Teste de Programação Orientada aos Objectos

MiEI e LCC
DI/UMinho

04/06/2016
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 - 5 VALORES

1. Considere que estamos a desenvolver um software para fazer a gestão de processos eleitorais, sendo necessário desenvolver a classe `ListaEleitoral` que permite registar os candidatos associados a um partido e fazer a gestão dos candidatos já eleitos/por eleger. Um dos programadores da empresa que está a desenvolver este software criou a classe que a seguir se apresenta.

```
public class ListaEleitoral {
  private String partidoPolitico;
  private Set<Candidato> eleitos;    // candidatos da lista já eleitos
  private List<Candidato>  porEleger;  // candidatos ainda por eleger
  ...
}
```

Implemente os métodos seguintes assumindo a existência da classe `Candidato`. Tenha em atenção que o código deve ser, sempre que possível, robusto não necessitando de desenvolver as classes de excepção, mas sendo necessário verificar se os métodos necessitam de lançar excepções.

(a) `public ListaEleitoral(String partido, Collection<Candidato> candidatos)` – construtor inicial que cria uma lista com os candidatos a serem eleitos.

(b) `public Candidato aEleger()` – devolve o próximo candidato a eleger.

(c) `public void elege()` – elege o próximo candidato (caso ainda existam candidatos por eleger). throws NoElegeVazioException

(d) `public void elege(int n)` – elege n candidatos se possível, caso contrário, deixa a lista inalterada.

(e) `public Collection<Candidatos> candidatos()` - que devolve uma colecção com todos os candidatos do partido.

Mude para uma nova folha

Parte II - 4 valores

2. Considere ainda a classe `ListaEleitoral` da parte I. Sabendo que a classe `Candidato` contém os seguinte métodos:

```
public class Candidato {
  ...
  public String getNome() {...}
  public int getIdade() {...}
  ...
}
```

   (a) Escreva o método `public TreeSet<Candidatos> eleitos()`, da classe `ListaEleitoral`, que devolve todos os candidatos já eleitos, ordenados segundo a ordem natural de Candidato.

   (b) Pretende-se que a ordem natural de `Candidato` seja a ordenação pelo nome do candidato. Escreva o método que a implementa.

   (c) Pretende-se agora que o método devolva um TreeSet em que os candidatos estão ordenados primeiro por idade e só depois por nome, se a sua idade for igual. Escreva o comparador necessário para garantir a ordenação pretendida.

Mude para uma nova folha

Parte III - 6 valores

3. Considere que a distribuição do Java que está instalada nas máquinas do laboratório, não possui uma implementação da interface `List`. Para que seja possível fazer a classe `ListaEleitoral` é necessário criar uma classe `LLCandidato`, que permita implementar uma lista ligada de candidatos.

   Desenvolva a classe `LLCandidato` (e eventuais classes auxiliares que sejam necessárias), com os seguintes métodos:

   (a) `public LLCandidato()`

   (b) `public int size()` – Saber o tamanho da lista

   (c) `public void add(Candidato c)`

   (d) `public Candidato get(int i) throws CandidatoException`

   (e) `public boolean equals(Object o)`.

Mude para uma nova folha

Parte IV - 5 valores

4. Considere agora que foi desenvolvida a classe `Parque` que implementa a interface `IParque`:

```
public inteface IParque {
  public void entra(String cartao, String matricula)
                                          throws SemPermissãoException;
  public void sai(String matricula) throws VeículoNãoExisteException;
  public boolean noParque(String matricula);
  public boolean equals();
}
```

   Desenvolva a classe `ParqueComRecusados` que, para além de implementar `IParque`, mantém o registo de todas as tentativas de entrada recusadas, associando a cada matrícula os cartões com que se tentou entrar e permitindo obter esse registo com o método: `Map<String, Set<String>> getRecusas()`.

5. Considere o seguinte método,

```
/**
 * Total de Km percorridos por todos os Motoristas.
 */
 public double qtsKmsTotal() {
     return this.empregados.values()
                           .stream()
                           .filter(e->e instanceof Motorista)
                           .mapToDouble(e->((Motorista)e).getNKms())
                           .sum();
 }
```

   Reescreva-o utilizando iteradores externos.

6. Relembre o que foi dito nas aulas sobre iteradores internos vs. externos. Explique de forma sucinta e concreta qual a vantagem que os interadores internos poderão ter sobre os iteradores externos.