Computer Science 134
Spring 2003

Midterm Examination
13 April 2003

Question	Points	Score	Description
1	12		Program Analysis
2	24		Classes
3	12		Style and Simplification
4	24		GUI/ActiveObjects
5	16		Recursion
6	12		Declarations
Total	100		

This examination is closed book. You have 90 minutes to complete the exam.

Please mark your lecture section:

9am _____ 10am _____

Please mark your lab section:

Monday _____ Tuesday _____

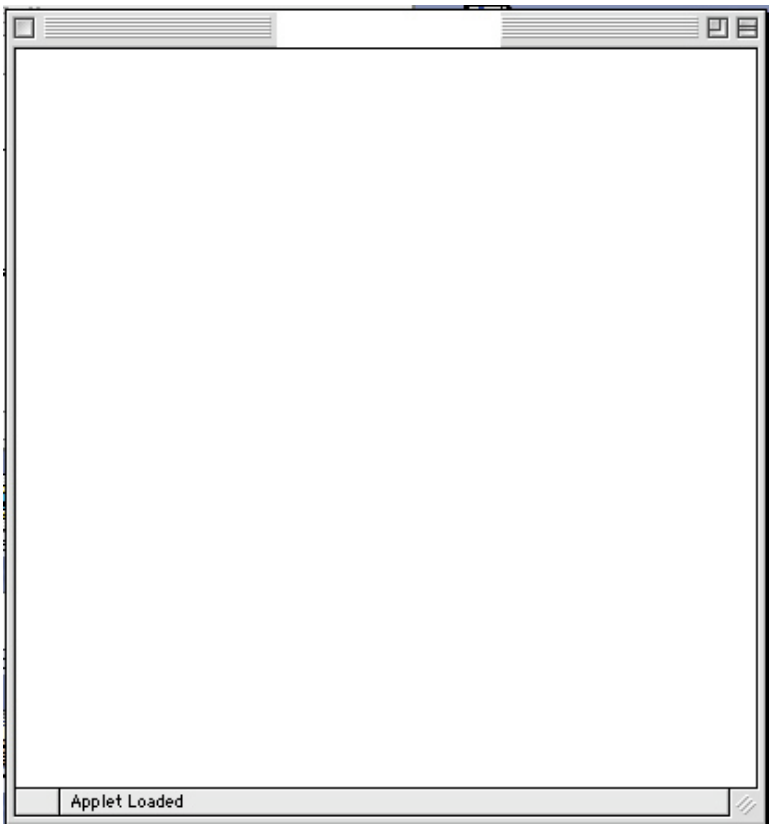
Your Name (Please print) _____

I have neither given nor received aid on this examination

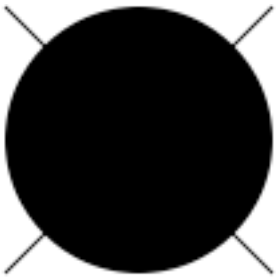
_____

1) The following is a simple but complete Java program that does nothing more than draw a picture on the canvas. Draw a sketch of the picture that will be produced on the canvas provided. The dimensions of the given canvas are 400x400.

```
import objectdraw.*;

public class MysteryDrawing extends WindowController {
  private static final int SPACING = 50;
  private static final int SIZE = 200;

   public void begin() {
     int position = 0;

     while (position <= SIZE) {
         new Line(0, position, SIZE, position, canvas);
         new Line(position, 0, position, SIZE, canvas);
         if (position < SIZE) {
           new FramedOval(position, position, SPACING, SPACING, canvas);
           new FramedOval(position, SIZE-SPACING-position, SPACING,
                          SPACING, canvas);
         }
         position = position + SPACING;
     }
   }
}
```



2) Spring has arrived, so we shift our attention away from snowmen to the sun. The class below is a partial implementation of a draggable "Sun-like" object that should look like this (except for the color):



You can think of this "sun" as having four rays emanating out (though you can draw them with just two lines, if you think about it a bit). The sun should be drawn initially with its upper left corner at position 100,100 on the canvas, with a width and height also of 100. It should be yellow. The `Sun` class should, however, be capable of drawing itself at any position on the canvas and with any size (you may assume that the height and the width are the same). Your sun should be "draggable," so it will need a `move` method and a `contains` method.

