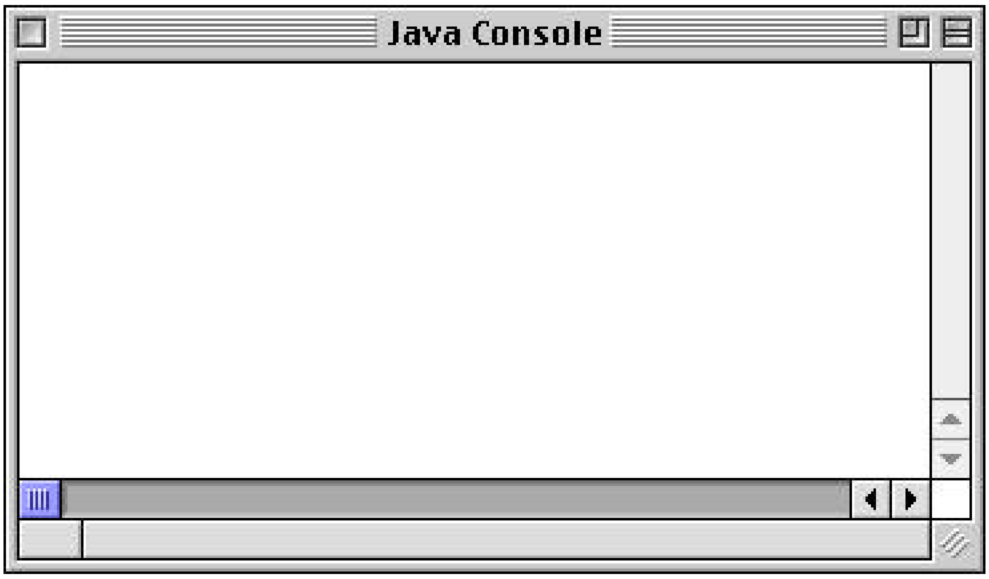
SURNAME: .... FORENAMES: ....

Question 1 (5 marks)

What is the output of the following Java application?

```
public class Q1 {

     public static void main(String[] args) {

          int i;
          double d;

          i = 6 % 10;
          System.out.println(i);

          i = 10 / 6;
          System.out.println(i);

          d = (int)(2.5 * 2.0);
          System.out.println(d);

          d = ((int)2.5 * (int)2.0);
          System.out.println(d);

          System.out.println(d + i);
     }
}
```

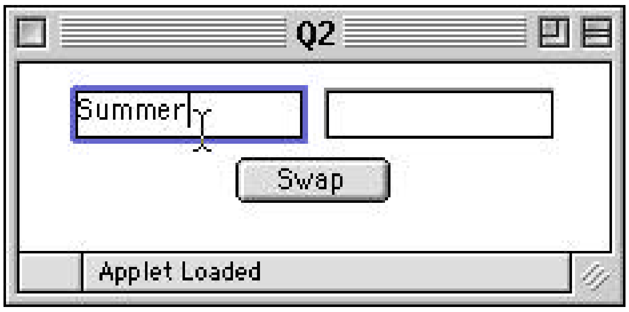


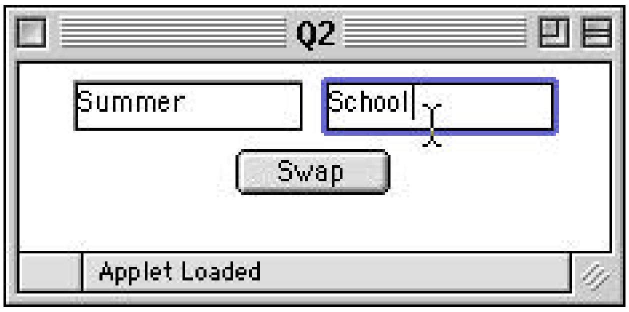
CONTINUED

SURNAME: .... FORENAMES: ....

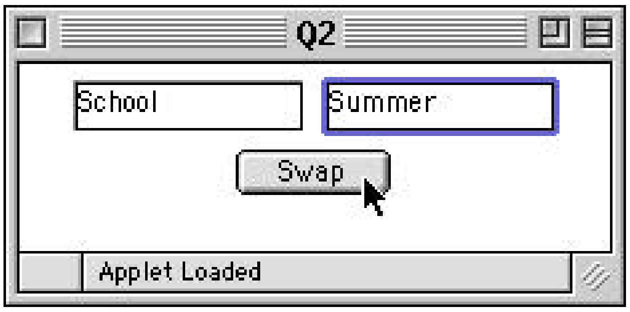
Question 2 (5 marks)

The applet below contains two `TextFields` and a `Button`. When the user presses the `Button` labelled "`Swap`", any text that has been typed into the `TextFields` is swapped from one `TextField` to the other. For example, in the screen shots below, the user has typed the words "`Summer`" and "`School`" into the `TextFields`.





