If the instructor is unavailable in the examination room and if doubt exists as to the interpretation of any problem, the candidate is urged to submit with the answer paper a clear statement of any assumptions made.

Proctors are unable to respond to queries about the interpretation of exam questions. Do your best to answer exam questions as written.

Please write your answers in the boxes provided and write your student number on each page. Extra space is available on the last page of the exam. The back of any page can be used for rough work. This exam is three hours long and refers exclusively to the use of the Java language. Comments are not required in the code you write. For full marks, code must be efficient as well as correct.

This is a closed book exam. No computers or calculators are allowed.
**Problem 1) [20 marks]: Glossary**
Using only one or two sentences each, explain the following terms or phrases from our glossary:

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstract Class</strong></td>
<td>A class that contains at least one abstract method, and may contain other concrete members. An abstract method has no body – it just consists of the return type, the method name and the parameter list.</td>
</tr>
<tr>
<td><strong>Anonymous Class</strong></td>
<td>A class created at the same time as it is instantiated without actually naming the class, which must be implementing an interface.</td>
</tr>
<tr>
<td><strong>Constructor</strong></td>
<td>A special method with no return type (not even void) that is invoked once when an object is instantiated. The method has the same name as the class and may or may not (the default constructor) accept arguments that will be used to set attributes.</td>
</tr>
<tr>
<td><strong>Encapsulation</strong></td>
<td>Describes a set of class design principles that provide for the abstraction and protection of data contained within the class. Proper encapsulation practices make sure that there is no way to create an illegal instance of an object as well as providing a set of standard methods that most objects should have.</td>
</tr>
<tr>
<td><strong>Garbage Collection</strong> (in the context of the JRE)</td>
<td>An automatic process carried out by the JRE where the memory used by objects that are no longer referenced by any pointers is returned to the memory pool for re-use.</td>
</tr>
</tbody>
</table>
Inheritance

A process built into Java that allows a child class to inherit all the public members of a parent class, thus removing the need to duplicate these member declarations in the child. This uses the extends keyword in the class header of the child.

Instantiation

When the new keyword is used to create an instance or a copy of a class using the class declaration as a “blue print” for the structure of the copy. The process often requires that attribute values are set by invoking a constructor during the instantiation process.

Method Overloading

Takes place when a method is declared with the same return type and the same name as another method in the current or a parent class. In the case of overloading the method must have a different parameter list – different in the number of parameters or the types of parameters.

Polymorphism

A process that is also called late binding that can be used with classes that are part of a hierarchy, or with a class that implements an interface. An object that has been declared a parent type, or the interface type, morphs into (or becomes) an object of a child or implementing type at runtime.
Bin sort is a simple, yet efficient sort that works well with integer data that has a limited range. It works by first locating the maximum in the dataset to be sorted, and then counting the occurrences of each value in the dataset into the bins array which is sized to the maximum value + 1. Then the sorted dataset is re-created using the bins count. For example, consider the following dataset of values between 0 and 9:

```
1 3 5 7 0 1 2 9 8 7 3 1 2 4 5 4 5 7 8 9 0 3 4 1 2 7 8 6 6 1 8 9 3 2 5 6 8 5 2
```

The largest value is 9, so the bins array will be of length 10. After counting the number of occurrences of each value in the dataset, the bins array will hold:

<table>
<thead>
<tr>
<th>Bin (index):</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count:</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Now, the dataset can be re-created in order by using the counts in each bin. Two zeros, five ones, five twos, four threes, etc.:

```
0 0 1 1 1 1 2 2 2 2 2 3 3 3 3 4 4 4 5 5 5 5 6 6 6 7 7 7 8 8 8 8 8 8 9 9 9
```

For this problem you must write three methods that will carry out a bin sort. You may assume that the supplied array of int will always consist of values ≥ 0, and that the largest value will not be too large (that is to say that the bins array will fit in memory). Start in the box provided below by writing a public static method that accepts an array of int, and then returns the largest value in the provided array:

```java
public static int findMax(int[] array) {
    int max = 0;
    for(int aVal : array)
        if (aVal > max)
            max = aVal;
    return max;
}
```
Problem 2, Cont.)
Next, write a method that uses the method from the previous page to create, populate and finally return the bins array, given a dataset array of any size:

```java
public static int[] generateBins(int[] array) {
    int[] bins = new int[findMax(array) + 1];
    for (int aVal : array)
        bins[aVal]++;
    return bins;
}
```