To make things slightly more interesting, the sun's rays should appear and disappear, and the sun should change color. While the sun is being dragged, its rays should disappear and the main body of the sun should turn orange. You can think of this as a setting sun. When the mouse is released, it should be returned to its original yellow. To support this, the `Sun` class must include a method `setting` , which make the rays disappear and changes the sun's color to orange, and a method `shining`, which makes the rays appear and restores the sun to its usual yellow color.

You are to fill in the classes below to implement this draggable shining sun. Fill in any needed constants, instance variables, local variables, formal parameters, and method bodies. You need not include comments.

```
import objectdraw.*;

public class StarGame extends WindowController {




  public void begin() {




  }
```

```
  public void onMousePress(Location point) {









  }
  public void onMouseDrag(Location point) {









  }
  public void onMouseRelease(Location point) {









  }
}
```

Sun.java:

```
import objectdraw.*;
import java.awt.*;

public class Sun {



  public Sun(                                                    ) {
```

```
 }
  public void move(                                        ) {









  }
  public void setting() {







  }
  public void shining() {







  }
  public boolean contains(Location point) {









  }
}
```

3) Style and Simplification

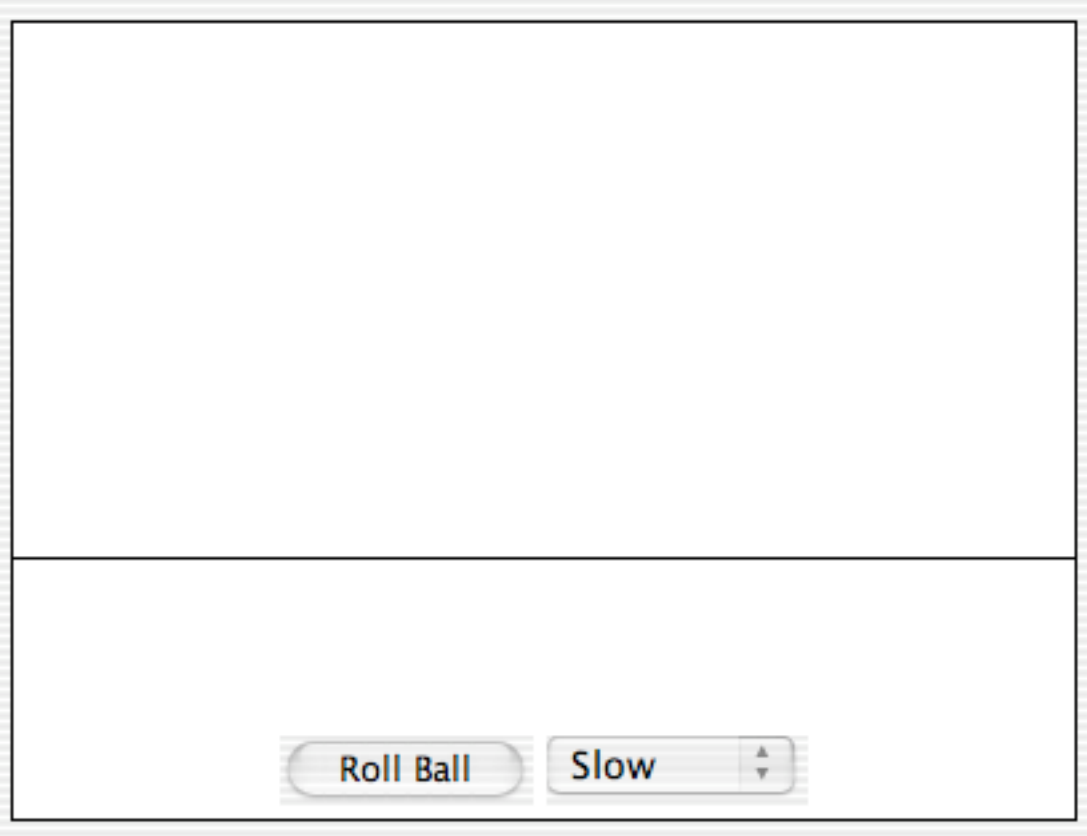
Please rewrite the following Java code so that it has the same effect, but it is written with better style, is written more concisely and understandably, and/or runs more efficiently:

