1.00   Introduction to Computers and Engineering Problem Solving
Quiz 2 – April 15, 2005

Name:	
E-mail Address:	
TA:	
Section:	

You have 80 minutes to complete this exam. For coding questions, you do not need to include comments, and you should assume that all necessary files have already been imported.

Question	Points
1. Inheritance	/10
2. Inheritance	/15
3. Matrix, exception	/20
4. Recursion	/20
5. Swing	/35
Total	/100

HW1	HW2	HW3	HW4	HW5	HW6

Question 1. Short answer/true-false: Inheritance (10 points)

1. A class can inherit from multiple abstract classes. Circle TRUE or FALSE.

   TRUE   **FALSE**

2. A class can implement multiple interfaces. Circle TRUE or FALSE

   TRUE   FALSE

3. What is the default visibility of the methods in an interface? Circle the correct answer.

   a. public
   b. protected
   c. private
   d. package

4. A subclass may call any non-private superclass method **someMethod** that has no arguments by using the syntax **super().someMethod();**

   TRUE   **FALSE**

5. If a subclass is derived from a superclass that has an abstract method, and if no definition is supplied in the subclass for that abstract method, then the subclass must be declared as an abstract class

   TRUE   FALSE

Question 2. Inheritance (15 points)

The `Account` class given below has characteristics similar to those of a bank account.

Line No	
1	`public class Account {`
2	`    private double balance;`
3	
4	`    public Account(double openingBalance) {`
5	`        balance = openingBalance;`
6	`    }`
7	`    public void deposit(double amt) {`
8	`        balance += amt;`
9	`    }`
10	`    public final double getPrevBalance(double amt) {`
11	`        return balance - amt;`
12	`    }//calculates the balance before the last`
13	`     //deposit (amt)`
14	`}`

```
Line No
  1      public class Account {
  2          private double balance;
  3
  4          public Account(double openingBalance) {
  5              balance = openingBalance;
  6          }
  7          public void deposit(double amt) {
  8              balance += amt;
  9          }
 10          public final double getPrevBalance(double amt) {
 11              return balance - amt;
 12          }//calculates the balance before the last
 13           //deposit (amt)
 14      }
```

Assume `class Account` is correct. The `class Checking` inherits from the `class Account`. Identify the two compilation errors in the `class Checking` given below.

```
Line No
  1      public class Checking extends Account {
  2          private int numTransactions;
  3
  4          public Checking(double myopeningbalance){
  5              super(myopeningbalance);
  6              numTransactions= 0;
  7          }
  8          public void deposit(double amt) {
  9              balance += amt;
 10              numTransactions++;
 11          }
 12          public double getPrevBalance(double amt) {
 13              if (numTransactions > 0)
 14                  return super.getPrevBalance(amt);
 15              else return 0;
 16          }
 17          public int getNumTransactions() {
 18              return numTransactions;
 19          }
 20      }
```

The two errors are:

1. The `getPrevBalance(…)` method of `class Account` was declared `final`. Therefore it cannot be overwritten in the subclass `MyChecking`.

2. The member `balance` is a private data member of the `class Account` and may not be accessed directly by instances of the `MyChecking` subclass.

DO NOT WRITE HERE – FOR GRADERS ONLY		
Points awarded:		/15

Question 3. Matrices and Exceptions (20 points)

Complete the class `SimpleMatrix` below by adding a method `transpose()` with no
arguments that transposes a square matrix by moving every element (i,j) to position (j,i)
in the output. A general example of transposing a matrix is shown below.

$$\text{If} \quad A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad \text{then} \quad A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

The method `transpose()` returns a new `SimpleMatrix` which is the transpose of
the original matrix. If the matrix is not square, your method must throw a
`BadMatrixException`. You do not need to check if the matrix has >0 rows or
columns.

```
public class SimpleMatrix {

    private double [][] data;

    public SimpleMatrix(int r, int c) {
        data= new double[r][c];
    }

    public double getElement(int i, int j) {
        return data[i][j];
    }

    public void setElement(int i, int j, double val) {
        data[i][j] = val;
    }

    //continued on the next page
```

// A. Write transpose():

```java
    public SimpleMatrix transpose() throws BadMatrixException {

        int rows = data.length;
        int columns = data[0].length;

        if (rows!=columns)
            throw new BadMatrixException();

        SimpleMatrix result = new SimpleMatrix(rows,columns);

        for (int i=0; i < data.length; i++){
            for (int j=0; j < data[0].length; j++) {
                result.data[i][j] = this.data[j][i];
            }
        }

        return result;
    }
```

}

B. Write class BadMatrixException with two (2) constructors:

```java
public class BadMatrixException extends Exception {

     public BadMatrixException() {
          super();
     }

     public BadMatrixException(String s) {
          super(s);
     }
}
```

DO NOT WRITE HERE – FOR GRADERS ONLY		
Points awarded:		/20

Question 4. Recursion (20 points)

Complete the method, `printBinary()`, which prints the binary equivalent of the input `int n` using a recursive algorithm. Assume the method is called with a positive integer argument `n > 0`.

The binary equivalent of integer `n` may be found by repeatedly dividing `n` by 2 and printing out the remainders. For example, the following procedure illustrates the steps to find the binary equivalent of 13.

```
13/ 2 = 6 remainder 1          (Note that 13 % 2 = 1)
6 / 2 = 3 remainder 0          (Note that  6 % 2 = 0)
3 / 2 = 1 remainder 1
1 / 2 = 0 remainder 1
```

Therefore, 13 in base 2 is 1101. Note the bits are in reverse order of the example calculation.

