CISC 212

QUEEN'S UNIVERSITY
SCHOOL OF COMPUTING

CISC212, FALL TERM, 2010
FINAL EXAMINATION
11 DECEMBER 2010, 9am

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 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.

<table>
<thead>
<tr>
<th>Student Number:</th>
</tr>
</thead>
</table>

Problem 1:    / 20  Problem 6:    / 10  
Problem 2:    / 10  Problem 7:    / 20  
Problem 3:    / 10  Problem 8:    / 20  
Problem 4:    / 10  Problem 9:    / 10  
Problem 5:    / 10  

TOTAL:    / 120
Problem 1) [20 marks]

The following complete program runs without error. Write the output for each println() statement beside that statement in the box provided:

```java
class Problem1 {
    public static void main(String[] args) {
        System.out.println(5 * 2 - 3);
        System.out.println(7 >= 6 && 5 != 3);
        System.out.println(5.0 / 10);
        System.out.println(5 / 10);
        System.out.println(1 / 2 * 10);
        System.out.println(1 / 2 * 10.0);
        System.out.println((double)(1 / 2) * 10);
        System.out.println(10 * 1 / 2);
        System.out.println(10.0 / 0.0);
        System.out.println(5 != 7 || 3 <= 10 || 7 > 20);
        System.out.println(Math.sqrt(-1.0F));
        System.out.println("100" + 10 + 1);
        System.out.println(Double.MAX_VALUE + 1 == Double.MAX_VALUE);
        System.out.println(Integer.MAX_VALUE + 1 == Integer.MIN_VALUE);
        System.out.println(0x10);
        System.out.println(0xA);
        System.out.println(5 * (2 - 3));
        System.out.println(100 + 10 + "1");
        System.out.println(010);
        System.out.println(0x10 + 010);
    }
}
```
Problem 2) [10 marks]

In the box provided below write the output of the second call to showArray() in the following complete program which runs without error. The output of the first call to showArray() is provided.

```java
public class Problem2 {
    public static void showArray(int[][] data) {
        int row, col;
        for (row = 0; row < data.length; row++) {
            for (col = 0; col < data[row].length; col++)
                System.out.printf("%4d", data[row][col]);
            System.out.println();
        }
    } // end showArray

    public static void fiddle(int[][] data) {
        int row, col;
        for (row = 0; row < data.length; row++)
            for (col = 0; col < data[row].length; col++)
                if (data[row][col] > 0)
                    data[row][col] = 10;
                else
                    data[row][col] = row;
    } // end fiddle

    public static void main(String[] args) {
        int[][] test = {{1, 0, 0, 0, 1},
                        {0, 0, 0, 0, 0},
                        {0, 0, 1, 0, 0},
                        {0, 0, 0, 0, 0},
                        {1, 0, 0, 0, 1}};
        showArray(test);
        fiddle(test);
        showArray(test);
    } // end main
}
```
Problem 3) [10 marks]

The following complete program runs without error. Write the output of the program in the box provided.

```java
public class Problem3 {

    public static void showArray(int[] data) {
        for (int i = 0; i < data.length; i++)
            System.out.printf("%4d", data[i]);
        System.out.println();
    } // end showArray

    public static void showArray(int[][] data) {
        for (int i = 0; i < data.length; i++)
            showArray(data[i]);
    } // end showArray

    public static int fiddle(int one, String two, int[] three, int[] four, int[][] five) {
        int sum = 0;
        int i;
        int[] inside = {5, 4, 3};
        for (i = 0; i < three.length; i++) {
            sum = sum + three[i];
            three[i] = three[i] * 2;
        }
        one = one + sum;
        two = two.toLowerCase();
        four = inside;
        for (i = 0; i < five.length / 2; i++) {
            inside = five[i];
            five[i] = five[five.length - i - 1];
            five[five.length - i - 1] = inside;
        }
        return one;
    } // end fiddle

    public static void main(String[] args) {
        int num = 10;
        String str = "HELLO";
        int[] test = {2, 3, 4};
        int[] nums = {20, 30, 40};
        int[][] test2D = {{1, 2, 3},
                          {4, 5, 6},
                          {7, 8, 9}};
        int result = fiddle(num, str, test, nums, test2D);
        System.out.println(num);
        System.out.println(str);
        showArray(test);
        showArray(nums);
        showArray(test2D);
        System.out.println(result);
    } // end main

} // end Problem3
```
Problem 4) [10 marks]
The following complete program runs without error. Write the output of the program in the box provided. Write the name of the sorting algorithm used here:

```java
public class Problem4 {
    public static void showArray(int[] data) {
        for (int i = 0; i < data.length; i++)
            System.out.printf("%4d", data[i]);
        System.out.println();
    } // end showArray

    public static void swap(int p1, int p2, int[] data) {
        int temp = data[p1];
        data[p1] = data[p2];
        data[p2] = temp;
    } // end swap

    public static int findMax(int start, int[] data) {
        int max = start;
        for (int i = start; i < data.length; i++)
            if (data[i] > data[max])
                max = i;
        return max;
    } // end findMax

    public static void sort(int[] data) {
        int max;
        for (int i = 0; i < data.length - 1; i++) { // fix the range
            max = findMax(i, data);
            if (max != i) // fix the test condition
                swap(i, max, data);
            showArray(data);
        }
    } // end sort

    public static void main(String[] args) {
        int[] values = {4, 12, 1, 3, 11, 8, 7, 6};
        showArray(values);
        sort(values);
    } // end main
}
```
Problem 6) [10 marks]
In lecture we saw a summation example:

\[ 1 = \sum_{i=1}^{n} \frac{i}{S_n} \quad \text{where:} \quad S_n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \]

Unfortunately, for large values of \( n \), the sum can be far away from one because some terms of the summation are simply too small to make a difference to the sum. The Kahan Summation Algorithm, also called “Compensated Summation” can solve this problem by changing the way the sum is calculated. The principle behind the algorithm is that each time the sum is calculated, the portion of the individual term that is not added to the sum is calculated and then summed back into the next term. In the box provided, write a method that returns the value of this summation in the most accurate way possible using the Kahan Summation Algorithm. Write just the method, no other code such as main is required. Use a `double` for your sum, and note that the method accepts \( n \) as a parameter.
Problem 7) [20 marks]
Write a fully encapsulated class called “Toothpaste” and an exception class called “ITException”. Here are the attributes that describe a tube of Toothpaste:

- **Brand**: “Crest” or “Colgate”.
- **Size**: between 50 and 150 mL inclusive.
- **Whitener**: `true` or `false`, the default is `false` (no whitener).

No other attributes are required. A Toothpaste object can be created with values for all three attributes or with just the brand and size, in which case the whitener attribute is assumed to be `false`. If an attempt is made to create a Toothpaste object, or modify an attribute with illegal values, then the constructors and mutators must throw an ITException with a relevant message. Write public mutators and accessors for all attributes.

Here are a couple of examples of how you should represent your Toothpaste object as a String:

Crest toothpaste, 100 mL size, with whitener.
Colgate toothpaste, 125 mL size, without whitener.

The Toothpaste class also needs the other standard methods: `toString()`, `compareTo()`, `equals()` and `clone()`. Toothpaste objects will be compared (for sorting, for example) on the basis of their size only. Equality is defined as all attributes being exactly equal. (Use the `String.equals()` method to compare the String attribute). The `clone()` method should return a deep copy of the current object. Your `equals()` method must override the `equals()` method from the base `Object` class. Use good encapsulation practices and prevent privacy leaks. Write the exception class on this page and your Toothpaste class on the following three pages. Do not write any other classes.
Problem 7, Cont.)
Problem 7, Cont.)
Problem 8) [20 marks]
Here is a table describing a few fasteners:

<table>
<thead>
<tr>
<th>Fastener:</th>
<th>Bolt</th>
<th>Nail</th>
<th>Staple</th>
<th>Screw</th>
<th>Bolt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>Phillips</td>
<td>n/a</td>
</tr>
<tr>
<td>Finish</td>
<td>no</td>
<td>galvanized</td>
<td>galvanized</td>
<td>bright</td>
<td>bright</td>
</tr>
<tr>
<td>Length (inch)</td>
<td>3</td>
<td>3.5</td>
<td>0.5</td>
<td>2.5</td>
<td>4</td>
</tr>
<tr>
<td>Price per lb</td>
<td>n/a</td>
<td>$10</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Price each</td>
<td>$0.50</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>$0.75</td>
</tr>
<tr>
<td>Price per 100</td>
<td>$40.00</td>
<td>n/a</td>
<td>$2</td>
<td>$4.50</td>
<td>$50.00</td>
</tr>
</tbody>
</table>

"n/a" means that this attribute is not applicable to this item. You will note that bolts have two prices, a bulk price per 100 and an individual price. As a service to the customer, bolt orders can be "sized up" in order to get a better price. For example, for the last Bolt in the table, it would be cheaper to price 480 bolts as 5 boxes * $50 (total $250), rather than 4 boxes * $50 + 80 * $0.75 (total $260).

