Ciências ULisboa | Informática

Desenvolvimento Centrado em Objetos

Trabalho Prático 1

2022/2023

1 Introdução

O objetivo deste trabalho é que os alunos exercitem a utilização das técnicas e dos princípios de programação OO lecionados na disciplina.

O trabalho centra-se numa aplicação que é um gestor e reprodutor de músicas em suporte digital. A aplicação vai ser desenvolvida incrementalmente e iterativamente. O primeiro incremento foca-se em funcionalidades básicas e músicas no formato *mp3* e a primeira iteração considera um conjunto básico de casos de uso em torno de bibliotecas de músicas e de *playlists*.

Muito resumidamente, a aplicação deve permitir carregar músicas para uma biblioteca e a partir da biblioteca, selecionar e pôr a tocar músicas individuais e fazer pesquisas. Existem pré-definidas várias *smart playlists* (listas de músicas geridas de forma automática) mas é também possível criar *playlists* manualmente. É possível gerir as músicas contidas numa *playlist* manual e pôr a tocar as músicas de uma qualquer *playlist*.

De seguida descrevem-se os principais elementos da solução de desenho concebida para o problema. A solução é incompleta em alguns aspetos; existem decisões de desenho que foram deixadas propositadamente em aberto.

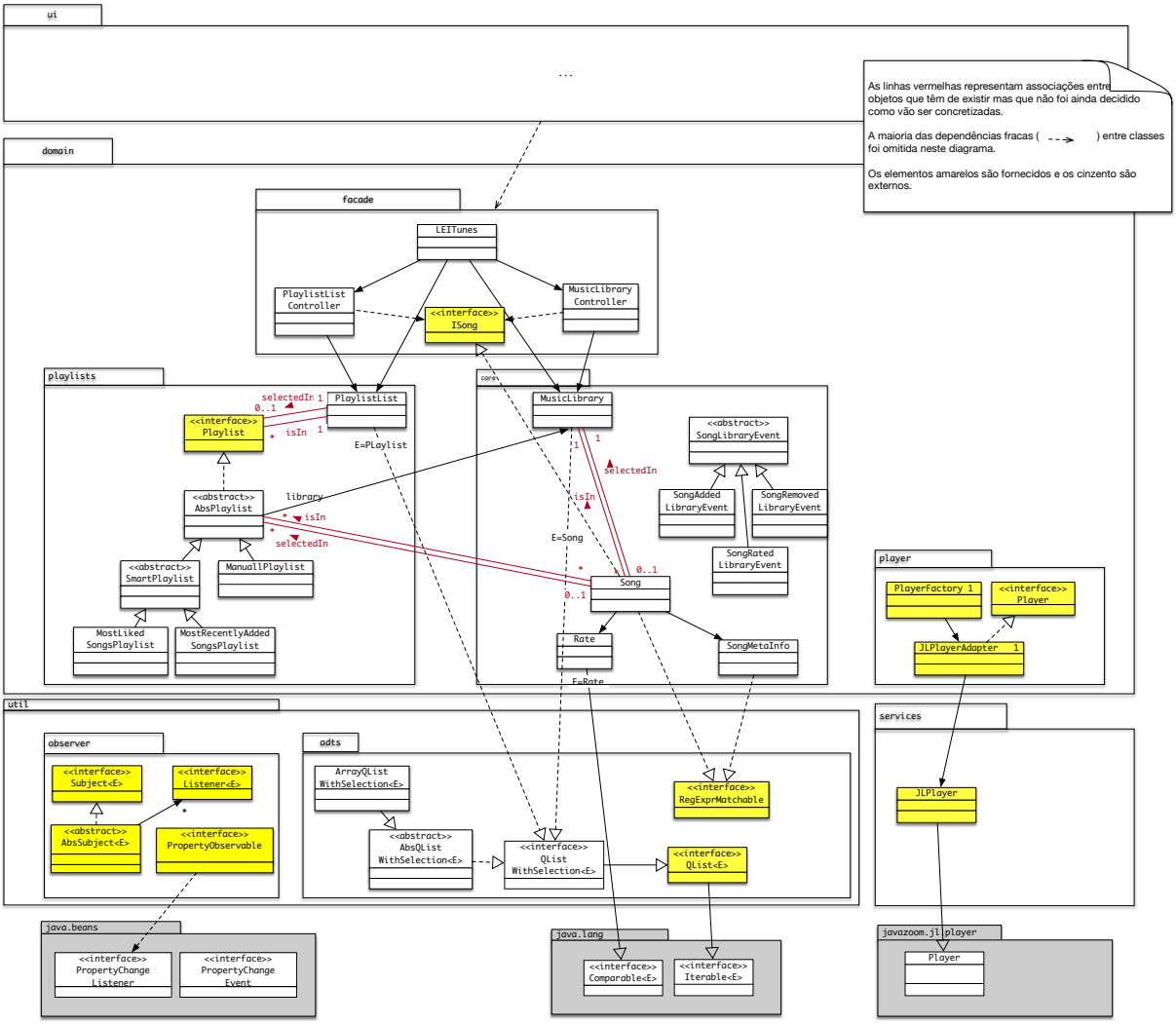
2 Solução de Desenho

As classes da solução foram organizadas em quatro pacotes:

- **ui**: com classes responsáveis exclusivamente pela interface da aplicação
- **domain**: com classes cuja responsabilidade se prende apenas com o domínio da aplicação
- **services**: com classes cuja responsabilidade se prende com a utilização de serviços externos à aplicação, como por exemplo os serviços que permitem pôr uma música a tocar e obter os seus meta-dados