As soon as the "`Swap`" button is pressed, the contents of the left `TextField` is swapped with the contents of the right `TextField`. The screen shot below illustrates this:



For this question, you need to complete the `actionPerformed()` method, which should swap the contents of the two `TextFields`.

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Q2 extends Applet implements ActionListener {

     TextField tOne, tTwo;
     Button bSwap;

     public void init() {
          tOne = new TextField(10);
          tTwo = new TextField(10);
          bSwap = new Button("Swap");
          bSwap.addActionListener(this);
          add(tOne);
          add(tTwo);
          add(bSwap);
     }
```

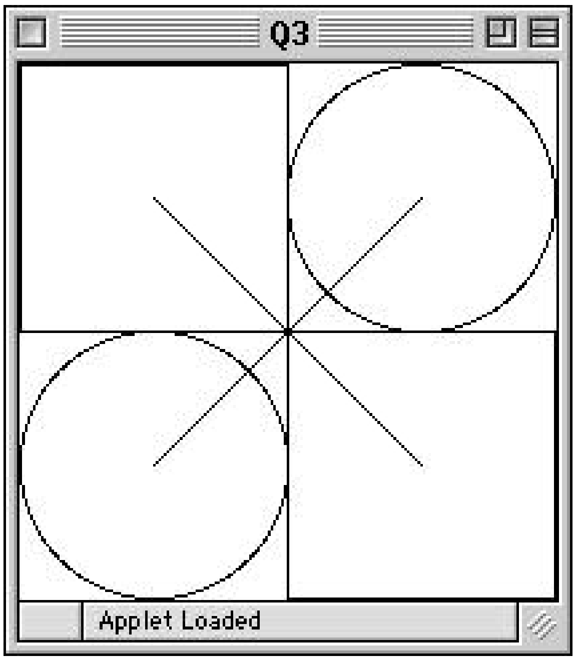
CONTINUED

SURNAME: .... FORENAMES: ....

```
    public void actionPerformed(ActionEvent e) {



    }

}
```

Question 3 (10 marks)

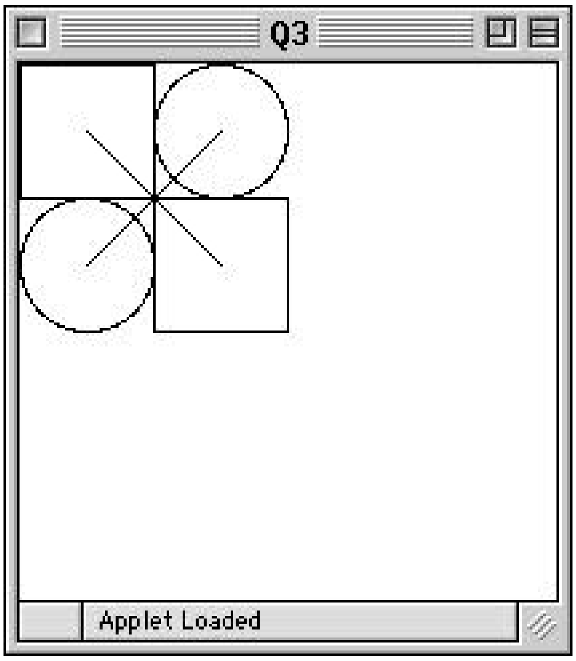
Complete the `paint()` method for the applet which produces the drawing below. All of the parameters to the drawing methods should be based on the constant `SIZE`, so that the picture can conveniently be scaled. In the screen shot below, the value of `SIZE` is 50. The applet window is 200 pixels wide and 200 pixels high:



CONTINUED

SURNAME: .... FORENAMES: ....

If the value of `SIZE` is changed to 25, the drawing should appear as below, where again the applet window is 200 pixels wide and 200 pixels high:



```
import java.awt.*;
import java.applet.*;
public class Q3 extends Applet {
     public void paint(Graphics g) {
          final int SIZE = 50;




