Name: Number:

Object Oriented Programming 2012/13

Final Exam
June 12th, 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 that students must attend to a series of seminars in order to pass some courses. Courses must have a name, an ID constituted by letters and digits, and a minimum number of seminars to attend. A seminar must have a title, a number and a date; it also contains the minimum number of students needed to be enrolled in order to work, a maximum number of students enrolled, as well as a classroom. Each course may require a different number of seminars and seminars may work in the context of different courses. To pass a given course students may attend a minimum number of seminars associated with the course; different students may attend a different set of seminars. A seminar has at most one instructor; some seminars might not have an instructor at all. An instructor may instruct several seminars and has a name and e-mail address.

Students must store their name, e-mail address and student number. They should hold the courses they are enrolled in. To pass in these courses they must attend a minimum number of seminars provided by the course, so students start by showing their interest in participating in some seminar. Seminars should hold a waiting list of interested students. When enough students show interest in a certain seminar they are enrolled automatically in that seminar. When students attend to seminars they receive marks. These individual marks are then used to compute the final mark of each pair student and course; individual and final marks should be stored somewhere.

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

b) [0.5 marks] Define an UML object diagram considering a seminar with two students interested in; this seminar has an instructor. Set all needed attributes/associations with some dummy values.

Part II -- Java (6.5 marks)

2 – A binary search tree (BST) is an ordered data structure with the following properties: (i) every node has a key-value pair; (ii) every node's key is larger than all keys in its left subtree and smaller than all keys in its right subtree. Provide an implementation of a BST, named `CBST`. The `CBST` class should be generic with type parameters `K` and `V`, for the key and value, respectively. In addition, `K` must implement `Comparable`.

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

a) **[0.5 marks]** Provide the skeleton of the necessary classes with fields and constructors. For the `CBST` class only a no-arg constructor is needed.

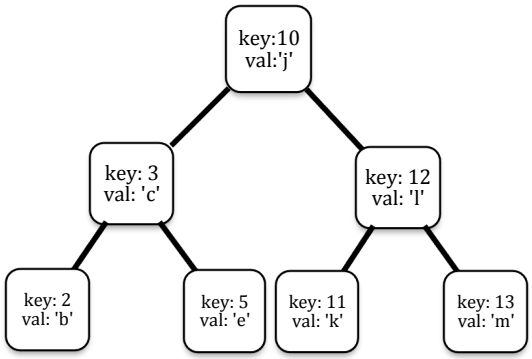
b) **[0.75 marks]** Provide a method `put` that receives two parameters, a `key` of type `K` and a `value` of type `V`, and inserts the `key-value` pair in the BST; it returns `void`. If the `key` being inserted already exists in the BST then the new `value` is stored within that `key`. Note: for simplicity, assume that the root of the BST is always the first pair inserted, and so the tree does not need not to be balanced.

c) **[0.5 marks]** Provide a method `get` that receives a parameter `key` of type `K` and returns a `value` of type `V` that contains the `value` associated with that `key` in the BST. If the `key` does not exist in the BST then `null` is returned.

d) **[0.75 marks]** Provide an implementation of the `toString` method. This method should return a string with the nodes in ascending order of the keys (see example in next page).

e) **[0.5 marks]** Provide a `main` method where (i) a `CBST` is created; (ii) a few elements are added to `CBST`; (iii) the `CBST` is printed to the terminal; and, (iv) some elements of the `CBST` are got and printed to the terminal.

Name: Number:



This is a BST where the key is an integer number and the value is a character. The `toString` method for this BST should return a string as

```
2:[2,b] 1:[3,c] 2:[5,e] 0:[10,j] 2:[11,k]
1:[12,l] 2:[13,m]
```

where the number `k` in `k:[i,j]` indicates the depth in the tree; `i` is the key and `j` is the value. The nodes are presented in ascending order of the keys.

```
public class CBST<K extends Comparable<K>,V> {

    private Node<K,V> root;

    /* Alternatively, it could be defined in a different source file! */
    public static class Node<K,V> {
      Node<K,V> left;
      Node<K,V> right;
      K key;
      V value;

      public Node(K key, V value){
            this.key=key;
            this.value=value;
      }

      public String toString(){
            return "["+key+","+value+"]";
      }
    }

    private Node put(Node<K,V> node, K key, V value){
      if (node==null) return new Node<K,V>(key,value);
      int res = key.compareTo(node.key);
      if (res==0) node.value = value;
      else if (res<0) node.left = put(node.left,key,value);
      else node.right = put(node.right,key,value);
      return node;
    }

    public void put(K key, V value){
      root = put(root,key,value);
    }
```

