COMP 249 – Final Exam - SOLUTION

Instructions:

Date: August 16, 2005
Time: 19:00 – 22:00 (3 hours)

No notes, books, calculators or other electronic devices are allowed.
Answer directly on this questionnaire.

Identification:

Last Name: _____

First Name : _____

Student ID : _____

Signature: _____

Marking scheme:

CONCEPTS

1	**/ 20**
2	**/ 20**

CODE COMPREHENSION

3	**/ 8**
4	**/ 8**
5	**/ 8**
6	**/ 8**

PROGRAMMING

7	**/ 8**
8	**/ 20**
TOTAL	**/ 100**

CONCEPTS

Question 1 (19pts): Complete the following sentences using the terms given below.

array	event	private	text file
ArrayList	graph	protected	Tree
Bag	HashSet	random access	type
binary file	Implementable	recursive	unchecked
binary search tree	javadoc	Serializable	writeable
BufferedReader	List	Stack	sequential
checked	listener	static	FileOutputStream
Comparable	package	static type	programmer-defined
dynamic type	private	stream	

A- (3pts)_____**Comparable Serializable List Queue** ___ are **all** built-in interfaces from the Java API: *(indicate **all** possible answers from the above list of terms).*

B- (3pts)_____**ArrayList __HashSet**_____ are all concrete classes from the Java Collections Framework. *(indicate **all** possible answers from the above list of terms).*

C- (2pts) A/an _____**checked exception** _____ exception must obey the catch-or-declare rule.

D- (2pts) With a/an _____random access_____ file, the programmer can modify a specific byte anywhere in the file.

E- (2pts) A/an _____**listener** _____ is often implemented using a inner class.

F- (2pts) The _____**static type** _____ of a variable is its type as it was declared in the program.

G- (2pts) In a/an _____**Stack**_____, objects are inserted at the same end as they are removed.

H- (3pts)_____**public protected** _____are **all** the access modifiers that, when used with a method, makes the method available to all the classes in the same package and to all the subclasses of the class. *(indicate **all** possible answers from the list of given terms).*

Question 2 (21pts): Give a brief explanation for each of the following questions.

A-(2pts): In the context of Java, what does it mean to « pass the buck » ?

Answer:

B- (2pts): You have written an applet that displays a cute icon, stored in the file `myCuteIcon.gif`. When you run it with the applet viewer on your own machine, everything is fine, but when you test your applet directly with your (correct) HTML file in your browser, the icon is not shown? Explain why.

Answer:

C- (3pts): If Java did not provide the `writeObject` and `readObject` methods, would you be able to write and read entire objects in binary form? Briefly Explain.

Answer:

D- (2pts): What is the basic difference between the `Reader / Writer` and `InputStream / OutputStream` sets of classes?

Answer:

E- (2pts): You need to write a program to manipulate the daily orders in a flower shop. You do not need these orders to be in a particular order, but they must be unique (no duplicates). What concrete class of the Java collection framework would be most suitable for this?

Answer :

PROGRAM COMPREHENSION

Question 3 (8pts): What is the output of the following program:

```
class Papa {
        public int AVar;

        public Papa() {
            AVar = 999;
            System.out.println("AAA");
            doSomething();
        }

        public void doSomething() {
            AVar = 1111;
            System.out.println("A.doSomething()");
        }
}

public class Baby extends Papa {
        public int BVar;

        public Baby() {
            System.out.println("BBB");
            doSomething();
            System.out.println("AVar=" + AVar);
            BVar = 222;
        }

        public void doSomething() {
            System.out.println("BVar=" + BVar);
        }

        public static void main(String[] args) {
            new Baby();
        }
}
```

Answer:

```
AAA

BVar=0

BBB

BVar=0

AVar=999
```

Question 4 (8pts): Consider the following class.

```
class Base {
   protected void m(int i) {
      // some code
   }
}

public class Child extends Base {

    // Method Here
}
```

For each of the following methods, indicate if it can or cannot be legally inserted in place of the comment **`//Method Here.`** If a method cannot be inserted, briefly explain why not.

