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.

Student Number:

Problem 1: / 10 Problem 5: / 24
Problem 2: / 6 Problem 6: / 14
Problem 3: / 14 Problem 7: / 8
Problem 4: / 14

TOTAL: /

90
Problem 1) [10 marks]
Write the output of each of the following console output statements beside the statement. If the code would result in an error, write “error” instead.

```java
public class TwoLiners {

    enum Possibilities {EXCEL, PASS, FAIL};

    public static void main(String[] args) {
        System.out.println("Today, I will: " + Possibilities.values()[0]);
        Possibilities myPast = Possibilities.PASS;
        System.out.println("Not just: " + myPast);
        System.out.printf("%5.2f\n", Math.PI);
        int aVal = 4.5e2;
        System.out.println(aVal);
        double aValAgain = 5 / 2;
        System.out.println(aValAgain);
        double anotherVal = 5 / (double)2;
        System.out.println(anotherVal);
        int[] evenNums = {2, 4, 6, 8, 10};
        System.out.println(evenNums.length);
        int[] oddNums = {1, 3, 5, 7, 9};
        System.out.println(oddNums[5]);
        int[][] pairs = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};
        System.out.println(pairs[3][1]);
        System.out.println(pairs.length);
    } // end main
} // end TwoLiners
```

```
Today, I will: EXCEL

Possibilities myPast = Possibilities.PASS;
System.out.println("Not just: " + myPast);
System.out.printf("%5.2f\n", Math.PI);
int aVal = 4.5e2;
System.out.println(aVal);
double aValAgain = 5 / 2;
System.out.println(aValAgain);
double anotherVal = 5 / (double)2;
System.out.println(anotherVal);
int[] evenNums = {2, 4, 6, 8, 10};
System.out.println(evenNums.length);
int[] oddNums = {1, 3, 5, 7, 9};
System.out.println(oddNums[5]);
int[][] pairs = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};
System.out.println(pairs[3][1]);
System.out.println(pairs.length);
```
Problem 2) [6 marks]

Write the console window output of the following program in the box provided.

```java
public class Tracing {

    public static int mysteryMethod (int[][] a, int[] b, int c) {
        int sum = 0;
        int i, j;

        for (i = 0; i < a.length; i++)
            for (j = 0; j < a[i].length; j++)
                if (i == j) {
                    a[i][j] = a[i][j] * b[i];
                    sum = sum + a[i][j];
                }

        c = a[a.length - 1][a[0].length - 1];
        return sum;
    }

    public static void main(String[] args) {
        int[][] array1 = {{1, 2, 3},
                          {4, 5, 6},
                          {7, 8, 9}};

        int[] array2 = {1, 2, 3};

        int aNum = 5;

        System.out.println(mysteryMethod(array1, array2, aNum));
        System.out.println(array1[2][2]);
        System.out.println(aNum);
    }

} // end Tracing
```

38
27
5
Problem 3) [14 marks]

a) Provide the value of the binary number 10111011 in base 10:

in base 16 (hex):

b) Provide the binary representation of the base 10 number 47.75

\[
\begin{array}{l}
\text{187} \\
\text{BB} \\
\text{101111.11}
\end{array}
\]

c) Is it possible to store the base 10 number 0.1 exactly on a computer using the IEE754 standard for double precision numbers? Why or why not:

No. 0.1 in base 10 results in an infinitely repeating series in binary and since there is only a limited number of bits for storage, some of this number is lost.

d) According to the IEE754 standard for double precision numbers, what value will result when you attempt to subtract one number from another when the absolute difference between the two numbers is less than machine epsilon?

0

e) Given two IEE754 double precision numbers, n1 and n2, where the value of n1 divided by n2 is less than machine epsilon, what will be the result of adding n1 to n2? Or is this value not defined? Explain your answer:

n2. The value in n1 is too small to make a difference to n2.

f) Consider a situation where you have to evaluate a slowly converging summation formula that has terms of alternating sign (ie. plus, minus, plus, minus, etc.). Describe the basis of an algorithm that would enhance the accuracy of the summation:

Store all positive terms in one sum and all negative terms in another sum. Once the summation has stopped, return the difference between the two sums.

g) Consider the summation algorithm that you have described in part f) – how would you determine when to stop the summation process?