Finally, write a `void` method that uses the above method to sort a supplied array of `int` of any size, *in situ*, using the bin sort algorithm as described above:

```java
public static void binSort(int[] array) {
    int[] bins = generateBins(array);
    int arrayPos = 0;
    int i, count;
    for (int binPos = 0; binPos < bins.length; binPos++) {
        count = bins[binPos];
        for (i = 0; i < count; i++) {
            array[arrayPos] = binPos;
            arrayPos++;
        }
    }
}
```
Problem 3) [35 marks]: Encapsulation

If the sides of a triangle are described by the positive values: a, b and c, then in order for the values to describe a valid triangle, they must satisfy what is called the "Triangle Inequality Theorem" which is a fancy name for the rule that states: "The sum of two sides must add up to be greater than the length of the remaining third side." In terms of the side lengths, these statements must be satisfied:

\[ a + b > c \]
\[ b + c > a \]
\[ a + c > b \]

The area of any triangle can be calculated using what is commonly called "Heron's Rule" after Heron of Alexandria. Given the three sides, a, b and c:

\[ s = \frac{a+b+c}{2} \]
\[ \text{area} = \sqrt{s(s-a)(s-b)(s-c)} \]

For this problem you need to encapsulate the side lengths of a triangle into an object called Triangle (no kidding!), like you did in Lab 6. You will need an exception class called IllegalTriangle that you can write in the box below. More details on your Triangle object are given on the next page.

```java
public class IllegalTriangle extends Exception {

    public IllegalTriangle (String message) {
        super(message);
    }

    public IllegalTriangle () {
        super("Attempt to create Triangle with illegal data!");
    }
}
```
Problem 3, Cont.)

Store the triangle dimensions as an array of double. You will need two constructors, one that takes three double values and another that takes an array of double. Throw an exception with an informative message if the parameters are illegal in any way. You do not need to have a public mutator, but must have a public accessor for the dimensions.

You will also need the rest of the standard encapsulation methods: toString(), equals(), compareTo() and clone(). A string representation of a Triangle object with the side dimensions 6.0, 2.0 and 7.0 would look like:

\[
\text{A Triangle with sides: 6.0, 2.0, 7.0}
\]

Equality is defined as all side dimensions being equal, even if they are not in the same order. Your equals() method must override the equals method inherited from the Object class. The compareTo() method compares the current Triangle with the provided Triangle on the basis of their areas. If the area of the provided triangle is higher, return -1, if lower return 1, if equal return 0. Finally, clone() must return a deep copy of the current Triangle object.

You can calculate the square root of a number using the static Math.sqrt() method. The java.util.Arrays.sort() method will sort an array of double in situ.

```java
import java.util.Arrays;

class Triangle {
    private double[] sides = new double[3];

    private void setSides (double a, double b, double c) throws IllegalTriangle {
        if (a > 0 && b > 0 && c > 0 &&
            a + b > c && b + c > a && a + c > b) {
            sides[0] = a;
            sides[1] = b;
            sides[2] = c;
        } else {
            throw new IllegalTriangle ("Illegal triangle sides: " + a + ", " + b + ", " + c);
        }
    } // end setSides mutator
}
```
Problem 3, Cont.

```java
public Triangle (double a, double b, double c) throws IllegalTriangle {
    setSides(a,b,c);
} // end full parameter constructor

public Triangle (double[] sides) throws IllegalTriangle {
    if (sides.length != 3)
        throw new IllegalTriangle("Must have three sides in array!");
    setSides(sides[0], sides[1], sides[2]);
} // end one parameter constructor

public double[] getSides () {
    return sides.clone();
} // end sides accessor

private double getArea () {
    double s = (sides[0] + sides[1] + sides[2]) / 2.0;
    double area = Math.sqrt(s * (s - sides[0]) *
        (s - sides[1]) * (s - sides[2]));
    return area;
} // end getArea

public boolean equals (Object otherTriangle) {
    if (otherTriangle instanceof Triangle) {
        Triangle otherT = (Triangle)otherTriangle;
        double otherSides[] = otherT.getSides();
        double currentSides[] = getSides();
        Arrays.sort(otherSides);
        Arrays.sort(currentSides);
        boolean check = true;
        for (int side = 0; side < 3; side++)
            check = check && (otherSides[side] == currentSides[side]);
        return check;
    } else
        return false;
} // end equals method
```
Problem 3, Cont.)

```java
public int compareTo(Triangle otherT) {
    if (getArea() < otherT.getArea()) return -1;
    else if (getArea() > otherT.getArea()) return 1;
    else return 0;
} // end compareTo

public String toString () {
    String output = "A Triangle with sides:" +
    String.format("%6.1f," , sides[0]) +
    String.format("%6.1f," , sides[1]) +
    String.format("%6.1f" , sides[2]);
    return output;
} // end toString

public Triangle clone() {
    Triangle otherT = null;
    try {
        otherT = new Triangle(sides);
    } catch (IllegalTriangle ite) {
        // Will never get here!
    }
    return otherT;
} // end clone

} // end Triangle class
```
Problem 5) [10 marks]:  