     }
}
```

CONTINUED

SURNAME: .... FORENAMES: ....

Question 4 (15 marks)

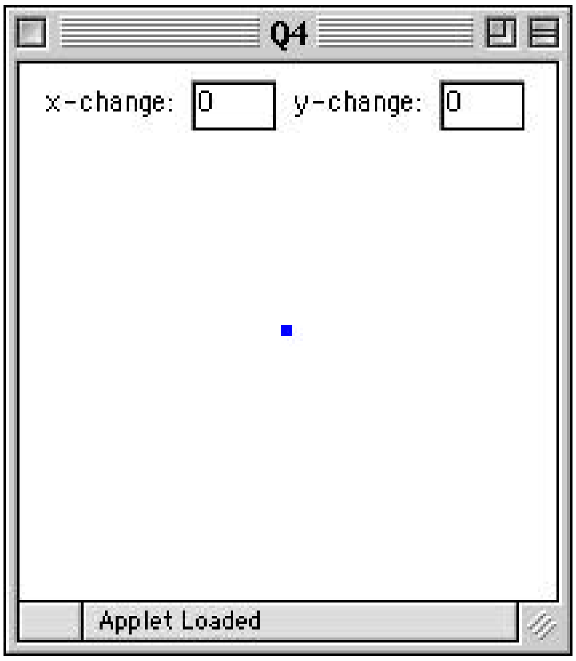
Complete the `paint()` and the `mousePressed()` methods (in the spaces provided) for the applet which executes as follows:

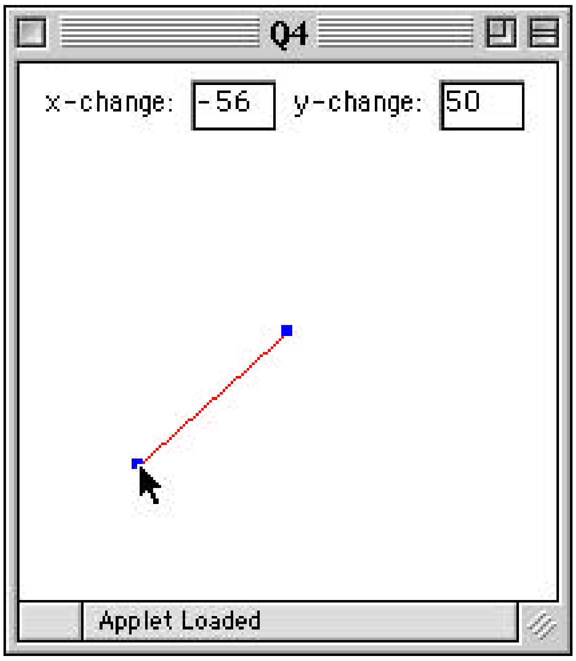
- Whenever the user presses the mouse, a line is drawn from the position of the previous mouse press to the position of the current mouse press. This applet only draws the line between the last two mouse presses.
- Two textfields show the change in the x and y directions between the current position of the mouse press and the previous position of the mouse press.

The screen shots below show how the applet should behave.

The screen shot on the left shows what the screen should look like when the applet first starts. The TextFields both initially display a 0, and a small blue square is drawn in the middle of the applet. The size of this square is given by the constant `SIZE` which is declared in the `paint()` method.

In the screen shot on the right below, the user has just pressed the mouse. A blue square is drawn at the location of the mouse press, and a red line is drawn from this new position to the original blue square. The TextFields display the values -56 and 50 because the position the mouse was pressed was 56 pixels to the left of and 50 pixels below the original blue square.



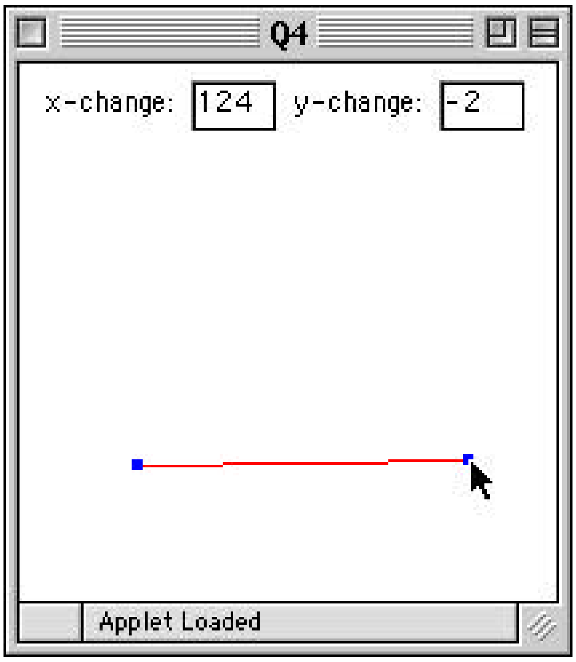


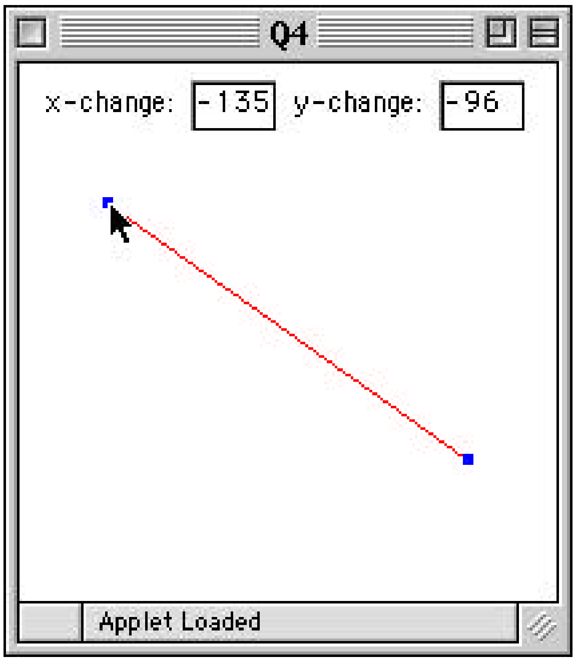
In the screen shots on the next page, the user presses the mouse two more times. In the screen shot on the left, the mouse is pressed 124 pixels to the right of and 2 pixels above the location of the previous mouse press. Notice that these values are displayed in the TextFields, and the red line is drawn between the locations of the latest two mouse presses.

CONTINUED

SURNAME: .... FORENAMES: ....

In the screenshot on the right below, the user has pressed the mouse near the top left corner of the applet. The line is now drawn between this location and the location of the previous mouse press, and the TextFields are updated appropriately.





You need to complete the `paint()` and `mousePressed()` methods for this applet:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Q4 extends Applet implements MouseListener {

     // The textfields
     private TextField tXChange, tYChange;

     // These variables store the locations of the mouse presses
     private int prevX, prevY, currentX, currentY;

     public void init() {

          tXChange = new TextField("0   ");
          tYChange = new TextField("0   ");

          add(new Label("x-change:"));
          add(tXChange);
          add(new Label("y-change:"));
          add(tYChange);

          // Initially, the boxes are drawn at position (100, 100):
          prevX = 100;
          prevY = 100;
          currentX = 100;
          currentY = 100;

          addMouseListener(this);
     }
```

