QUEEN'S UNIVERSITY  
SCHOOL OF COMPUTING  

CISC212, FALL TERM, 2008  
FINAL EXAMINATION  
7pm to 10pm, 17 DECEMBER 2008, Grant Hall  

Instructor: Alan McLeod  

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. 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: | / 10 | Problem 4: | / 40 |
| Problem 2: | / 10 | Problem 5: | / 10 |
| Problem 3: | / 10 |

TOTAL: / 80
Problem 1) [10 marks]

In lecture we found that the machine epsilon for a float is 5.9604645e-8. Also, Float.MAX_VALUE is 3.4028235e38. Write the output of the following program for each println() statement. Remember that a float literal has a capital F appended to the number.

```java
public class Problem1 {
    public static void main(String[] args) {
        System.out.println(1.0F / 0.0F); // Infinity
        System.out.println(0.0F / 0.0F); // NaN
        System.out.println(2.0F + Float.MAX_VALUE > Float.MAX_VALUE); // false
        System.out.println(Math.sqrt(-4.0F)); // NaN
        System.out.println(1.0F + 6.0E-8 == 1.0F); // false
        System.out.println (2.0F / 3.0F - 0.666666666661F == 0.0F); // true
        System.out.println(1.0F + 1.0e-12F == 1.0F); // true
        System.out.println(1.0F - 1.0e-12F == 1.0F); // true
        System.out.println(1.0 + 1.0e-12 == 1.0); // false
        System.out.println(1.0e12F + 1.0F == 1.0F); // false
    }
}
```
Problem 2) [10 marks]

The main method for this program is on the next page. In the box provided on the next page, complete the console output of this program. The output of the first call to printTwoD() is given so you can see how this method works.

public class Problem2 {

    public static int attr = 0;

    public static int method1(int[][] twoD, int[] oneD, int noD) {

        int max = 0;
        for(int i = 0; i < twoD.length; i++)
            if (sumRow(twoD[i]) > max) {
                max = sumRow(twoD[i]);
                noD = i;
            } // end if

        for(int i = 0; i < twoD[noD].length; i++) {
            oneD[i] = twoD[noD][i];
            twoD[noD][i] = 0;
        } // end for

        return max;
    } // end method1

    public static int sumRow(int[] oneD) {

        int sum = 0;
        for(int i = 0; i < oneD.length; i++)
            sum = sum + oneD[i];

        attr = attr + sum;

        return sum;
    } // end sumRow

    public static void printOneD(int[] oneD) {

        for(int i = 0; i < oneD.length; i++)
            System.out.print(oneD[i] + " ");
    } // end printOneD

    public static void printTwoD(int[][] twoD) {

        for(int i = 0; i < twoD.length; i++) {
            printOneD(twoD[i]);
            System.out.print("\n");
        } // end for
    } // end printTwoD
public static void main(String[] args) {
    int[][] test2D = {{0, 0, 0, 1},
                     {0, 0, 1, 1},
                     {1, 1, 1, 1},
                     {1, 0, 1, 1}};
    int[] test1D = {2, 1, 2, 1};
    int aNum = 0;
    printTwoD(test2D);  // This output is supplied
    System.out.println(method1(test2D, test1D, aNum));
    printTwoD(test2D);
    printOneD(test1D);
    System.out.println("\naNum = " + aNum);
    System.out.println("attr = " + attr);
} // end main
} // end Problem2

Complete the output from here down:

```
  0 0 0 1
  0 0 1 1
  1 1 1 1
  1 0 1 1
  4
  0 0 0 1
  0 0 1 1
  0 0 0 0
  1 0 1 1
  1 1 1 1
aNum = 0
attr = 17
```
**Problem 3) [10 marks]**

In lecture we saw a summation example that used one of the Ramanujan formulae. This summation should converge exactly to zero. It does not, if coded as shown on the next page, because the summation is storing small differences between two relatively large numbers.

\[
0 = \sum_{k=0}^{\infty} (-1)^k \frac{(2k+1)^2 + (2k+1)^3}{k!}
\]

or,

\[
0 = \frac{1^2 + 1^3}{0!} - \frac{3^2 + 3^3}{1!} + \frac{5^2 + 5^3}{2!} - \frac{7^2 + 7^3}{3!} + \ldots
\]

