

THE UNIVERSITY OF AUCKLAND

FIRST SEMESTER, 2001
City Campus

COMPUTER SCIENCE

Principles of Programming

(Time allowed: TWO hours)

Surname:	
Forenames:	
Student ID number:	

INSTRUCTIONS:

- Attempt **ALL** questions - write your answers in the space provided
- Marks for each question are shown, and total 100.
- There is extra space on the last page of the booklet if you need it
- There is an appendix at the end of this booklet which lists some useful methods and instance variables
- Calculators are **NOT** permitted

Examiner to complete:

Question	Mark
1	(/10)
2	(/10)
3	(/10)
4	(/5)
5	(/10)
6	(/10)

Question	Mark
7	(/5)
8	(/5)
9	(/10)
10	(/10)
11	(/10)
12	(/5)

TOTAL:

(/100)

CONTINUED

SURNAME: FORENAMES:

Question 1 (10 marks)

- a) Declare a local variable of type `double` called `rate` and assign it the value 3.

(2 marks)

- b) What is the *scope* of an instance variable?

(2 marks)

- c) What does the word “void” mean in the following method header?

```
private void printResults(int a, int b) {
```

(2 marks)

- d) Write a Java statement (or statements) to print out the *length* of the `String` stored at element 0 of a `Vector` called `v`.

(2 marks)

- e) Rewrite the following “while” loop as a “for” loop:

```
int i = 4;
while (i < lastHouseNo) {
    System.out.println("House No." + i);
    i = i + 2;
}
```

(2 marks)

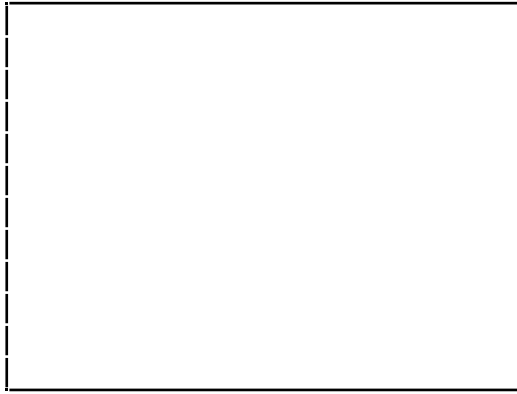
CONTINUED

SURNAME: FORENAMES:

Question 2 (10 marks)

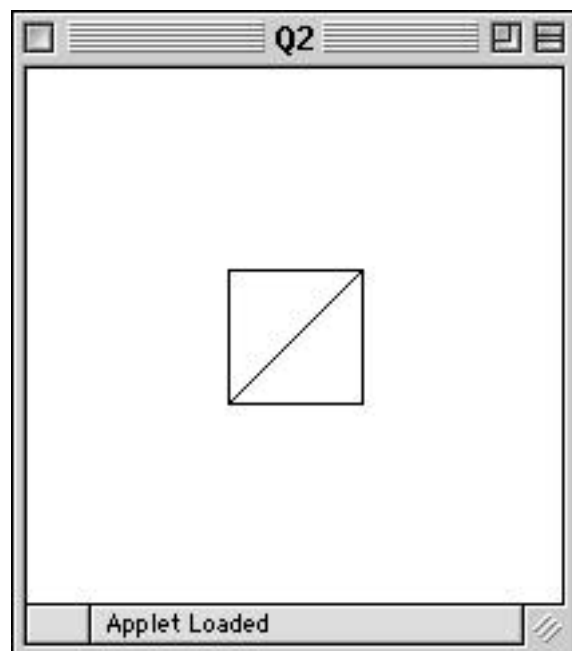
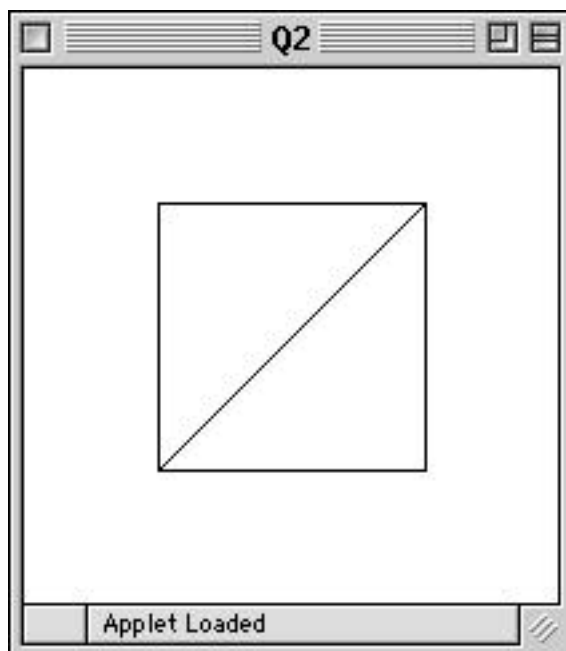
You need to complete the `paint()` method of the following applet:

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

public class Q2 extends Applet {
    public void paint(Graphics g) {
        final int SIZE = 50;
        
