Projeto Final - Programação Orientada ao Objeto

Gustavo de Oliveira, João Lucas Melo

1. CONTEXTO E MOTIVAÇÃO

Trata-se de aplicação que simula, utilizando os conceitos do Paradigma de Programação Orientada ao Objeto, um controlador de biblioteca de músicas, contendo todas as funções que se esperam de tal, como criação e manipulação de coleções de músicas, assim como funções para usuários e artistas que dela participam. A motivação se deu quando nós dois, estávamos conversando sobre o Spotify, famoso aplicativo e serviço em geral de streaming de música, podcasts e etc. Então, idealizamos baseado nele e suas funções de manipulações de biblioteca de músicas.

Planejamos o sistema de maneira que, tal como convenciona o paradigma de orientação ao objeto, procurar entidades reais presentes em serviços reais de manipulação de coletâneas de músicas, e abstraí-los em classes com seus métodos e atributos, e fazer com que todas elas se relacionassem, utilizando de conceitos como composição, herança e polimorfismo para deixar os código mais funcional e escalável possível.

2. CLASSES UTILIZADAS

2.1 Comportamentos e conceitos pertinentes recorrentes na aplicação

2.1.1 Muitas classes possuem o atributo controleId e o método incrementarId, eles servem para poder criar e controlar o atributo estático id, para que cada instância de tal classe possua um identificador singular para si.

2.1.2 Todas as classes possuem o método retornaDados, que retorna string contendo todas as informações presente para ser mostrada na tela ou apenas retornada.

2.2 Interfaces, classes, enums, classes abstratas

2.2.1 Coletanea (classe abstrata): Trata-se da representação mais abstrata de literalmente uma coletânea de músicas, um conjunto que possui como atributos: nome (nome da coletânea), qtdMusicas (quantidade total de músicas), duracao (duração total da coletânea baseado na soma da duração de todas as músicas), listaMusicas (lista de músicas), listaArtistas (lista de artistas que participam desta coletânea). Possui os métodos: retornaDados (uma string contendo todas as informações da coletânea) adicionarMusica, removerMusica, adicionarArtista, incrementarId.

2.2.2 Album (classe): Classe filha de Coletanea, trata-se da representação de um Album real, sua diferenciação nada mais é que apenas quem pode criar é um Artista. Utiliza os métodos de sua superclasse para poder manipular quais músicas estão presentes no álbum.

2.2.3 Playlist (classe): Assim como Album, trata-se de uma classe filha de Coletanea, a diferença que quem a faz é apenas são Usuario.

2.2.4 Pessoa (classe abstrata): Trata-se da representação mais abstrata possível de todas aquelas que irão utilizar a aplicação. Possui como atributo: nome,

controleId, listaMusicas (lista de músicas), listaGeneros (lista de gêneros que o usuário alguma vez adicionou em sua lista de músicas), listaAlbuns (lista de álbuns).

2.2.5 Usuario (classe): Representação de um usuário, subclasse de Pessoa, absolutamente todos aqueles que manipulam conjuntos de músicas.

2.2.6 Artista (classe): Representa aquelas que podem além de manipular os conjuntos de músicas para si, podem criar músicas, subclasse de Usuario, afinal, todo Artista está utilizando o sistema, logo é usuário, porém nem todo usuário é um artista. Possui como atributo musicasProprias (lista de músicas que ele mesmo criou) e albunsProprios (lista de álbuns que ele criou), além de métodos para que ele possa manipular suas criações como aidiconarMusicaAlbum.

2.2.7 Audio (interface): Ao pensar em um áudio, nota-se que há certos comportamentos e manipulações que o usuário pode fazer, logo foi implementada uma interface para manipulação de todos estas possíveis manipulações possuindo: adicionarGenero (adicionar um gênero a tal Audio), aumentarVelocidade e diminuirVelocidade (ambas para alterar o atributo duracao).

2.2.8 Musica (classe): Classe que implementa a interface Audio, possui todos os atributos que possui uma música real: nome, duracao, album (Album a qual ela pertence), artista (Artista que criou a música), listaGeneros (lista de todos os gêneros que tal música possui). Além de todos os métodos para além de validar e retornar, além de aqueles que possuem a interface, como retornaDados e incrementarId.

2.2.9 EnumGenero (enum): Todos os gêneros válidos na aplicação, desta maneira não cabe ao usuário ou qualquer outro, manipular ou criar de maneira indevida algum gênero.

2.2.10 Genero (classe): Composta pelo atributo enumGenero, permite manipular os gêneros e facilitar sua inserção em listas, além do método retornaDados.

2.2.11 Aplicacao (classe): Classe central, que interliga todos as classes e unifica como uma real aplicação. Possui apenas listas como atributos, estas que servem para controlar tudo aquilo que é cadastrado na aplicação, ou seja: listaUsuario, listaArtista, listaMusica, listaAlbum, listaPlaylist, listaGenero (esta possui uma peculiaridade, pois todos os gêneros são criados previamente, não cabendo ao usuário ou artista alterá-la). Como métodos, estes sim todos aqueles que permitem manipular todos as músicas e bibliotecas: cadastrarUsuario, cadastrarArtista, cadastrarMusica, cadastrarAlbum, criarGenero (método para criar todos os generos), consultarUsuario, consultarArtista, consultarMusica, consultarAlbum, consultarGenero, consultarPlaylist (todas essas atráves de uma busca pelo id de cada um), definirGeneroPlaylist, adicionarMusicaPlaylist, removerMusicaPlaylist, aposentarArtista (atráves de cast, transforma um Artista em Usuario), acelerarMusica, desacelerarMusica, showInstrucoes (função com print para mostrar como utilizar a aplicação).