MAC5754 – Conceitos de Linguagens de Programação

Prova 2

20/06/2016

Você tem 100 minutos para realizar esta prova. Utilize seu tempo com cuidado, resolva antes as questões que considerar mais fáceis.

Utilize o espaço reservado para as questões. Ele é um indicativo da quantidade de informação esperada na resposta.

|  |  |  |
| --- | --- | --- |
| Questão | Valor | Nota |
| 1 | 2 |  |
| 2 | 2 |  |
| 3 | 2 |  |
| 4 | 1 |  |
| 5 | 1 |  |
| 6 | 2 |  |
|  |  |  |
| Total | 10 |  |

1. (2 pontos)Explique o conceito de avaliação por demanda. Qual sua relação com efeitos colaterais? Como ele é implementado? Seja detalhado, ocupe todo o espaço de resposta.
2. (2 pontos) O que faz com que a implementação de C++ seja muito mais eficiente do que a de Java? Em particular qual o mecanismo que garante que o envio de métodos, em tempo de execução seja apenas um call indireto? Como isso é Possível em C++. quando declaramos uma variável de classe A, e ela recebe um objeto de uma subclasse que reimplementa um método da Classe A, que um método desta classe chame o método redefinido pela subclasse?
3. (2 pontos) Em classe vimos várias implementações de objetos e classes. Veja a implementação descrita abaixo e diga o que ela faz e o que não é feito em relação à versão final? Seja detalhado

(define ClasseA (lambda ()

(letrec ( [valorA 0]

[func1 (lambda (parlist) (+ 5 (self 'm2 '())))]

[func2 (lambda (parlist) 10)]

[self (lambda (msg parlist)

(case msg

[(m1) (func1 parlist)]

[(m2) (func2 parlist)]))])

self)))

(define ClasseB (lambda()

(letrec ( [Superobj (ClasseA)]

[valorB 0]

[\_m2 (lambda (parlist) 5)]

[self (lambda (msg parlist)

(case msg

[(m2) (\_m2 parlist)]

[ else (Superobj msg parlist)]))])

self)))

1. (1 ponto) Em classe vimos uma implementação de herança em orientação a objetos. Imagine agora uma implementação ligeiramente diferente. Quando mandamos uma mensagem a um objeto da classe A, o método é procurado e se não é encontrado a classe do objeto é temporariamente mudada para a superclasse de A, e assim sucessivamente. A classe original é guardada e reinstalada após a execução da mensagem. Há algum problema com este esquema? Ele muda de alguma maneira a semântica da linguagem? Se muda, ela fica parecida com qual linguagem?
2. (1 ponto) Em classe vimos que Smalltalk é uma linguagem puramente orientada a objetos, ao contrario de Java e C++ que têm características de linguagens híbridas. Isso significa que em smalltalk não há condicionais nem operações aritméticas, apenas classes e objetos. Explique como Smalltalk implementa execução condicional de comandos tipo (IF-THEN-ELSE) e como as operações sobre números são vistas pela linguagem.
3. (2 pontos) Existem duas maneiras de ser reutilizar código em uma linguagem Orientada a Objetos: herança e delegação. Quando devemos usar herança?