MC322 – Programação Orientada a Objetos
Laboratório 02 – 2s2021

Leonardo Montecchi (Professor) Thales Eduardo Nazatto Ângelo Renato Pazin Malaguti
Leonardo de Sousas Rodrigues

Para perguntas ou dúvidas usem o Discord (`https://discord.gg/Xzpz9epNTG`)

1 Objetivos e submissão

Objetivos

- Aplicar os conceitos de programação a objetos e declaração de classes.

Data de entrega

- **29/08/2021** até às **23h59**.

Submissão

- Ao criar o projeto Java, selecionar a versão **JavaSE-11** no JRE
- IMPORTANTE: Nomear o projeto na forma **RA_Lab02** e o pacote base na forma **com.unicamp.mc322.lab02**. Substitua RA com o seu Registro Acadêmico (matrícula).
- Submeta o trabalho no link de entrega na página do Classroom da disciplina, em formato de arquivo compactado (zip) com o nome **RA_Lab02.zip**. Substitua RA com o seu Registro Acadêmico (matrícula).
- O arquivo compactado deve conter o projeto inteiro ("File / Export" no Eclipse, ou crie o arquivo manualmente).

Critérios de avaliação

- Este laboratório **não vale** nota.

2 Exercício : Musicfy

Desenvolver um programa capaz de instanciar usuários, músicas e *playlists*. O objetivo é armazenar as músicas em playlists adequadas para cada usuário. As classes sugeridas são:

- **Song.java** - A classe Song deverá armazenar as informações da música, como nome, gênero musical, artista, duração da música. Além de métodos para a alteração de qualquer um desses atributos;
- **User.java** - A classe User deverá armazenar as informações do usuário, como nome, cpf, data de nascimento, gênero e se o usuário é assinante ou não. Deverá, também, possuir métodos para adição e remoção de playlists, além de possibilitar a transferência de uma playlist para outro usuário. Se o usuário for assinante, poderá possuir até 10 playlists, caso contrário poderá possuir apenas 3. A qualquer momento o usuário pode cancelar a assinatura; a alteração deve ser refletida nas limitações no número de playlists, assim como o tamanho dessas (veja a classe Playlist).

- **Playlist.java** - Uma Playlist possui um nome e o gênero musical da playlist. Caso o usuário seja assinante, a playlist poderá armazenar até 100 músicas, senão deverá armazenar apenas 10. Ao serem adicionadas à playlist, as músicas devem ser ordenadas alfabeticamente pelo nome. A playlist deverá implementar métodos para adicionar e remover músicas, além de métodos para retornar: 1) a música de menor duração; 2) a música de maior duração; 3) a duração média das músicas da playlist; 4) a duração total da playlist; 5) o artista que possui mais músicas na playlist. Uma playlist deverá possuir o método **play()**; toda vez que este método for chamado, retornará a próxima música da playlist. No entanto, se o método **play()** for chamado com o parâmetro **shuffle** igual a verdadeiro, a música retornada deverá ser aleatória, porém diferente da música atual.

A aplicação principal é representada pela classe **Musicfy**, que possui o método **main()**. Abaixo é fornecido um código que serve como base para a construção do programa. Note que esse é apenas um exemplo de execução, deverão ser implementadas as funcionalidades listadas acima. Tudo o que não está especificado neste texto fica como livre escolha do aluno.

```
package com.unicamp.mc322.lab02;

public class Musicfy {

  public static void main(String[] args) {
    User user1 = new User("Marcos Paulo", "777.777.777-77");
    User user2 = new User("Cookiezi", "111.111.11-11");

    Song song1 = new Song("Seven Nation Army", "Rock", "The White Stripes");
    Song song2 = new Song("Crazy Train", "Rock", "Ozzy Osbourne");
    Song song3 = new Song("Feels", "Pop", "Calvin Harris");
    Song song4 = new Song("Roar", "Pop", "Katy Perry");
    Song song5 = new Song("Anima", "Hardcore", "Xi");
    Song song6 = new Song("Freedom Dive", "Hardcore", "Xi");
    Song song7 = new Song("Teo", "Hardcore", "Omoi");
    Song song8 = new Song("Sleepwalking", "Metalcore", "Bring Me The Horizon");

    Playlist rockPlaylist = new Playlist("Awesome Rock Songs", "Rock");
    rockPlaylist.addSong(song1);
    rockPlaylist.addSong(song2);

    Playlist osuPlaylist = new Playlist("Osu Memories", "hardcore");
    osuPlaylist.addSong(song5);
    osuPlaylist.addSong(song6);
    osuPlaylist.addSong(song7);

    Playlist metalcorePlaylist = new Playlist("Best of Metalcore", "Metalcore");
    metalcorePlaylist.addSong(song8);

    user1.addPlaylist(rockPlaylist);
    user1.addPlaylist(metalcorePlaylist);
    user2.addPlaylist(osuPlaylist);

    System.out.println(user1.showPlaylists());
    System.out.println("");
    System.out.println(user2.showInformation());

    Song asong1 = osuPlaylist.play();
    Song asong2 = osuPlaylist.play();
    Song asong3 = osuPlaylist.play(true);
  }
}
```

Código Fonte 1: Classe Musicfy com o método main.

Como pode ser visto no código acima, a impressão dos detalhes se dá pelos métodos **showInformation()** e **showPlaylists()** da classe **User**, que para retornar as informações pertinentes ao usuário (eg, nome, cpf, etc.) e às playlists do usuário, respetivamente.

- Exemplo de saída esperada para o método **showInformation()**:

```
Nome: Fulano de Tal
CPF: 123.456.789-10
...
```

- Exemplo de saída esperada para o método **showPlaylists()**:

```
User: Marcos Paulo
Number of Playlists: 2
Playlist 1: Awesome Rock Songs
    Number of Songs: 2
    Songs:
    - Seven Nation Army - The White Stripes;
    - Crazy Train - Ozzy Osbourne;
Playlist 2: Best of Metalcore
    Number of Songs: 1
    Songs:
    - Sleepwalking - Bring Me The Horizon;
```

Um arquivo de texto com o código da classe Musicfy está anexado no Classroom.

3 Documentação de ajuda e dicas

3.1 Dicas

- Existem várias classes em Java para representar datas. Podem também criar uma própria classe personalizada para representar esse conceito.
- Tipos enumerados (`enum`) são uma boa escolha para representard dados que tem um número fixo de valores (e.g., dias da semana). `https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html`
- Tente focar apenas nos métodos que forem necessários para implementar as funcionalidades descritas no texto.

3.2 Boas Práticas em Java

- Nomes de classes devem começar com letra maiúscula;
- Nomes de variáveis e métodos devem começar com letra minúscula;
- Nomes de classes devem ser substantivos;
- Nomes de métodos devem ser verbos ou começar com verbo;
- Nomes compostos por mais de uma palavra devem ser escritos na forma *CamelCase*. Isto é, com a primeira letra de cada uma das demais palavras em maiúsculo. Por exemplo: "get playlists"deve ser escrito como **getPlaylists** (método), e uma classe que representa uma lista de músicas poderia ser declarada como **SongList**.