a.
```
if (x < 0) {
   System.out.println("The answer is an outlier: "+x);
else if (x >= 100) {
   System.out.println("The answer is an outlier: "+x);
else {
   System.out.println("The answer is ok");
}
```

b.
```
if (x < 0) {
} else {
   sum = sum + x;
}
```

c.
```
int sum = 0;
int n = 1;
while (n <= 30) {
   if (sum < 100) {
       sum = sum + n;
   }
   n = n + 1;
}
System.out.println(sum);
```

4) This question tests your knowledge both of GUI components and `ActiveObjects`. When the program begins, the following picture should be drawn:



The bottom of the window contains a panel that holds a button labeled "Roll Ball" and a Choice button (menu) that offers choices of "Slow" and "Fast". When the user pushes the "Roll Ball" button, a ball appears at the left edge of the window, with its bottom resting on the horizontal line. The ball then rolls (moves) smoothly to the right until its right edge reaches the right side of the screen. The ball should then burst into flames (turn red) for two seconds (2000 milliseconds) and then be removed from the screen. The ball, of course, is an `ActiveObject`.

The speed of the ball is determined by the setting of the Choice button. If "Slow" is showing when the "Roll Ball" button is pushed then the ball moves at a rate of 1 pixel every 30 milliseconds. If "Fast" is showing then it moves 2 pixels every 30 milliseconds. Changing the setting of the Choice button has no impact on the speed of a ball that has already been constructed.

