Instituto Superior Técnico

Programação Avançada – 2009/2010

Segundo Exame – 13/7/2010

Número: _____

Nome: _____

Escreva o seu número em todas as folhas da prova. O tamanho das respostas deve ser limitado ao espaço fornecido para cada pergunta. Pode usar os versos das folhas para rascunho. A prova tem **5** páginas e a duração é de **2.0 horas**. A cotação de cada questão encontra-se indicada entre parêntesis. Boa sorte.

1. (1.0) O que é a reflexão? Quais são os graus de reflexão que conhece? Explique e exemplifique.

   Reflexão é a capacidade que um sistema computacional tem de manipular uma representação da sua estrutura e comportamento durante a sua própria execução.

   Existem dois* graus *de Reflexão:

   - ***A Introspecção é a capacidade de um sistema de* observar *a sua própria estrutura e comportamento. Por exemplo, saber "Quantos parâmetros tem a função*** `foo`***?" é introspecção.***
   - ***A Intercessão é a capacidade de um sistema de* modificar *a sua própria estrutura e comportamento. Por exemplo, "Mudar a classe desta instância para*** `Bar`***" é intercessão.***

2. (1.0) De forma a permitir reflexão, a linguagem Java reificou vários conceitos. Indique três desses conceitos.

   Java reificou o conceito de classe, o conceito de método e o conceito de atributo (*field*).

3. (1.0) O que é a meta-programação? Explique.

   A meta-programação é a escrita de programas que produzem outros programas. Um meta-programa manipula uma estrutura de dados que representa outro programa.

4. (2.0) Há quem defenda que as interfaces da linguagem Java podem ser usadas para implementar herança múltipla. Concorda? Explique a sua resposta.

5. (2.0) Em CLOS, qual é a diferença entre uma função genérica, um método primário e um método auxiliar? Explique.

   A função genérica é uma função que pode ser particularizada para diferentes tipos de argumentos (ou para diferentes argumentos). Para além disso, ela armazena todos os métodos que a especializam. Um método primário é o método que define o comportamento fundamental do método efectivo e determina o valor a retornar. Um método auxiliar é um método que modifica esse comportamento fundamental.

6. (1.0) A função `gensym` existe há já muitos anos em quase todos os dialectos de Lisp. O que faz? Onde é usada e para quê? Explique.

   A função `gensym` ***devolve um novo símbolo único de cada vez que é invocada. É usada fundamentalmente para a definição de macros de modo a evitar problemas de múltipla avaliação e de captura indevida de* bindings.**

7. (1.0) Considere a seguinte função definida em Common Lisp:

```
(defun foo (x f)
  (funcall f (bar #'g x)))
```

Reescreva a definição anterior em Scheme.

```
(define (foo x f)
  (f (bar g x)))
```

8. (3.0) Pretende-se que desenvolva um mecanismo para instrumentar programas Java que permita medir a sua *performance*. Em particular, estamos interessados em saber, para cada método, qual o tempo gasto na sua execução.

Proponha um mecanismo de intercessão baseado em Javassist que permita contabilizar o tempo gasto com a invocação de cada método de uma determinada classe. O seu mecanismo deverá disponibilizar uma forma de, num dado ponto do programa, o programador inserir uma instrução cuja execução provoque a escrita do estado actual dessa informação. Não é necessário descrever exaustivamente a implementação do seu mecanismo mas inclua toda a informação que considerar relevante para que outra pessoa (conhecedora de Javassist) possa realizar essa implementação sem problemas.

9. (2.0) A linguagem Pascal permite definir funções dentro de outras funções mas não permite que as funções possam ser retornadas como resultados ou guardadas em variáveis. Por outro lado, a linguagem C permite retornar funções como resultados e permite guardá-las em variáveis, mas não permite definir funções dentro de outras funções.

   (a) (1.0) Qual é o problema que está por detrás destas limitações do Pascal e do C? Explique.

   (b) (1.0) O que é que sugere que se faça num avaliador meta-circular para eliminar aquelas limitações das linguagens Pascal e C? Explique.

   Associar a cada função o ambiente léxico em que foi definida.

10. (1.0) Algumas linguagens consideram uma ordem de avaliação dos operandos de um operador (por exemplo, da esquerda para a direita) enquanto que outras deixam essa ordem por especificar. Em que condições considera uma das opções preferível à outra? Explique.

    Quando uma linguagem promove o uso de efeitos secundários, é normalmente preferível que o programador possa contar com uma ordem de avaliação da esquerda para a direita.

11. (2.0) Quais são os problemas que a programação orientada a aspectos pretende resolver? Como é que o consegue fazer? Explique.

12. (1.0) Descreva, por palavras suas, qual o efeito do seguinte aspecto em AspectJ:

```
aspect ShapeListening {
    private Vector<Screen> Shape.listeners = new Vector<Screen>();

    pointcut changeShape(Shape s):
      call(void Shape+.set*(..)) && target(s);

    after(Shape sh): changeShape(sh) {
        for(Screen sc : sh.listeners) {
            sc.redraw(sh);
        }
    }
}
```

13. (2.0) Um triplo Pitagórico são três números inteiros positivos i, j, e k, tais que $i^2 + j^2 = k^2$. Utilizando os operadores não-determinísticos `amb` e `fail`, defina uma função (e as funções auxiliares que achar por bem definir) que recebe um parâmetro n e computa não-deterministicamente um triplo Pitagórico com $i \leq j \leq k \leq n$.

```
(define (an-integer-between a b)
  (if (> a b)
      (fail)
      (amb a
           (an-integer-between (+ a 1) b))))

(define (a-pythagorean-triple n)
  (let ((i (an-integer-between 1 n)))
    (let ((j (an-integer-between i n)))
      (let ((k (an-integer-between j n)))
        (if (= (+ (* i i) (* j j)) (* k k))
          (list i j k)
          (fail))))))
```