    }
}
```

(10 marks)

All of the parameters to the drawing methods you call in `paint()` must be based on the constant `SIZE`. When the value of `SIZE` is 50 the applet should produce the drawing on the left below, where the width of the square that is drawn is exactly half the width of the applet window. The applet window is exactly 200 pixels wide and 200 pixels high:



The square that is drawn must always be centred at exactly the middle of the applet window, and must have a line drawn from its bottom left to its top right corner.

The constant `SIZE` can be used to scale the drawing. For example, if the value of `SIZE` is changed to 25, the drawing on the right above should be produced. In this drawing, the width of the square is exactly a quarter the width of the applet window.

CONTINUED

SURNAME: FORENAMES:

Question 3 (10 marks)

We wish to provide a commentary to a user based on how well they are progressing in a game of finding a hidden spot. To do this we track the user's progress for their last guess as well as for their current guess. We define three levels of progress: SAME; CLOSER; and FURTHER (represented by integer constants).

Then, based on their progress we will give them a comment.

The method `progressMessage()`, detailed below, determines what message will be returned to the user. The parameters `lastGuess` and `thisGuess` will only hold values represented by the integer constants SAME, CLOSER, and FURTHER.

```
final int SAME = 1;
final int CLOSER = 2;
final int FURTHER = 3;

final String START1 = "No improvement to ";
final String START2 = "From good to ";
final String START3 = "Bad luck to ";
final String END1 = "no gain.";
final String END2 = "incredible.";
final String END3 = "stink.";

public String progressMessage (int lastGuess, int thisGuess) {
    if (lastGuess == SAME)
        if (thisGuess == SAME)
            return START1 + END1;
        else if (thisGuess != FURTHER)
            return START1 + END2;
        else
            return START1 + END3;
    else if (lastGuess == CLOSER)
        if (thisGuess < CLOSER)
            return START2 + END1;
        else if (thisGuess <= CLOSER)
            return START2 + END2;
        else
            return START2 + END3;
    else
        if (thisGuess >= FURTHER)
            return START3 + END3;
        else if (thisGuess != SAME)
            return START3 + END2;
        else
            return START3 + END1;
}
```

After you have looked at this method, answer the questions on the next page...

SURNAME: FORENAMES:

What will be printed by the following statements which use this method:

a) `System.out.println(progressMessage(SAME, SAME));`

(2 marks)

b) `System.out.println(progressMessage(SAME, CLOSER));`

(2 marks)

c) `System.out.println(progressMessage(FURTHER, FURTHER));`

(2 marks)

d) `System.out.println(progressMessage(FURTHER, SAME));`

(2 marks)

e) `System.out.println(progressMessage(CLOSER, SAME));`

(2 marks)

SURNAME: FORENAMES:

Question 4 (5 marks)

What is the output of this applet? The program compiles and runs without error.

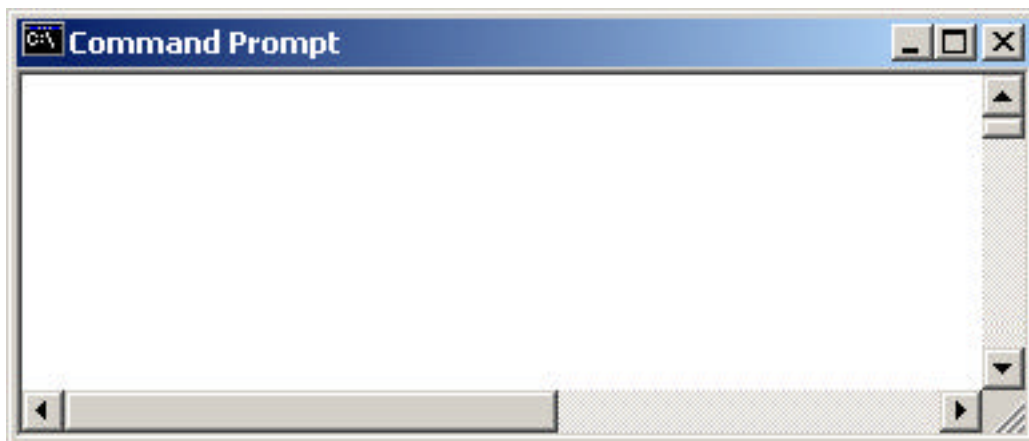
```
import java.awt.*;
import java.applet.*;

public class Q4 extends Applet{
    int i = 100;
    public void init() {
        System.out.println(whichOne());
    }

    private boolean flip(boolean in){
        System.out.println(in);
        return !in;
    }

    private boolean testIt(int value){
        boolean out;
        switch (value) {
            case 1: case 2: case 3: case 4:
            case 5:
                out = true;
                System.out.println(out);
            default:
                out = false;
                System.out.println(out);
                return out;
        }
    }

    private int whichOne(){
        int i = 10;
        if (flip(!testIt(5))){
            System.out.println(i);
            i--;
            return i%2;
        }
        else {
            System.out.println(i);
            i++;
            return i*i;
        }
    }
}
```



(5 marks)

CONTINUED

SURNAME: FORENAMES:

Question 5 (10 marks)

Complete the method `leftMostXValue()` which is passed an array of `Point` objects as a parameter, and returns the smallest x coordinate value, in other words, the left most position of any `Point` in the array. You can assume that every element of the array refers to a valid `Point` object, and that the size of the array is at least one.