- **util**: com classes que foram concebidas especificamente para a aplicação, mas potencialmente reutilizáveis noutros contextos

Porque o desenho da interface da aplicação não é o foco de DCO, é fornecida uma implementação rudimentar de um interface gráfico (implementado com o *swt*[1]) que suporta algumas das funcionalidades cobertas nesta iteração. Não é necessário alterarem esse código. O diagrama seguinte dá uma visão geral da estrutura de classes do resto da solução. Alguns elementos não fundamentais foram omitidos, nomeadamente para manter a legibilidade do diagrama.

[1] https://www.eclipse.org/swt/



Pacote `util`

Este pacote subdivide-se em dois:

- **`util.adts`** — tem os elementos relacionados com tipos de dados abstratos úteis para a implementação das bibliotecas de música e *playlists*
- **`util.observer`** — tem elementos de suporte a duas formas diferentes de implementação do padrão observador (uma básica, e pouco flexível, e outra mais poderosa baseada nas propriedades do *java.beans*)

`util.adts`

- Interface **`RegExpMatchable`** (fornecida) representando objetos que suportam o *match* com uma expressão regular
- Interface **`QList<E>`** (fornecida) representando listas que suportam apenas a inserção no fim e são iteráveis
- Interface **`QListWithSelection<E>`** representando listas **`QList<E>`** que podem ter zero ou um elemento selecionado e permitem a remoção deste elemento. Deve estender **`QList<E>`** com:
  - **`void select(int i)`** que, assumindo que **`0≤i<size()`**, seleciona o elemento na posição i da lista
  - **`void add(E e)`** que adiciona o elemento que passa a ser o elemento selecionado
  - **`boolean someSelected()`** que indica se há um elemento selecionado

