415.101FC PRINCIPLES OF PROGRAMMING

Test – Tuesday 11th April 2000
6:30pm - 8:00pm

INSTRUCTIONS

This test constitutes 15% of your final grade for the course.

You have 5 minutes reading time.

Do not write until you are told.

No one is to leave in the last 10 minutes.

The test is closed book.

No calculators are allowed.

There are 100 marks in this test. The marks for each question are at the end of the question.

Answer all questions in the answer book provided.

Write your name and the day and time of your tutorial on the cover sheet and your ID number on the inside of the cover sheet.

Write your name on every page.

When you are asked to write code you do **NOT** have to include Javadoc comments.

See the last page for a list of methods and constants you may find useful in your answers.

You may use the back of the answer book for rough working.

1. Draw the output of the following HTML code in the browser window in the answer book. Do not worry about the exact size of fonts, but do show differences between different levels of heading and different styles of text and normal text.

```
<html><head>
<title>
How to pass a test
</title></head>

<body>
<h1>Want to pass?</h1>

Follow these simple steps:

<ul>
<li>relax</li>
<li>make sure you understand <a href =
http://www.cs.auckland.ac.nz/lectures/>these</a></li>
<li><b>don't</b> spend too long on question one</li>
</ul>
<hr>

</body></html>
```

(5 marks)

2. The following Java application is supposed to simulate the tossing of a coin. Roughly half the time this application is run, the output should be "A head was thrown.", and the other half, the output should be: "A tail was thrown.".

```
public class CoinToss {
    public void main (String args[]) {
           toss = (int)Math.random()*2;
           if (toss = 0)
                System.out.println("A head was thrown.")
           else
                System.out.println("A tail was thrown.");
     }
}
```

However, there are 5 errors in the code above. 4 of these errors are syntax errors which prevent the code from compiling. The fifth error is a logic error, so that even though the code compiles, the program does not behave correctly.

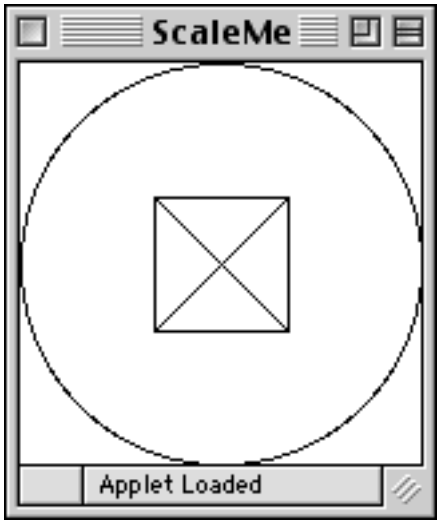
Fix all the errors by writing the program out correctly and CIRCLE the changes. You will lose marks for making unnecessary changes.

(5 marks)

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

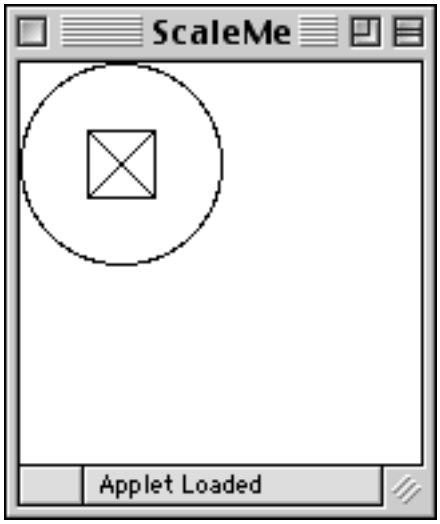
```
import java.awt.*;
import java.applet.*;

public class ScaleMe extends Applet {

     public void paint(Graphics g) {
          final int SIZE = 50;

          ...

     }

}
```

All of the parameters to the drawing methods you call in the `paint()` method must be based on a single constant, called `SIZE`. When the value of `SIZE` is 50, the applet should produce the following drawing, where the applet window is exactly 150 pixels wide and 150 pixels high:



The diameter of the circle is 3 times as large as the side length of the square. The square is positioned exactly in the center of the circle.

If the value of the constant `SIZE` is changed to 25, and the applet is run again, the following drawing should be produced:



Again, the applet window is exactly 150 pixels wide and 150 pixels high in the picture above.

(10 marks)

4. What is the exact output of the following application:

```
public class Types {

    public static void main (String args[]) {

           int i;
           double d;

           i = 4;
           d = 10/i;
           System.out.println("a) " + d);

           i = 3 % 9;
           d = 1.5;
```

```
            System.out.println("b) " + (d * i));

            i = 10;
            i += (int)d;
            d = i;
            System.out.println("c) " + d);

            i = 5;
            d = 2;
            System.out.println("d) " + (i / (int)d));

            d = 7.99999;
            i = (int)Math.round(d);
            System.out.println("e) " + i);
      }
}
```

(5 marks)

5. What is the output of this program. (As this is an Applet the print statements would be seen in the Java console window.) You will get some credit for showing correct working even if your output is wrong.