```
private int leftMostXValue(Point[] pts) {
```

(10 marks)

For example, the output of the following applet:

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

public class Q5 extends Applet {
    public void init() {
        Point[] ps = new Point[5];
        ps[0] = new Point(100, 5);
        ps[1] = new Point(20, 100);
        ps[2] = new Point(140, 200);
        ps[3] = new Point(70, 100);
        ps[4] = new Point(25, 0);

        int leftX = leftMostXValue(ps);
        System.out.println("Smallest x value: " + leftX);
    }

    private int leftMostXValue(Point[] pts) {
        ....
    }
}
```

would be:

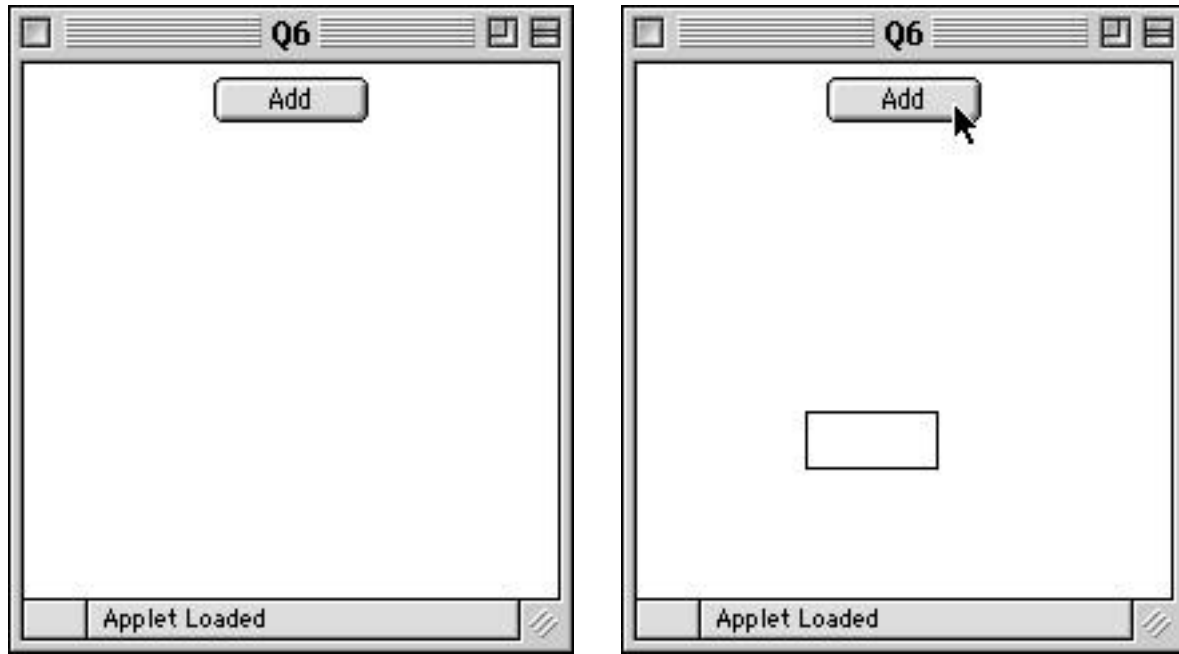
Smallest x value: 20

CONTINUED

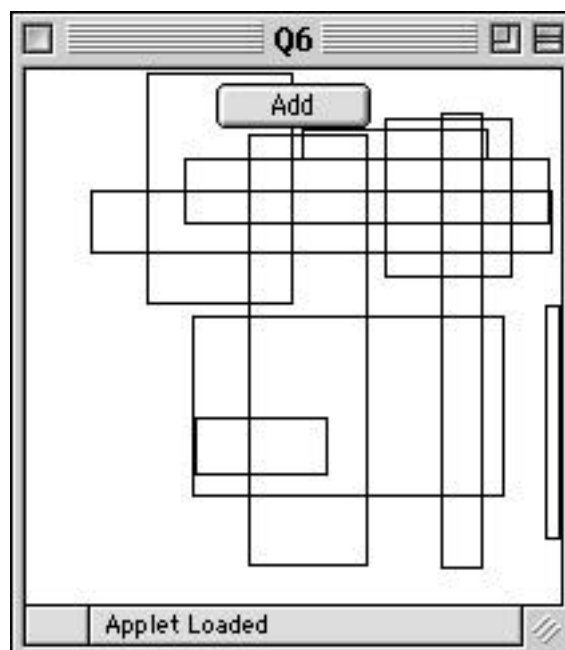
SURNAME: FORENAMES:

Question 6 (10 marks)

You need to complete the code for the following applet, which uses a Vector to store a collection of Rectangle objects. When the applet first starts, a Button appears with the label "Add", as shown in the screen shot on the left below. As soon as the user presses the Button, a rectangle appears at a random location on the applet, as shown in the screen shot on the right below:



Every time the "Add" button is pressed, another rectangle is created and drawn to the applet. The screen shot below shows the applet after the "Add" button has been pressed a further 9 times:



Complete the source code on the following page for the applet as described above. A method, `randomRect()`, has been written for you that you can use to create `Rectangle` objects with random locations and sizes. The applet window is 200 pixels wide and 200 pixels high.