  - `int getIndexSelected()` que, assumindo que `someSelected()`, devolve a sua posição na lista
  - `void next()` que seleciona o elemento na posição seguinte à selecionada, se `getIndexSelected()<size()-1;` senão, deixa de haver um elemento selecionado
  - `void previous()` que seleciona o elemento na posição anterior à selecionada, se `getIndexSelected()>0;` senão, deixa de haver um elemento selecionado
  - `void remove()` que apaga o elemento selecionado, se `someSelected`() (caso em que não fica nenhum elemento selecionado) e não faz nada no caso contrário
  - `E getSelected()` que, assumindo que há um elemento selecionado, o devolve
- Classe abstrata `AbsQListWithSelection<E>` que implementa `QListWithSelection<E>` e fornece uma implementação esqueleto com um construtor que desencadeia a criação da lista sem, no entanto, se comprometer com nenhuma implementação específica para estas listas; oferece implementações por omissão para todos os métodos.
- Classe `ArrayQListWithSelection<E>` que estende `QListWithSelection<E>` e fornece uma implementação de `QListWithSelection<E>` baseada em *arrays*. Esta classe não admite ser estendida, mas pode ser usada de outras formas.

`util.observer`

- Interface `PropertyObservable` (fornecida) representando objetos que têm propriedades que podem ser observadas por objetos do tipo `java.beans.PropertyChangeListener`
- Interface `Event` (fornecida) representando objetos que são eventos
- Interface `Listener<E>` (fornecida) representando objetos que se registam para receber certos eventos de objetos `Subject`
- Interface `Subject<E>` (fornecida) representando objetos que emitem eventos para os seus *listeners*
- Classe abstrata `AbsSubject<E>` (fornecida) com uma implementação esqueleto de `Subject<E>`

Pacote `domain`

Este pacote subdivide-se em quatro:

- `domain.core` –– tem os elementos fundamentais do domínio
- `domain.playlists` –– tem os elementos do domínio específicos das *playlists*
- `domain.player` –– tem os elementos do domínio relacionados com o suporte à reprodução das músicas
- `domain.facade` –– tem os elementos que constituem a interface que o `domain` expõe para o seu exterior

`domain.core`

- *Record* `SongMetaInfo` cujos objetos representam alguma informação meta das músicas como o título da música, o género, uma lista com o nome de um ou mais artistas e o nome do álbum. Implementa `RegExpMatchable` considerando que alguma informação disponível (o nome, algum dos nomes dos artistas, o género ou o nome do álbum) sobre o objeto adere à *regexp* dada.
- Classe `Rate` cujos objetos, imutáveis, representam os possíveis valores das classificações das músicas.
- Classe `Song` cujos objetos representam músicas. Cada música tem o nome de um ficheiro onde a música se encontra (com o formato mp3), meta-informação, quantas vezes já foi tocada e uma classificação. A classe deve implementar `ISong` e `RegExpMatchable` e oferecer o seguinte construtor:
  - `Song(SongMetaInfo info, String fileName)`
- Classe `MusicLibrary` cujos objetos representam bibliotecas de música, sendo que em cada momento pode estar selecionada no máximo uma destas músicas. A classe deve implementar `QListWithSelection<Song>`, `Subject<LibraryEvent>` e `PropertyChangeListener` e pode ou não estender uma das duas classes abstratas que implementam os dois primeiros interfaces; esta é uma decisão de desenho que ficou em aberto e sobre a qual terá de decidir o que é mais apropriado.

Uma biblioteca deve notificar os seus observadores sempre que (1) é removida uma música, (2) sempre que é adicionada uma música e (3) sempre que uma música é *rated*, com eventos apropriados para o efeito. Por outro lado, uma biblioteca regista-se para ser notificada de mudanças do estado do objeto responsável pela reprodução das músicas, nomeadamente para contabilizar o número de vezes que cada música é tocada até ao fim. A classe deve oferecer ainda os seguintes métodos de instância:

  - `void play()` que, assumindo que `someSelected()`, determina a interrupção da música que estiver a tocar (se for esse o caso) e começa a tocar a música selecionada
  - `boolean isPlaying()` que indica se está a tocar uma música que foi posta a tocar através da biblioteca
  - `void stop()` que, assumindo que `isPlaying()`, para a música que estava a ser tocada.
  - `void incRateSelected()` que passa a classificação da música selecionada para o valor imediatamente acima ao que possui correntemente (ou fica na mesma se esse valor já for o máximo)
  - `void decRateSelected()` que passa a classificação da música selecionada para o valor imediatamente abaixo ao que possui correntemente (ou fica na mesma se esse valor já for o mínimo)
  - `Iterable<ISong> getMatches(String reexp)` que devolve uma estrutura iterável com as músicas da biblioteca que emparelham com a expressão regular dada.
  - `Iterable<ISong> getSongs()` que devolve uma estrutura iterável com as músicas da biblioteca na ordem própria.