You need to change how the method on the next page operates so that the sum will properly converge to zero. Delete code by crossing it out if necessary, and add code where needed. Do not add any other methods.
Problem 3, Cont.)

```java
public static double summation () {
    int k = 0;
    double sum = 0;
    double prevSum;
    double kfactorial = 1;
    int sign = 1;
    int twoKPlusOne, twoKPlusOneSquared, twoKPlusOneCubed;
    double term;
    double negSum = 0;
    double posSum = 0;
    double prevNegSum;
    double prevPosSum;

    do {
        prevNegSum = negSum;
        prevPosSum = posSum;
        prevSum = sum;
        twoKPlusOne = 2 * k + 1;
        twoKPlusOneSquared = twoKPlusOne * twoKPlusOne;
        twoKPlusOneCubed = twoKPlusOneSquared * twoKPlusOne;
        if (k > 0)
            kfactorial *= (double)k;
        term = (twoKPlusOneSquared + twoKPlusOneCubed) / kfactorial;
        if (sign > 0)
            posSum += term;
        else
            negSum += term
        sum = sum + sign * term;
        sign = -sign;
        k++;
    } while (!(negSum == prevNegSum && posSum == prevPosSum));
    } while (sum != prevSum);
    return sum;
    return posSum - negSum;
} // end summation
```
Problem 4) [40 marks]

For this problem you need to build a partial hierarchy of classes to hold information about fuels. Here is a verbal description of the hierarchy containing all the required class names in **bold**:

- Diesel is a Fuel
- Gasoline is a Fuel
- FuelOil is a Fuel
- BronzeGas is a Gasoline
- GoldGas is a Gasoline

You must also write and use an Exception class called *IllegalFuel*. All Fuels have the following attributes:

- `pricePerLitre` – Legal range is $0.50 to $2.00 per litre, exclusive.
- `sourceID` – A string name of the fuel’s vendor. Cannot be empty or null.

In addition, a Gasoline object also has an attribute, `ethanolContent`, which must lie between 0 and 20%, inclusive.

Each attribute must have an **accessor**, but no mutators are required. Objects must have a `toString()` method, but do not need `equals()`, `compareTo()` or `clone()` methods.

The Fuel class must contain the following abstract method declaration:

```
public abstract double getCostForWeight(double weightKg);
```

A concrete implementation of this method returns the cost for a supplied weight of the fuel. Use the following densities for your calculations (temperature dependencies are ignored!):

<table>
<thead>
<tr>
<th></th>
<th>Diesel</th>
<th>Bronze Gas</th>
<th>Gold Gas</th>
<th>Fuel Oil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density (kg/litre)</td>
<td>0.85</td>
<td>0.72</td>
<td>0.78</td>
<td>0.893</td>
</tr>
</tbody>
</table>

Consider carefully where you declare your attributes and methods to minimize the volume of code you must write.

On the next page is a complete program listing that illustrates how this hierarchy must behave.

Write your classes on the following pages. You will see a partial class header like “**public class Gasoline**” for each object that you must complete. Finish the headers and write the balance of the code for each object following that class header.
Problem 4, Cont., Demo Code

public class TestFuelHierarchy {

    public static void main(String[] args) {

        Fuel fail1, fail2;
        try {
            double illegalPrice = 3.00; // dollars per litre!
            fail1 = new BronzeGas(illegalPrice, "Esso", 10);
        } catch (IllegalFuel e) {
            System.out.println(e.getMessage());
        } // end test1
        try {
            String illegalSource = ""; // empty string
            fail2 = new BronzeGas(0.762, illegalSource, 10);
        } catch (IllegalFuel e) {
            System.out.println(e.getMessage());
        } // end test2

        Fuel[] testFuels = new Fuel[100];
        // For simplicity all instantiations are in a single try/catch block.
        // Rather poor form! But all are legal.
        try {
            testFuels[0] = new Diesel(1.016, "Petro1");
            testFuels[1] = new BronzeGas(0.817, "Petro1", 0);
            testFuels[2] = new BronzeGas(0.796, "ElCheapo", 10);
            testFuels[3] = new GoldGas(0.911, "Petro2", 0);
            testFuels[4] = new FuelOil(0.876, "HomeHeatingCo");
        } catch (IllegalFuel e) {
            System.out.println(e.getMessage());
        }

        for(int i = 0; i < 5; i++) {
            System.out.println("\n" + testFuels[i]);
            System.out.printf("Cost of 20 kg: $%.2f",
                    testFuels[i].getCostForWeight(20));
        } // end for

    } // end main
} // end TestFuelHierarchy

/* OUTPUT:
Illegal price: 3.0
Illegal source ID.