CONTINUED

SURNAME: FORENAMES:

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

public class Q6 extends Applet implements ActionListener {
    Vector rects;
    Button bAdd;

    private Rectangle randomRect() {
        final int APPLET_SIZE = 200;
        int randX, randY, randWidth, randHeight;
        randX = (int)(Math.random() * (APPLET_SIZE - 5));
        randY = (int)(Math.random() * (APPLET_SIZE - 5));
        randWidth = (int)(Math.random()*(APPLET_SIZE-randX-5)) + 5;
        randHeight = (int)(Math.random()*(APPLET_SIZE-randY-5))+5;
        return new Rectangle(randX, randY, randWidth, randHeight);
    }

    public void init() {
        
    }

    public void actionPerformed(ActionEvent e) {
        
    }

    public void paint(Graphics g) {
        
    }
}
```

(10 marks)

CONTINUED

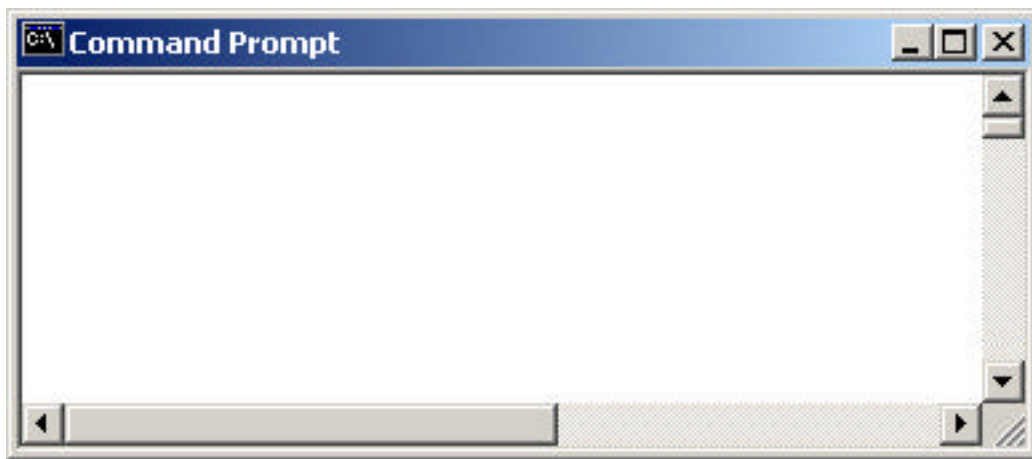
SURNAME: FORENAMES:

Question 7 (5 marks)

What is the output of these Java code segments? They compile and run without error.

a)

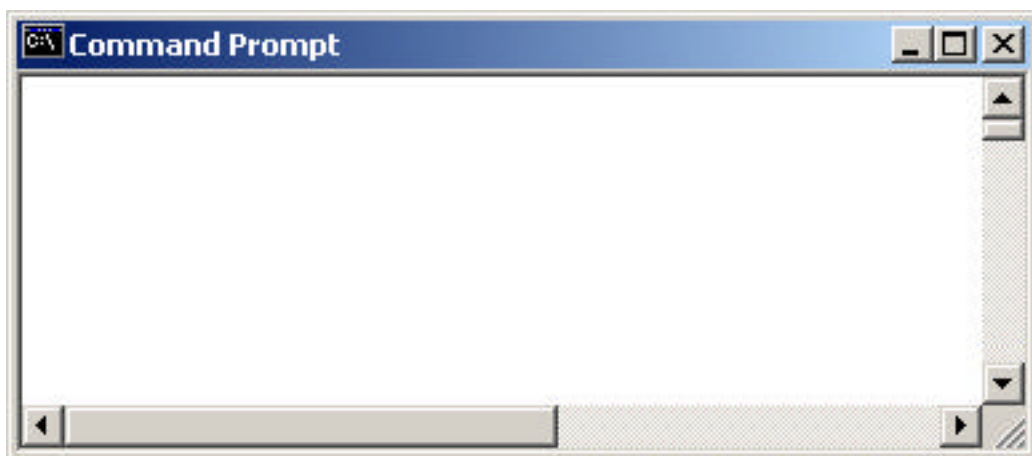
```
int value = 0;
for (int i = 1; i <= 8; i++)
    for (int j = 0; j <= 10; j += 2)
        value++;
System.out.println(value);
```



(2 marks)

b)

```
double value = 0.0;
for (int j = 0; j < 10; j++)
    for (int i = j; i < 10; i++)
        value += 1.0;
System.out.println(value);
```



(3 marks)

SURNAME: FORENAMES:

Question 8 (5 marks)

What is the output of this applet?

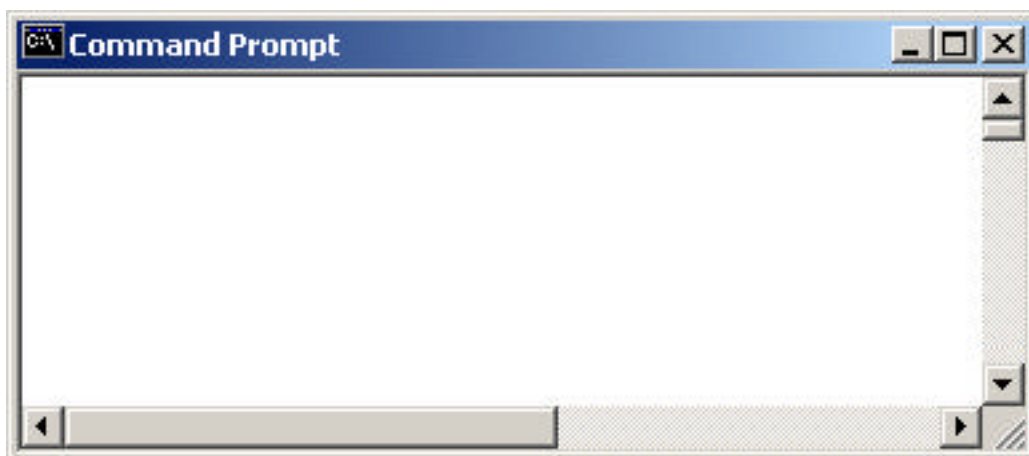
```
import java.applet.*;
import java.util.*;