- Classe abstrata `SongLibraryEvent` que estende `Event` e representa um evento genérico sobre uma música de uma biblioteca; carrega informação sobre a música e a *library*. Os eventos concretos são `SongAddedLibraryEvent`, `SongRemovedLibraryEvent` e `SongRatedLibraryEvent`.

`domain.playlists`

- Interface `Playlist` (fornecido) que representa *playlists* com músicas de uma dada `MusicLibrary`; uma *playlist* tem um nome e, em cada momento, uma das suas músicas pode estar selecionada e uma das suas músicas pode estar a tocar. A interface estende `Listener<SongLibraryEvent>`, `PropertyChangeListener` e `Iterable<ISong>`
- Classe abstrata `AbsPlaylist` com uma implementação esqueleto de *Playlist* fornecendo uma implementação por omissão de todos os métodos de `Playlist`.
- Classe abstrata `SmartPlaylist` com uma implementação esqueleto de *playlists* que não permitem a adição e a remoção manual de músicas nem a mudança de posição. A classe deve estender `AbsPlaylist` e oferecer:
  - `SmartPlaylist(String name, MusicLibrary library)`
  - `protected void addAutomatic(ISong song)` que serve para as inserções automáticas que são feitas numa *smartplaylist*
  - `protected void removeAutomatic(int index)` que serve para as remoções automáticas que são feitas numa *smartplaylist*
- classe `MostLikedSongsPlaylist` cujos objetos representam uma *smartplaylist* onde estão as *N* músicas com *rating* mais alto de uma determinada biblioteca (onde *N* denota uma constante). A classe deve estender `SmartPlaylist` e oferecer um construtor com a assinatura
  - `MostPlayedSongsPlaylist(MusicLibrary library)`
- classe `MostRecentlyAddedSongsPlaylist` cujos objetos representam uma *playlist* onde estão as *N* músicas mais recentemente adicionadas a uma determinada biblioteca. A classe deve estender `SmartPlaylist` e oferecer um construtor com a assinatura
  - `MostRecentlyAddedSongsPlaylist(MusicLibrary library)`
- classe `ManualPlaylist` cujos objetos representam uma *playlist* onde estão músicas que foram manualmente lá colocadas; estas *playlists* permitem a remoção e mudança de posição das músicas. A classe deve estender `Playlist` e oferecer um construtor com a assinatura
  - `ManualPlaylist(String name, MusicLibrary library)`
- classe `PlaylistList` cujos objetos representam uma lista de *playlists* de uma `MusicLibrary`. Em cada momento pode estar selecionada no máximo uma destas *playlists*. A classe deve implementar

`QListWithSelection<Playlist>` e pode ou não estender `AbsQListWithSelection<Playlist>`. Deve adicionalmente oferecer:

- `PlaylistList(MusicLibrary library)` que coloca uma instância de cada um dos tipos de *smartplaylist* na lista
- `void play()` que, assumindo que `someSelected()`, se existir alguma música selecionada na *playlist* selecionada, determina a interrupção da música que estiver a tocar (se for esse o caso) e começa a tocar, por ordem, todas as músicas da lista a partir da música selecionada (ou até ser interrompida de alguma forma).
- `boolean isPlaying()` que indica se está a ser tocada alguma música que foi posta a tocar nalguma das suas *playlists*
- `void stop()` que, assumindo que `isPlaying()`, pára a música que estava a ser tocada.

