Teste de Programação Orientada aos Objectos

MiEI e LCC
DI/UMinho

30/05/2017
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 - 7 VALORES

1. Considere que estamos a desenvolver um sistema que guarda faixas de música numa playlist (cf. exercício da Ficha 5). A classe `Faixa` foi definida como tendo a seguinte estrutura:

```
public class Faixa {
  private String nome;
  private String autor;
  private double duracao;
  private int classificacao;
  private ArrayList<String> letra;  //letra da música
  private int numeroVezesTocada; //número de vezes que foi tocada
  private LocalDateTime ultimaVez; //regista quando foi tocada pela últi

  ...
}
```

   Para a classe Faixa codifique os seguintes métodos:

   (a) `public Faixa(Faixa f)` – construtor de cópia de faixa.

   (b) `public boolean equals(Object o)` – método de igualdade standard.

   (c) método que implemente a ordem natural de `Faixa`, em que se ordena os elementos por ordem crescente de número de vezes que a faixa foi tocada.

   (d) indique, e codifique, como fazer para criar um mecanismo que ordene as faixas por ordem cronológica de última vez que foram tocadas. O método `isBefore`, da classe `LocalDateTime`, aceita um `LocalDateTime` como parâmetro e devolve um boolean como resultado.

2. Considere agora que criamos a classe `PlayList` que tem a seguinte estrutura:

```
public class PlayList {
    private String nome;
    private Map<String, List<Faixa>> musicas;
```

em que a cada nome de autor corresponde a lista das faixas por ele interpretadas.

Para a classe `PlayList` codifique os seguinte métodos:

(a) `public List<Faixa> getFaixas(String autor) throws AutorInexistenteException`, que devolve a lista de músicas do autor passado como parâmetro.

(b) `public double tempoTotal(String autor) throws AutorInexistenteException`, que devolve o tempo total das faixas de música do autor indicado.

Mude para uma nova folha

Parte II - 8 valores

3. Continue a considerar a classe `PlayList` apresentada na pergunta anterior. Codifique agora os métodos:

   (a) `public List<Faixa> todasAsFaixas()`, que devolve numa lista todas as faixas existentes na playlist.

   (b) `public Map<Integer,List<Faixa>> faixasPorClass()`, que devolve um Map que associa a cada valor de classificação existente a lista das faixas com essa classificação. Este método deve ser feito preferencialmente recorrendo a iteradores internos (caso não o consiga apresente uma outra solução).

   (c) considere agora que a classe `Faixa` deverá implementar a interface `Playable`, definida como:

   ```
   public interface Playable {
      public void play();
   }
   ```

   Assumindo que existe um objecto chamado `System.audio`, que apresenta um comportamento similar ao `System.out`, e que transforma em som a letra da faixa, altere a classe `Faixa` de modo a que implemente a interface `Playable`.

4. Considere que se criaram novos tipos de conteúdo e passaram a fornecer-se além da música os vídeoclips de suporte. Pretende-se criar um novo tipo de conceito `MusicaComVideo`, que acrescenta à informação existente o conteúdo do filme. Por conveniência assuma que o conteúdo de vídeo é uma sequência de caracteres que devem ser enviados para o écran.

   Codifique a classe `MusicaComVideo`, apresentando:

   (a) as suas variáveis de instância;

   (b) o construtor parametrizado;

   (c) o método `play`, sabendo que num conteúdo destes deve ser reproduzido o som e a imagem (assuma, por simplificação, que podem ser feitos por esta ordem).

Mude para uma nova folha

Parte III - 5 valores

5. Relembre a classe `Hotel` fornecida nas aulas práticas

```
public abstract class Hotel implements Comparable<Hotel> {
    /** O código do hotel */
    private String codigo;
    /** O nome do hotel */
    private String nome;
    /** Localidade do hotel */
    private String localidade;
    /** Preço base por quarto */
    private double precoBaseQuarto;
    /** Numero de quartos */
    private int numeroQuartos;
    /** Estrelas **/
    private int estrelas;
    ...
}
```

Considere também que existem as subclasses concretas `HotelStandard`, `HotelPremium` e `HotelDiscount`.

Considere agora que se pretende desenvolver uma classe `AgenciaViagens`. Esta classe deverá permitir manter um registo dos Hotéis com que a agência trabalha. Associada a cada código de hotel, deverá ainda manter uma lista dos NIF dos clientes que ficaram no hotel.

Responda às seguintes questões:

(a) Escreva a declaração da classe `AgenciaViagens`, considerando apenas as variáveis necessárias (não necessita indicar métodos nem construtores).

(b) Escreva o método de `AgenciaViagens` que permite gravar em ficheiro, em modo texto, os hotéis de um dado tipo passado como parâmetro, desde que estes tenham clientes registados.

(c) Escreva o método de `AgenciaViagens` que permite criar uma instância da classe a partir de um ficheiro de objectos anteriormente gravado. Considere que, caso não seja possível ler a instância do ficheiro, deverá ser devolvida uma nova instância vazia.

(d) Que alterações teria que fazer nas classes `AgenciaViagens` e `Hotel` para o método anterior funcionar?