public class Q8 extends Applet {

    Vector v;

    public void init() {
        String a, b, c;
        v = new Vector();
        a = new String("hello");
        b = "";
        c = a;
        v.addElement(a);
        v.addElement(new String("X"));
        v.addElement(b);
        v.addElement(new String("Y"));
        v.addElement(c);

        a = (String)v.elementAt(1);
        b = (String)v.elementAt(3);
        System.out.println(b.length());
        v.removeElement(a);
        System.out.println(b);
        b = (String)v.elementAt(3);
        System.out.println(b.length());
        System.out.println(a);
        System.out.println(c);
    }
}
```

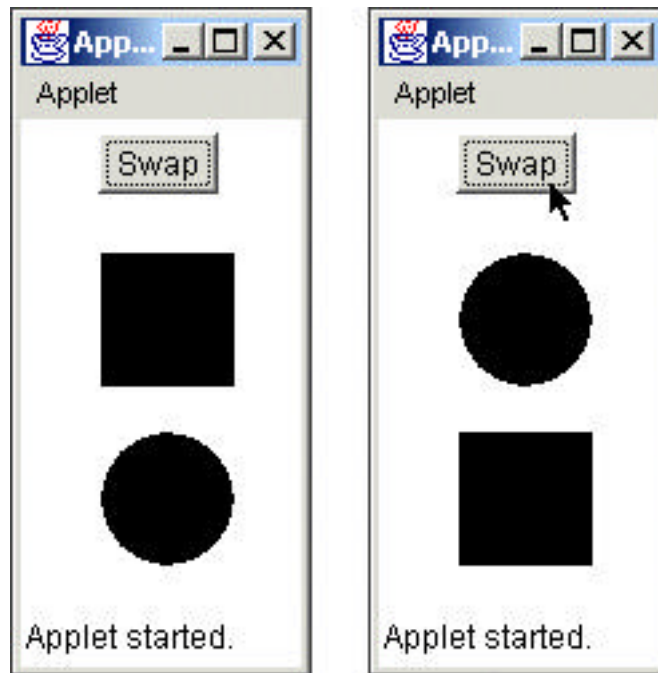


(5 marks)

SURNAME: FORENAMES:

Question 9 (10 marks)

The following applet uses a `Thing` class. It creates two `Thing` objects and draws them. Whenever the user clicks the button labelled "Swap", the bottom shape moves to the position of the top shape, and vice versa, as shown in the screen shots below. The screen shot on the left shows what the applet looks like when it initially starts, and the screen shot on the right shows what happens when "Swap" is clicked. You need to complete the code for the `Thing` class.



The source code for the applet class is given below:

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

public class Q9 extends Applet implements ActionListener {
    private Button bSwap;
    private Thing shape1, shape2;

    public void init() {
        bSwap = new Button("Swap");
        add(bSwap);
        shape1 = new Thing(30, 80, 50, Thing.SQUARE);
        shape2 = new Thing(30, 150, 50, Thing.CIRCLE);
        bSwap.addActionListener(this);
    }

    public void paint(Graphics g) {
        shape1.draw(g);
        shape2.draw(g);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == bSwap)
            shape1.swapPosition(shape2);
        repaint();
    }
}
```

CONTINUED

SURNAME: FORENAMES:

Complete the Thing class below. You are required to write the parameters for the constructor and methods, write the return type of the methods, and complete the constructor and methods.

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

```
public class Thing {
```

```
    public static final int SQUARE = 0;
    public static final int CIRCLE = 1;
```

```
    private int posX;    // x position of the centre of the shape
    private int posY;    // y position of the centre of the shape
    private int size;
    private int shape;
```

```
    public Thing(_____ ) {
```

```
    }
```

```
    public _____ swapPosition(_____ ) {
```

```
    }
```

```
    public _____ draw(Graphics g) {
```

```
    }
```

```
}
```

(10 marks)

CONTINUED

SURNAME: FORENAMES:

Question 10 (10 marks)

In an online supermarket shopping system the user is able to type in their requirements as a list of either: quantity and item pairs; or weight and item pairs. A typical list might look like the one below:

```
1 Loaf of white bread
3 Tins of tomatoes
5kg Rua potatoes
500g Mushrooms
```

You need to complete the `splitShoppingLine()` method below to split a single line into its quantity (or weight) and the item description. Note that the *first* space character on a line is the divider between a quantity (or weight) and the item description. There will only ever be one space dividing these two portions of the string. You should ensure that both sub-strings are converted to upper case when they are split up.

