Name: Number:

Object Oriented Programming 2012/13

Final Exam
June 27th, 2013

Directions (read carefully):

- CLEARLY print your name and ID on every page.
- The exam contains 8 pages divided into 4 parts. Make sure you have a complete exam.
- The point value of each problem is indicated next to the problem and below.
- You have two hours.
- It is wise to skim all the problems before beginning, to best plan your time.
- This is a closed book exam. No notes of any kind are allowed. Do all work in the space provided, using the backs of sheets if necessary.
- **Over the table it should ONLY be this exam, a pen and an ID.**
- Turn off the mobile phone. The use of a mobile phone annuls the exam.

Part	Problem	Description	Page	Marks
I	1 a) b)	UML	2	1.5
II	2 a) b) c) d) e)	Java development	3	3.0
	3 a) b) c) d) e)	Java multiple choice	6	2.5
	4	Java miscellaneous	7	1.0
III	5	XML	8	1.0
IV	6	Internet	8	1.0
Total				**10.0**

Version A

Name:	Number:

Part I -- UML (1.5 marks)

1 – Consider a club that offers a set of services to its associates. An associate has a number, name and address. The club provides services of roadside and health assistance. All services contain information about the day and hour, as well as a textual description. Each service has a number and indicates the associate that requested it. Both the associate and service numbers must identify unequivocally the associate and the service, respectively. The club should store a list of associates and two lists of services -- pending and executed services.

Roadside assistance may include towing (“reboque”), helping with a flat tire (“pneu furado”) and providing a small amount of fuel when the vehicle runs unexpectedly out of it (“falta de combustível”). To all these services the license plate number and an address are needed. For the flat tire service, the model of the car is also needed; for the fuel service, the type of the fuel is needed.

Health assistance may include ambulatory (“ambulatório”) and domiciliary (“médico ao domicílio”) care. For each health service, the name of the doctor as well as the name and address of the patient is needed. In the case of domiciliary care it is also needed the specialty of the doctor required to provide the care; the club only provides general practice, pediatrics and orthopedics.

a) [1.0 mark] Define the UML class diagram for the presented problem.

Name:	Number:

b) [0.5 marks] Define an UML object diagram considering an ambulatory service; this service is pending. Do not consider the associate. Set all needed attributes/associations with some dummy values.

Part II -- Java (6.5 marks)

2 – Consider the class `SortedSet` of the package `java.util`. You should give an implementation similar to it with the characteristics described in the following. Note: the code that you are going to offer is intended to compile and execute, so providing only the constructors and the methods is not enough. Your implementation should not rely on any data structure from `java.util`.
Provide a class named `CSortedSet` that defines an ordered set implemented through a linked list. In a set there are no repeated elements. The class should be generic with type parameter `T`.

a) **[0.5 marks]** Provide a no-arg constructor that builds an empty ordered set. The ordering maintained by the sorted set should be offered by the type `T`; therefore, type `T` should implement the `Comparable` interface. In addition, provide a second constructor with a `Comparator` object as parameter.

```
public interface Comparable<T> {
      int compareTo(T o);
}
```

```
public interface Comparator<T> {
      int compare(T o1, T o2);
}
```

b) **[0.6 marks]** Provide an `add` method that receives an object of type `T` and inserts it in the set; it returns `true` iff the object was not already present in the sorted set and `false` otherwise.

Name:	Number:

c) **[0.7 marks]** Provide a `Comparator` of objects of type `Ingredient`. An ingredient contains a name and a weight in grams; define the `Ingredient` and a `Comparator` that allows to compare ingredients by weight.

d) **[0.7 marks]** Provide a `headSet` method that returns a view of the portion of the sorted set whose elements are strictly less than an element of type `T` received as parameter. This method returns another `CSortedSet<T>`.

e) **[0.5 marks]** Provide a `main` method where: (i) two objects of type `CSortedSet` are built. One of these sets will use the comparator defined in c), and so it will be an ordered set of ingredients. The second `CSortedSet` should be a set of `String`'s (recall that `String` implements the `Comparable` interface).

```
import java.util.Comparator;

public class CSortedSet<T> {

    private final Comparator<? super T> comparator;
    private Node<T> head;

    private static class Node<T> {
      Node<T> next;
      T elem;
      Node(T o, Node<T> n){
            elem=o;
            next=n;
      }
    }

    public CSortedSet(){
      this.comparator = null;
    }

    public CSortedSet(Comparator<T> comp){
      this.comparator = comp;
    }

    private int isSmaller(T o1, T o2){
      if (comparator==null){
            Comparable<? super T> comparable = (Comparable<? super T>) o1;
            return comparable.compareTo(o2);
      }
      else return comparator.compare(o1,o2);
    }

    public boolean add(T o) {
      if (head==null) {
            head = new Node<T>(o,null);
            return true;
      } else {
            int cmp = isSmaller(o,head.elem);
            if (cmp<0) {
                  head = new Node<T>(o,head);
                  return true;
            } else if (cmp==0) {
                  return false;
            }
```