The summation would stop when neither sum value is changing.
Problem 4) [14 marks]

In Lab 2 you learned how to obtain numeric console input in a very rigorous manner. That is to say that you learned how to obtain a number from the user in such a way that no matter what he entered he could not cause your method to crash. Neither will this robust method give up until it has obtained and returned a value of the proper type.

a) Write a method called getInt() that accepts a String prompt as a parameter and will return an integer value. If the user enters something that does not represent a valid int then he should be re-prompted until he does. Remember that the Scanner class must be instantiated with the System.in object for console input. The Scanner class has a nextInt() method that throws an InputMismatchException if it is given something that does not represent a valid int. It also has a nextLine() method that can be used to obtain any String from the user or to empty the console input buffer. You might wish to use the Integer.parseInt() method which throws a NumberFormatException if it cannot convert a supplied string to a valid integer.

```
public static int getInt(String prompt) {
    Scanner input = new Scanner(System.in);
    boolean inputOK = false;
    int aVal = 0;
    String dummy;
    while(!inputOK) {
        try {
            System.out.print(prompt);
            aVal = input.nextInt();
            inputOK = true;
        } catch (InputMismatchException e) {
            dummy = input.nextLine();
            System.out.println("Not a valid int, please try again!");
        }
    }
    return aVal;
}
```
Problem 4, Cont.)

b) Overload the getInt(String prompt) method you wrote for part a) by writing three more methods. These methods not only ensure that a valid int is supplied but that it lies between supplied numeric bounds. One method must ensure that the value is greater than or equal to a lower limit, the second method must ensure that the value is less than or equal to an upper limit and the third method must ensure that the value lies between a lower and an upper limit, inclusive. As for part a), the user should be re-prompted if he does not supply a number in the proper range. These methods may certainly invoke each other and the method you wrote for part a) in order to reduce code duplication. You might wish to use the static constants Integer.MAX_VALUE and Integer.MIN_VALUE.

```java
public static int getInt(int low, String prompt, int high) {
    boolean inputOK = false;
    int aVal = 0;
    while(!inputOK){
        aVal = getInt(prompt);
        if (aVal >= low && aVal <= high)
            inputOK = true;
        else
            System.out.println("Outside range, please re-enter.");
    }
    return aVal;
}

public static int getInt(int low, String prompt) {
    return getInt(low, prompt, Integer.MAX_VALUE);
}

public static int getInt(String prompt, int high) {
    return getInt(Integer.MIN_VALUE, prompt, high);
}
```
Problem 5) [24 marks]

Desperate for work after graduation (so you can start paying back your OSAP loan...), you have accepted a contract job with Acme Bats, Inc. They want you to design a database to hold a description and an inventory of their products. One of their better sellers is named the “Dingbat”, a club-shaped thing used to bat dings. Different Dingbat designs are characterized by an overall length in metres and a set of five integer numbers that represent critical profile diameters in mm. The length must be 0.5 metre (the “IttyBittyBat”), 0.75 metre (the “ClassicBat”) or 1.0 metre (the “Whopper”). No individual diameter measurement can be less than 15 mm or greater than 80 mm. The five profile diameters are stored in sequence in an array and the sequence must increase from low index to high when supplied to the constructor. All diameters must be different.

For example, a legal Dingbat could be described by the attribute values:

- 0.75 metre length
- 25, 30, 32, 36, 40 mm diameters

The following set is illegal for a number of reasons:

- 1.2 metre length
- 22, 10, 32, 120, 50, 36 mm diameters

In the parameter set listed above, the length is not equal to one of the three acceptable values. The array of diameters has one too many values, is not increasing for each value and has one value greater than 80 mm, and another less than 15 mm.

