Computer Science 134
Fall 2001

Midterm Examination
1 November 2001

Question	Points	Score	Description
1	6		Arrays & Loops
2	12		Class definition
3	10		Program analysis
4	12		Programming - AWT
5	8		Declarations
6	12		Short Answers
Total	60		

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

Please mark your lecture and laboratory sections:

Lecture: 9am _____ 10am _____

Laboratory: Monday _____ Tuesday _____

Your Name (Please print) _____

I have neither given nor received aid on this examination

_____

1a. Suppose that `elements` is an array of `Strings` with `8` elements. A beginning programmer is trying to write code to delete the string at the beginning of the array, shift the next 4 Strings down to fill the gap, and then insert "#" as the new 5th element.

For example, if `elements` contains "A", "B", "C", "D", "E", "F", "G", "H", then after this operation, the array should contain "B", "C", "D", "E", "#", "F", "G", "H".

Unfortunately our programmer is having a lot of trouble. Only one of the three fragments below does what she wants. Which is correct?

i.
```
for (int counter = 0; counter < 4; counter++) {
    elements[counter] = elements[counter+1]
}
elements[4] = "#";
```

ii.
```
for (int counter = 0; counter < 5; counter++) {
    elements[counter] = elements[counter+1]
}
elements[5] = "#";
```

iii.
```
for (int counter = 4; counter > 0; counter--) {
    elements[counter-1] = elements[counter]
}
elements[4] = "#";
```

b. Please show what the array looks like after the two incorrect fragments if the array starts with the values "A", "B", "C", "D", "E", "F", "G", "H",

2. **Class Definition**:

We would like you to fill in the instance variable definitions, constructor, and the methods for a class that can be used to represent a car that can move horizontally across the screen. To make it interesting, the class is provided with two Images of a car, one pointing to the right, and the other to the left. When the car is moving from left to right, the image that faces right should be shown. When the car is moving from right to left, the image that faces left should be shown.

The constructor for the car should be provided with the two Images, an initial position, and an initial speed. It should display the appropriate image based on whether the initial speed is positive (i.e. the car is moving to the right) or negative.

The car is *not* an active object. Instead, it implements four methods named move, setSpeed, hide and show. The setSpeed method should take a double parameter which determines the speed of the car. If setSpeed changes the sign of the car's speed, then the image displayed should be changed. The move method takes no parameters. Every time the move method is called, the car should move horizontally an amount equal to the car's current speed (i.e., to the left if speed is negative and to the right if it is positive). Finally, hide() and show() should hide or show the appropriate image.

An outline of the class can be found on this and the next page. Please fill in the variable and parameter declarations, the constructor body, and the four method bodies and their return types.

```
Public class Car {


  public Car(                                                    ) {









  }
```

```
  // Move the car speed units
  public void move() {









  }

  public       setSpeed(                ) {









  }

  // show the appropriate car image
  public       show()
  {









  }


  // hide the car image
  public       hide()
  {









  }
}
```

3. Explain what the program on the following 2 pages does:
   a. When it starts:

   b. When the user presses and continues holding the mouse:

   c. When the user releases the mouse:

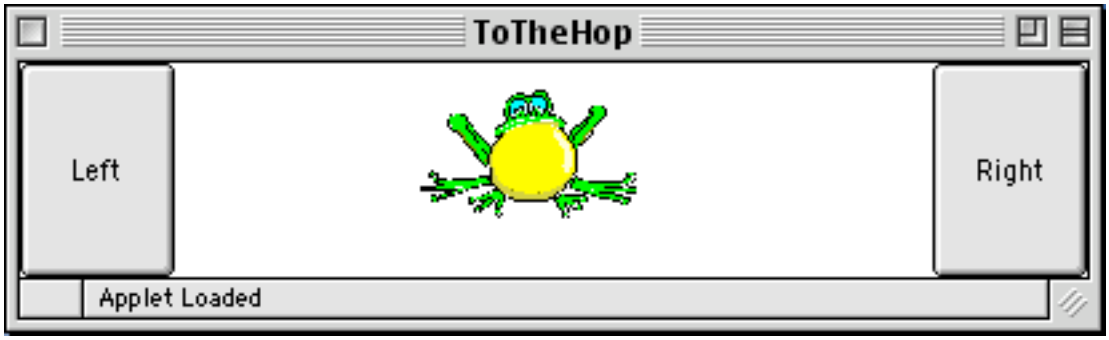
```
import java.awt.*;
import objectdraw.*;

public class Mystery extends WindowController
{
     private static final Location textPlace =
                                 new Location(150,450);
     private MysteryObject thing;


     public void begin() {
          new Text("hold the mouse down for a while",
                       textPlace,canvas);
     }

     public void onMousePress(Location point) {
          thing = new MysteryObject(point,canvas);
     }

     public void onMouseRelease(Location point) {
          thing.release();
     }

}
```

