Teste de Programação Orientada aos Objectos

MiEI e LCC
DI/UMinho

23/05/2019
Duração: **2h**

Leia o teste com muita atenção antes de começar
Assuma que gets e sets estão disponíveis, salvo se forem explicitamente solicitados.

Responda a cada Parte em folhas separadas.

Parte I - XX valores

1. Considere que se pretende desenvolver um programa para manipular polinómios (tipo `Poly`). Uma vez que existem várias formas de representar polinómios, todas as possíveis implementações têm de disponibilizar pelo menos os seguintes métodos:

```
public void  addMonomio(int grau , double coef);
public double calcula(double x);
public int grau();
public Poly derivada();
```

Considere que se pretende implementar uma solução em que o polinómio é representado por uma lista de coeficientes e a posição na lista dá o expoente correspondente. Por exemplo, o polinómio $2.2x^3 + 1.3x - 4.7$ é representado pela lista $[-4.7, 1.3, 0, 2.2]$).

Apresente as definições necessárias para ter a classe *PolyAsList* (uma representação de um polinómio baseado numa lista). Apresente também a(s) variável(is) de instância necessárias e os métodos referidos acima.

Parte II - XX valores

2. Relembre o exercício de uma das fichas práticas em que se pedia para criar uma classe para representar grafos dirigidos. Para tal, foi decidido utilizar uma lista de adjacência que associa, a cada vértice (representado como sendo uma *String*), os vértices que podem ser visitados a partir dele. Foi já definida a seguinte estrutura base:

```
import java.util.Set;
import java.util.Map;
import java.util.HashMap;

public class Grafo {
    // variáveis de instância
    private Map<String, Set<String>> adj;
}
```

Complete a classe definindo:

(a) Os construtores `Grafo()` e `Grafo(Map<String, Set<String>> adj)`.

(b) `void addArco(String vOrig, String vDest)`, método que adiciona um arco ao grafo. Note que todos os vértices do grafo devem ter uma entrada na lista de adjacência (que eventualmente poderá ser vazia).

(c) `boolean isSink(String v)`, método que determina se um vértice é um *sink* (não existem arcos a sair dele, ou seja, está-lhe associado um conjunto vazio de vértices na "lista"de adjacência).

(d) `int size()`, método que calcula o tamanho do grafo (o tamanho de um grafo com n vértices e m arcos é $n + m$).

(e) `boolean haCaminho(String vOrig, String vDest)`, método que determina se existe um caminho entre os dois vértices passados como parâmetro. Tenha em consideração que poderão existir ciclos no grafo.

(f) `Set<Map.Entry<String, String>> fanOut (String v)`, método que calcula o conjunto de todos os arcos que saem de um vértice. Neste método os arcos são representados como sendo objectos do tipo `Map.Entry`.

Parte III - XX valores

3. Considere que se criou um programa para registar alugueres de imóveis (à semelhança do que fazem algumas plataformas acessíveis por apps). Como entidades principais o programa considera os tipos `Imovel`, `Cliente` e `Aluguer`.

```
public abstract class Imovel implements Serializable {
  private String codImovel;
  private String morada;
  private String nifProprietario;
  private double area;
  private double precoBase;

  private abstract double precoDia();
  ...
}

public class Apartamento extends Imovel {
  private String andar;
  private double factorQualidade;
  ...
}

public class Moradia extends Imovel {
  private double areaPrivativa;
  private double areaExterior;
  ...
}

public class Bungalow extends Imovel {
  private double factorQualidade;
  private double espessuraParedes;
```

```
  ...
}


public class Cliente implements Serializable {
  private String nome;
  private String codCliente;
  private List<Aluguer> meusAlugueres;
  ...
}

public class Aluguer implements Serializable {
  private String codCliente;
  private String codImovel;
  private LocalDate dataInicio;
  private LocalDate dataFim;
}

 public class POOAirBnB implements Serializable {
   private Map<String, Imovel> imoveis;
   private Map<String, Cliente> clientes;
   ...

 }

}
```

Sabe-se também que o preço por dia de um apartamento é função do factor de qualidade em relação ao preço base, de uma moradia é um rácio de 30% sobre a área privativa e 70% sobre a área exterior e que o preço por dia de um bungalow é calculado com base em iguais parcelas de factor de qualidade geral e da espessura das paredes.

Codifique os seguintes métodos:

(a) `public double precoDia()`, nas classes em que tal seja necessário.

(b) `public void insereImovel(Imovel i) extends ImovelJaExistente`, da classe `POOAirBnB`. Codifique a classe de excepção.

(c) `public double valorTotalAluguerCliente(String codCliente)`, que determina o valor total dos alugueres pagos pelo cliente indicado no parâmetro. Considere que o cliente pode não existir.

(d) `public String clienteMaisGastador()`, que determina o cliente que mais gastou em alugueres de imóveis.

(e) `public Map<String, Set<String>> clientesPorImovel()`, que devolve um map onde se associa a cada imóvel o conjunto dos clientes que alguma vez o alugaram.

Parte IV - XX valores

4.