Name: Number:

```
    public V get(K key){
      Node<K,V> node = root;
      int res;
      while (node!=null){
            res = key.compareTo(node.key);
            if (res==0) return node.value;
            else if (res<0) node = node.left;
            else node = node.right;
      }
      return null;
    }

    private String visit(Node<K,V> node, int level){
      String res;
      if (node==null) return "";
      res = visit(node.left,level+1);
      res += level+":"+node.toString()+" ";
      res += visit(node.right,level+1);
      return res;
    }

    public String toString(){
      return visit(root,0);
    }

    public static void main(String[] args){
      CBST<Integer,Character> bst = new CBST<Integer,Character>();
      bst.put(10,'j');
      bst.put(3,'c');
      bst.put(2,'b');
      bst.put(5,'e');
      bst.put(12,'l');
      bst.put(13,'m');
      bst.put(11,'k');
      System.out.println(bst);
      System.out.println(bst.get(5));
      System.out.println(bst.get(12));
    }

}
```

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.

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

a) **[0.5 marks]** What is the value of the fields `x` and `y` after the construction of an object `X`?

```
class X {
      static int x;
      int y=x;
      static{x++;}
      {y=y+x;}
}
```

A. `x=0 and y=2`
B. `x=0 and y=1`
C. `x=1 and y=2`
D. `x=1 and y=1`
E. None of the above

b) **[0.5 marks]** What is printed in the terminal?

```
Integer[] tiVar1= new Integer[3];
int[] tiVar2 = new int[3];
System.out.print(java.util.Arrays.toString(tiVar1));
System.out.print(java.util.Arrays.toString(tiVar2));
```

A. `[null,null,null][null,null,null]`
B. `[null,null,null][0,0,0]`
C. `[0,0,0][0,0,0]`
D. `[0,0,0][null,null,null]`
E. None of the above

c) **[0.5 marks]** If `Crossword` extends `Puzzle` and implements `Solvable`, and

```
Puzzle p = new Puzzle();
Crossword cw = new Crossword(10, 20);
```

are declared, which of the following statements will cause a compile-time error?

A. `p = cw;`
B. `cw = new Puzzle();`
C. `p = new Crossword(10, 20);`
D. `Solvable x = new Crossword(10, 20);`
E. All of the above will compile with no errors.

d) **[0.5 marks]** Which of the following statements is true?

A. A static variable cannot be initialized in a constructor.
B. A static variable must be declared final.
C. An instance variable can't be declared final.
D. A static method can't access an instance variable.
E. Only a static method can access a static variable.

e) **[0.5 marks]** The method `mixup` is defined as follows:

```
String mixup(String word){
      int len = word.length();
      if (len==1) return "";
      else return mixup(word.substring(0,len-1))+ word.charAt(len-1);
}
```

What is the value of the string returned by `mixup("IDEAL")`?

A. `IDEAL`
B. `IDEA`
C. `LEAD`
D. `LEDA`
E. `DEAL`

`public String substring(int beginIndex, int endIndex)`
Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex-1`. Thus the length of the substring is `endIndex-beginIndex`.

4 – [1.0 marks] Many hold that the `instanceof` operator should be used only as a last resort. Justify this assertion focusing in two aspects: (i) a better alternative; (ii) a common exception in this guideline.

From Effective C++, by Scott Meyers: "Anytime you find yourself writing code of the form *if the object is of type T1, then do something, but if it's of type T2, then do something else*, slap yourself.

(i) Use polimorphism → needs a deeper explanation!

(ii) Equals method → needs a deeper explanation!

Name: Number:

Part III -- XML (1 mark)

5 – [1.0 marks] Consider a XML with root `Node` containing two elements, `Ypto` followed by `Xpto`. `Xpto` is an empty element with attribute `x` which is a reference to another attribute called `y` of the element `Ypto`. The element `Ypto` contains a string. Both attributes (`x` and `y`) are mandatory. Provide the DTD (including the DOCTYPE) and two XML documents, one valid and other invalid.

```
<!DOCTYPE Node [
  <!ELEMENT Node (Ypto,Xpto)>
  <!ELEMENT Xpto EMPTY>
  <!ELEMENT Ypto (#PCDATA)>
  <!ATTLIST Xpto x IDREF #REQUIRED>
  <!ATTLIST Ypto y ID #REQUIRED>
]>

Valid XML:
<Node>
      <Ypto y="a1">Hello!</Ypto>
      <Xpto x="a1"/>
</Node>

Invalid XML:
<Node>
      <Xpto x="a1"/>
      <Ypto y="a1">Hello!</Ypto>
</Node>
```

Part IV -- Internet (1 mark)

6 – [1.0 marks] Consider a `JApplet` class, named `MyApplet`, that receives two parameters, namely, `color` and `size`. Provide an HTML page with the appropriate tag to load the applet code. Fill all required and non-specified elements and attributes with dummy values.

```
<html>
<title>My Applet</title>
<hr>
<applet code="MyApplet.class" width="300" height="300">
<param name="color" value="blue">
<param name="size" value="30">
</applet>
<hr>
</html>
```