The two instance variables (`quantityOrWeight` and `itemDescription`) should contain the extracted sub-strings when the method completes. Between 3 and 5 lines of code are sufficient to answer this question.

```
public class ShoppingItem {
    private String quantityOrWeight;
    private String itemDescription;
    public ShoppingItem(String shoppingLine) {
        splitShoppingLine(shoppingLine);
    }
    public void splitShoppingLine(String shoppingLine) {
        int spacePosition;
        
    }
    public void printItem() {
        System.out.println(quantityOrWeight + " - " + itemDescription);
    }
}
```

(10 marks)

An example of the use of this class is shown in the code below.

```
ShoppingItem item1 = new ShoppingItem("1 Loaf of white bread");
ShoppingItem item2 = new ShoppingItem("5kg Rua potatoes");
item1.printItem();
item2.printItem();
```

In the example above the following output would be produced:

```
1 - LOAF OF WHITE BREAD
5KG - RUA POTATOES
```

CONTINUED

SURNAME: FORENAMES:

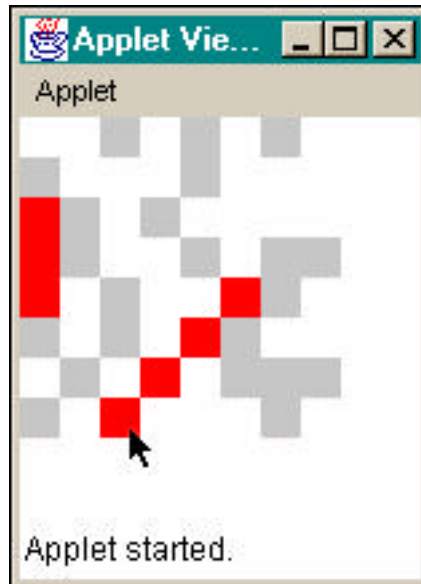
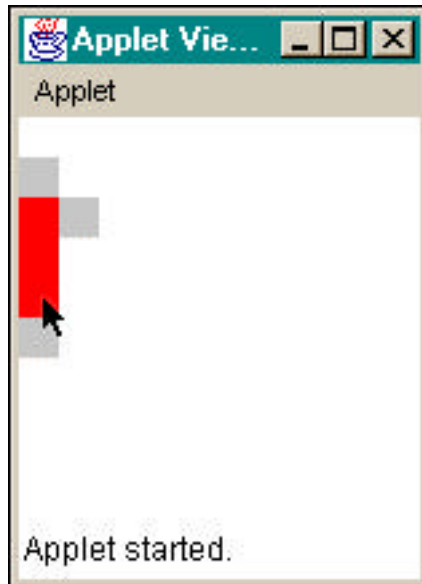
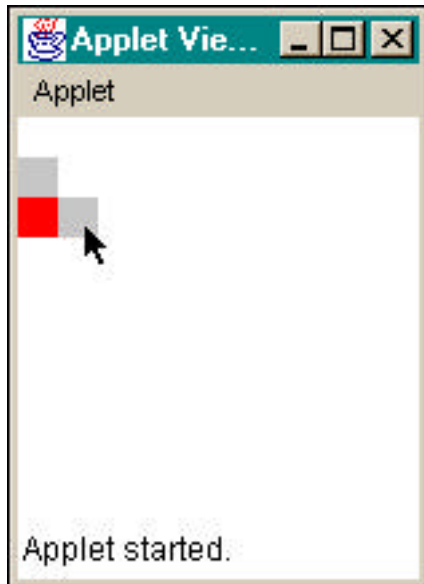
Question 11 (10 marks)

The applet below provides a mouse driven interface for the battleships game. In this game the user selects a point on the screen which will either be a hidden ship, or the sea. The locations of the ships are arranged in an 8x8 grid. There is an array called `layout` which stores the position of the hidden ships, and a second array called `shots` which records whether a shot was fired into a particular grid position.

When the game first starts, the applet is blank as in the screen shot to the right.

Whenever the mouse is pressed, the corresponding grid location is drawn in either red or blue, depending on whether that grid location corresponds to a ship or the sea.

An example of the game in play is shown in the following series of screenshots, where the darker rectangles (red) represent a hit ship and the lighter rectangles (blue) are missed shots into the sea:



You must complete the `paint()` method for this applet. You will need to cycle through all of the elements in the `shots` array. For each element that is `true`, which indicates a shot has been fired, you need to draw a rectangle in the corresponding grid location. The colour of the rectangle should be red if the corresponding element of the `layout` array represents a ship, or blue if the corresponding element of the `layout` array represents the sea.

The `GRID_SIZE` constant determines how wide and high the grid rectangle should be drawn. Take care with X (for columns) and Y (for rows) positions in respect to the `shots` array.