CONTINUED

SURNAME: .... FORENAMES: ....

```
    public void paint(Graphics g) {
        // This constant give the size of the blue boxes
        final int SIZE = 4;

        // Draw the red line between the locations of the previous
        // and current mouse presses








        // Draw the blue boxes at the location of the previous
        // and the current mouse press











    }

    public void mousePressed(MouseEvent e) {





















    }
    public void mouseReleased(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
}
```

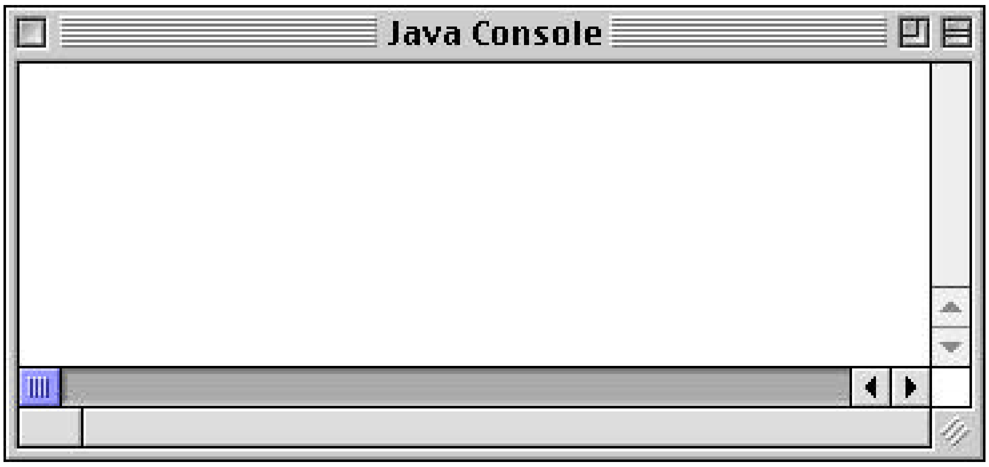
CONTINUED

SURNAME: .... FORENAMES: ....

Question 5 (10 marks)

What is the output of the following Java application?

```
public class Q5 {

    public static void main(String[] args) {

        boolean isValid = true;
        int value = 45;

        if (isValid && (value < 50)) {
            isValid = !isValid;
            value = value * 2;
        }

        else {
            isValid = true;
            value = value / 2;
        }

        System.out.println("1. value: "+ value);

        if (isValid) {
            value = value / 2;
            if (value % 2 == 0)
                value = value + 10;
            else
                value = value - 10;
            System.out.println("isValid: " + isValid);
        }

        else {
            value = value + 3;
            if (value % 2 == 0)
                value = value + 10;
            else
                value = value - 10;
            System.out.println("isValid: " + isValid);
        }

        System.out.println("2. value: " + value);

    }
}
```