`domain.player`

- Interface `Player` (fornecido) cujos objetos sabem tocar músicas. É um objeto deste tipo que é usado sempre que seja preciso mandar carregar/tocar/parar uma música (obtido através da `PlayerFactory`)
- Classe `PlayerFactory` (fornecida) que é um *singleton* cuja instância tem a responsabilidade de criar o `Player` que vai ser usado e o fornece a quem precisar através de um método
  - `Player getPlayer()`
- Classe `JLPlayerAdapter` (fornecida) que é um *singleton* e que implementa `Player` para leituras de ficheiros em formato *mp3* adaptando esta interface à oferecida pela classe `services.JLPlayer`, baseada em `javazoom.jl.player.Player`

`domain.facade`

- Interface `ISong` (fornecido) cujos objetos representam as músicas
- Classe `MusicLibraryController` cujos objetos representam um controlador das interações com um objeto do tipo `MusicLibrary`. A classe deve oferecer os seguintes construtores e métodos de instância:
  - `MusicLibraryController(MusicLibrary library)`
  - `int numberOfSongs()` que dá o número de músicas na biblioteca controlada
  - `addSong(String filename)` que, assumindo que `filename` é um ficheiro mp3, adiciona a música na biblioteca controlada obtendo do ficheiro a meta-informação sobre a mesma (se existir); a decisão de como deve ser concretizada a leitura da meta-informação ficou em aberto e cabe-vos a vós tomar
  - `void selectSong(int i)` que se `0≤i<numberOfSongs()`, despacha o pedido de seleção para a biblioteca controlada; caso contrário não faz nada
  - `Optional<ISong> getSelectedSong()`, que devolve a música selecionada na biblioteca (se existir)
  - `void removeSelectedSong()`, que apaga a música selecionada na biblioteca (se esta existir)
  - `void play()` que, se estiver alguma música selecionada, determina a interrupção da música que estiver a tocar (se for esse o caso) e começa a tocar a música correntemente selecionada na biblioteca controlada; o contador de uma música é incrementado sempre que a música é tocada até ao fim; se nenhuma música está selecionada não faz nada
  - `void stop()` que para a música que estava a ser tocada, se essa música foi posta a tocar através da biblioteca; senão, não faz nada
  - `void incRateSelected()` que passa a classificação da música selecionada na biblioteca controlada (se esta existir) para o valor imediatamente acima ao que possui correntemente (ou fica na mesma se esse valor já for o máximo)
  - `void decRateSelected()` que passa a classificação da música selecionada na biblioteca controlada (se esta existir) para o valor imediatamente abaixo ao que possui correntemente (ou fica na mesma se esse valor já for o mínimo)
  - `Iterable<ISong> getMatches(String reexp)` que devolve uma estrutura iterável com as músicas da biblioteca controlada que emparelham com a expressão regular dada
  - `Iterable<ISong> getSongs()` que devolve uma estrutura iterável com as músicas da biblioteca