SURNAME: FORENAMES:

```

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

public class Battleship extends Applet implements MouseListener {

    static final int GRID_SIZE = 15; // Size that grids are drawn
    static final int X_DIMENSION = 8; // X dimension for the arrays
    static final int Y_DIMENSION = 8; // Y dimension for the arrays
    static final int SEA = 0; // Value for a sea position
    static final int SHIP = 1; // Value for a ship position

    // Array containing positions of ships and sea
    static final int[][] layout = {
        {SEA, SEA, SEA, SEA, SEA, SEA, SEA, SEA},
        {SEA, SEA, SEA, SEA, SEA, SEA, SHIP, SHIP},
        {SHIP, SEA, SEA, SEA, SEA, SEA, SEA, SEA},
        {SHIP, SEA, SEA, SEA, SEA, SEA, SEA, SEA},
        {SHIP, SEA, SEA, SEA, SEA, SHIP, SEA, SEA},
        {SEA, SEA, SEA, SEA, SHIP, SEA, SEA, SEA},
        {SEA, SEA, SEA, SHIP, SEA, SEA, SEA, SEA},
        {SEA, SEA, SHIP, SEA, SEA, SEA, SEA, SEA}
    };

    // Array to record if a shot has been made in a grid position
    private boolean[][] shots=new boolean[Y_DIMENSION][X_DIMENSION];

    // Initialise the shots array to false (no shots)
    public void init() {
        for (int i = 0; i < shots.length; i++)
            for (int j = 0; j < shots[i].length; j++)
                shots[i][j] = false;
        addMouseListener(this);
    }

    /**
     Gets the mouse position
     If it is inside the playing area then record a shot by setting
     the corresponding element in the shots array to true, and
     repaint the screen
     @param e used to find location of mouse press
    */
    public void mousePressed(MouseEvent e) {
        int mouseX = e.getX();
        int mouseY = e.getY();
        if ( (mouseX < X_DIMENSION * GRID_SIZE) &&
            (mouseY < Y_DIMENSION * GRID_SIZE) )
            shots[mouseY/GRID_SIZE][mouseX/GRID_SIZE] = true;
        repaint();
    }
}

```

CONTINUED

SURNAME: FORENAMES:

/**

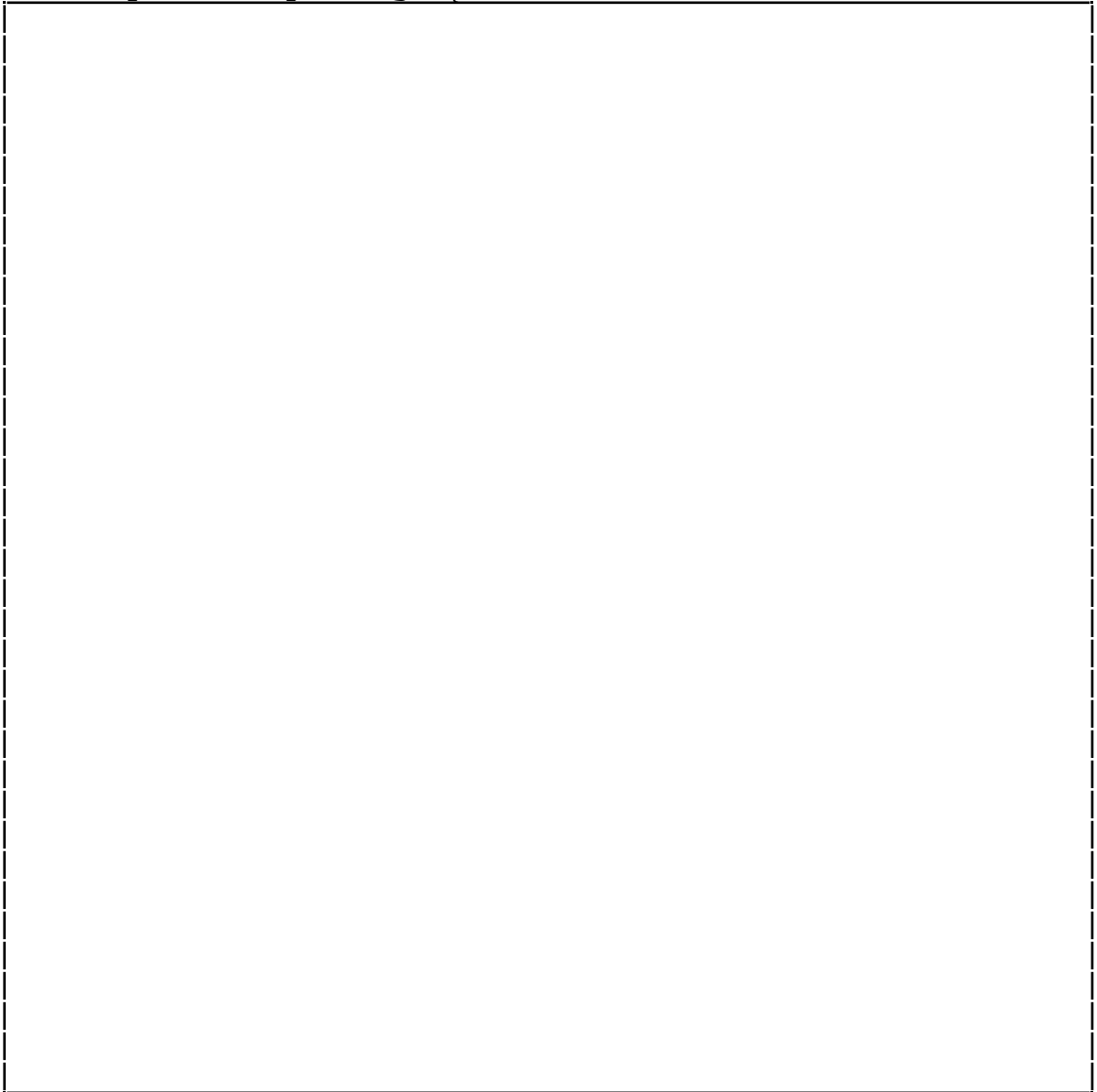
Examine every element in the shots array:

For each element that is true, examine the corresponding element in the layout array. Draw a rectangle in red if this represents a ship, and draw a rectangle in blue if this represents the sea. For each element that is false, don't draw anything.

