MC322 – Programação Orientada a Objetos
Laboratório 02 – 1s2020

Leonardo Montecchi (Professor)
leonardo@ic.unicamp.br

Iury Cleveston (PED)
iurycl@gmail.com

Matheus Rotta (PAD)
matheusrotta7@gmail.com

Matheus Mazon (PAD)
matheusmazon@outlook.com

24/03/2020

1 Objetivos

O objetivo deste laboratório é aplicar os conceitos de programação orientada a objetos aprendidos em sala de aula.

2 Boas Práticas

1. Nomes de classes devem começar com letra maiúscula;
2. Nomes de variáveis e métodos devem começar com letra minúscula;
3. Nomes de classes devem ser substantivos;
4. Nomes de métodos devem ser verbos ou começar com verbo;
5. Nomes compostos por mais de uma palavra devem ser escritos na forma *CamelCase*. Isto é, com a primeira letra de cada palavra maiúscula. Por exemplo: ”get playlists”deve ser escrito como **getPlaylists** (método), e uma classe que representa música poderia ser declarada como **Song**.

3 Exercício - Musicfy

Desenvolva um programa que seja capaz de instanciar usuários, músicas e playlists. Crie as seguintes classes:

1. **User.java** - A classe User deverá armazenar as informações do usuário, como nome, cpf, data de nascimento, gênero e se o usuário é assinante. 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, senão deverá possuir apenas 3. Além disso, a classe User deverá possuir um método para permitir a alteração do tipo de assinatura, bem como as implicações nas limitações de armazenamento de músicas (veja item 3).

2. **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;

3. **Playlist.java** - A classe Playlist deverá armazenar nome e gênero da playlist. Caso o usuário seja assinante, a playlist poderá armazenar até 100 músicas, senão deverá armazenar 10. 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. Cada playlist deverá possuir o método **play()**, que retornará uma música, toda vez que este método for chamado, retornará a próxima música da playlist. As músicas devem ser ordenadas alfabeticamente pelo nome. 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.

Para isso, a classe **Musicfy**, que possui o método **main()**, deverá ser criada. Abaixo é fornecido um código para servir de base para a construção do programa, note que os demais atributos e métodos requeridos na descrição deverão ser adicionados. Tudo que não está especificado fica a critério do aluno.

```
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");

    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);

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

```
    System.out.println(user1.getPlaylists()[0].getDetails());
    System.out.println("");
    System.out.println(user2.getDetails());

    osuPlaylist.play();
    osuPlaylist.play();
    osuPlaylist.play(true);
  }
}
```

Tiramos os números das linhas para que vocês possam copiar o código, porém a indentação será perdida. Tentem usar um beautifier online (`https://www.techiedelight.com/tools/java`) ou a função **Reindent** do Eclipse (Ctrl-Shitf-F).

Perceba que os construtores de cada classe deverão ser implementados de maneira apropriada, e também será necessário escrever o método **getDetails()** nas três classes, para retornar as informações pertinentes a cada classe (e.g Total de músicas; Nome das músicas; Nome das playlist, etc).

Exemplo de saída esperada para o método **getDetails()**:

```
Playlist Awesome Rock Songs has songs: Seven Nation Army, Crazy Train.

User Cookiezi has playlists: Osu Memories.
```

4 Submissão

Submeta o trabalho no link de entrega na página do Classroom da disciplina, em formato de arquivo compactado (zip). Envie o arquivo com o nome {**ra**}_**Lab02.zip**. Arquivos a serem submetidos:

- Musicfy.java
- User.java
- Playlist.java
- Song.java

Este laboratório **não vale** nota. Entrega para o dia **31/03/2020**.