Trabalho Prático 1

LEITunes

@author: Daniela Camarinha fc58199
@author: Rafael Ribeiro fc58193

Decisões de implementação:

domain.core

Rate: A nossa Classe Rating representa objetos que descrevem o rating das músicas. Podem ter dois estados (UNRATED, LIKED), isto pois queríamos aproximar esta característica o mais possível com o "coração" da aplicação *Spotify.* No *Spotify*, quando uma música tem o coração a verde, esta música é colocada na "Liked Playlist", tal e qual como no LEITunes. Mas quando não tem "coração" não podemos dizer nada sobre ela, simplesmente ouví-la, deste modo o estado UNRATED torna-se apropriado.

Ao contrário do que está explícito no enunciado, os nossos objetos RATE são mutáveis. Quando são chamada a partir de funções como decRateSelected e IncRateSelected estas alteram o estado do nome (this.rate = RATE.UNRATED) e o próprio estado interno do Rate (dando forwarding para funções internas do enumerado incRating(), decRating()) que alteram o número associado ao estado.

Acreditamos que o efeito de alterar o Rate podia ter sido implementado de outra maneira, podíamos ter colocado na classe Song um método chamado setRate, chamado aquando da chamada das funções decRateSelected e IncRateSelected que criasse um novo objeto Rate com um estado diferente do original e trocasse o atributo Rate da Song.

Esta decisão foi tomada devido á possibilidade de criação de múltiplos objetos Rate em memória que podem já não estar a ser usados. Apesar do JAVA ter a capacidade de apontar para objetos iguais em memória, reutilizando-os, achamos que o Rate deve ser um estado flutuante, intrínseco a cada Song.

MusicLibrary: A nossa classe Music Library estende AbsQListWithSelection, e implementa as interfaces descritas no enunciado. Esta decisão foi tomada para facilitar a reutilização de código. Ao herdar métodos da classe abstrata, a implementação dos métodos da interface QListWithSelection são essencialmente forwardings para os métodos de AbsQListWithSelection. É nos também fornecido com esta extensão os atributos list e selectedIndex, que representam a Music Library em si.

domain.playlists

PlaylistList: Esta classe estende AbsQListWithSelection e implementa as interfaces descritas no enunciado. Da mesma forma que havia reutilização de código na Music Library, a extensão de AbsQListWithSelection na PlaylistList funciona exatamente da mesma forma. Há forwarding para os métodos da abstrata aquando da implementação dos métodos de QListWithSelection, com a única diferença de que nos métodos remove e add são registadas ou apagadas dos Listeners da Music Library as Playlists correspondentes. Por herança somos capazes de utilizar os atributos que irão representar a própria PlaylistList.

Leitura dos meta-dados: Feita com o auxílio da biblioteca open-source fornecida no enunciado.