For this problem, you will make a small start on this database by writing a class called “Dingbat” to encapsulate the description of a Dingbat, and a class that extends the Exception class called “IllegalDingbat” to be used when an attempt is made to provide the Dingbat class with illegal attribute values. Do not write any other classes, nor any testing code to use the Dingbat class. Do not write a main method. Your class must follow the design specifications listed here:

- Use good encapsulation techniques, including information hiding and the prevention of privacy leaks.
- Use just a single, full parameter constructor that has the following types for its parameters:

  double, int[]

- Supply an accessor for each attribute.
- Supply a mutator for each attribute.
- The Dingbat class’ methods must throw the IllegalDingbat exception where appropriate if an attempt is made to set any parameter to an illegal value. When throwing the exception supply an informative message describing the exact error. It is sufficient to throw an exception for the first error found; you don’t have to keep looking at the parameters to throw exceptions for every error.
Problem 5, Cont.)

- Override the equals() method from the Object class. That is to say you must write an equals method with the method header:

  public boolean equals (Object obj)

  Equality is defined as all attribute contents being exactly equal.

- Supply a compareTo() method that compares Dingbats on the basis of just their length in mm. compareTo() should return an int value in mm.

- Supply a toString() method that provides a String representation of the current object. For example, the first set of parameters shown above would provide the String representation:

  A Dingbat of length 0.75 metres, with the diameter set: 25, 30, 32, 36, 40 mm.

- Supply a clone() method that returns a deep, non-aliased copy of the current object.

No other methods are required – this is enough work!

Write your exception class below and the Dingbat class on the following three pages.

(Actually a Dingbat is an ornament or spacer used in typesetting, sometimes more formally known as a "printer's ornament", but that's not as much fun!)

```java
public class IllegalDingbat extends Exception {

    public IllegalDingbat() {
        super("Illegal Dingbat parameters.");
    }

    public IllegalDingbat(String s) {
        super(s);
    }

} // end IllegalDingbat
```
public class Dingbat {

    private double length;
    private int[] diameters = new int[5];

    public void setLength(double l) throws IllegalDingbat {
        if (l == 0.5 || l == 0.75 || l == 1.0) {
            length = l;
        } else {
            throw new IllegalDingbat("Illegal length value: " + l);
        }
    } // end setLength

    public void setDiameters(int[] d) throws IllegalDingbat {
        if (d == null || d.length != 5) {
            throw new IllegalDingbat("Diameters array not legal.");
        }
        for (int i = 0; i < 5; i++) {
            if (d[i] < 15 || d[i] > 80) {
                throw new IllegalDingbat("Illegal diameter: " + d[i]);
            }
            if (i > 0 && d[i] <= d[i - 1]) {
                throw new IllegalDingbat("Diameters not in order or
two diameters are the same.");
            }
            diameters[i] = d[i];
        } // end for
        //or outside of loop: diameters = d.clone();
    } // end setDiameters

    public Dingbat(double l, int[] d) throws IllegalDingbat {
        setLength(l);
        setDiameters(d);
    } // end Dingbat constructor

    public double getLength() {
        return length;
    } // end getLength

    public int[] getDiameters() {
        return diameters.clone();
    } // end getDiameters
public boolean equals(Object obj) {
    if (obj instanceof Dingbat) {
        Dingbat otherDingbat = (Dingbat)obj;
        if (otherDingbat.length == length) {
            for (int i = 0; i < 5; i++) {
                if (diameters[i] != otherDingbat.diameters[i]) {
                    return false;
                }
            }
            return true;
        } // end if
    } // end if
    return false;
} // end equals

public int compareTo(Dingbat otherDingbat) {
    return (int)(1000 * (length - otherDingbat.length));
} // end compareTo

public String toString() {
    String s = "A Dingbat of length " + length;
    s += " metres, with the diameter set: ";
    for (int i = 0; i < 4; i++) {
        s += diameters[i] + ", ";
    }
    s += diameters[4] + " mm."
    return s;
} // end toString

public Dingbat clone() {
    int[] temp = diameters.clone(); // not really necessary here
    Dingbat d = null;
    try {
        d = new Dingbat(length, temp);
    } catch (IllegalDingbat e) {
        // do nothing
    } // end try/catch
    return d;
} // end clone

} // end Dingbat
Problem 6) [14 marks]

You have been hired by a real-estate agency to create a database to store information on properties that are up for sale. Here is a table that gives one example of each property type that will be stored in the database. An “n/a” means that a certain attribute is “not applicable” to the property being described.

<table>
<thead>
<tr>
<th>Attribute:</th>
<th>Vacant Land</th>
<th>Farmed Land</th>
<th>Industrial</th>
<th>Office</th>
<th>Home</th>
<th>Cottage</th>
<th>Condo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hectares:</td>
<td>120</td>
<td>100</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Crop</td>
<td>n/a</td>
<td>corn</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Lot Width (metres):</td>
<td>n/a</td>
<td>n/a</td>
<td>100</td>
<td>30</td>
<td>20</td>
<td>50</td>
<td>n/a</td>
</tr>
<tr>
<td>Lot Depth (metres):</td>
<td>n/a</td>
<td>n/a</td>
<td>200</td>
<td>60</td>
<td>40</td>
<td>100</td>
<td>n/a</td>
</tr>
<tr>
<td>Building coverage (m²)</td>
<td>n/a</td>
<td>n/a</td>
<td>800</td>
<td>180</td>
<td>240</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Lake frontage (metres):</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>50</td>
<td>n/a</td>
</tr>
<tr>
<td>Number of bedrooms</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>List Price $k</td>
<td>120</td>
<td>270</td>
<td>350</td>
<td>80</td>
<td>235</td>
<td>230</td>
<td>289</td>
</tr>
</tbody>
</table>

You must build an object hierarchy that starts from a single root class, which will be the base type of your database. You can do this by either drawing a class diagram or by writing code. Either way you must identify the name of each class and which attributes are defined in which classes. If you do this by writing code, just declare the attributes and a single constructor for each class. Do not do any error checking or write any other methods. If you use a class diagram you must clearly indicate the relationship between classes. You will need at least 7 classes – one for each property type shown above. It is not acceptable for an attribute to hold “n/a” or `null` after a class has been instantiated – it must be able to hold a real value. So, you must figure out which extra classes you need and how they are related to build the most efficient structure possible. In this context, efficiency means minimizing the duplication of attribute definition.
Problem 6, Cont. – Alternate Solution)

```java
public class Property {
    private int price;
    public Property(int price) {
        this.price = price;
    }
}

public class Land extends Property {
    private int hectares;
    public Land(int price, int hectares) {
        super(price);
        this.hectares = hectares;
    }
}

public class VacantLand extends Land{
    public VacantLand(int price, int hectares) {
        super(price, hectares);
    }
}

public class FarmedLand extends Land {
    private String crop;
    public FarmedLand(int price, int hectares, String crop) {
        super(price, hectares);
        this.crop = crop;
    }
}

public class Developed extends Property {
    private int coverage;
    public Developed(int price, int coverage) {
        super(price);
        this.coverage = coverage;
    }
}

public class Business extends Developed {
    private int width;
    private int depth;
    public Business(int price, int coverage, int width, int depth) {
        super(price, coverage);
        this.width = width;
        this.depth = depth;
    }
}

public class Office extends Business {
    public Office(int price, int coverage, int width, int depth) {
        super(price, coverage, width, depth);
    }
}
```
Problem 6, Cont. – Alternate Solution, Cont.)

```java
public class Industrial extends Business {
    public Industrial(int price, int coverage, int width, int depth) {
        super(price, coverage, width, depth);
    }
}

public class Residential extends Developed {
    private int numBedrooms;
    public Residential(int price, int coverage, int numBedrooms) {
        super(price, coverage);
        this.numBedrooms = numBedrooms;
    }
}

public class Detached extends Residential {
    private int width;
    private int depth;
    public Detached(int price, int coverage, int numBedrooms, int width, int depth) {
        super(price, coverage, numBedrooms);
        this.width = width;
        this.depth = depth;
    }
}

public class Cottage extends Detached {
    private int lakeFront;
    public Cottage(int price, int coverage, int numBedrooms, int width, int depth, int lakeFront) {
        super(price, coverage, numBedrooms, width, depth);
        this.lakeFront = lakeFront;
    }
}

public class Condo extends Residential {
    public Condo(int price, int coverage, int numBedrooms) {
        super(price, coverage, numBedrooms);
    }
}
```
Problem 7) [8 marks]