Diesel fuel, source: Petro1, $1.016 per litre.
Cost of 20 kg: $23.91
Bronze gasoline, with 0.0% ethanol, source: Petro1, $0.817 per litre.
Cost of 20 kg: $22.69
Bronze gasoline, with 10.0% ethanol, source: ElCheapo, $0.796 per litre.
Cost of 20 kg: $22.11
Gold gasoline, with 0.0% ethanol, source: Petro2, $0.911 per litre.
Cost of 20 kg: $23.36
Fuel oil, source: HomeHeatingCo, $0.876 per litre.
Cost of 20 kg: $19.62
*/
Problem 4, Cont., Your Code)

```java
public abstract class Fuel {

    private double pricePerLitre;
    private String sourceID;

    public Fuel (double ppl, String source) throws IllegalFuel {
        if (ppl < 0.5 || ppl > 2.0)
            throw new IllegalFuel("Illegal price: " + ppl);
        if (source == null || source.length() == 0)
            throw new IllegalFuel("Illegal source ID.");
        pricePerLitre = ppl;
        sourceID = source;
    }

    public double getPricePerLitre() { return pricePerLitre; }
    public String getSourceID() { return sourceID; }

    public String toString() {
        return "source: " + sourceID + ", $" + pricePerLitre + 
            " per litre."
    }

    public double getCostForWeight(double sg, double weightKg) {
        double volume = weightKg / sg;
        return pricePerLitre * volume;
    }

    public abstract double getCostForWeight(double weightKg);

} // end Fuel
```
Problem 4, Cont., Your Code

```java
public class IllegalFuel extends Exception {
    public IllegalFuel() {
        super("Attempt to create an illegal Fuel.");
    }
    public IllegalFuel(String msg) {
        super(msg);
    }
}

public abstract class Gasoline extends Fuel {
    private double ethanolContent;
    public Gasoline(double ppl, String source, double ethanol)
            throws IllegalFuel {
        super(ppl, source);
        if (ethanol < 0 || ethanol > 20)
            throw new IllegalFuel("Illegal ethanol level.");
        ethanolContent = ethanol;
    }
    public double getEthanolContent() { return ethanolContent; }
    public String toString() {
        return "gasoline, with " + ethanolContent + "% ethanol, " +
               super.toString();
    }
} // end Gasoline
```
Problem 4, Cont., Your Code

```java
public class BronzeGas extends Gasoline {

    public BronzeGas(double ppl, String source, double ethanol) throws IllegalFuel {
        super(ppl, source, ethanol);
    }

    public String toString() {
        return "Bronze " + super.toString();
    }

    public double getCostForWeight(double weightKg) {
        return getCostForWeight(0.72, weightKg);
    }

} // end BronzeGas

public class GoldGas extends Gasoline {

    public GoldGas(double ppl, String source, double ethanol) throws IllegalFuel {
        super(ppl, source, ethanol);
    }

    public String toString() {
        return "Gold " + super.toString();
    }

    public double getCostForWeight(double weightKg) {
        return getCostForWeight(0.78, weightKg);
    }

} // end GoldGas
```
public class Diesel extends Fuel {

    public Diesel(double ppl, String source) throws IllegalFuel {
        super(ppl, source);
    }

    public String toString() {
        return "Diesel fuel, " + super.toString();
    }

    public double getCostForWeight(double weightKg) {
        return getCostForWeight(0.85, weightKg);
    }
}

public class FuelOil extends Fuel {

    public FuelOil(double ppl, String source) throws IllegalFuel {
        super(ppl, source);
    }

    public String toString() {
        return "Fuel oil, " + super.toString();
    }

    public double getCostForWeight(double weightKg) {
        return getCostForWeight(0.893, weightKg);
    }
}

