Introdução à Programação (IF-669)
2º Exame Escrito

Adriano Sarmento
22 de junho de 2009

1ª Questão

Por que podemos dizer que "em Java toda variável do tipo referência é potencialmente polimórfica? Como se dá o polimorfismo em Java? (2,0 p)

2ª Questão

Considere as classes abaixo:

```
public class Medico {
   private String nome;
   private double salarioBase;

   public Medico(String nome,double salario) {...}

   public double calcularSalario(){
      return salarioBase;
   }

   public static String getProfissao() {
      return "Medico";
   }

   public static String getTurno() {
      return "Diurno";
   }
   ...
}
```

```
public class Plantonista extends Medico {
   private double adicionalNoturno;

   public Plantonista(String nome,double salario, double adicional){...}

   public double calcularSalario(){
      return (super.calcularSalario() + adicionalNoturno);
   }

   public static String getTurno() {
      return "Noturno";
   }
   public double getAdicional() { return this.adicionalNoturno;}
}
```

Considere agora o trecho de código abaixo:

```
Medico m1 = new Medico("Jose",2000);
Medico m2 = new Plantonista("Julia",2000,200);
Plantonista p1 = new Plantonista("Ana",3000,300);
```

Para cada linha de comando abaixo adicionado a este trecho de programa, descreva o que ocorrerá, justificando a resposta. No caso de erro, escreva qual é o tipo de erro (compilação ou execução) e explique o porquê do erro (2,0p).

(a) System.out.println("Profissao de Ana: "+ p1.getProfissao());

(b) System.out.println("Turno de Julia: "+ m2.getTurno());

(c) System.out.println("Salario de Julia: "+ ((Medico)m2).calcularSalario());

(d) System.out.println("Adicional de José: "+ ((Plantonista)m1).getAdicional());

3ª Questão

Considere que exista uma classe `Chamada` que representa uma chamada telefônica e que possua dois métodos: `String getNumeroChamado()` que retorna o número chamado; e `int duracao()` que retorna a duração em minutos da chamada.

Agora considere a classe `ColecaoChamada`, que representa uma coleção de chamadas telefônicas, e a interface `IteratorChamada`:

```
public class ColecaoChamada {
    ...
    public ColecaoChamada() {...}
    public IteratorChamada getIterator() {...}
    ...
}
```

```
public interface IteratorChamada {
    public Chamada next();
    public boolean hasNext();
}
```

Finalmente, assuma agora que exista um tipo `PlanoTelefone` que pode ser uma interface, classe abstrata ou classe concreta. Este tipo tem 3 métodos que são listados abaixo e que podem ser abstratos ou não:

```
public String getNomeCliente();
public String getNumeroTelefone();
public double calcularFaturamento();
```

(a) Defina o tipo `PlanoTelefone` em Java. Justifique a sua escolha entre interface, classe abstrata ou classe concreta. (2,0p)

(b) Defina a classe `PlanoPrePago` que é uma implementação ou especialização de `PlanoTelefone`. Esta classe deve armazenar informação sobre o valor pré-pago. O valor do faturamento é igual ao valor pré-pago. (1,0p)

(c) Defina a classe `PlanoPosPago` que é uma implementação ou especialização de `PlanoTelefone`. Esta classe deve armazenar informações sobre a tarifa inicial do plano, o número de minutos garantidos pela tarifa inicial, a tarifa por minuto extra e uma coleção de chamadas feitas. O valor do faturamento vai ser calculado a partir destas informações da seguinte forma:

- Se o total das durações das chamadas for dentro do número de minutos garantidos pela tarifa inicial, o valor do faturamento será igual à tarifa inicial
- Se o total das durações das chamadas ultrapassar o número de minutos garantidos pela tarifa inicial, deve-se multiplicar os minutos extras pela tarifa por minuto extra e somar com a tarifa inicial para calcular o valor do faturamento

O construtor desta classe deve ter como um dos parâmetros uma coleção de chamadas. (3,0p)

Obs 1: Todas as classes devem ter construtores

Obs 2: Devem-se empregar bem os modificadores de visibilidade (acesso) para não violar o princípio de *"information hiding"*