@param g the Graphics object

*/

public void paint(Graphics g) {



}

(10 marks)

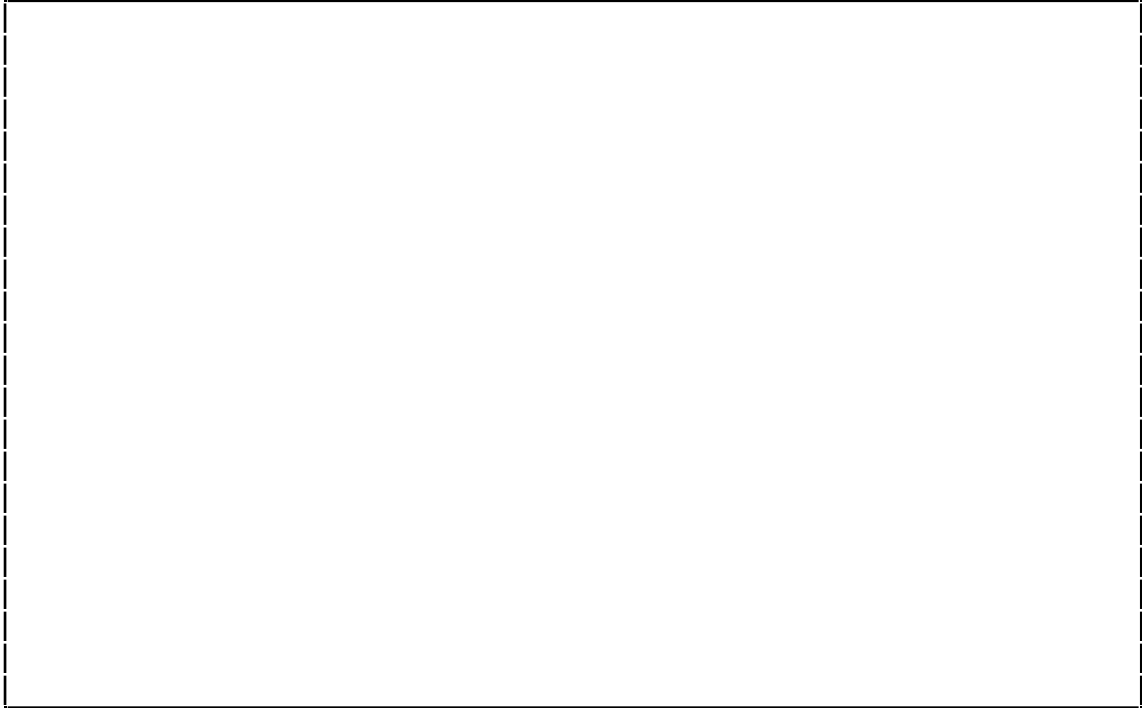
```
public void mouseClicked(MouseEvent e) {}  
public void mouseEntered(MouseEvent e) {}  
public void mouseExited(MouseEvent e) {}  
public void mouseReleased(MouseEvent e) {}
```

}

SURNAME: FORENAMES:

Question 12 (5 marks)

Name a principle held by the Internet community in general, and briefly explain how the principle applies to a practical case, for example in e-commerce activities, network architecture, educational activities, or other areas of Internet activity.

*(5 marks)*

SURNAME: FORENAMES:

OVERFLOW PAGE

CONTINUED

SURNAME: FORENAMES:

APPENDIX:

Useful methods and variables:

Vector

```
public void addElement(Object obj)
public boolean removeElement(Object obj)
public Object elementAt(int index)
public int size()
```

String

```
public int indexOf(int c)
public int indexOf(String string)
public char charAt(int index)
public String substring(int beginIndex, int endIndex)
public int length()
public boolean equals(String comparison)
```

Graphics

```
public void fillRect(int x, int y, int width, int height)
public void drawRect(int x, int y, int width, int height)
public void fillOval(int x, int y, int width, int height)
public void drawOval(int x, int y, int width, int height)
public void drawLine(int x1, int y1, int x2, int y2)
public void setColor(Color c)
```

Applet

```
public void init()
public void paint(Graphics g)
public void add(Component component)
```

Button

```
public void setBounds(int x, int y, int width, int height)
public void addActionListener(ActionListener listener)
```

Color

```
Color.black
Color.white
Color.yellow
Color.red
Color.blue
Color.green
```

Point

```
public int x
public int y
```

Rectangle

```
public int x
public int y
public int width
public int height
```

Math

```
public static double random()
```