Write the minimum amount of code necessary to build a hierarchy to store these fastener descriptions and calculate costs for given order sizes. You are supplied with the base Fastener class, which you cannot change. If you are careful, there will not be any redundancy in the declaration of your attributes. No attribute should be null, empty or zero after instantiation. Do not carry out any error checking on the arguments. Write the minimum amount of code to satisfy the operation of the testing program shown below. The output of this program is shown on the next page.

```java
public class Problem8 {
    public static void showItemCost(Fastener fast, int orderSize) {
        String output = orderSize + fast.getDescription();
        output += String.format(" Cost = $%.2f", fast.getCost(orderSize));
        System.out.println(output);
    } // end showItemCost

    public static void main(String[] args) {
        // First three parameters are length, finish and unit price:
        Fastener nails = new Nail(3.5, "galvanized", 10);
        Fastener bolts1 = new Bolt(3, "no", 0.50, 40);
        Fastener bolts2 = new Bolt(4, "bright", 0.75, 50);
        Fastener staples = new Staple(0.5, "galvanized", 2);
        Fastener screws = new Screw(2.5, "bright", 4.50, "Phillips");

        showItemCost(nails, 3);  // 3 lbs
        showItemCost(bolts1, 210); // 210 bolts
        showItemCost(bolts2, 480); // 480 bolts
        showItemCost(staples, 4); // 4 boxes of 100 each
        showItemCost(screws, 3); // 3 boxes of 100 each
    } // end main
} // end Problem8
```
/*OUTPUT:
3 lb. of 3.5 inch nails, galvanized finish. Cost = $30.00
210 of 3.0 inch bolts, no finish. Cost = $85.00
480 of 4.0 inch bolts, bright finish. Cost = $250.00
4 boxes of 100 each 0.5 inch staples, galvanized finish. Cost = $8.00
3 boxes of 100 each Phillips head 2.5 inch screws, bright finish. Cost = $13.50 */

public abstract class Fastener {

    private double length;
    private String finish;
    private double unitPrice;

    public Fastener(double len, String fin, double price) {
        length = len;
        finish = fin;
        unitPrice = price;
    }

    public double getLength() { return length; }
    public String getFinish() { return finish; }

    public double getCost(int numUnits) {
        return unitPrice * numUnits;
    }

    public abstract String getDescription();

} // end Fastener
Problem 8, Cont.)
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class GUIProblem extends JFrame {

private final int WIDTH = 800;
private final int HEIGHT = 160;

private String message1 = "Almost Over";
private String message2 = "VACATION TIME!";
private JLabel messageLabel1 = new JLabel(message1);
private JLabel messageLabel2 = new JLabel(message2);

public GUIProblem () {
    super();

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setTitle("GUI Problem");
    setSize(WIDTH, HEIGHT);

    messageLabel1.setFont(new Font("Lucida Handwriting Italic", Font.PLAIN, 26));
    messageLabel2.setFont(new Font("Lucida Handwriting Italic", Font.PLAIN, 26));
    JButton switchMessageButton = new JButton("Switch");
    switchMessageButton.addActionListener(new MessageButtonListener());
    JTextField textBox = new JTextField(8);
    setLayout(new BorderLayout());
    JPanel bottomPanel = new JPanel(new FlowLayout());
    JPanel midPanel = new JPanel(new FlowLayout());

    midPanel.add(messageLabel1);
    midPanel.add(textBox);
    midPanel.add(messageLabel2);
    bottomPanel.add(switchMessageButton);

    add(midPanel, BorderLayout.CENTER);
    add(bottomPanel, BorderLayout.SOUTH);
}

// end constructor
private class MessageButtonListener implements ActionListener {

    public void actionPerformed (ActionEvent e) {

        if (messageLabel1.getText().equals(message1)) {
            messageLabel1.setText(message2);
            messageLabel2.setText(message1);
        } else {
            messageLabel1.setText(message1);
            messageLabel2.setText(message2);
        }
    } // end actionPerformed

} // end MessageButtonListener

} // end GUIProblem class

You can assume that there is another class with a main method which instantiates this one and then shows the window. Sketch the appearance of this window after the “Switch” button has been clicked once:

![Image of a window with a label and a text box]

Don’t worry about what the font looks like. Use the following key to draw your components, changing the text where necessary:

A Label  A Text Box  A Button