);

28          pack();
29      }

30      /** Creates the File menu for the frame. */
31      private JMenu createFileMenu() {
32          JMenu ret = new JMenu("File");
33          JMenuItem quit = new JMenuItem("Quit");
34          quit.addActionListener(this);
35          ret.add(quit);
36          return ret;
37      }

38      /** Creates the Tool menu for the frame. */
39      private JMenu createToolMenu(Canvas canv) {
40          // First, create list of tools to have in menu.
41          ArrayList tools = new ArrayList();
42          tools.add(new RectangleTool());
43          tools.add(new CircleTool());

44          // Then, set current tool for canvas.
45          canv.setTool((Tool) tools.get(0));

46          // Finally, create menu to return.
47          JMenu ret = new JMenu("Tool");
48          ButtonGroup bgroup = new ButtonGroup();
49          for(int i = 0; i < tools.size(); i++) {
50              Tool tool = (Tool) tools.get(i);
51              ToolMenuItem item
52                      = new ToolMenuItem(canv, tool);
53              bgroup.add(item);
54              ret.add(item);
55          }
56          return ret;
57      }

58      /** Creates the Color menu for the frame. */
59      private JMenu createColorMenu(Canvas canv) {
```



```
60          // First, create list of colors to have in menu,
61          // with a parallel list of names for the colors.
62          ArrayList colors = new ArrayList();
63          ArrayList names = new ArrayList();
64          colors.add(Color.red);     names.add("Red");
65          colors.add(Color.yellow);  names.add("Yellow");
66          colors.add(Color.green);   names.add("Green");
67          colors.add(Color.cyan);    names.add("Cyan");
68          colors.add(Color.blue);    names.add("Blue");
69          colors.add(Color.magenta); names.add("Magenta");
70          colors.add(Color.black);   names.add("Black");
71          colors.add(Color.white);   names.add("White");

72          // Finally, create menu to return.
73          JMenu ret = new JMenu("Color");
74          ButtonGroup bgroup = new ButtonGroup();
75          for(int i = 0; i < colors.size(); i++) {
76              Color color = (Color) colors.get(i);
77              String name = (String) names.get(i);
78              ColorMenuItem item
79                  = new ColorMenuItem(name, canv, color);
80              bgroup.add(item);
81              ret.add(item);
82          }
83          return ret;
84      }

85      // ActionListener method (for Quit menu item only)
86      public void actionPerformed(ActionEvent e) {
87          System.exit(0);
88      }

89      // WindowListener methods
90      public void windowActivated(WindowEvent e) { }
91      public void windowClosed(WindowEvent e) { }
92      public void windowClosing(WindowEvent e) {
93          System.exit(0);
94      }
95      public void windowDeactivated(WindowEvent e) { }
96      public void windowDeiconified(WindowEvent e) { }
97      public void windowIconified(WindowEvent e) { }
98      public void windowOpened(WindowEvent e) { }

99      //
100     // class methods
```

```
101     //
102     public static void run() {
103         MainFrame frame = new MainFrame();
104         frame.show();
105     }
106 }
```

Class ToolMenuItem

```
 1 import java.awt.event.ActionListener;
 2 import java.awt.event.ActionEvent;
 3 import javax.swing.JRadioButtonMenuItem;

 4 public class ToolMenuItem extends JRadioButtonMenuItem
 5         implements ActionListener {
 6     private Canvas canvas;
 7     private Tool tool;

 8     /** (Constructor) Creates a ToolMenuItem for a
 9      * particular tool. */
10     public ToolMenuItem(Canvas myCanvas, Tool myTool) {
11         super(myTool.getName());
12         canvas = myCanvas;
13         tool = myTool;
14         addActionListener(this);
15         setSelected(canvas.getTool().equals(tool));
16     }

17     /** Changes the tool associated with the canvas. */
18     public void actionPerformed(ActionEvent e) {
19         if(isSelected()) {
20             canvas.setTool(tool);
21         }
22     }
23 }
```

Class ColorMenuItem

```
 1 import java.awt.Color;
 2 import java.awt.event.ActionListener;
 3 import java.awt.event.ActionEvent;
 4 import javax.swing.JRadioButtonMenuItem;

 5 public class ColorMenuItem extends JRadioButtonMenuItem
 6         implements ActionListener {
```



```
 7     private Canvas canvas;
 8     private Color color;

 9     /** (Constructor) Creates a ColorMenuItem for a color
10      * with the given name. */
11     public ColorMenuItem(String name, Canvas myCanvas,
12                 Color myColor) {
13         super(name);
14         canvas = myCanvas;
15         color = myColor;
16         addActionListener(this);
17         setSelected(canvas.getColor().equals(color));
18     }

19     /** Changes the color associated with the canvas. */
20     public void actionPerformed(ActionEvent e) {
21         if(isSelected()) {
22             canvas.setColor(color);
23         }
24     }
25 }
```

Assignment

Modify the Drawing program to include a tool for deleting: If the user clicks on a shape, the shape should be removed from the canvas. You should define clicking the shape as being when the user's drag begins and ends on the same shape.

To do this, you should add a `contains` method to the Shape interface for querying whether that shape contains a point. You can easily figure out whether a circle contains a point by computing the distance to the circle's center, and comparing this result to the circle's radius.

(If you're feeling really ambitious, other more difficult projects are a tool for moving components, a tool for drawing lines, a tool for drawing text (using a KeyListener), and an undo menu option.)