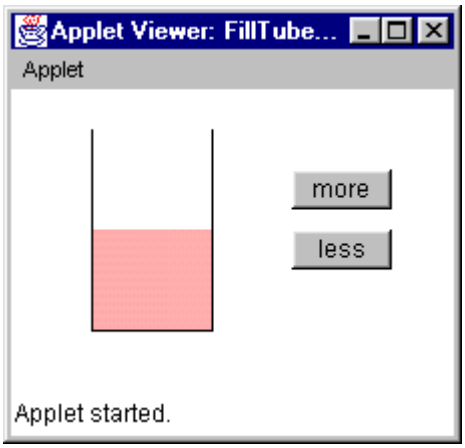
```
import java.applet.*;

public class MethodTester extends Applet {

      public void init() {
            int sum;
            sum = 0;
            sum = first(sum);
            sum = second(sum);
            sum = third(sum);
            System.out.println("From init:" + sum);
      }

      private int third(int number) {
            number = first(number);
            number = second(number);
            System.out.println("From third:" + number);
            return number;
      }

      private int second(int number) {
            number = first(number);
            System.out.println("From second:" + number);
            return number;
      }

      private int first(int number) {
            number = number + 1;
            System.out.println("From first:" + number);
            return number;
      }

}
```

(20 marks)

6. Complete the `FillBeaker` applet. The applet starts by drawing a beaker in black which is half filled with pink goo as in the following picture. There are two buttons in the applet: one labelled "more" and the other labelled "less".

   - When the user presses the "more" button and the beaker is not full the pink goo goes up in the beaker by 10 pixels. If the beaker is full, nothing happens.

   - When the user presses the "less" button and the beaker is not empty the pink goo goes down in the beaker by 10 pixels. If the beaker is empty, nothing happens.



You must use the constants which are included in the following code to position the beaker and the buttons in the applet. The beaker is 100 pixels high and the pink goo starts off at 50 pixels high. Don't worry about being off by 1 pixel.

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

public class FillBeaker extends Applet implements ActionListener {

      /** Left side of the beaker. */
      static final int LEFT = 40;
      /** Right side of the beaker . */
      static final int RIGHT = 100;
      /** Top of the beaker . */
      static final int TOP = 20;
      /** Bottom of the beaker . */
      static final int BOTTOM = 120;
      /** Left position of the buttons. */
      static final int BUTTON_LEFT = 140;
      /** Width of the buttons. */
      static final int BUTTON_WIDTH = 50;
      /** Height of the buttons. */
      static final int BUTTON_HEIGHT = 20;
      /** Top of the "more" button. */
      static final int MORE_TOP = 40;
      /** Top of the "less" button. */
      static final int LESS_TOP = 70;
      …
}
```

(30 marks)

7. In assignment 5 there were several tests to see if the mouse pointer was within small rectangles on the screen.

   Complete the method called `isInsideRect` which takes six parameters: the x and y position of the mouse pointer, and the x and y position and width and height of a rectangle.

   The `isInsideRect` method returns true if the `(mouseX, mouseY)` point is inside the rectangle with the top-left position `(rectX, rectY)` and dimensions `width` and height. Otherwise the `isInsideRect` method returns false. Don't worry about being off by one pixel.

```
private boolean isInsideRect(int mouseX, int mouseY, int rectX, int rectY,
                                    int width, int height) {
     …
}
```

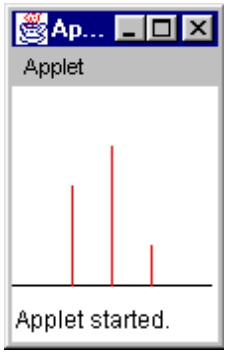
(10 marks)

8. Complete the `VerticalLine` class. A `VerticalLine` object has an x position and a length. All lines start at the y position `100` and are drawn up the screen to the y position `100 - length` in the color red.

   You have to write the constructor and the `draw` method for the `VerticalLine` class in such a way that the following applet can use the `VerticalLine` class to produce the output shown. You will also need to have some instance variables in your `VerticalLine` class.

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

public class LotsOfLines extends Applet {

     private VerticalLine one;
     private VerticalLine two;
     private VerticalLine three;

     public void init() {
          one = new VerticalLine(30, 50);
          two = new VerticalLine(50, 70);
          three = new VerticalLine(70, 20);
     }

     public void paint(Graphics g) {
          g.setColor(Color.black);
          g.drawLine(0, 100, 100, 100);
          one.draw(g);
          two.draw(g);
          three.draw(g);
     }
}
```

Produces this output:



(15 marks)

Methods you may find useful:

Graphics class

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

Button and TextField classes

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

TextField class

```
String getText()
```

Applet class

```
void paint(Graphics g)
void init()
void showStatus(String message)
void setLayout(LayoutManager layout);
```

String class

```
String String.valueOf(double number)
String String.valueOf(int number)
```

ActionListener interface

```
void actionPerformed(ActionEvent e)
```

Constants you may find useful:

Color class

```
Color.red
Color.pink
Color.blue
Color.white
Color.black
Color.gray
Color.green
```