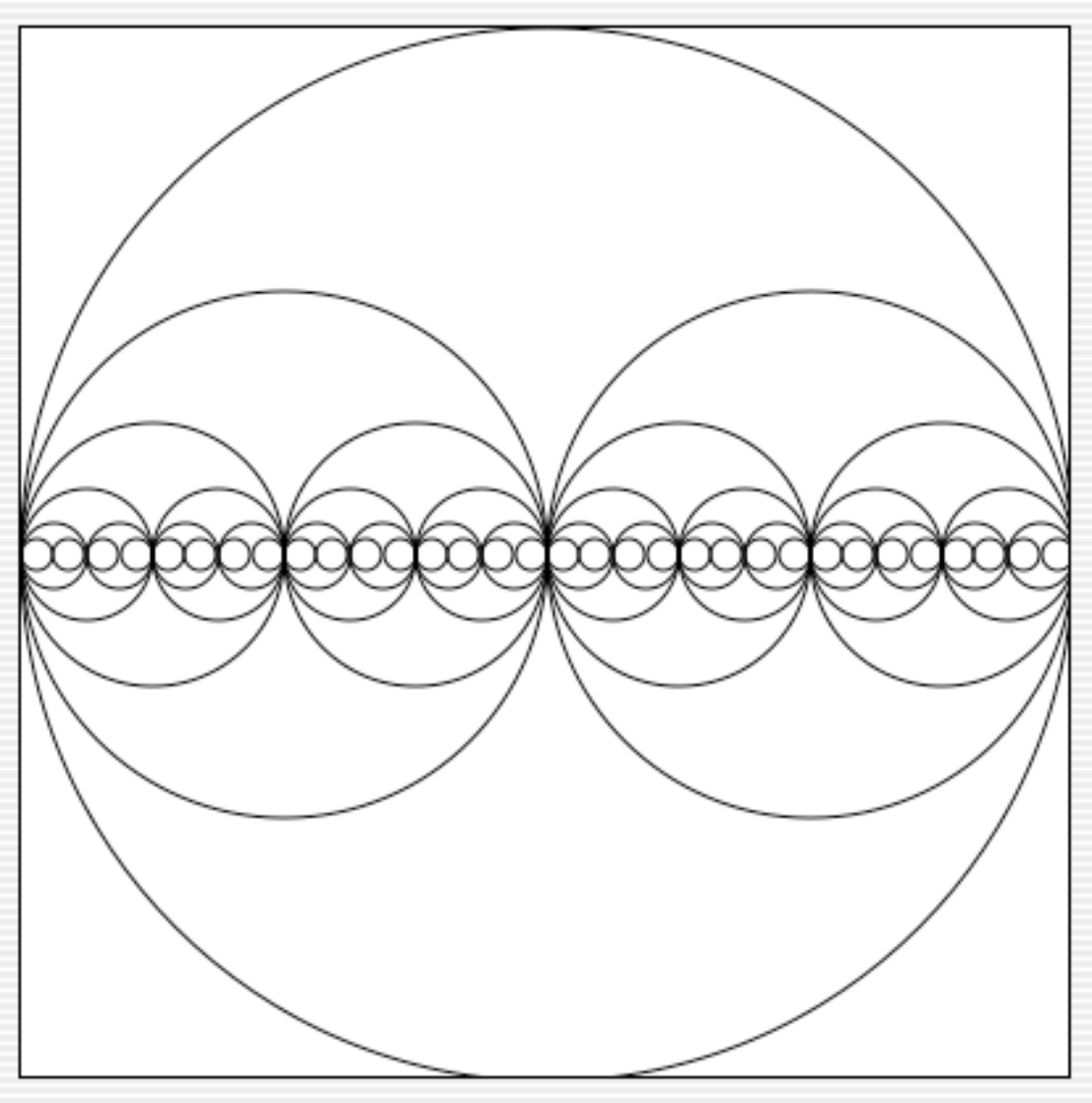
The window is 400 pixels wide and 300 pixels high. The horizontal line is drawn 200 pixels below the top of the screen. The ball should have diameter of 30,

Please complete the definitions of the two classes on the next two pages.

```
public class GUIBall extends WindowController
{
   private static final double BALL_SIZE = 30;



   public void begin() {









      validate();
   }









}
```

```
public class Ball extends ActiveObject {

   public Ball(double x, double y, double width,
               int speed, DrawingCanvas canvas) {

   }

   public void run() {

   }
}
```

5) This question asks you to write a **recursive** constructor for a class called `NestedEyes`. A `NestedEyes` object is a graphical object made up of circles in a pattern as in the picture below:



The picture consists of an outer circle with two copies of `NestedEyes` inside, each of which is half the size of the original. No new circles are drawn inside a circle whose radius is 10 or smaller.

The constructor for the `NestedEyes` class should take 5 parameters:

- `x` – the x coordinate of the left edge of the object
- `y` – the y coordinate of the middle of the object (e.g., 200 if the canvas height is 400).
- `radius` – the radius of the outer circle
- `canvas` – the drawing canvas

Here is the controller class that draws the above picture:

```
public class NestedEyesController extends WindowController {

   private NestedEyes pict;
   private Location lastMouse;
   private boolean dragging;

   public void begin() {
      pict = new NestedEyes(0,200,200,canvas);
   }

   public void onMousePress(Location pt) {
      dragging = pict.contains(pt);
      lastMouse = pt;
   }

   public void onMouseDrag(Location pt) {
      if (dragging) {
         pict.move(pt.getX()-lastMouse.getX(),
         pt.getY()-lastMouse.getY());
      }
      lastMouse = pt;
   }
}
```

You are to write the `NestedEyes` class, including the constructor, `contains`, and `move` methods. Continue on the back of this page, if you need more space.

6) Below and on the following two pages, we have included the (almost) complete code for a solution to the Ski Ball problem from the first test program.

While the program is nearly complete, a few of the names that are used in the program are never declared. In particular, if you examine the program carefully, you will notice that although the names

biggest	big	middle	top	display	start	trail
ball	pullBack		pullSize		endLocn	score

are used in the code, there are no declarations for any of these names in the program. Perhaps this program was written by a frustrated CS134 student who got tired of losing points for declaring variables incorrectly and decided not to bother at all. To make it easier to find the uses of these names within the program, we have displayed them in **bold font**.

For this problem, we would like you to add any declarations for the names shown above that are needed to make the program correct. In doing this remember:

- The only modification you should make to the code is the addition of declarations.
- You should declare each name in the most appropriate way (i.e., as an instance variable or as a variable local to a method or as a formal parameter to a method).
- You should make each name's declaration as local as possible while making the program correct.

We have attempted to leave enough extra space in each area of the program where declarations could be added to enable you to write the declarations you believe are needed in their appropriate places.

```
import objectdraw.*;

public class SkiBall extends WindowController {

   private static final Location CENTER = new Location(200,200);
   private static final double BIGGEST_DIAMETER = 150;
   private static final double BIG_DIAMETER = 100;
   private static final double MIDDLE_DIAMETER = 50;
   private static final double TOP_DIAMETER = 25;

   private static final double POWER = 4;
   private static final double LINE_Y = 500;
   private static final double BALL_RADIUS = 5;
```

```
public void begin() {









   biggest = new FramedOval(CENTER.getX()-BIGGEST_DIAMETER/2,
                            CENTER.getX()-BIGGEST_DIAMETER/2,
                            BIGGEST_DIAMETER, BIGGEST_DIAMETER, canvas);

   big = new FramedOval(CENTER.getX()-BIG_DIAMETER/2,
                            CENTER.getX()-BIG_DIAMETER/2,
                            BIG_DIAMETER, BIG_DIAMETER, canvas);

   middle = new FramedOval(CENTER.getX()-MIDDLE_DIAMETER/2,
                            CENTER.getX()-MIDDLE_DIAMETER/2,
                            MIDDLE_DIAMETER, MIDDLE_DIAMETER, canvas);

   top = new FramedOval(CENTER.getX()-TOP_DIAMETER/2,
                            CENTER.getX()-BIGGEST_DIAMETER/2,
                            TOP_DIAMETER, TOP_DIAMETER, canvas);

   new Line(0,LINE_Y,canvas.getWidth(),LINE_Y,canvas);

   display = new Text("Drag to shoot", CENTER.getX() - 50, 450,canvas);
}

public void onMousePress(
                        Location point) {









   start = new Location(point.getX(),LINE_Y);
   trail = new Line(start,point,canvas);
   ball = new FilledOval(start,2*BALL_RADIUS,2*BALL_RADIUS,canvas);
   ball.move(-BALL_RADIUS,-BALL_RADIUS);
}
```

```
   public void onMouseDrag(
                         Location point) {



      trail.setEnd(point);
   }

   public void onMouseRelease(
                            Location point) {









      pullBack = point.getY() - trail.getStart().getY();
      pullSide = point.getX() - trail.getStart().getX();
      endLocn = new Location(trail.getStart().getX()-pullSide*POWER,
                                    LINE_Y - pullBack*POWER);
      trail.removeFromCanvas();
      ball.moveTo(endLocn);
      ball.move(-BALL_RADIUS,-BALL_RADIUS);
       setScore(endLocn);
   }

   private void setScore(                                              ) {









      score = 0;
      if (top.contains(endLocn)) {
         score = 50;
      } else if (middle.contains(endLocn)) {
         score = 30;
      } else if (big.contains(endLocn)) {
         score = 20;
      } else if (biggest.contains(endLocn)) {
        score = 10;
      }
      display.setText("You scored "+ score + " points.");
   }

}
```