// end Diesel

// end FuelOil
Problem 5) [10 marks]
Here is a complete GUI program, sprawling over this and the next two pages:

```java
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Font;

public class Problem5 extends JFrame {

    private final int WINDOW_WIDTH = 300;
    private final int WINDOW_HEIGHT = 140;

    private JTextField txtWidth = new JTextField(8);
    private JTextField txtHeight = new JTextField(8);

    private JLabel lblWidthError = new JLabel("Illegal Width!");
    private JLabel lblHeightError = new JLabel("Illegal Height!");
    private JLabel lblResult = new JLabel("Area =");

    public Problem5() {
        super();
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
        setTitle("Problem 5");
        setLocation(200, 200);
        setResizable(false);

        Font myFont = new Font("Arial", Font.BOLD, 16);

        JPanel topPanel = new JPanel(new FlowLayout());
        JPanel centrePanel = new JPanel(new FlowLayout());

        JLabel lblWidth = new JLabel("Width:");
        lblWidth.setFont(myFont);
        JLabel lblHeight = new JLabel("Height:");
        lblHeight.setFont(myFont);
        JTextField txtWidth = new JTextField(8);
        txtWidth.setFont(myFont);
        JTextField txtHeight = new JTextField(8);
        txtHeight.setFont(myFont);
        JLabel lblWidthError = new JLabel("Illegal Width!");
        lblWidthError.setFont(myFont);
        JLabel lblHeightError = new JLabel("Illegal Height!");
        lblHeightError.setFont(myFont);
        JLabel lblResult = new JLabel("Area =");
        lblResult.setFont(myFont);

        JButton btnCalculate = new JButton("Calculate");
        btnCalculate.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String widthStr = txtWidth.getText();
                String heightStr = txtHeight.getText();

                int width = Integer.parseInt(widthStr);
                int height = Integer.parseInt(heightStr);

                if (width > 0 && height > 0) {
                    lblWidthError.setVisible(false);
                    lblHeightError.setVisible(false);
                    lblResult.setText(String.format("Area = %.2f", width * height));
                } else {
                    lblWidthError.setVisible(true);
                    lblHeightError.setVisible(true);
                }
            }
        });

        JPanel bottomPanel = new JPanel(new FlowLayout());
        bottomPanel.add(btnCalculate);

        topPanel.add(lblWidth);
        topPanel.add(txtWidth);
        topPanel.add(lblWidthError);
        topPanel.add(lblResult);
        topPanel.add(lblHeight);
        topPanel.add(txtHeight);
        topPanel.add(lblHeightError);
        topPanel.add(bottomPanel);

        centrePanel.add(lblWidth);
        centrePanel.add(txtWidth);
        centrePanel.add(lblWidthError);
        centrePanel.add(lblResult);
        centrePanel.add(lblHeight);
        centrePanel.add(txtHeight);
        centrePanel.add(lblHeightError);

        getContentPane().add(topPanel, BorderLayout.NORTH);
        getContentPane().add(centrePanel, BorderLayout.CENTER);
    }

    public static void main(String[] args) {
        Problem5 frame = new Problem5();
        frame.setVisible(true);
    }
}
```
Problem 5, Cont.)

topPanel.add(lblWidth);
topPanel.add(txtWidth);
topPanel.add(lblWidthError);

centrePanel.add(lblHeight);
centrePanel.add(txtHeight);
centrePanel.add(lblHeightError);

add(topPanel, BorderLayout.NORTH);
add(centrePanel, BorderLayout.CENTER);

JPanel bottomPanel = new JPanel(new BorderLayout());

lblResult.setFont(myFont);

JButton calculateButton = new JButton("Calculate");
calculateButton.setFont(myFont);
calculateButton.addActionListener(new CalculateListener());

JButton exitButton = new JButton("Close");
exitButton.setFont(myFont);
exitButton.addActionListener(new EndingListener());

bottomPanel.add(calculateButton, BorderLayout.WEST);
bottomPanel.add(lblResult, BorderLayout.CENTER);
bottomPanel.add(exitButton, BorderLayout.EAST);

add(bottomPanel, BorderLayout.SOUTH);

} // end Problem5 constructor

private class EndingListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {
        System.exit(0);
    }

} // end actionPerformed method

} // end EndingListener class
Problem 5, Cont.)

private class CalculateListener implements ActionListener {

    public void actionPerformed(ActionEvent e) {

        boolean widthOK, heightOK;
        double width = 0, height = 0, area;

        lblWidthError.setVisible(false);
        lblHeightError.setVisible(false);

        lblResult.setText(""");

        widthOK = true;
        try {
            width = Double.parseDouble(txtWidth.getText());
            if (width <= 0)
                widthOK = false;
        } catch(NumberFormatException except) {
            widthOK = false;

        }

        heightOK = true;
        try {
            height = Double.parseDouble(txtHeight.getText());
            if (height <= 0)
                heightOK = false;
        } catch(NumberFormatException except) {
            heightOK = false;

        }

        if (!widthOK)
            lblWidthError.setVisible(true);
        if (!heightOK)
            lblHeightError.setVisible(true);

        if (widthOK && heightOK) {
            area = width * height;
            lblResult.setText("Area = " + area);
        }

    } // end actionPerformed method
} // end CalculateListener class

public static void main(String[] args) {

    Problem5 gui = new Problem5();
    gui.setVisible(true);

} // end main

} // end Problem5
Problem 5, Cont.)

Sketch the appearance of this window for each of the three following cases. Use the following key to draw your components:

A Label:  A Text Box  A Button

a) As the window first appears, without any user entry or actions:

![Window without user input]

b) If the user enters 2 for the width and 3 for the height and clicks on "Calculate":

![Window with user input]

c) If the user enters “Happy” for the width and “Holidays!” for the height and clicks on “Calculate”: