PARADIGMAS DE PROGRAMAÇÃO IV – LESI
e
PROGRAMAÇÃO ORIENTADA AOS OBJECTOS - LCC

2º ANO/2º SEMESTRE – 2006/2007

EXAME de 1ª CHAMADA – 15 de Junho de 2007
Cotação: 20 valores Duração: 2h00m

RESPONDA A CADA PARTE EM FOLHAS SEPARADAS

PARTE I (8 valores)

A classe **Disciplina** apresentada a seguir possui um código, um conjunto de números de alunos inscritos, uma sequência das aulas previstas e um livro de presenças que vai associando a cada data de aula a folha de presenças contendo os números dos alunos que estiveram presentes na mesma.

```
public class Disciplina implements Serializable {
   // Variáveis de Instância
   private String codigo;
   private TreeSet<String> inscritos;
   private ArrayList<GregorianCalendar> previstas;
   private TreeMap<GregorianCalendar, TreeSet<String>> livroPres;
   …
}
```

Desenvolva o código dos seguintes métodos:

1) Método **aulasParaDar()** que determina o número de aulas por leccionar;

2) Método **perctPres()** que devolve a percentagem global de presenças, ou seja, o quociente entre a frequência verificada e a frequência total possível;

3) Método **aulasFreqAluno(String num)** que determina e devolve o conjunto das datas das aulas a que um dado aluno assistiu;

4) Método **naoFreq(String num)** que verifica se um aluno frequentou alguma aula;

5) Método que insere a data de uma aula e o conjunto dos números dos alunos presentes na aula;

6) Método **getLivroPres()** que devolve uma cópia do livro de presenças da disciplina;

PARTE II (12 valores)

Pretende-se desenvolver um sistema de gestão de um **Stand** de veículos. O stand comercializa de momento automóveis, motos, bicicletas e barcos. Sobre cada veículo é registada a seguinte informação: matrícula, marca, cor e preço base. Para os automóveis, motos e barcos acrescenta-se ainda a informação sobre consumo e cilindrada. Para os automóveis distingue-se também o tipo de combustível.
O preço final de um veículo deve ser facilmente determinável. Para os veículos com cilindrada registada, o preço final resulta de se somar o preço base com o produto do valor fixo de imposto de cilindrada para esse tipo de veículo multiplicado pela sua cilindrada.

A classe **Cliente** representa toda a informação necessária sobre um cliente registado, designadamente, o seu código, nome, telefone e lista de produtos já comprados.

```
public class Cliente implements Serializable {
       private String codigo;
       private String nome;
       private String telf;
       private ArrayList<String> compras;   // códigos dos produtos já comprados

   // métodos getX(), setX(..), toString(), equals() e clone() estão disponíveis
}
```

A classe **Stand** é estruturalmente definida como possuindo um nome, um ficheiro de clientes indexado por código e um ficheiro de veículos para venda indexado por matrícula, cf. se representa na declaração seguinte:

```
public class Stand implements Serializable {
       private String nome;                          // nome do stand
       private HashMap<String, Cliente> clientes;    // clientes por código
       private TreeMap<String, Veiculo> veiculos;    // veículos por matrícula
       ….
   // métodos getX(), setX(..), toString(), equals() e clone() estão disponíveis

}
```

1) Defina o mais completamente possível em estrutura e comportamento (excepto os métodos *equals(), gets e sets*) a classe abstracta **Veiculo** que servirá para representar o tipo de todos os veículos à venda no stand;
2) Defina de forma completa a classe **Automovel** subclasse de **Veiculo** (excepto *toString()* e *equals(),* bem como *gets* e *sets*);
3) Defina um método que permita inserir uma nova viatura no sistema;
4) Defina o construtor de cópia **public Stand(Stand st)**;
5) Defina um método que acrescente uma nova compra à lista de compras de um dado cliente.
6) Defina um método que determine o valor total dos veículos já vendidos a um dado cliente;
7) Defina um método que determine o número total de veículos de dado tipo;
8) Defina um método que determine o telefone do cliente de nome dado;

Prof. F. Mário Martins