Here is a complete GUI program, listed on this and the next page:

```java
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JTextField;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.Font;
public class Problem7 extends JFrame {
    private final int WINDOW_WIDTH = 300;
    private final int WINDOW_HEIGHT = 150;
    private JTextField txtEntry1 = new JTextField(8);
    private JTextField txtEntry2 = new JTextField(8);
    private JTextField txtEntry3 = new JTextField(8);
    private JLabel lblAllDone = new JLabel("All done!");
    private JLabel lblEntry1 = new JLabel("Enter1:");
    private JLabel lblEntry2 = new JLabel("Enter2:");
    private JLabel lblEntry3 = new JLabel("Enter3:");
    private JButton exitButton = new JButton("Exit");

    public Problem7() {
        super();
        setLayout(new BorderLayout());
        setTitle("Problem 7");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
        setResizable(false);
        setLocation(200, 200);

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

        JPanel topPanel = new JPanel(new FlowLayout());
        JPanel centrePanel = new JPanel(new GridLayout(2, 2));
        JPanel bottomPanel = new JPanel(new BorderLayout());

        txtEntry1.setFont(myFont);
        txtEntry2.setFont(myFont);
        txtEntry3.setFont(myFont);
        lblAllDone.setFont(myFont);
        lblEntry1.setFont(myFont);
```

```java
        bottomPanel.add(txtEntry1, BorderLayout.NORTH);
        bottomPanel.add(txtEntry2, BorderLayout.NORTH);
        bottomPanel.add(txtEntry3, BorderLayout.NORTH);
        bottomPanel.add(lblAllDone, BorderLayout.SOUTH);
        bottomPanel.add(lblEntry1, BorderLayout.EAST);
        bottomPanel.add(lblEntry2, BorderLayout.EAST);
        bottomPanel.add(lblEntry3, BorderLayout.EAST);
        bottomPanel.add(exitButton, BorderLayout.SOUTH);

        centrePanel.add(bottomPanel, BorderLayout.CENTER);
        centrePanel.add(lblEntry1, BorderLayout.CENTER);
        centrePanel.add(lblEntry2, BorderLayout.CENTER);
        centrePanel.add(lblEntry3, BorderLayout.CENTER);

        topPanel.add(centrePanel, BorderLayout.CENTER);

        add(topPanel, BorderLayout.NORTH);
    }
}
```
lblEntry2.setFont(myFont);
lblEntry3.setFont(myFont);
exitButton.setFont(myFont);

topPanel.add(lblEntry1);
topPanel.add(txtEntry1);

centrePanel.add(lblEntry2);
centrePanel.add(lblEntry3);
centrePanel.add(txtEntry2);
centrePanel.add(txtEntry3);

bottomPanel.add(lblAllDone, BorderLayout.WEST);
bottomPanel.add(exitButton, BorderLayout.EAST);

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

} // end Problem7 constructor

public static void main(String[] args) {
    Problem7 gui = new Problem7();
gui.setVisible(true);
} // end main
} // end Problem7

Sketch the appearance of this window after the program is run. Use the following key to draw your components:

A Label:  A Text Box  A Button

![Window Sketch]