- classe `PlaylistListController` cujos objetos representam um controlador de uma `PlaylistList` e da interação do exterior com esta. A classe deve oferecer os seguintes construtores e métodos de instância:
  - `PlaylistListController(PlaylisList playlists, MusicLibrary library)`
  - `void createPlaylist(String name)` que insere uma nova *playlist* manual com o nome dado na lista controlada; essa passa a ser a *playlist* seleccionada
  - `void selectPlaylist(int i)` que selecciona a i-ésima *playlist* se `0≤i<size()` e não faz nada no caso contrário
  - `boolean somePlaylistSelected()` que indica se há alguma *playlist* selecionada
  - `Playlist getSelectedPlaylist()`, que assumindo que `somePlaylistSelected()`, devolve a *playlist* selecionada
  - `void removePlaylist()` que remove a *playlist* selecionada na lista controlada se esta existir, senão não faz nada
  - `Iterator<Playlist> iterator()` que devolve um iterador com as *playlists* na lista controlada.
  - `int numberOfSongs()` que, assumindo `somePlaylistSelected()`, dá o número de músicas na *playlist* selecionada
  - `void addSong()` que, assumindo `somePlaylistSelected()`, insere a música selecionada na biblioteca da *playlist* selecionada (se a operação for possível). Essa música passa a ser a selecionada. Se não existir nenhuma música selecionada na biblioteca não faz nada.
  - `void selectSong(int i)` que, assumindo que `somePlaylistSelected()`, se `0≤i<getSelectedPlaylist().numberOfSongs()`, seleciona a i-ésima música na *playlist* selecionada.
  - `boolean someSongSelected()` que verifica se `somePlaylistSelected()` e, em caso positivo, se também `getSelectedPlayList().someSelected()`
  - `void removeSelectedSong()` que, assumindo que `someSongSelected()`, apaga a música selecionada na *playlist* selecionada.
  - `void nextSong()` que, assumindo `somePlaylistSelected()`, despacha o pedido de seleção da próxima música para a *playlist* selecionada.
  - `void previousSong()` que, assumindo `somePlaylistSelected()`, despacha o pedido de seleção da música anterior para a *playlist* selecionada.
  - `void play()` que, se `someSongSelected()`, determina a interrupção da música que estiver a tocar (se for esse o caso) e começa a tocar, por ordem, todas as músicas da *playlist* correntemente selecionada, a partir da música correntemente selecionada. O contador de uma música é incrementado sempre que a música é tocada até ao fim. Se !`someSongSelected()`,não faz nada.
  - `void stop()` que pára a música que estava a ser tocada, se essa música foi posta a tocar através de uma *playlist*. Senão, não faz nada.
- classe `LEITunes` que fornece o objeto inicial do sistema contendo uma `MusicLibrary` e uma `PlaylistList`. Adicionalmente, contém um `MusicLibraryController` e um `PlaylistListController`.

Pacote `services`

Este pacote deve conter as classes que fornecem serviços de baixo nível à aplicação. Dado que ainda não foi decidido que solução usar para obter os meta-dados das músicas, atualmente tem apenas:

- Classe `JLPlayer` (fornecida) que estende a classe `javazoom.jl.player.Player` e que é quem sabe carregar ficheiros mp3s e reproduzir o seu conteúdo.

3 O que há a fazer?

O trabalho a realizar consiste em implementar a solução fornecida e tomar as decisões em falta, de forma a obter uma versão funcional da aplicação.

É preciso tomar em atenção que a solução de desenho descrita não é suficientemente detalhada para que não tenha de tomar ainda algumas decisões ao nível da implementação (por exemplo, a acessibilidade das classes, a factorização de código através de definição e utilização de classes abstratas, a redefinição apropriada dos métodos herdados de *Object*). Na implementação dos métodos *toString* devem seguir a representação textual fornecida em anexo. Devem ser tomadas as decisões que forem consideradas mais adequadas relativamente a tudo o que não estiver definido.

Relativamente à leitura dos meta-dados das músicas a partir de ficheiros *mp3*, existem várias soluções. Sugere-se a utilização da biblioteca de código aberto disponível em https://github.com/mpatric/mp3agic (o *jar* desta biblioteca está incluída no projeto fornecido).

As classes devem estar convenientemente documentadas com ***JavaDoc***. Devem usar o plugin ***SonarLint*** para o Eclipse (ou alternativa equivalente para o vosso IDE favorito) para vos ajudar a controlar a qualidade do vosso código. Tendo em vista facilitar o teste do código que vão desenvolver, é fornecido um programa **cliente** que exercita algumas funcionalidades da aplicação e também um interface gráfico. Isto, porém, não dispensa a escrita de testes. É esperado que entregue testes unitários ***JUnit*** para`ArrayQListWithSelection` e `Song`.

Devem usar o sistema de controlo de versões ***git***, para vos ajudar no desenvolvimento cooperativo. É importante que cada um dos elementos do grupo torne a sua atividade no trabalho visível no repositório pois esta informação será usada para a nota do trabalho, que é individual.