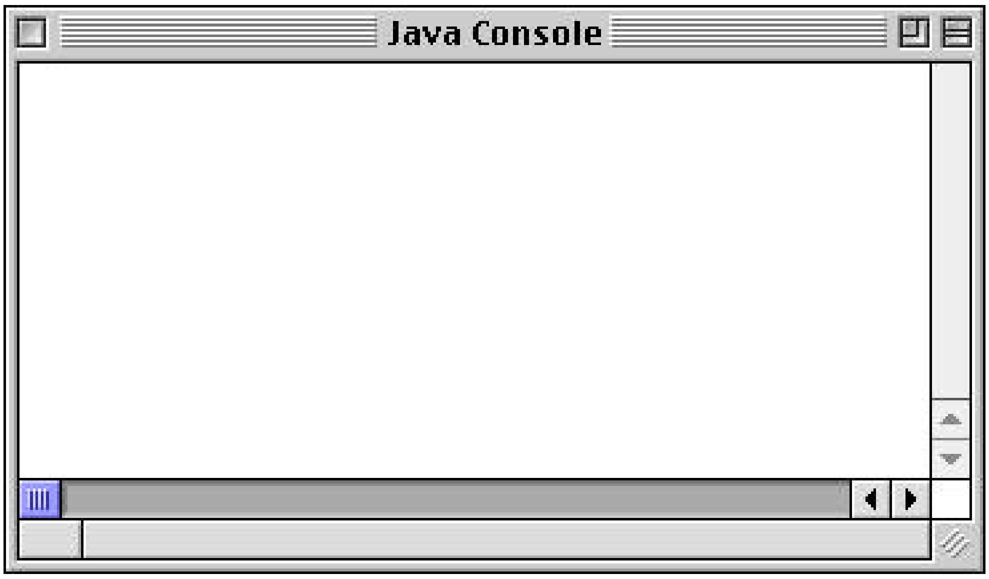
CONTINUED

SURNAME: .... FORENAMES: ....

Question 6 (10 marks)

(i) What is the output of the following application? (5 marks)

```
public class Q6 {

     public static void main(String[] args) {

          int i = 10;
          int j = 0;
          int value = 1;

          while (i < 17) {
               j = 15;
               value = value + 3;
               if (value < j)
                    System.out.println("value: "+ value);
               else
                    value = value +  1;
               i++;
               j++;
          }

          System.out.println("value: "+value);
     }
}
```



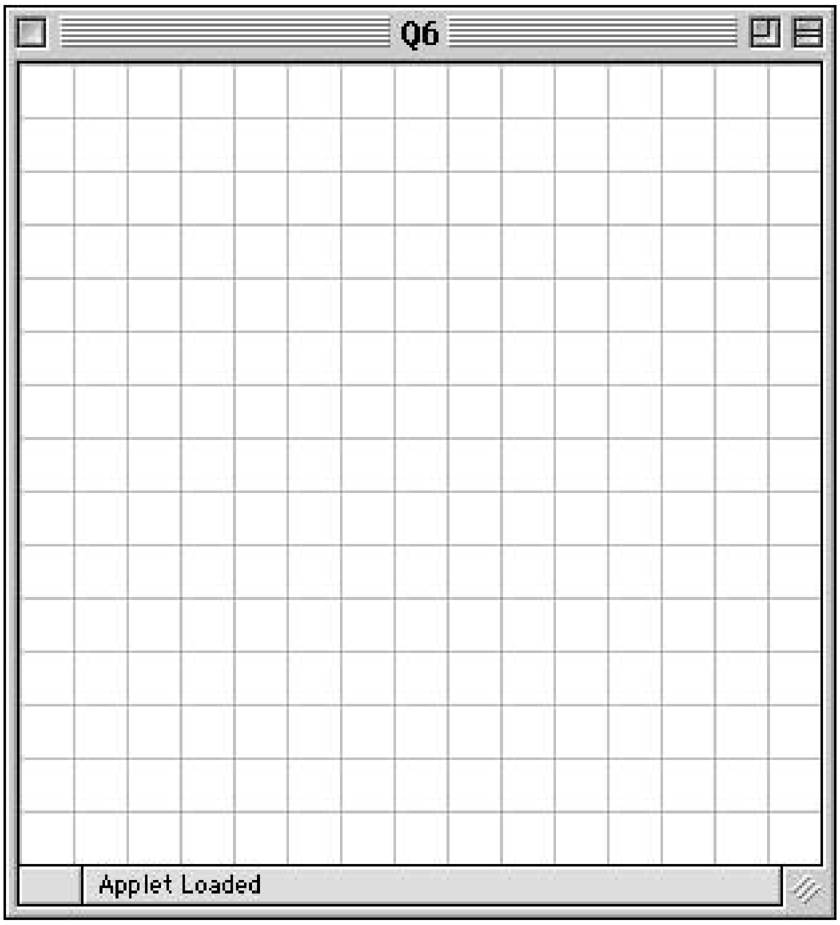
CONTINUED

SURNAME: .... FORENAMES: ....

(ii) What is the output when the following `paint()` method is executed? (5 marks)

```
public void paint(Graphics g) {

     final int SIZE = 20;

     g.setColor(Color.black);

     int i = 0;
     int currentX = 10;
     int currentY = 10;

     while (i < 4) {
          g.drawRect(currentX, currentY, SIZE, SIZE);
          currentX += SIZE;
          currentY += SIZE / 2;
          i++;
     }
}
```

Draw the output in the window below. The window is 150 pixels wide and 150 pixels high. To help you, gridlines have been drawn - each small square is 10 pixels wide and 10 pixels high:



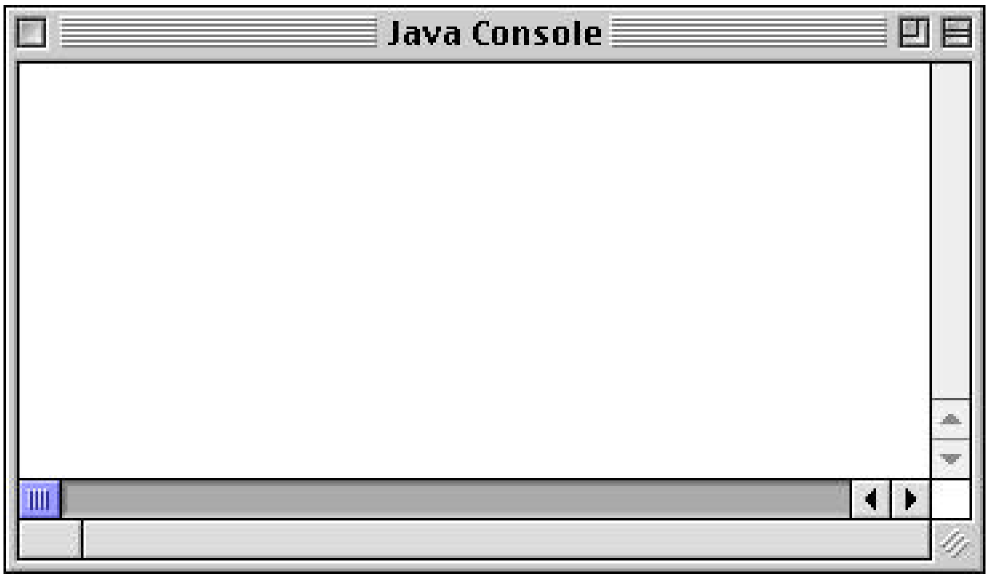
CONTINUED

SURNAME: .... FORENAMES: ....

Question 7 (15 marks)

What is the output of the following applet:

```
import java.awt.*;
import java.applet.*;

public class Q7 extends Applet {

     int one;
     double two;
     String three;

     public void init() {

          one = method1(5, 10);
          three = "3";
          method2(one);

          System.out.println("one: " + one);
          System.out.println("two: " + two);
          System.out.println("three: " + three);
     }

     public int method1(int a, int one) {
          two = a * one;
          System.out.println(one);
          one = (int)(a + two);
          return (one + one);
     }

     public void method2(int x) {
          System.out.println(two);
          two = x + two;
          three = three + (two/4);
     }
}
```



CONTINUED

SURNAME: .... FORENAMES: ....

Question 8 (15 marks)

The source code for an applet which paints a `String` on the screen at the location the user presses the mouse button is given below. This applet makes use of a class called `DrawableString`.

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Q8 extends Applet implements ActionListener, MouseListener {

     DrawableString myWord;
     TextField tInput;

     public void init() {
          myWord = new DrawableString("Hello", 100, 100);
          tInput = new TextField(10);
          add(tInput);
          tInput.addActionListener(this);
          addMouseListener(this);
     }

     public void actionPerformed(ActionEvent e) {
          myWord.setText(tInput.getText());
          repaint();
     }

     public void mousePressed(MouseEvent e) {
          myWord.setPos(e.getX(), e.getY());
          repaint();
     }

     public void paint(Graphics g) {
          myWord.draw(g);
     }

     public void mouseReleased(MouseEvent e) {}
     public void mouseEntered(MouseEvent e) {}
     public void mouseExited(MouseEvent e) {}
     public void mouseClicked(MouseEvent e) {}