Use `System.out.print()`, not `System.out.println()` to output the bits on one line. Hint: Can you write the result directly when `n = 1` ?

Write your code for `printBinary()` here:

```java
public static void printBinary(int n) {

        if (n > 1)
            printBinary(n/2);
        System.out.print(n % 2);
    }


    // Alternate solution following pink-yellow example

    public static void printBinary2(int n) {
        if (n == 1)
            System.out.print(1);
        else {
            printBinary2(n/2);
            System.out.print(n % 2);
        }
    }

}
```
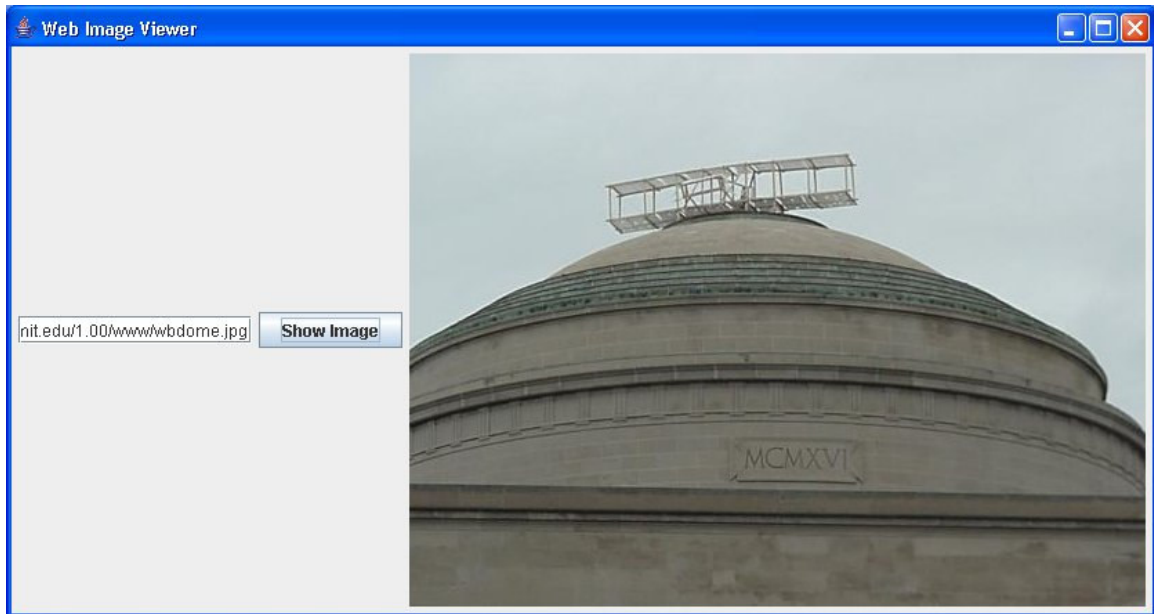
DO NOT WRITE HERE – FOR GRADERS ONLY		
Points awarded:		/20

Question 5. Swing web image viewer application (35 points)

The application you are given to develop has an interface like one shown in the figure below. This application takes the URL of an image file from the text field (entered by the user), and loads that image to a `JLabel` component.

Your application must do the following two things:
- When the "Show Image" button is clicked, the image is loaded from the URL onto the `JLabel` next to the text field
- If the URL is invalid, catch the exception, blank the text field. (The user will need to enter a correct URL.)

Extend the given fragment of code as follows.
- A.   Create and add the necessary components to the interface. Also size the `PictureViewer`.
- B.   Using an anonymous inner class, write and set an event listener for the "Show Image" click event.

```java
import java.awt.event.*;
import javax.swing.*;
import java.awt.*;
import java.net.*;

public class PictureViewer extends JFrame  {
      JLabel imageLabel;
      JPanel picturePanel;
      JTextField urlText;
      JButton showImageButton;

      public PictureViewer() {
            setTitle("Web Image Viewer");
```

```java
// Part A: add components to the user interface
// Create text field, button, panel
// Recall that JPanel uses FlowLayout)
// Add panel to contentPane
// Add text field, button to panel
// Call pack()
```

```java
        urlText = new JTextField(15);
        showImageButton = new JButton("Show Image");
        picturePanel = new JPanel();
        picturePanel.add(urlText);
        picturePanel.add(showImageButton);
        Container c = getContentPane();
        c.add(picturePanel);
        pack();
```

```
// Part B: Write, register event listener for the "Show Image"
// click event using anonymous inner class
// See Lecture 17 notes for image viewer example
// 1 Create URL object; its constructor takes String from text box
// 2 Create ImageIcon object; its constructor takes URL argument
// 3 Create JLabel object; its constructor takes ImageIcon as first
//    argument, position as second (use SwingConstants.CENTER)
// 4 Add the JLabel object to your GUI. Proceed as usual…
// 5 URL constructor throws MalformedURLException that you must catch
```

```java
            showImageButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e){
                    try {
                            URL source = new URL(urlText.getText());
                            ImageIcon image = new ImageIcon( source );
                            imageLabel = new JLabel( image,
                                SwingConstants.CENTER );
                            picturePanel.add(imageLabel);
                            repaint();
                            pack();
                    }
                    catch (MalformedURLException ex) {
                            urlText.setText("");
                    }
                }
            });
    }
        public static void main(String[] args) {
                PictureViewer app = new PictureViewer();
                app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                app.setVisible(true);
        }
    }
```

DO NOT WRITE HERE – FOR GRADERS ONLY		
Points awarded:		/35