Teste de Paradigmas da Programação por Objectos

2012.06.05
Duração: **2h**

Leia o teste com muita atenção antes de começar.
RESPONDA A CADA PARTE EM FOLHAS SEPARADAS.

PARTE I - 7 valores

Considere que se pretende desenvolver um sistema de gestão de agendas electrónicas. As classes relevantes são a classe Tarefa e Agenda.

```
public class Tarefa {
  private String descricao;    // descrição da tarefa
  private float prioridade;
       // prioridade é um valor entre 0..1, sendo 1 a prioridade máxima
  private ArrayList<String> participantes;
      //lista dos nomes dos envolvidos na tarefa
  private GregorianCalendar inicio;
      //data e hora do início da tarefa
  private GregorianCalendar fim;
     //data e hora do término previsto da tarefa
  private boolean terminada;     //tarefa concluída?

  ...
}
```

1. Codifique o método `public boolean activa()`, que determina se uma tarefa ainda está em actividade, ou seja se já passou a data de início, ainda não chegou à data de término e não está terminada.

2. Seja agora a classe Agenda

```
public class Agenda {
  private String titular;    // nome do dono da agenda
  private ArrayList<Tarefa> tarefas;     //lista das tarefas agendadas

  ...
}
```

   (a) Desenvolva o método `public void addTarefa(Tarefa t)`.

(b) Codifique o método public Iterator<Tarefa> tarefasActivas(Comparator<Tarefa> ct), que devolve as tarefas activas de acordo com o critério de comparação desejado.

(c) Codifique um comparador de tarefas para que estas fiquem ordenadas em primeiro lugar por prioridade e depois por data de término.

(d) Desenvolva o método `public Set<Tarefa> tarefasEntreDatas(GregorianCalendar di, GregorianCalendar df)`, que devolve as tarefas que terminam no intervalo de datas especificado.

Parte II - 8 valores

Considere a arquitectura de classes definida na Figura 1, com as seguintes definições:

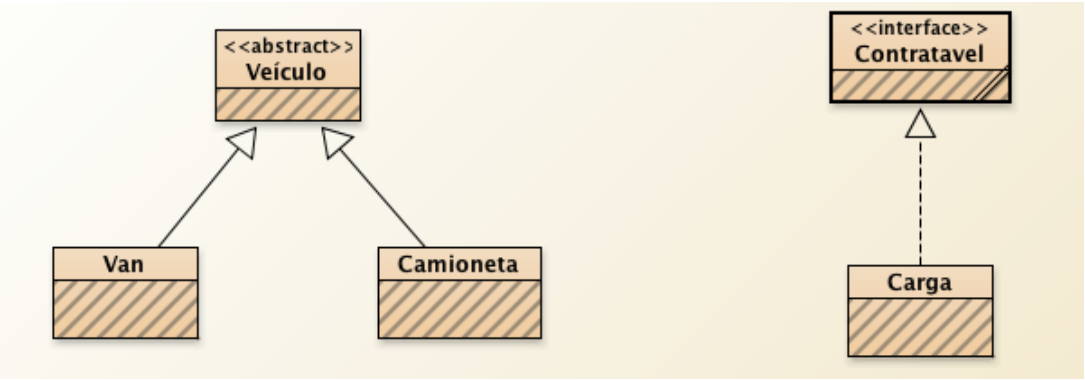

Figura 1: Veículos e Carga

```
public abstract class Veiculo {
  private String matricula;
  private ArrayList<Contratavel> servicosContratados;
     // relação das cargas já transportadas pelo veículo
  private double custoPorKm;

  ...
  public abstract double custoPorContrato(Contratavel ct);
     // calculo do custo do transporte da carga indicada

}


public interface Contratavel {
  public String getCliente(); //devolve o cliente que envia a carga
  public double getPeso();    //devolve o peso da carga
  public double getDistancia(); // devolve a distância a percorrer
}
```

3. Sabendo que uma Van acrescenta à informação de veículo a capacidade útil de carga e que o custo de transporte é calculado pela fórmula $custoPorKm * distancia * peso\ da\ carga/capacidade\ util\ do\ veiculo$ e que uma Camioneta acrescenta o nome do motorista, o peso do camião e um factor que aumenta o preço em função da emissão de gases para a atmosfera (dependente de camioneta para camioneta), ou seja o custo é calculado pela fórmula $custoPorKm * distancia * factor\ emissao\ gases$. Codifique as classes Van e Camioneta, identificando as variáveis de instância, os construtores e a concretização do método `double custoPorContrato()`, para cada classe.

4. Seja agora a classe Empresa declarada como:

```
public class Empresa {
   private String nomeEmpresa;
   private TreeMap<String, Veiculo> veiculos;  // Matricula -> Veiculo

   ...

}
```

   implemente os métodos:

   (a) `public double valorTodosContratos()`, que calcula o valor de todos os contratos efectuados pela empresa.

   (b) `Map<String,ArrayList<Veiculo>> veiculosPorCliente()`, que determina a tabela que relaciona cada cliente com os veículos que transportaram cargas suas.

   (c) `public String veiculoComMaisKms()`, que devolve a matrícula do veículo com mais kms acumulados nos serviços contratados.

   (d) `public boolean existeServicoComDistanciaMaiorQue(double dist)`, que determina se existe pelo menos um serviço contratado na empresa com distância superior a `dist`.

PARTE II - 5 valores

5. Imagine que a classe HashMap não existe na biblioteca de classes Java que tem instalada na sua máquina. No entanto, precisa dessa classe na medida em que o seu programa utiliza maps. Declare a classe MyMap que implementa a interface `Map<String, Veiculo>`, e apresente a implementação dos métodos `put`, `get`, `containsKey` e `values`. Escolha a estrutura interna que lhe pareça mais adequada (sem utilizar Maps!!).

6. Codifique um método, designado por `public void gravaMyMap(String filename, double valorRef) throws IOException`, que grava numa stream de objectos a informação que consta do map, gravando apenas os veículos com um valor de todos os contratos superior ao valor passado por parâmetro (`valorRef`).