GUI Construction

The following GUI program sprawls over this page and the next two and runs without error:

```java
public class Problem5 extends JFrame {

    private final int WIDTH = 600;
    private final int HEIGHT = 400;
    private final int LEFT = 200;
    private final int TOP = 100;

    private Timer animTimer;
    private int tickCount = 0;
    private JLabel message;
    private JTextField enterText;
    private DrawPanel canvas;

    public Problem5() {
        super();
        setSize(WIDTH, HEIGHT);
        setLocation(LEFT, TOP);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Problem 5 Window");
        setLayout(new BorderLayout());

        Font myFont = new Font("Arial", Font.PLAIN, 20);
        Box topBox = Box.createHorizontalBox();
        JLabel leftLabel = new JLabel("Enter here:");
        leftLabel.setFont(myFont);
        topBox.add(leftLabel);
        enterText = new JTextField(8);
        enterText.setFont(myFont);
        topBox.add(enterText);
        JButton goButton = new JButton("Start Show");
        goButton.setFont(myFont);
        goButton.addActionListener(new GoButtonListener());
        topBox.add(goButton);
```
canvas = new DrawPanel();
message = new JLabel("Hello");
message.setFont(myFont);
message.setHorizontalAlignment(JLabel.CENTER);
canvas.add(message, BorderLayout.CENTER);

add(topBox, BorderLayout.NORTH);
add(canvas, BorderLayout.CENTER);

// The delay is 1000 milliseconds or one second:
animTimer = new Timer(1000, new AnimateListener());

} // end Problem5 constructor

private class GoButtonListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        message.setText(enterText.getText() + " is done!");
        animTimer.start();
    } // end actionPerformed method
} // end GoButtonListener

private class AnimateListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        tickCount++;
        canvas.repaint();
        if (tickCount > 3)
            animTimer.stop();
    } // end actionPerformed method
} // end AnimateListener

private class DrawPanel extends JPanel {
    public DrawPanel() {
        setLayout(new BorderLayout());
    }
    public void paint(Graphics g) {
        super.paint(g);
        for (int rectCount = 0; rectCount < tickCount; rectCount++) {
            // Rectangles will be centred on, and will completely surround and frame
            // the message label. All rectangles will fit in the canvas panel.
            int lblHeight = 30;
            int lblWidth = message.getText().length() * 10;
            int height = lblHeight + rectCount * 60;
            int width = lblWidth + rectCount * 60;
            int canvasHeight = getHeight();
            int canvasWidth = getWidth();
            g.setColor(Color.BLACK);
            // The drawRect method accepts the parameters (left, top, width, height),
            // and draws a rectangle of the given width and height with the left, top
            // corner at (left, top)
            g.drawRect((canvasWidth - width) / 2,
                        (canvasHeight - height) / 2, width, height);
Problem 5 class

In the empty frame shown below, sketch the window as it first appears. Do not worry about absolute pixel sizes, the shape of the letters or drawing straight lines, just try to get the relative positions of the components correct.
**Problem 5, Cont.**  
Sketch the window again, after you have typed “This exam” into the text box, clicked on the button and one second has passed:

![Problem 5 Window](image1)

Sketch the window one last time after another 10 seconds have passed:

![Problem 5 Window](image2)