Method	**yes**: it can be inserted OR **no**: it cannot be inserted	If it cannot be inserted, why not?
`void m(int i) throws Exception{}`	**no**	**// not OK... weaker priviledge**
`void m(char c) throws Exception{}`	**yes**	**// OK overloading**
`public void m(int i){}`	**yes**	
`protected void m(int i) throws Exception{}`	**no**	**// overriding... should not throw anything**

Question 5 (8pts): What is the output of the following program.

```
import java.util.*;

class Secret {
    public static void main(String [] args)  {
        Collection[] c = new Collection[4];

        try {
             c[0] = new ArrayList();
             c[1] = new LinkedList();
             c[2] = new TreeSet();
             c[3] = new HashSet();
             for (int i = 0; i < c.length; i++)  {
                  c[i].add("1");
                  c[i].add("4");
                  c[i].add("1");
                  c[i].add("3");
                  c[i].add("1");
             }
             for (int i = 0; i < c.length; i++)   {
                  Iterator i1 = c[i].iterator();
                  while (i1.hasNext()) {
                       System.out.print(i1.next());
                  }
                  System.out.println();
             }
        }
        catch (ArrayIndexOutOfBoundsException e) {
             System.out.print("Array out of bounds");
        }
        catch (Exception e) {
             System.out.print("Some other exception");
        }
        System.out.println("the End");
    }
}
```

Answer:

```
14131

14131

134

341

the End
```

Question 6 (6pts): Assume that you have a linked list of nodes. Each node contains a value (a character) and a reference to the next node. Each node is an instance of the following class:

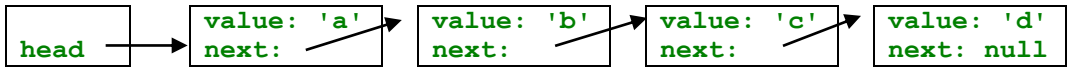
```
class Node {
    public char value;
    public Node next;

    public  Node(char c, Node n){
        this.value = c;
        this.next = n;
    }
}
```

Assume the following program.

```
public class Test {
   public static void main(String[] arg) {
     Node head;
     head = new Node('a',new Node('b',new Node('c',new Node('d',null))));
      // point A
     Node ptr1, ptr2, p;
     ptr1 = head;
     ptr2 = head.next;
     ptr1.next = ptr2.next.next;
     ptr2.next.next = ptr1;
     p = head;
     head = head.next;
     while (p!=null)
     {
         System.out.print(p.value + " ");
         p = p.next;
     }
   }
}
```

A- (4pts) Draw the linked list (each node, its value and what it points to) that the reference head points to at the point marked "point A"

Answer:



B- (4pts) After the following code executes, what is printed?

Answer:

```
a d
```

PROGRAMMING

Question 7 (8pts): The following is an incomplete version of the class `Tree` seen n class to represent a binary tree. Complete the `getNbLeaves()` method to return the number of leaves (nodes with no children) in the tree. Your method **must be recursive**.

```
class Node {
        private Node left;
        private Node right;
        private int value;

        public Node(int value) { this.value = value; }
        Node getLeft() { return left; }
        Node getRight() { return right; }
        int getValue() { return value; }
}

public class Tree {
        Node root;

        public Tree(Node root) { this.root = root; }

        //  getNbLeaves starts at the root
        int getNbLeaves() { return getNbLeaves(root); } }

         int getNbLeaves(Node currentNode)
         {

         int getNbLeaves(Node currentNode){

         if(currentNode.getLeft() == null && currentNode.getRight() == null)
             return 1;

             int nb = 0;
         if(currentNode.getLeft() != null)
            nb += getNbLeaves(currentNode.getLeft());
          if(currentNode.getRight() != null)
                nb += getNbLeaves(currentNode.getRight());

         return nb;

    }



     }
}
```

Question 8 (20pts): In this question, you must use the appropriate object oriented design techniques covered in class (e.g., inheritance, polymorphism, method overloading, etc). In fact, in order to get a full grade for this question, the code must be well designed.

