Segunda Prova de Paradigmas de Linguagens de Programação
Universidade Federal de São Carlos
23 de maio de 2007
Prof. José de Oliveira Guimarães

1. (2.4) Considere o método

```
proc testFish(f : Fish) : boolean
   begin
   return f.livesInSea() or f.livesInLake();
   end
```

Na chamada "`f.livesInSea()`", escreva que conferências/buscas seriam feitas nos seguintes casos:

(a) em Java e em tempo de compilação;

(b) em Java e em tempo de execução;

(c) em POOL-I e em tempo de compilação;

Resposta:

(a) o compilador verifica se a classe declarada de `f`, `Fish`, possui um método chamado `livesInSea`, sem parâmetros. Se não tiver, a busca por este método continua na superclasse de `Fish` (se existir), superclasse da superclasse e assim por diante. Se o método não for encontrado, o compilador sinaliza um erro;

(b) o sistema de tempo de execução procura por um método `livesInSea` sem parâmetros na classe do objeto referenciado pela variável `f`. Se o método não for encontrado lá, a busca continua na superclasse desta classe, superclasse da superclasse e assim por diante;

(c) idem ao item (b).

2. (2.6) Considere o método

```
proc testFish(f : Fish) : boolean
      var a : Animal;
   begin
   a = f;
   ...
   end
```

Na atribuição "`a = f`", escreva que conferências/buscas seriam feitas nos seguintes casos:

(a) em Java e em tempo de compilação;

(b) em POOL-I e em tempo de compilação.

Resposta:

(a) se `Animal` é uma classe, o compilador confere se `Fish` é uma subclasse direta ou **indireta** de `Animal`. Se `Animal` é uma interface, o compilador confere se `Fish` 1) implementa esta interface direta ou indiretamente (confere se `Fish` implementa uma interface que herda direta ou indiretamente de `Animal`) ou 2) herda direta ou indiretamente de uma classe que implementa `Animal`;

(b) o compilador confere se a classe `Fish` é um subtipo de `Animal`; isto é, se `Fish` possui pelo menos todos os métodos públicos de `Animal`. Mas especificamente, o tipo de uma classe é o conjunto das assinaturas de seus métodos públicos. O compilador confere se o tipo de `Animal` está contido no tipo de `Fish`.

3. (3.0) Com relação a linguagens funcionais, responda às seguintes questões:

(a) (1.0) explique o que é transparência referencial e faça um exemplo de seu uso;

(b) (2.0) porquê estas linguagens são de alto nível ? Cite três motivos (os motivos relacionados ao gerenciamento de memória contam como um motivo).

Resposta:

(a) uma linguagem funcional possui transparência referencial quando o valor de retorno de cada função depende apenas dos valores dos parâmetros. Por exemplo, a expressão "`f(x, y) + f(x, y)`"pode ser substituída por "`2*f(x, y)`". Ou a expressão "`f(x) + h(y, z)`"pode ser avaliada por dois processadores: um pode avaliar a chamada a `f` e o outro a chamada a `h`. Uma não interferirá na outra;

(b) 1) a alocação e a desalocação de memória são automáticos. O programador não precisa se preocupar com detalhes irrelevantes para o problema como gerenciamento de memória; 2) não há atribuição e portanto podemos entender o significado de uma função estaticamente, sem imaginar o que ocorre durante a execução. As partes de uma função podem ser compreendidas independentemente umas das outras; 3) a recursão é utilizada no lugar da iteração, sendo que programas recursivos, sem efeitos colaterais, são mais fáceis de entender e 4) várias linguagens funcionais oferecem mecanismos que aumentam a abstração. Como exemplo, em SML/Haskell temos a definição de funções usando "ou", que substitui o if, diminuindo os detalhes do programa e aumentando a abstração. Em Haskell temos a avaliação preguiçosa, que retira do programador a preocupação se certo trecho será avaliado ou não. Em FP temos diversas formas funcionais que facilitam a programação e permitem realizar diversas tarefas com um único comando.

4. (2.0) Qual o tipo da função `f` em SML dada abaixo ? Qual o resultado de `f([1,2,3], 7, [-1,0,1])` ?

```
proc f([], x, []) is [] |
     f( h::t, x, y::u )
        is
          if y >= 0
          then
             h::x::f(t, x, u)
          else
             h::x::x::f(t, x, u);
```

Resposta: O tipo da função é `'a list` $\times$ `'a` $\times$ `integer list` $\longrightarrow$ `'a list`. O resultado é `[1, 7, 7, 2, 7, 3, 7]`.