Programação Orientada pelos Objectos

LEI – Exame de Época Normal

2012/06/30 Duração: 2 h 30 m

Instruções importantes:

- **Responda a cada grupo em folhas separadas.**
- Verifique que todas as folhas estão identificadas com o **número de caderno.**
- Antes de começar a resolver, leia o enunciado do princípio até ao fim.
- **Pode** usar caneta ou **lápis.**
- Não é permitido consultar quaisquer elementos para além deste enunciado.
- As regras do **código de ética** do departamento de informática serão integralmente aplicadas na classificação deste exame.

Introdução aos problemas para os grupos I, II, III e IV:

Os grupos I-IV deste enunciado estão relacionados por um tema comum mas foram concebidos de modo a que a resolução de cada grupo possa ser feita independentemente.

Na Figura 1 mostra-se as interfaces necessárias para gerir a informação de albuns de música. No exame vão definir algumas classes que implementam algumas destas interfaces. Se não for pedida a classe para implementar determinada interface, assuma que já existe.
Há vários tipos de album: SimpleAlbum (uma música de um único artista) e CompilationAlbum (uma colectânea de músicas de diferentes artistas).

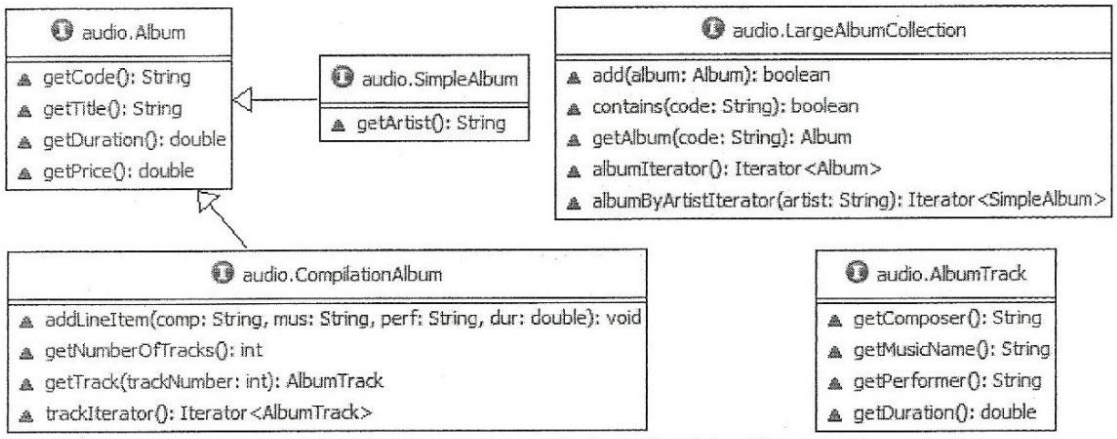

Figura 1– Entidades para a gestão de albuns de música (Grupos I-IV)

Grupo I – AbstractAlbumClass e SimpleAlbumClass

a) Implemente a classe abstracta AbstractAlbumClass que implementa a interface Album (Figura 1) e contém os seguintes itens de informação, comum a todos os tipos de album:
   - Código (ou número de série) do album – tipo String.
   - Preço – tipo double.
   - Duração total do album – tipo double (não é realista; é para simplificar).
   - Título do album – tipo String.

Não deve ser possível instanciar esta classe directamente em expressões usando **new**. A classe inclui um único construtor que recebe o código, o título e o preço, por essa ordem. Esses dados

poderam ser consultados através das operações getCode(), getTitle() e getPrice() respectivamente. Note que o construtor não recebe a duração do album.

b) Um dos tipos de album efectivamente usado é SimpleAlbum, que representa um album com um único artista (grupo ou "artista" que é simultaneamente autor das músicas (compositor) e seu intérprete). A operação getArtist() devolve o nome desse artista.
A interface SimpleAlbum estende a interface Comparable<SimpleAlbum> (não mostrada na Figura 1) e consequentemente herda o método compareTo que tem a seguinte assinatura:

```
int compareTo(SimpleAlbum other);
```

O critério de comparação entre albuns de um único artista (SimpleAlbum) é baseado na ordem resultante da comparação das *strings* retornadas por getArtist(), ou seja o nome do artista.

Implemente a classe concreta SimpleAlbumClass que implementa a interface SimpleAlbum (Figura 1). A classe inclui um único construtor que recebe o código, o título, o preço, o artista e a duração total do album, por essa ordem.

Grupo II –CompilationAlbumClass

Implemente a classe concreta CompilationAlbumClass que implementa a interface CompilationAlbum (Figura 1), que representa um album-colectânea com músicas de vários artistas. Para simplificar o problema esta colectânea pode ter músicas repetidas. O constructor desta classe deve ter três argumentos: código, titulo e preço, por esta ordem.

Uma instância desta classe contém também os dados de cada uma das faixas (*tracks*), que correspondem à entidade AlbumTrack (Figura 1). Note que a colecção de faixas deve ser mantida, pelo que utilize alguma das interfaces e classes da API do Java (ver anexo) que conheça (de preferência, aquelas que melhor se adequem ao problema).

Na implementação de CompilationAlbumClass parta do princípio que já existe uma classe AlbumTrackClass que implementa a interface AlbumTrack e fornece um construtor com a assinatura mostrada abaixo. Note que não tem de implementar AlbumTrackClass.

```
public AlbumTrackClass(String composer, String music, String performer, double duration)
```

Assim, cada objecto da classe AlbumTrackClass inclui os seguintes itens de informação:

- Nome do compositor (ou autor) da música da faixa – tipo String.
- Nome da música – tipo String.
- Intérprete (em Inglês: *performer*: aplica-se também a grupos) – tipo String.
- Duração da faixa – tipo double.

As operações de CompilationAlbumClass (ver CompilationAlbum na Figura 1) são as seguintes:

- void addLineItem(String composer, String music, String performer, double duration) cria uma nova faixa no album. As faixas são guardadas na ordem pela qual são acrescentadas.
- int getNumberOfTracks() devolve o número de faixas do album.
- double getDuration() devolve a duração total do album, que corresponde ao somatório da duração de todas as suas faixas.

- AlbumTrack getTrack(int trackNumber) devolve a faixa a que corresponde o número passado como argumento. Para facilitar, números válidos de faixas começam com **zero** inclusive.
- Iterator<AlbumTrack> trackIterator() devolve um iterador (interface da API do Java) que permite percorrer as diversas faixas do album, na ordem pela qual foram inseridas.

Implemente a classe CompilationAlbumClass, que implementa a interface CompilationAlbum da Figura 1.

Grupo III – LargeAlbumCollectionClass

Pretende-se implementar uma classe LargeAlbumCollectionClass que implementa a interface LargeAlbumCollection (Figura 1). A classe destina-se a cenários em que um objecto LargeAlbumCollection pode conter/manter vários milhares de albuns. Na coleção não existem albuns repetidos, ou seja, com o mesmo código (identificador de album). As operações são as seguintes:

- boolean add(Album album) insere o album dado como parâmetro na colecção de albuns, caso não exista já um album com o mesmo código. Retorna true no caso de sucesso na inserção. Caso exista um album com o mesmo código, não insere e retorna false.
- boolean contains(String code) recebe o código de um album e diz se existe (devolve true) ou não (devolve false).
- getAlbum(String code) recebe o código de um album e devolve o objecto Album que representa esse album. Pressupõe a pré-condição de que existe sempre um e um só album com o código fornecido como argumento.
- Iterator<Album> albumIterator() devolve um iterador (interface da API do Java) para percorrer os albuns, ordenadas por código (ordenação usada por omissão para as *strings*,i.e., ordem natural).
- Iterator<SimpleAlbum> byArtistIterator(String artist) devolve um iterador (interface da API do Java) para percorrer os albuns que são instâncias de SimpleAlbum e cujo artista seja o recebido como argumento. A ordem de travessia dos albuns é a determinada pela operação compareTo declarada em SimpleAlbum.

Grupo IV – Excepções e testes unitários

a) Na implementação da operação de getAlbum(String) pretende-se substituir a abordagem baseada numa pré-condição por uma baseada numa excepção verificada. Esboce uma implementação de getAlbum(String) baseada numa excepção verificada AlbumNotFoundException. Para esse efeito, apresente a implementação completa de AlbumNotFoundException e um esboço da nova implementação de getAlbum(String), incluindo o novo cabeçalho do método.
   Note que não tem de mostrar as partes respeitantes à funcionalidade de getAlbum(String) ou de repetir alguma implementação que já tenha criado para outro grupo deste enunciado. Basta representar essas partes com um simples comentário de linha.

b) Implemente um método de teste à operação getDuration() da entidade CompilationAlbum da Figura 1, usando o JUnit. Note que para a resolução desta questão não é forçoso que tenha resolvido os 2 primeiros grupos.
   Para ser considerado completo, o teste deve montar um cenário em que é criado um objecto CompilationAlbum e vários objectos AlbumTrack. O teste verifica que a duração de antes de ser inserida a primeira faixa é zero e que após sucessivas inserções de faixas a duração

corresponde sempre ao somatório das faixas inseridas – e apenas dessas.
Para simplificar, considere sempre uma escala decimal para os valores de teste (e.g., a soma do tempo 3.45 e do tempo 5.20 dá 8.65).

Grupo V – Modelação

Pretende-se construir uma aplicação que mantenha informações de conectividade sobre dispositivos móveis. A aplicação deve suportar diferentes tipos de dispositivos móveis, nomeadamente *telemóveis*, *portáteis* e *tablets*. Todos os tipos de dispositivos móveis estão equipados com antena *bluetooth*. Vamos considerar que os telemóveis estão equipados apenas com *bluetooth* e que os tablets, para além de *bluetooth*, estão também equipados com antena *wi-fi* e que os portáteis, para além de *bluetooth*, estão equipados com antena *wi-fi* e infra-vermelhos. O comportamento associado às funcionalidades dos vários aparelhos para o mesmo tipo de ligação é igual.
Seguidamente apresentam-se as operações suportadas por cada tipo de aparelho. Para simplificar pode assumir que nenhuma das operações falha.

Telemóvel:

- Alterar o estado da ligação *bluetooth* (não recebe nada, não retorna nada)
- Consultar o estado da ligação *bluetooth* (não recebe nada, retorna um boolean)
- Obter o identificador *bluetooth* (não recebe nada, retorna uma String)
- Alterar o identificador *bluetooth* (recebe uma String, não retorna nada)
- Obter o próprio número de telefone (não recebe nada, retorna uma String)

Tablet:

- Alterar o estado da ligação *bluetooth* (não recebe nada, não retorna nada)
- Consultar o estado da ligação *bluetooth* (não recebe nada, retorna um boolean)
- Obter o identificador *bluetooth* (não recebe nada, retorna uma String)
- Alterar o identificador *bluetooth* (recebe uma String, não retorna nada)
- Ligar a rede *wi-fi* a uma determinada rede (recebe uma String, não retorna nada)
- Obter o nome da rede *wi-fi* (não recebe nada, retorna uma String)

Laptop:

- Alterar o estado da ligação *bluetooth* (não recebe nada, não retorna nada)
- Consultar o estado da ligação *bluetooth* (não recebe nada, retorna um boolean)
- Obter o identificador *bluetooth* (não recebe nada, retorna uma String)
- Alterar o identificador *bluetooth* (recebe uma String, não retorna nada)
- Ligar a rede *wi-fi* a uma determinada rede (recebe uma String, não retorna nada)
- Obter o nome da rede *wi-fi* (não recebe nada, retorna uma String)
- Alterar o estado da ligação infra-vermelhos (não recebe nada, não retorna nada)
- Consultar o estado da ligação infra-vermelhos (não recebe nada, retorna boolean)

Apresente um diagrama de classes e interfaces para modelar todos os dispositivos móveis descritos em cima, onde devem constar os nomes das classes/interfaces, as relações entre as mesmas, as variáveis de instância e as assinaturas dos métodos (mas não é necessário especificar os construtores). O desenho deve favorecer implementações que **minimizam** o número de linhas de código necessárias, seguindo a metodologia dada nas aulas.
Nota: Não é necessário implementar nenhuma das operações.
Nota: Para efeitos de clareza da apresentação represente as relações com uma seta etiquetada, por exemplo A →extends B significa A extends B (analogamente para implements, eventualmente com linha tracejada mas sempre etiquetada com implements).

Algumas das interfaces abaixo reproduzidas poderão ser úteis na resolução deste exame: