COMP348/1 CC: Principles of Programming Languages
(Summer 2, 2009)

Final Exam

Professor: J. Bentahar
Date: Monday, August 17, 2009
Duration: 3 hours

INSTRUCTIONS:

- The use of ENCS calculators is permitted.
- This exam is 7 pages long, including the cover page. Check that your copy is complete.
- This exam is out of 100 points.
- This is a closed book examination.

I- True or False (12 Points)

1- An abstract class in Java should have at least one abstract method

2- In Java, all methods in an abstract class are automatically abstract

3- The methods in Java interfaces should be preceded by the keyword abstract.

4- In Java, all methods in a final class are automatically final

5- In prolog, variables should be capital letters

6- Suppose `x` is an object. `puts x` and `puts x.to_s` will produce the same thing in Ruby

II- Exercises

1- Without using or defining size or length functions, write down a Prolog program that returns true only if a list contains exactly one element. (4 Points)

Examples
[1, 2, 3, 8] → false
[8] → true
[[]] → true
[[4, 5]] → true
[[4, 5], 4] → false

2- Write down a Prolog program that returns true if a list is sorted. Suppose that the list does not contain internal lists. (5 Points)

Examples
[1, 2, 2, 3, 8] → true
[8, 4, 2] → false
[7] → true
[] → true

3- Write down a Prolog procedure: `series(N, Total)`, which sums up the series
1 + 1/2 + 1/3 + 1/4 + ... + 1/(N-1). (4 Points)

4- Consider the following code: (25 Points)

```
class Myclass1
private
   def privMethod1
   end
protected
  def protMethod1
  end
public
   def pubMethod1
   end
end # class Myclass1

class Myclass2 < Myclass1
  def initialize (id)
    @@count += 1
    @id = id
  end
private
   def privMethod2
   end
protected
   def protMethod2
   end
public
   attr_accessor :@@count = 0
   def pubMethod2
   end
   def Myclass1.Method2
   end
   def to_s
     return "Object id: ", @id
   end
end # class Myclass2

obj1 = Myclass2.new("3")
obj2 = Myclass2.new(8)
obj3 = Myclass1.new()
puts Myclass2.@@count
```

a- In which language this code is written?

b- What is the default access control of "`initialize`"?

c- Indicate what statements are wrong in this code.

d- Suppose that each class is stored in a separate file (one file for `Myclass1` and another file for `Myclass2`). Do we need to add some code in this case? If yes, what is the code we need to add and at which level.

e- Suppose that the code is correct, give the output of the following:

```
1-  ObjectSpace.each_object(Myclass2) {|x| puts x.to_s}
2-  puts obj1.respond_to?("privMethod1")
3-  puts obj1.respond_to?("protMethod1")
4-  puts obj1.respond_to?("pubMethod1")
5-  puts obj1.respond_to?("privMethod2")
6-  puts obj1.respond_to?("protMethod2")
7-  puts obj1.respond_to?("pubMethod2")
8-  puts obj3.respond_to?("privMethod1")
9-  puts obj3.respond_to?("protMethod1")
10- puts obj3.respond_to?("pubMethod1")
11- puts obj3.respond_to?("privMethod2")
12- puts obj3.respond_to?("protMethod2")
13- puts obj3.respond_to?("pubMethod2")
14- puts obj1.respond_to?("Method2")
15- puts Myclass1.respond_to?("Method2")
16- puts Myclass2.respond_to?("Method2")
```

5- Give the output of the following: (8 Points)

a-

```
car (cdr (cdr '(410 (510 (250 20) (30)) (160 (280 70) (340)))))
```

b-

```
puts [1,2,3] & [3,4,5]
```

c-

```
puts [1,2,3] - [3,4,5]
```

d-

```
alpha = ["a","b","c","d","e","f"]
puts "pop=" + alpha.pop
```

6- Write down a Lisp function that takes as argument a list and returns another list that contains the same elements but without repetition. (4 Points)

Examples
(3 4 3 2 4 5) → (3 2 4 5)
(1 2 1 4 3 4 5 5 6) → (2 1 3 4 5 6)
(8 7 9 4 1) → (8 7 9 4 1)

7- We would like to display the following: (5 Points)

```
anu1 cau1 doh1 bef1.
```

To do that you need to complete the following Ruby code by adding only **ONE** statement.

```
a = ["ant", "cat", "dog", "bee"]
```

8- Let us consider the following code: (7 Points)

```
(defclass circle ()
  ((radius :accessor circle-radius)
   (center :accessor circle-center)))
(setf c (make-instance 'circle))
(setf circle-radius c 1
```

a- In which language this code is written?
b- The last statement has only one "`(`". Complete this statement by adding the missing parentheses and justify your answer (if you add a parenthesis, justify why it is added).
c- `circle-radius` has a specific name in the language in which the code is written. Indicate this name.
d- Give the output of this program.

9- Consider the class definitions below. Use the code in `class Dispatch` to briefly explain *all explicit responsibilities of the compiler (if ok why, if not ok why) and the run-time system*. If a statement does not compile, provide a brief explanation and consider it as being commented out. (14 Points)
NOTE: Give the answer under the form of a table like:

Line	STATEMENT	COMPILER	RUN-TIME SYSTEM

```
   public interface MyIF {
     public void call();
    }
    public class C1 {
     public static String name="--static variable in C1";
     public void callme() {System.out.println("--callme() in
C1.");}
     public void callme(String s) {System.out.println(s);}
    }

    public class C2 extends C1 implements MyIF{
     public static String name="--static variable in C2";
     public void callme() {System.out.println("method callme()
in C2");}
     public void call() {System.out.println("--call() from
interface");}
    }

1   public class Dispatch {
2     public static void main(String[] args) {
3          C1 c = new C2();
4          C2 cc1, cc2;
5          cc1 = c;
6          cc2 = (C2)c;
7          MyIF i = new C2();
8          MyIF j = new C1();
9          c.callme("Hello there!");
10         c.call();
11         i.call();
12         i.callme();
13         cc2.callme();
14         System.out.println(c.name);
15    }
16   }
```

10- Write down a CLOS program that represents a queue system, such that the head of the list is considered as the rear of the queue. This implies that during enqueue, an element is added to the head of the list, and during dequeue, the last element of the list is removed. (7 Points)

11- Write down a Ruby class, "Array2", with one operation: (5 Points)

`same-elements`: Testing if two arrays have the same elements (the order is not important).

Example
[1, 8, 7, 9] and [9, 8, 1, 7] have the same elements. However, [1, 8, 7, 9] and [9, 8, 1] do not have the same elements; and [1, 8, 7, 7] and [1, 7, 8] do not have the same elements.