Exame de Programação Orientada aos Objectos

MiEI e LCC
DI/UMinho

21/06/2016
Duração: **2h**

Leia o teste com muita atenção antes de começar e lembre-se de preservar sempre o encapsulamento das variáveis de instância.

RESPONDA A CADA PARTE EM FOLHAS SEPARADAS.

PARTE I - 6 VALORES

1. Considere as seguintes definições JAVA desenvolvidas para implementar um Dicionário:

```
public class Dicionario {
  private String nomeDic;                    //nome do dicionário
  private Map<String, Entrada> entradas; //termo -> definição

  public Dicionario(String nomeDic) {
    this.nomeDic = nomeDic;
    this.entradas = new HashMap<>();
  }
  ...
}

public interface Entrada {
  public String getTermo();
  public String getDefinicao();
  public boolean equals (Object o);
  public Entrada clone();
}
```

Tendo em conta estas declarações implemente os seguintes métodos de `Dicionario`:

(a) `void add(Entrada ed) throws ExistingEntryException` – adiciona uma nova entrada (o termo não pode estar já definido).

(b) `boolean exists(String termo)` – testa se um termo está definido.

(c) `Entrada get(String termo) throws EntryDoesNotExistException` – devolve a entrada relativa a um termo, caso exista.

(d) `Collection<Entrada> getAll()` – devolve uma colecção com todas as entradas do dicionário.

(e) `boolean sinonimos(String termo1, String termo2)` — determina se os dois termos têm a mesma definição.

Mude para uma nova folha

Parte II - 4 valores

2. Pretende-se agora implementar o método `Map<String, List<String>> getSinonimos()`, da classe `Dicionario`, que mapeia cada definição existente no dicionários para a lista de termos com essa definição:

   (a) Implemente o método utilizando iteradores externos.

   (b) Implemente o método utilizando iteradores internos.

3. Considere agora de definição:

```
public class EntradaBase implements Entrada, Comparable {
   ...
}
```

   Com base apenas na informação acima, escreva o método que define a ordem natural de `EntradaBase` como sendo a ordem decrescente dos termos.

4. Recorde a matéria leccionada nas aulas teóricas. Diga por palavras suas o que entende serem as principais diferenças entre os conceitos de *classe abstracta* e *interface*. Descreva em que situações de modelação de um problema é que consideraria mais vantajoso utilizar um ou outro.

Mude para uma nova folha

Parte III - 6 valores

5. Considere que se pretende desenvolver a classe `Matriz`. Foi decidido representar a matriz como uma lista de listas:

```
public class Matriz {
   private List<<List<Object>> linhas;
   ...
}
```

   Codifique cada um dos seguintes métodos:

   (a) `public Matriz(int n, int m)` — construtor que cria uma matriz n×m (os elementos da matriz deverão ficar com o valor `null`).

   (b) `public Object get(int l, int c) throws ArrayIndexOutOfBoudsException` — devolve o valor do elemento na posição (l,c)[1].

   (c) `public int size()` — calcula o número de elementos da matriz.

   (d) `public int count(Object o)` — conta quantos elementos da matriz são iguais ao objecto passado como parâmetro.

   (e) `public boolean equals(Object o)` — comparação de duas matrizes.

[1] Note que a classe `ArrayIndexOutOfBoundsException` é uma classe existente no Java pelo que não necessita defini-la.

Mude para uma nova folha

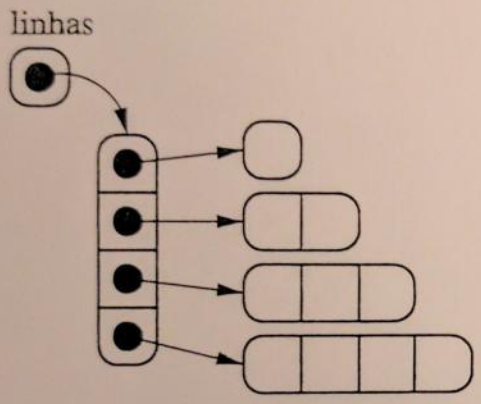
Parte IV - 4 valores

6. Considere agora que se pretende implementa a funcionalidade de guardar matrizes em ficheiro.

   (a) Escreva o método `public void save(String f)` que guarda a matriz num ficheiro em formato objecto.

   (b) Altere a declaração da classe `Matriz` para que o método da alínea anterior possa ser executado com sucesso.

7. Considere ainda a classe `Matriz`. Note que, com a definição apresentada acima, cada linha da matriz poderá ter um tamanho diferente das outras. Podemos, por exemplo, ter uma matriz triangular:



   (a) Escreva o construtor `public Matriz(int n)` que constroi uma matriz triangular de `n` linhas (os elementos da matriz deverão ficar com o valor `null`).

   (b) Escreva o método `public boolean tri()` que testa se a matriz é triangular.