Exame de Paradigmas da Programação IV e Programação Orientada aos Objectos
1ª chamada

2006.06.23
Duração: **2h**

Leia o teste com muita atenção antes de começar.
Responda a cada Parte em folhas separadas. Se utilizar a notação do Java 1.5 por favor indique-o em cada pergunta.

Parte I - 8 valores

1. Considere as seguintes definições Java:

```
public class EntradaCatalogo {
  private String codigo;
  private String descricao; //descricao do item

  public EntradaCatalogo(String t, String d) {
    this.codigo = t;
    this.descricao = d;
  }

  public String getCodigo() { return this.codigo;}
  public String getDescricao() {return this.descricao;}

  //...
}
```

sendo que o Catálogo é implementado como sendo uma tabela de entradas do tipo `EntradaCatalogo`.

```
public class Catalogo {
  private String nomeCatalogo;   //nome do catalogo
  private HashMap entradas;      //HaspMap com as entradas

  public Catalogo(String nomeCatalogo) {
    this.nomeCatalogo = nomeCatalogo;
    this.entradas = new HashMap();
  }
 //...
}
```

Tendo em conta esta modelação implemente os seguintes métodos de `Catalogo`:

(a) `void inserirEntrada(EntradaCatalogo ec) throws ExistingEntryException`

(b) `EntradaCatalogo getDescricao(String codigo) throws EntryDoesNotExistException`

(c) `boolean existeCodigo(String codigo)`

(d) `boolean equals(Object o)`

2. Considere agora que se pretendem criar outro tipo de Catálogos em que a informação associada a cada código, para além da descrição, contém também uma lista de *produtos semelhantes*. Crie as classes necessárias para o efeito, sabendo que instâncias da classe `CatalogoInteligente` deverão responder aos métodos de `Catalogo`, bem assim como aos métodos:

(a) `void addSemelhante(String codigo, String semelhante) throws EntryDoesNotExistException`

(b) `List getSemelhantes(String codigo) throws EntryDoesNotExistException`

(c) `boolean saoMesmoSemelhantes(String codigoA, String codigoB) throws EntryDoesNotExistException`

Parte II - 7 valores

3. Relembre o trabalho prático. Algumas das soluções apresentadas para a hierarquia das localidades previam a existência de um HashMap de pontos de interesse histórico nesta classe. A justificação dada era que se a localidade fosse histórica, este HashMap teria informação, caso contrário estaria vazio.

Seja então a classe Localidade escrita como:

```
public class Localidade {
  private String nome;
  private String distrito;
  private HashMap pontosInteresse;
  private boolean temPontosDeInteresse; //flag que diz se a localidade
                                        //é ou não histórica

  //...
}
```

Tendo em conta esta definição desenvolva os seguintes métodos de `Localidade`:

(a) `void inserirPontoInteresse(PontoInteresse pe) throws ???Exception`

(b) `PontoInteresse getPontoInteresse(String nomePontoInteresse) throws ???Exception`

(c) `boolean getTemPontosdeInteresse()`

(d) `void setTemPontosdeInteresse(boolean flag) throws ???Exception`

(e) `boolean invarianteEstado()`, que determina se a informação existente no estado de uma localidade é coerente.

(f) `void gravarFichObjectos(String filename) throws ???Exception`.

Parte III - 5 valores

Relembre o contexto do problema da Biblioteca Multimédia que foi sendo realizado nas aulas práticas. Como sabe, é declarada uma superclasse chamada ItemMultimedia e as subclasses CD e DVD.

Por forma a conseguir levar algums dos itens da biblioteca para reprodução no carro, criou-se um dispositivo que funciona implementando uma fila onde se acrescentam instâncias de ItemMultimedia. Crie uma classe, BibliotecaMobile, que implemente uma fila de espera de objectos multimédia, com os seguintes métodos:

(a) `insere(List itens)`, que insere a lista parâmetro

(b) `remove(int n)`, que remove os N primeiros itens da fila

(c) `ItemMultimedia seguinte()`, que retira o próximo item a ser tocado

4. Suponha agora que alguns dos itens existentes na Biblioteca Multimédia implementam a interface ComImagem.

```
public interface ComImagem {
public void addFoto(String foto); //a foto é mapeada de alguma forma numa String
public String getFoto();
}
```

Crie uma classe que seja uma BibPhotoMobile, isto é, que além da funcionalidade sobre musica, permite fazer um mosaico com a concatenação de todas as fotografias dos conteúdos existentes (desde que estes suportem fotografia).

Implemente o método `String mosaico()`.