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

public class MysteryObject extends ActiveObject {
    private static final int START_DIAMETER = 4;
    private static final double INCREASE = 2;
    private static final int PAUSE_TIME = 50;

    private FramedOval circle;
    private boolean growing = true;

    public MysteryObject(Location point,
                         DrawingCanvas canvas)
   {
        circle = new FramedOval(
                        point.getX()-START_DIAMETER/2,
                        point.getY()-START_DIAMETER/2,
                        START_DIAMETER,START_DIAMETER,canvas);
        start();
    }

    public void release() {
        growing = false;
    }

    public void run() {
        while (growing) {
          circle.setWidth(circle.getWidth()+INCREASE);
          circle.setHeight(circle.getHeight() + INCREASE);
          circle.move(-INCREASE/2,-INCREASE/2);
          pause(PAUSE_TIME);
        }

        while(circle.getY() + circle.getHeight() >= 0) {
          circle.move(0,-INCREASE);
          pause(PAUSE_TIME);
        }
    }
}
```

4. A very enthusiastic CS 134 student named Herb became so excited when he learned about GUI components that he wanted to go back and revise the programs he wrote for some of the earlier labs to use them. For example, Herb wanted to change the Frogger program so that the player would have to click on appropriate buttons to move the frog rather than just clicking on the canvas. Unfortunately, despite his enthusiasm, Herb doesn't really understand how to write programs using GUI components. So, we would like you to help him get started on the task of revising his Frogger by writing a program to illustrate how a set of buttons could control the hopping of the frog.

   We don't want you to write all the code Herb will need. After all, this should be a learning experience for Herb. Instead, we just want you to write a simple program that will display the frog and two buttons on the screen. Clicking one of the buttons should make the frog "hop" to the left. Clicking the other button should make the frog hop to the right.

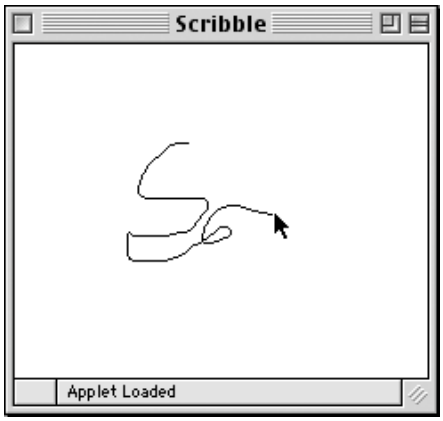
   We have included a picture of what the display should look like while the program is running below. In particular, in this picture we show how we would like the buttons to appear in the display.



   A "starter" for this program can be found on the next two pages. Please write all the additional code needed to implement the program described on this starter.

```
public class ToTheHop extends WindowController
                      implements ActionListener
{
  //  position frog is displayed initially
  private static final int FROGX = 160;
  private static final int FROGY = 10;

  // distance frog should hop
  private static final int FROGWIDTH = 70;
  // place your instance variable declarations here









  public void begin()
  {
    frog = new VisibleImage( getImage("froggy.gif"),
                             FROGX, FROGY, canvas );









}

//  place additional method defs on the following page.
```

5. During one of the very first classes of this course, we showed you a sample program named "Scribble". This program drew lines following the path of the mouse as the mouse was dragged around the screen. As a result, it let you create simple "pencil drawings" like the one shown below.



On the next page, we have provided an almost complete version of a new class called VanishingLine. The class VanishingLine extends ActiveObject. When a VanishLine is created, it draws a line on the screen and then it enters a loop that gradually changes the color of the line from black to gray to white until the line disappears. Accordingly, the VanishingLine class can be used to draw lines that will disappear from the screen in a specified amount of time.

Below, we show a WindowController that uses VanishingLine. This program is identical to the "Scribble" program shown in class except that it draws using VanishLines rather than Lines. So, like Scribble, it draws lines following the mouse as it is dragged about the screen, but, after a short while, only the most recently drawn lines remain.

```
public class Scribble extends WindowController {
  private final static double LIFETIME = 300;

    // first coordinate of a line segment
  private Location lineStarts;

    // save the first coordinate of the line
  public void onMousePress(Location point) {
    lineStarts = point;
  }

    // Draw a line from previous point to current point.
  public void onMouseDrag(Location point) {
    new VanishingLine(LIFETIME, lineStarts, point, canvas);
    lineStarts = point;
  }
}
```

All that it is missing from the VanishLine class definition on the next page are the declarations of a few names. In particular, while you can see that the names:

pauseTime
timeToLive
darkness
myLine

are used in the program, they are never declared. Please complete the program by adding the missing declarations.

```
public class VanishingLine extends ActiveObject {
  private static final int WHITE = 255;
  private static final int BRIGHTENING = 80;




  public VanishingLine(
                      Location start,
                      Location end,
                      DrawingCanvas canvas)
  {


    myLine = new Line( start, end, canvas);
    pauseTime = timeToLive/6;
    start();
  }

  public void run(                          ) {


    pause(2*pauseTime);

    darkness = 0;

    while ( darkness < WHITE ) {
      myLine.setColor( new Color( darkness, darkness,
                                                 darkness ) );
      darkness = darkness + BRIGHTENING;
      pause( pauseTime );
    }

    myLine.removeFromCanvas();
  }
}
```

6. Short questions:
   a. Please rewrite the following method so that it does the same thing but uses better style (and is more compact):

```
public boolean sameNums(int x, int y) {
  if (x == y) {
    return true;
  } else {
    return false;
  }
}
```

   b. Convert the following while-loop into an equivalent for-loop:

```
int counter = 12;
while (counter < 93) {
  System.out.println("counter is: "+counter);
  counter++;
}
```

c. Draw a picture of what will be displayed if the code below is executed.

```
int count1 = 10;
while (count1 < 40) {
  int count2 = 0;
  while (count2 < 20) {
    new FramedRect(count1,count1+count2, 5, 5, canvas);
    count2 = count2 + 10;
  }
  count1 = count1 + count2;
}
```

d. Simplify the following code in which `continuing` is a boolean variable:

```
if (continuing) {

} else {
  System.out.println("Still working")
}
```