You must write the appropriate classes to model a game involving *aliens* and a *colony of aliens*.

In this game, two types of aliens exist: snakes and ogres. All aliens have:

1. a type -- ogre or snake
2. a health level -- an integer from 0% (dead) to 100% (full strength)
3. a name -- a string

Of course, aliens are bad; so they attack innocent humans. The level of damage inflicted on an innocent victim is represented by points.
When snakes attack, they inflict (10pts × their health level) of damage on their victim.
When ogres attack, they inflict (6pts × their health level) of damage on their victim.
For example, a snake with a health level of 50% will inflict 5pts of damage to their victim.

A colony of aliens (a group of aliens) must include a method called `showDamage()` that:

1. prints in a text file (called `damage.txt`) the level of damage inflicted (the number of points) by every aliens in the colony, and

2. displays on the screen the sum of the damage inflicted by all the aliens in the colony.

For example, if the colony contains 1 snake with 100% strength, 1 ogre with 50% and 1 snake with 10%, then the method should store in the file:

```
10.0
3.0
1.0
```

and on the screen: `14.0pts`

Write the necessary classes to properly represent an `alien` and a `colony` of aliens.

Notes:

- Make sure you write proper exceptions handlers if necessary.
- You are not required to write any comments.
- You can consult the appendix for examples of I/O declarations.

Answer:

```
import java.io.*;

abstract class Alien {
    public int health;
    public String name;
    public Alien(int h, String n) {
        health = h;
        name = n;
    }

    abstract public double getDamage();
}

 class Snake extends Alien {
```

```
    public Snake(int h, String n) {
        super(h, n);
    }
    public double getDamage() {
        return (health/100.0) * 10;
    }
}
 class Ogre extends Alien {
    public Ogre(int h, String n) {
        super(h, n);
    }
    public double getDamage() {
        return (health/100.0) * 6;
    }
}
  class AlienPack2 {

    private Alien[] pack;

    public AlienPack2(int n) {
        pack = new Alien[n];
    }

    public void showDamage() {
        PrintWriter f = null;
        try {
             f = new PrintWriter(
              new BufferedWriter(
              new FileWriter("damage.txt")));

        double total = 0;
        for (int i= 0; i< pack.length; i++) {
            f.println(pack[i].getDamage());
            total += pack[i].getDamage();
        }
        System.out.println(total);
        }
        catch (IOException e) {
            System.out.print(e.getMessage());
        }
        finally {
            if (f!=null) f.close();
        }
    }
    public void fill() {
            pack[0] = new Snake(100, "snake1");
            pack[1] = new Ogre(50, "ogre2");
            pack[2] = new Snake(10, "snake3");
    }
}

public class  AlienPack {
    public static void main(String arg[]) {
        AlienPack2 col = new AlienPack2(3);
        col.fill();
        col.showDamage();
        }
        }
```

End of the exam

APPENDIX: EXAMPLES OF I/O DECLARATIONS

```
BufferedReader f = new BufferedReader(new FileReader("fileName"));
```

```
BufferedWriter f = new BufferedWriter(new FileWriter("fileName"));
```

```
File f = new File("fileName");
```

```
FileInputStream f = new FileInputStream("fileName");
```

```
FileOutputStream f = new FileOutputStream("fileName");
```

```
FileReader f = new FileReader("fileName");
```

```
FileWriter f = new FileWriter("fileName");
```

```
ObjectInputStream f = new ObjectInputStream(new FileInputStream("fileName"));
```

```
ObjectOutputStream f = new ObjectOutputStream(new FileOutputStream("fileName"));
```

```
PrintWriter f = new PrintWriter(
                  new BufferedWriter(
                    new FileWriter("fileName")));
```

```
PrintWriter f = new PrintWriter("fileName");
```

```
RandomAccessFile f = new RandomAccessFile("fileName", "rw");
```

```
Scanner f = new Scanner(new File("fileName"));
```

```
Scanner f = new Scanner(new FileReader("fileName"));
```