}
```

For this question you need to write the `DrawableString` class. As you can see from the source code above, in your `DrawableString` class you will need to define a constructor method, a `setText()` method, a `setPos()` method and a `draw()` method.

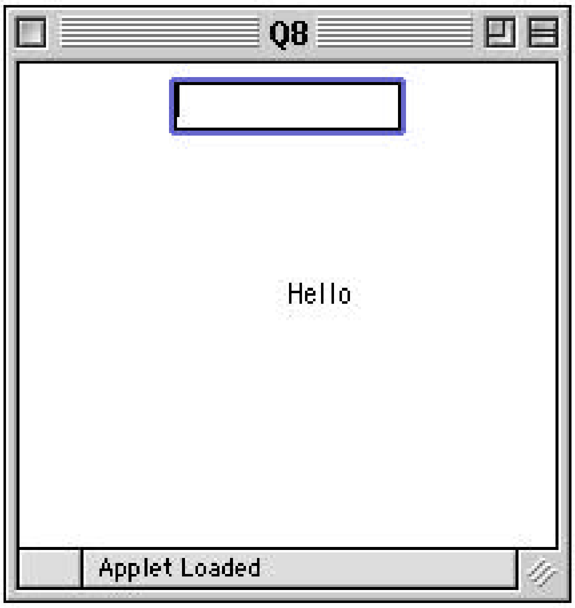
A detailed description of how the applet behaves is given on the next page.

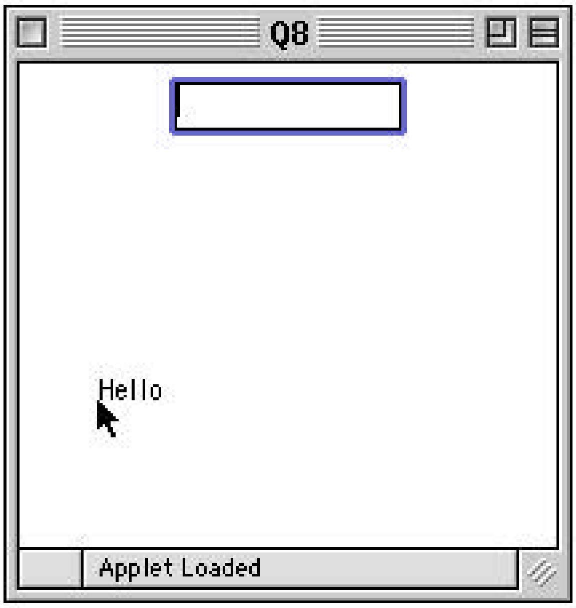
CONTINUED

SURNAME: .... FORENAMES: ....

When the applet first starts, the TextField is empty and the word "`Hello`" appears at location (100, 100), as shown in the screen shot on the left. In the screen shot on the right, the user has just pressed the mouse button, and the position of the `String` has moved to where the mouse was pressed.

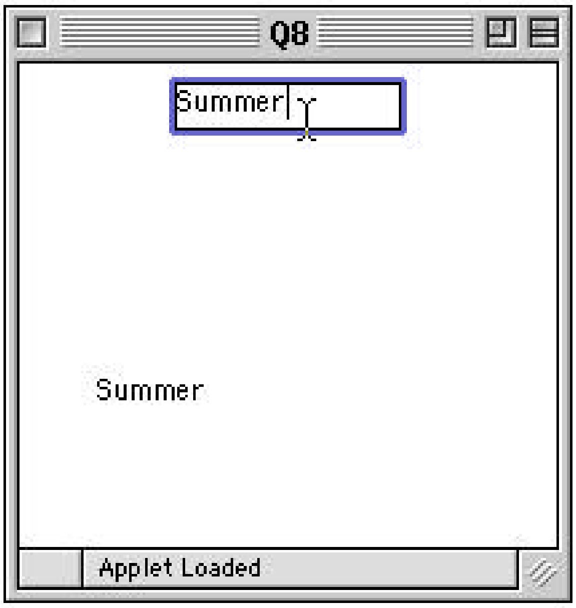
In the screen shots below, the applet window is 200 pixels wide and 200 pixels high.

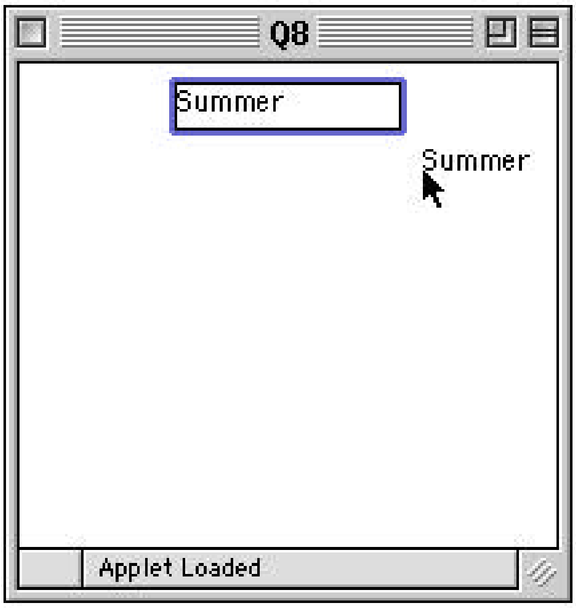




The `String` which is drawn on the applet is an instance of a class called `DrawableString`. You need to write the code for this `DrawableString` class.

The user can change the text of the `DrawableString` object by entering new text into the `TextField`. The screen shot on the left below shows the applet after the user has pressed return in the `TextField`. The text of the `DrawableString` object changes straight away. The object can still be moved around in the same way as before by pressing the mouse button, as shown in the screen shot on the right.





The instance variables for the `DrawableString` class have been declared for you. Complete the the rest of this class over the page.

CONTINUED

SURNAME: .... FORENAMES: ....

