Teste de Programação Orientada aos Objectos

MiEI e LCC
DI/UMinho

04/06/2018
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 - 6 VALORES

1. Considere que um dado grupo apresentou a solução arquitectural (não necessariamente ideal) esquematizada abaixo para o trabalho prático de POO:

```
public interface Bonificado {
    public double reducaoImposto();
}

public class Actividade { ... }

public class Fatura {
    private String numero;
    private LocalDate data;
    private double valor;
    private Actividade actividade;  // null caso não esteja definida
    public double getValor() { ... }
    ...
}

public abstract class Entidade {
    private String nif; private String name;
    private Map<LocalDate, Fatura> faturas;
    ...
    public String getNIF() { ... }
    ...
}
public class Contribuinte extends Entidade {
    private List<Contribuinte> dependentes;
    ...
    public List<Contribuinte> getDependentes() { ... }
        ...
}
public class FamiliaNumerosa extends Contribuinte implements Bonificado { ...
```

```
public class Negocio {
    private Map<String, Contribuinte> contribuintes;
    ...
}
```

Codifique os seguintes métodos:

(a) o método `public printFamiliasNumerosas(String nomeFich)` da classe `Negocio`, que deverá escrever, no ficheiro com nome `nomeFich`, linhas de texto com o Nome, NIF e valor da redução de imposto de todos os contribuintes que sejam bonificados. Sabendo que no futuro poderão ser adicionados mais sub-tipos de `Contribuinte`, desenvolva uma solução genérica. Não se esqueça de considerar as excepções relevantes.

(b) o método que implemente a ordem natural de `Fatura`, em que se ordenam os elementos por ordem crescente da data.

(c) o método `public Map<Actividade, List<Fatura>> porActividade()` da classe `Entidade`, que deverá calcular um `Map` de actividade para lista de facturas dessa actividade.

Parte II - 6 valores

2. Considere a definição de `Lampada`, cf solicitada numa ficha prática. Uma **Lâmpada** pode estar ligada ou desligada ou em modo *eco*. Em qualquer um desses estados tem um consumo por milissegundo que é característica da lâmpada (quando desligada o consumo é zero!).

```
public class Lampada {
  private String ident;
  private LocalDateTime inicio;
  private LocalDateTime parcial;
  private double consumoLigada;
  private double consumoEco;
  private double consumoTotal;
  private double consumoParcial;
  private int estado; // 0 - desligada, 1 - ligada, 2 - eco

  public Lampada(String ident, double consumoLigada, double consumoEco) {...}
  // liga a lâmpada em modo consumo máximo
  public void lampON() {...}
   // liga a lâmpada em modo eco
  public void lampECO() {...}
  // desliga a lâmpada
  public void lampOFF() {...}
  // devolve o consumo desde o início
  public double totalConsumo() {...}
  //devolve o consumo desde o último reset
  public double periodoConsumo() {...}
  public void efectuarResetConsumo() {...}
}
```

Crie agora a classe `CasaInteligente` que internamente associa a cada identificador de lâmpada o objecto respectivo, e faz a gestão das operações que a casa disponibiliza. Considere que todos os métodos `equals`, `clone`, `toString` estão definidos.

Resolva os seguintes exercícios:

(a) declaração de variáveis de instância e o construtor que aceita uma colecção de lâmpadas a adicionar à casa, `public CasaInteligente(Collection<Lampada> novasLampadas)`

(b) `public int qtEmEco()`, que determina quantas lâmpadas é que estão ligadas em modo económico.

(c) `public void removeLampada(String id) throws ...`, que remove a lâmpada identificada (apresente todas as definições necessárias).

(d) `public double consumoTotal()`, que determina o consumo total da casa.

Parte III - 5 valores

3. Ainda em relação ao exercício anterior, codifique os seguintes métodos:

   (a) `public Set<String> topConsumo(int x)`, que devolve o conjunto das `x` lâmpadas que mais gastam.

   (b) Considere agora que se criou um novo tipo de lâmpada, a `LampadaLed` Uma lâmpada deste tipo tem um consumo que é menor em relação às lâmpadas normais em função de um parâmetro (uma percentagem) que é fornecido aquando da criação destas. Crie a classe `LampadaLed`, com as variáveis de instância, um construtor parametrizado e os métodos que tem de ser reescritos (relativamente aos que foram escritos acima).

   (c) Codifique na classe `CasaInteligente` o método que grava em formato binário, num ficheiro de nome a ser fornecido, as lâmpadas que são `LampadaLed`. Este método deverá apenas lançar a excepção `LampLedWriteFail`.

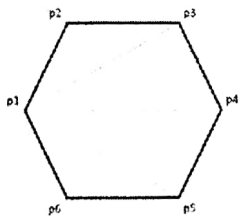
Parte IV- 3 valores

4. Considere que a conhecida classe `Ponto` está já definida tal como feito na aulas. Entre outros disponibiliza o construtor `public Ponto(double x, double y)`

   e o método `public double distancia(Ponto p)`

   (a) Defina a classe `Poligono` que utiliza uma lista para representar a lista de pontos de um polígono. Codifique o construtor parametrizado, e os métodos `public void addPonto(Ponto p)` que adiciona um ponto ao polígono, `public boolean eFechada()` que indica se o poligono esta (correctamente) definido por uma linha poligonal fechada (último ponto é igual ao primeiro), `public double perimetro()` e delega nas subclasses o método que calcula a área.

   (b)



   Considere que está definida a classe `PoligonoConvexo` (um polígono onde nenhum segmento de recta que liga dois pontos de seu perímetro passa por fora do polígono. Ver figura anexa.). Considere que existe a classe `Triangulo` que disponibiliza um construtor

   `public Triangulo (Ponto x, Ponto y, Ponto z)` e um método `public double area()`. Defina o método `public double area()` da classe `PoligonoConvexo`.