Name: Number:

```
            else {
                  Node<T> aux = head.next, prev = head;
                  while (aux!=null){
                        cmp = isSmaller(o,aux.elem);
                        if (cmp<0){
                              prev.next = new Node<T>(o,aux);
                              return true;
                        } else if (cmp==0) {
                              return false;
                        } else {
                              prev = aux;
                              aux = aux.next;
                        }
                  }
                  prev.next = new Node<T>(o,null);
                  return true;
            }
      }
    }

    private static class Ingredient {
      String name;
      float weight;
      public Ingredient(String name, float weight){
            this.name = name;
            this.weight = weight;
      }
    }

    private static class IngredientComp implements Comparator<Ingredient> {
      public int compare(Ingredient i1, Ingredient i2){
         return i1.weight < i2.weight ? -1 : i1.weight == i2.weight ? 0 : 1;
     }
    }

    public CSortedSet<T> headSet(T o){
        CSortedSet<T> res;
        if (comparator == null) res = new CSortedSet<T>();
        else res = new CSortedSet<T>((Comparator<T>)comparator);
        Node<T> aux = head;
        while(aux!=null){
            if (isSmaller(aux.elem,o)<0)
                res.add(aux.elem);
            else
                return res;
            aux = aux.next;
        }
        return res;
    }

    public static void main(String[] args){
      CSortedSet<Ingredient> set1 = new CSortedSet<Ingredient>(new
                                                IngredientComp());
      CSortedSet<String> set2 = new CSortedSet<String>();
    }
}
```

Name:	Number:

3 – Fill the answers of multiple choice questions in the following table (use only capital letters). If you want to correct your answer scratch out and write the correct answer. Each correct question is marked 0.5 points. A question not answered is marked 0 points, whereas a wrong answer discounts 0.2 points. If you think NONE of the options are correct, write NONE.

Question	a)	b)	c)	d)	e)
Answer	B	C	C	C	C

a) **[0.5 marks]** What is the value returned by the method `y3()` the first time it is called?

```
public class Y{
   private Y(){}
   private static int v=0;
   double y1(double f) { return (f-1.6); }
   void y2(){ v+=(int)y1((int)1.6+3); }
   public static int y3(){
      (new Y()).y2();
      (new Y()).y2();
      return v;
   }
}
```

A. `3`
B. `4`
C. `6`
D. Compile-time error: "method `y1` cannot be applied to type `int`".

b) **[0.5 marks]** Which of the following statements is true?
A. A class has always either zero or one constructors.
B. Primitive types have methods.
C. An object of type `String` is immutable, that is, its value cannot change.
D. Data structures of type `Collection` can store primitive types.

c) **[0.5 marks]** Which of the following statements is true?
A. An abstract class may have fields whereas an interface does not.
B. An interface may contain implemented methods.
C. A class that implements an interface needs not to implement all methods of that interface.
D. It is possible to instantiate an abstract class, but the same is not true for interfaces.

d) **[0.5 marks]** Consider the following code and identify what happens when we try to compile/execute it.

```
try { }
catch (Exception e) { }
catch (IOException e) { }
```

A. It compiles and executes without any problem.
B. There is a compile-time error because the try block does not have body.
C. There is a compile-time error for a reason different from B.
D. There is a run-time error and an exception is thrown.

e) **[0.5 marks]** Consider the following code and identify what happens when we try to compile it.

```
public class X {
   int x1(int s) {return 0;}
   float x1(int i) {return 1;}
}
```

A. It compiles without errors.
B. An error occurs: "constructor not defined".
C. An error occurs: "method `x1` defined twice".
D. Both errors described in B and C occur.

Name: Number:

4 – [1.0 marks] Describe the difference between a field in a class (`c_xpto` in the code) and a variable declared inside a method (`m_xpto` in the code), where `SOMETYPE` in the code represents any type offered by Java or defined by the user:

```
public class Xpto {
      SOMETYPE c_xpto;

      public void xpto(){
            SOMETYPE m_xpto;
            // ...
      }

      // ...
}
```

In particular, identify the modifiers that both can have, as well as its meaning, and default initialization.

Name: Number:

Part III -- XML (1 mark)

5 – [1.0 marks] Consider an element `Xpto` whose content is a string, with an attribute `x` of type string with a fixed value “`xpto`”. Provide the respective DTD.

```
<!ELEMENT Xpto (#PCDATA)>
<!ATTLIST Xpto x CDATA #FIXED “xpto”>
```

Part IV -- Internet (1 mark)

6 – [1.0 marks] Develop a java applet that contains a button with text “Click!” and a text field (not editable) that shows the number of clicks done so far. Do not provide the HTML to load the applet.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class MyApplet extends Applet implements ActionListener {

        static final String click ="Click!";
        private TextField text;
        private int nClicks=0;

        public void init(){
                Panel panel = new Panel();
                Button button = new Button(click);
                button.addActionListener(this);
                panel.add(button);

                text = new TextField(nClicks+" clicks!");
                text.setEditable(false);
                panel.add(text);

                add(panel);
        }

        public void actionPerformed(ActionEvent e) {
                nClicks++;
                text.setText(Integer.toString(nClicks)+" clicks!");
        }

}
```