```
import java.awt.*;
public class DrawableString {
     private String text;
     private int xPos, yPos;



}
```

CONTINUED

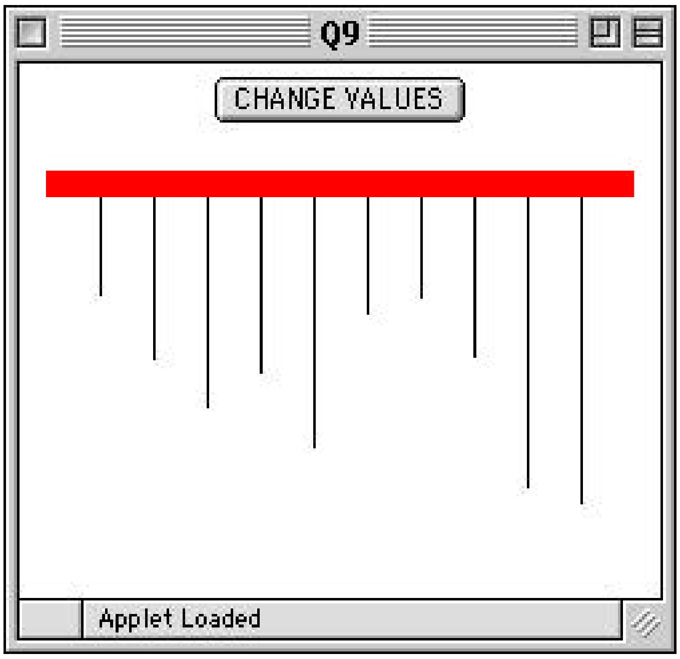
SURNAME: .... FORENAMES: ....

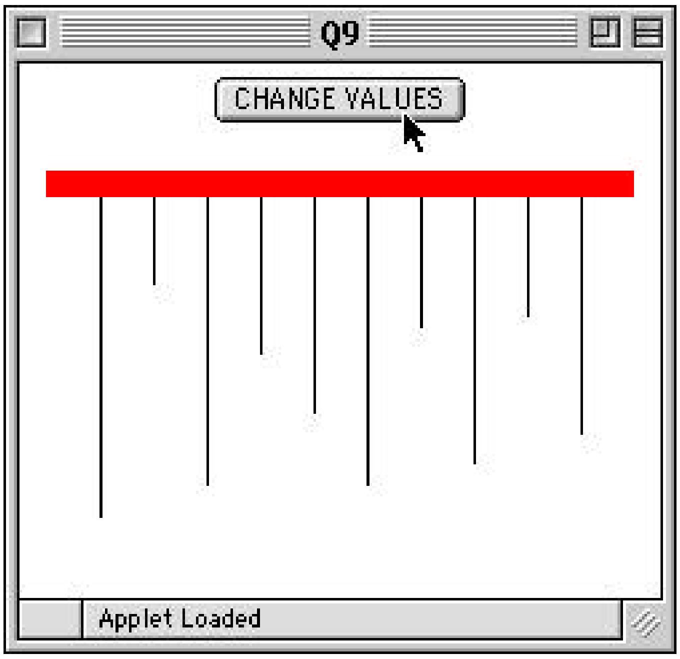
Question 9 (15 marks)

The following applet displays a series of vertical lines hanging from a bar and 20 pixels apart; the length of each line is given by the value stored in an array of `int` values.

For example the array containing the numbers 36, 60, 78, 65, 93, 43, 37, 59, 108, 114 would produce the drawing to the right.

In the drawing, the leftmost vertical line is 36 pixels long, the next line is 60 pixels long, the next line is 78 pixels long and so on.





Each time the user presses the "CHANGE VALUES" button, ten new random values (between 30 and 120) are stored in the array and the length of the lines hanging from the bar reflect the new values stored in the array.

For example in the screen shot to the left, the user has just clicked on the "CHANGE VALUES" button, and the `int`s stored in the array are changed to random numbers between 30 and 120. The length of the lines hanging from the bar change to reflect the new values stored in the array.

Your code should use loops so that if the size of the array `theNums` is changed, the applet will still work.

You need to complete the `paint()` and `actionPerformed()` method of the applet, in the spaces provided:

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Q9 extends Applet implements ActionListener {
     // The array of ints
     private int[] theNums = {36, 60, 78, 65, 93, 43, 37, 59, 108, 114};
     private Button bChangeValues;
```

CONTINUED

SURNAME: .... FORENAMES: ....

```
    public void init() {
         bChangeValues = new Button("CHANGE VALUES");
         bChangeValues.addActionListener(this);
         add(bChangeValues);
    }
    public void actionPerformed( ActionEvent e) {
         // Assign new random values to each element of the array using
         // a loop, and then repaint() the screen













    }
    public void paint(Graphics g) {
         final int BAR_LEFT = 10;      /** left of bar*/
         final int BAR_RIGHT = 230;    /** right of  bar*/
         final int BAR_TOP = 40;       /** top of bar*/
         final int BAR_BOTTOM = 50;    /** bottom of bar*/

         // horizontal step between each line
         final int X_STEP = 20;

         // x position of first line
         final int FIRST_LINE_POS = BAR_LEFT + X_STEP;

         //draw the top bar in red
         g.setColor(Color.red);
         g.fillRect(BAR_LEFT, BAR_TOP, BAR_RIGHT - BAR_LEFT,
                                            BAR_BOTTOM - BAR_TOP);

         // Draw the lines hanging from the bar in black, using a loop
         g.setColor(Color.black);















         }
    }
}
```

CONTINUED