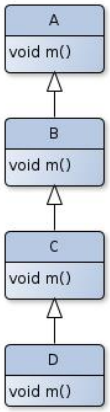
MC336 - Paradigmas de Programação
Exame - 10/12/2013

Questão 1 (Valor 4,0) Considere o diagrama de classes a seguir:



Todas as classes mostradas tem um método público `void m()` em comum. Suponha que as definições deste método nas respectivas classes são:

// A.java `public void m() {` `  System.out.print("a");` `}`	*// B.java* `public void m() {` `  System.out.print("b");` `}`
// C.java `public void m() {` `  System.out.print("c");` `}`	*// D.java* `public void m() {` `  System.out.print("d");` `}`

Responda: Quais das seguintes atribuições **NÃO** gerarão erro de compilação?

```
a = b; b = c; c = d; a = c; b = d; a = d;
b = a; c = b; d = c; c = a; d = b; d = a;
```

Veja a segunda página para as questões de Prolog e Lisp.

Nas questões de Prolog abaixo, faça com que os predicados pedidos falhem em tentativas de ressatisfação, a não que ser haja instruções explícitas em contrário no enunciado da questão.

Nas questões de Prolog e Lisp abaixo, podem ser definidos funções ou predicados auxiliares, além daqueles pedidos na questão. Neste caso, explique o que faz cada função ou predicado auxiliar. Você pode também utilizar funções ou predicados pré-definidos nas linguagens, vistos em classe ou na apostila.

Questão 2 (Valor 3,0) Escreva em Prolog um predicado `salta(L,R)` que é satisfeito quando `R` é uma lista contendo apenas os elementos das posições ímpares em `L`, sendo que a primeira posição é considerada de índice zero e portanto par. Exemplos:

```
?- salta([3,4,8,1,2,7,9], Z).
Z = [4,1,7]
?- salta([ ], Z).
Z = [ ]
?- salta([5], Z).
Z = [ ]
?- salta([6,5], Z).
Z = [5]
```

Questão 3 (Valor 3,0) Escreva em Lisp uma função `junta` que recebe duas listas `X` e `Y` de números ordenadas em ordem não decrescente (podendo haver portanto números repetidos) e retorna uma outra lista contendo todos os números das duas listas em ordem não decrescente, ou seja, o resultado possui todos os elementos de `X` mais os de `Y` e todos em ordem não-decrescente. Exemplos:

```
(junta '(2 3) '(-1 0 1))           ≡ (-1 0 1 2 3)
(junta '(-1 1 2 3) '(-5 0 0 0 1)) ≡ (-5 -1 0 0 0 1 1 2 3)
(junta '() '(3 7 9))               ≡ (3 7 9)
(junta '(2 3 4) '())               ≡ (2 3 4)
```

Boa sorte a todos!