Teste de Programação Orientada aos Objectos

2013.06.12
Duração: **2h**

Leia o teste com muita atenção antes de começar.
Responda a cada Parte em folhas separadas.

Parte I - 6 valores

1. Recorde o projecto dos *Empregados*. No enunciado do projecto referia-se que:

   > Os empregados de uma empresa são caracterizados pela seguinte informação: código, nome e dias de trabalho efectivos no mês. Todos os empregados normais possuem um salário-dia fixo. No entanto, os gestores têm um prémio de gestão que multiplica o seu vencimento normal (ex: 1.25). Os motoristas, têm definido um valor por Km percorrido, e têm a si associado o número de Kms percorridos nesse mês para cálculo do seu salário final.

   Assuma que é necessário desenvolver a classe EmpresaSet, que deve ter como estrutura interna uma implementação de conjunto. A classe Empresa deve ter como funcionalidades os seguintes métodos:

   (a) `public boolean existeEmpregado(String cod)`, que verifica se um empregado de código dado existe;

   (b) `public Empregado getEmpregado(String cod)`, que devolve a informação de um empregado existente dado o seu código;

   (c) `public void addEmpregado(Empregado e)`, que insere um novo empregado (caso um empregado com o mesmo código já exista, o novo empregado não deve ser inserido);

   (d) `public Iterator<Empregado> listaPorOrdemDecrescenteSalario()`, que devolve uma estrutura ordenada decrescentemente pelo valor do salário (seja o mais completo possível).

Parte II - 5 valores

2. Considere agora que é necessário definir mais um tipo de empregado: o CEO. O CEO recebe por dia três vezes o salário normal de um empregado (que é calculado em função do número de dias de trabalho).

   Sabendo que tem a seguinte implementação da classe Empregado:

```
public abstract class Empregado implements Serializable {
   private static double salDia = 50.00;
   public static double getSalDia() { return salDia; }

   private String codigo;
   private String nome;
   private int dias;
   ...
   public abstract double salario() {...}
   public abstract String toString() {...}
   public abstract Empregado clone();
}
```

   defina a classe CEO, apresentando o seu construtor de cópia, a redefinição dos métodos `salario` e `toString` e a definição do método abstract (`clone`) da classe `Empregado`.

3. Quer Empresas, quer Empregados, necessitam registar-se nas finanças.

   Sabendo que a classe DGCI está implementada do seguinte modo:

```
public class DGCI {
   private Map<String, Contribuinte> utentes;  // NIF -> Contribuinte

   public void regista(Contribuinte c) throws ContribuinteJaExiste {...}
   ...
}
```

   em que `Contribuinte` garante que todos os objectos na lista utentes possuem o método `public String getNIF()`, que devolve o número de contribuinte, responda às seguintes questões:

   (a) defina `Contribuinte`, justificando a escolha que faz

   (b) mostre (justificando) o que alteraria nas classes `Empregado` e `EmpresaSet` por forma a que possam ser registadas na DGCI

Parte III - 5 valores

4. Considere que se pretende gerir um livro de receitas culinárias. Este livro de receitas é constituído com sendo uma associação entre o nome da receita e a sua definição. Uma receita é constituída pelo seu nome, descrição, o conjunto de ingredientes (em que cada ingrediente tem o seu nome, quantidade, unidade de medida, e calorias) e ainda por uma sequência de passos de preparação (que são linhas de texto).

   Um aluno de POO definiu a classe `Ingrediente` como sendo:

```
public class Ingrediente {
  private String nome;
  private int quantidade;
  private String unidades;
  private int calorias;
  ...
}
```

   Responda às seguintes questões:

   (a) declare as classes `LivroDeReceitas` e `Receita`, definindo as suas variáveis de instância (ou de classe, caso existam);

   (b) implemente o construtor parametrizado de `Receita`;

   (c) codifique o método da classe `LivroDeReceitas`, `public int totalCalorias(String nomeReceita) throws ReceitaInexistenteException`, que determina o total de calorias de uma receita.

Parte IV - 4 valores

5. Relativamente ainda ao tema do grupo anterior, e à classe `LivroDeReceitas`, responda às seguintes questões:

   (a) codifique o método, `public Map<String, Set<String>> receitasPorIngrediente()`, que devolve uma estrutura de dados em que se associa a cada nome de ingrediente o nome das receitas onde ele é necessário;

   (b) implemente o método, `public void gravaObj(String filename, int calorias)`, que grava em modo binário, no ficheiro indicado, as receitas com número de calorias igual ou superior ao passado como parâmetro.