4 Por onde começar?

Devem importar o projeto fornecido e analisá-lo. Trata-se de um projeto Eclipse com o código fornecido. Devem especialmente começar por ler o README do projeto.

Um dos elementos do grupo de trabalho deverá criar um repositório git com o nome *leitunes_XXXXX_YYYYY*, onde *XXXXX* e *YYYYY* são os números dos membros do grupo, e definir a visibilidade do projeto como privada. Além de dar acesso ao colega de grupo com o nível *Maintainer*, deve adicionar à lista de membros do projeto o utilizador dco000 com o nível de *Reporter*.

A relação de dependência entre as classes define naturalmente uma ordem de implementação: primeiro desenvolvem-se as classes sem dependências, e depois as que apenas dependem destas, e assim sucessivamente

5 Como e quando entregamos?

Identifiquem o *commit* de entrega com `git tag entrega` e coloquem essa identificação no servidor (`git push origin entrega`). O *deadline* para entrega é **3 de Abril**.

O repositório deve conter:

1. um documento (txt/md/pdf) a descrever sumariamente as decisões que tomaram;
2. o código fonte do vosso projeto

6 Critérios de Correção

Para ajudar a guiar o vosso esforço indicam-se alguns critérios de correção que vamos usar:

- Se o código compila e o programa cliente fornecido executa até ao fim.
- Em que medida o comportamento do sistema, em cada cenário de teste, é o esperado (isto é feito através da execução do programa cliente fornecido e da execução de testes automáticos)
- Em que medida a organização e legibilidade do vosso código segue as boas práticas e está de acordo com o que vos foi fornecido.
- Em que medida as classes e comportamentos definidos estão de acordo com o que vos é pedido: cada classe deve suportar os métodos descritos e com os parâmetros adequados, o comportamento dos métodos dever ser o indicado.
- Em que medida as decisões sobre acessibilidade das classes e fatorização do código são adequadas e estão de acordo com o princípio do *Information Hiding*.
- Em que medida os métodos herdados de *Object* foram redefinidos de forma apropriada, nomeadamente os metódos *toString()* seguem o exemplo dado.

- Em que medida a documentação de interfaces/classes é adequada, nomeadamente se há contratos a descrever o comportamento dos construtores e métodos acessíveis.
- Em que medida os testes unitários *JUnit* pedidos foram realizados, seguem as melhores práticas e têm uma cobertura adequada das funcionalidades esperadas.
- Em que medida a leitura do relatório permite saber as decisões que tomaram.

