UDESC - Fundação Universidade do Estado de Santa Catarina
Professor: Fabiano Baldo
Disciplina: Programação Orientada a Objetos

O trabalho deve ser realizado individualmente. Os arquivos devem ser compactados em um arquivo *.zip* ou *.tar* contendo o nome do aluno. O arquivo compactado deverá conter apenas os códigos-fontes (pasta src do Eclipse) e os diagramas UML em formato *.jpg* ou *.png*. **Não serão aceitos projetos com códigos-fonte no formato *.class*!**

Trabalho de POO

Tema 1: Aplicativo de Músicas

Uma equipe de desenvolvedores está planejando criar uma aplicativo de músicas, tal como o Spotify. No aplicativo existirão usuários, músicas, artistas e playlists. Usuários podem favoritar suas músicas preferidas, além de criar playlists e adicionar músicas a elas. Cada música possui um ou mais artistas e cada artista pode possuir várias músicas dos quais ele é autor. O usuário deve ser capaz de fazer *upload* de músicas .mp3 e tocar uma prévia dessa música (código para realizar isso ao final desse enunciado).

O aplicativo deve possuir as seguintes funcionalidades:

- Permitir que o usuário faça upload de novas músicas, selecionando um ou mais artistas que participam da música;
- Permitir que o usuário liste as músicas cadastradas juntamente com o artista que elaborou a música.
- Permitir que o usuário liste os artistas cadastrados juntamente com as músicas que este artista participa.
- Permitir que o usuário liste as playlists cadastradas juntamente com as músicas que a compõe;
- Permitir que o usuário toque uma prévia da música em qualquer lugar que seja possível vê-la (seja na hora de listar as músicas, ou na hora de listar as músicas de um artista ou na hora de listar uma playlist);
- Também deve ser possível que o usuário exclua e adicione novas músicas a uma playlist, além de poder alterar ou remover músicas e artistas do aplicativo;
- O aplicativo deve manter dados para múltiplos usuários, isto é, cada usuário terá suas músicas e playlists, além de solicitar a autenticação (usuário e senha) do usuário ao se conecte ao aplicativo;

Implemente o sistema descrito anteriormente, modelando as classes com atributos que façam sentido ao contexto da classe.

Para tocar uma música usando Java, veja o trecho de código a seguir:

```
try {
    FileInputStream fileInputStream = new FileInputStream("musica.mp3");
    Player player = new Player(fileInputStream);
    System.out.println("Song is playing...");
    player.play(300);
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (JavaLayerException e) {
    e.printStackTrace();
}
```

O método **play(int frames)** determina a quantidade de frames que serão tocados. Ao utilizar 300, garante-se que irá tocar apenas uma prévia da música. Para executar esse código é necessário adicionar as bibliotecas **jlayer** e **jl** ao **build path**. A biblioteca **jl** pode ser encontrada neste link. Já a biblioteca **jlayer** pode ser obtida neste link do site jar-download. Ambos os jars devem ser adicionados ao **build path** ou o arquivo da música não será tocado.

Para persistir as informações no Postgres, veja um exemplo tirado da própria documentação (note que o exemplo utiliza um bytea, i.e., um array de bytes, utilize tal tipo de dado para armazenar as músicas):

Para persistência:

```
CREATE TABLE images (imgname text, img bytea);
```

```
File file = new File("myimage.gif");
FileInputStream fis = new FileInputStream(file);
PreparedStatement ps = conn.prepareStatement("INSERT INTO images VALUES (?, ?)");
ps.setString(1, file.getName());
ps.setBinaryStream(2, fis, file.length());
ps.executeUpdate();
ps.close();
fis.close();
```

Para consulta:

```
PreparedStatement ps = con.prepareStatement("SELECT img FROM images WHERE imgname = ?");
ps.setString(1, "myimage.gif");
ResultSet rs = ps.executeQuery();
if (rs != null) {
    while (rs.next()) {
        byte[] imgBytes = rs.getBytes(1);
        // use os dados de alguma forma
    }
    rs.close();
}
ps.close();
```