ANEXO

O output indicativo do *SimpleClient* fornecido.

```
------------------------------------------------
Adding songs to library
------------------------------------------------
------------------------------------------------
Playlist On the Go created
------------------------------------------------
------------------------------------------------
Playlist Relax created
------------------------------------------------
------------------------------------------------
Songs [0, 2, 5, 7] added to On the Go
------------------------------------------------
------------------------------------------------
Songs [9, 2, 8] added to Relax
------------------------------------------------
*****MUSIC LIBRARY****
0 [Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0 -- 0
1 [Meu Fado Meu, Transparente, Pop, [Mariza]] --- 0 -- 0
2 [Fado Tordo, Transparente, Pop, [Mariza]] --- 0 -- 0
3 [Odeio, Cê, Latin, [Caetano Veloso]] --- 0 -- 0
4 [Homem, Cê, Latin, [Caetano Veloso]] --- 0 -- 0
5 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
6 [o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
7 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0->
8 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
9 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
***** PLAYLISTS *****
*-- Playlist Most Liked--*
*-- Playlist Most Recently Added--*
0 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
1 [o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
2 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
3 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
4 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
*-- Playlist On the Go--*
0 [Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0 -- 0
1 [Fado Tordo, Transparente, Pop, [Mariza]] --- 0 -- 0
2 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
3 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
*-- Playlist Relax--*
0 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
1 [Fado Tordo, Transparente, Pop, [Mariza]] --- 0 -- 0
2 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
------------------------------------------------
Search songs in library by .*VO.*
------------------------------------------------
[o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
[Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
------------------------------------------------
Select song 2 from library and remove it
------------------------------------------------
*****MUSIC LIBRARY****
0 [Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0 -- 0
1 [Meu Fado Meu, Transparente, Pop, [Mariza]] --- 0 -- 0
2 [Odeio, Cê, Latin, [Caetano Veloso]] --- 0 -- 0
3 [Homem, Cê, Latin, [Caetano Veloso]] --- 0 -- 0
4 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
5 [o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
6 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
7 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
8 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
***** PLAYLISTS *****
*-- Playlist Most Liked--*
*-- Playlist Most Recently Added--*
0 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
1 [o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
2 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
3 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
4 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
*-- Playlist On the Go--*
0 [Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0 -- 0
1 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
```

```
2 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
*-- Playlist Relax--*
0 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
1 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
-------------------------------------------------
Playing from library
-------------------------------------------------
Select and play 0
Selected Optional[[Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0
-- 0]
Wait 5 seconds and select 2
Selected Optional[[Odeio, Cê, Latin, [Caetano Veloso]] --- 0 -- 0] and wait 5seconds
Let's change the mood and play selected
I like this music! Let's increase its rank twice
Lets wait a bit and stop it
*****MUSIC LIBRARY****
0 [Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0 -- 0
1 [Meu Fado Meu, Transparente, Pop, [Mariza]] --- 0 -- 0->
2 [Odeio, Cê, Latin, [Caetano Veloso]] --- 2 -- 0
3 [Homem, Cê, Latin, [Caetano Veloso]] --- 0 -- 0
4 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
5 [o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
6 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
7 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
8 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
***** PLAYLISTS *****
*-- Playlist Most Liked--*
0 [Odeio, Cê, Latin, [Caetano Veloso]] --- 2 -- 0
*-- Playlist Most Recently Added--*
0 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
1 [o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
2 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
3 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
4 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
*-- Playlist On the Go--*
0 [Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0 -- 0
1 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
2 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
*-- Playlist Relax--*
0 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
1 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
-------------------------------------------------
Playing from a playlist
-------------------------------------------------
Library selected
*-- Playlist Relax--*
0 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 0
1 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 0
Select first and play all in playlist
*****MUSIC LIBRARY****
0 [Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0 -- 0
1 [Meu Fado Meu, Transparente, Pop, [Mariza]] --- 0 -- 0->
2 [Odeio, Cê, Latin, [Caetano Veloso]] --- 2 -- 0
3 [Homem, Cê, Latin, [Caetano Veloso]] --- 0 -- 0
4 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
5 [o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
6 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
7 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 1
8 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 1
***** PLAYLISTS *****
*-- Playlist Most Liked--*
0 [Odeio, Cê, Latin, [Caetano Veloso]] --- 2 -- 0
*-- Playlist Most Recently Added--*
0 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
1 [o mundo é já aqui, unkown, Pop, [OVO]] --- 0 -- 0
2 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
3 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 1
4 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 1
*-- Playlist On the Go--*
0 [Bach: Goldberg Variations, BWV 988 - Variatio 13 A 2 Clav., Bach: Goldberg Variations, Classical, [Glenn Gould]] --- 0 -- 0
1 [Mozart: Exsultate, Jubilate, K 165 - 2. Tandem Advenit Hora, Mozart: Exsultate Jubilate!, Classical, [Carolyn Sampson; Robert King:
The King's Consort & Choir]] --- 0 -- 0
2 [Dormir, OVO, Pop, [Ovo]] --- 0 -- 0
*-- Playlist Relax--*
0 [unkown, Suite April 2020, unkown, [Brad Mehldau]] --- 0 -- 1
1 [unkown, West Side Story, Soundtrack, [Various artists]] --- 0 -- 1
```