Documentação do Projeto ED1 - PobreFy

1. Descrição Geral

Esse documento visa mostrar a estrutura, métodos e lógicas aplicadas na implementação do conceito de listas duplamente encadeadas circulares em um programa de criação de playlists de músicas. Nosso trabalho foi feito com a linguagem de programação Java. Esse projeto Java conta com 6 Classes:

- `Main`
- `Playlist`
- `Musica`
- `ArquivoObjeto`
- `ArquivoTexto`
- `Menu`

---

2. Classes

2.1 `Main`

- **Descrição**: Classe que contém o método main(). É esse método que gerencia a chamada dos métodos das classes utilizadas de acordo com a opção selecionada pelo cliente através do Console.

2.2 `Playlist`

- **Descrição**: Classe que define a estrutura da lista encadeada dupla aplicada neste projeto, através dela é possível acessar tanto as músicas como também outras playlists. Ela funciona como uma lista encadeada dupla de músicas da classe Musica, e também como um nó duplamente encadeado para outra playlist da classe Playlist.

2.3 `Musica`

- **Descrição**: Classe que contém a estrutura básica das músicas utilizadas no software, além disso funciona como um “nó” para a lista duplamente encadeada.

2.4 `ArquivoObjeto`

- **Descrição**: Aponta para a playlist anterior na lista circular de playlists.

2.5 `ArquivoTexto`

- **Descrição**: Classe que serve para a manipulação do arquivo de texto que contém as músicas utilizadas pelo programa.

2.6 `Menu`

- **Descrição**: Classe que contém os métodos referentes a leitura de dados do teclado, também possui as implementações de textos e símbolos mostrados no Console durante a execução do programa.

---

3. Atributo das Classes

3.1 `Playlist`

- **Musica** cabeca;
  - Descrição: Recebe a referência da primeira música (Nó) que faz parte da lista de músicas encadeadas, caso esteja sem música, recebe null.
- **String** nome;
  - Descrição: Nome da playlist.
- **Playlist** cabecaPlaylist;
  - Descrição: Recebe a referência da primeira lista de músicas da lista duplamente encadeada de playlists.
- **Playlis**t proximo;
  - Descrição: Recebe a referência da próxima lista de músicas.
- **Playlist** anterior;
  - Descrição: Recebe a referência da lista de músicas anterior.

3.2 `Musica`

- **String** nome;
  - Descrição: Nome da música.
- **String** cantor;
  - Descrição: Nome do cantor ou banda.
- **String** tempo;
  - Descrição: Tempo de duração da música.
- **String** album;
  - Descrição: Nome do álbum em que a música pertence.
- **String** genero;
  - Descrição: Nome do gênero da música.
- **Musica** proximo;

    - Descrição: Referência para a próxima música ou próximo nó da lista encadeada.
- **Musica** anterior;
    - Descrição: Referência para o endereço da música anterior ou nó anterior.

3.3 `ArquivoObjeto`

- **File** arquivo;
    - Descrição: Referência para o arquivo que será usado para leitura e escrita binária de objetos.

3.4 `Menu`

- **Scanner** sc;
    - Descrição: Objeto usado para leitura de dados do teclado.

---

4. Métodos das Classes

Nesse documento está presente somente a descrição dos métodos das classes que apresentam a estrutura do conceito explorado por esse projeto, lista duplamente encadeada circular. As Classes não detalhadas possuem apenas ferramentas que auxiliam na execução do software, mas não são essenciais para o objetivo principal.

4.1. Classe Musica

4.1.1 `getters e setters dos atributos`

- **Descrição**:Definição e retorno dos respectivos atributos das classes, permite a manipulação desses dados..
- **Retorno**:No caso dos getters eles retornam exatamente o atributo ao qual fazem referências.

4.1.2 `public Musica(String cantor,String nome, String album,String genero,String tempo)`

- **Descrição**: Construtor da classe, configuram os atributos de acordo com as informações fornecidas e fazem os atributos proximo e anterior apontarem para si mesmos.
- **Parâmetros**:
    - `String cantor:`Nome do cantor ou banda.
    - `String nome:` Nome da música.
    - `String album:` Nome do álbum.
    - `String genero:` Nome do genero.
    - `String tempo:` Tempo de duração da música.

4.1.3 `public String toString()`

- **Descrição**:Retorna uma string com a descrição de uma música, escrito todos os seus atributos.
- **Retorno**: A descrição da música: Cantor(artista/banda) + Nome(nome da música) + Álbum + Gênero + Tempo

---

4.2. Classe Playlist

4.2.1 `getCabecaPlaylist() e setCabecaPlaylist(Playlist cabecaPlaylist)`

- **Descrição**: Definição e retorno da primeira playlist da lista de playlists.
- **Retorno**:No caso do get ele retorna exatamente o atributo ao qual faz referência.
- **Parâmetros (setCabecaPlaylist())**:
    - `Playlist cabecaPlaylist:` Playlist que será configurada para ser a primeira na lista duplamente encadeada circular de playlists.

4.2.2 `getters e setters dos atributos proximo e anterior`

- **Descrição**:Definição e retorno do proximo elemento ou do elemento anterior da lista de playlists.
- **Retorno**:No caso dos getters eles retornam exatamente o atributo a referência ou do próximo elemento ou do elemento anterior.

4.2.3 `getCabeca() e setCabeca(Musica cabeca)`

- **Descrição**: Definição e retorno da primeira música da lista de músicas.

- **Retorno**:No caso do get ele retorna exatamente a referência ao nó música.
- **Parâmetros (setNome())**:
    - `Musica cabeca:` Objeto da classe Musica que será a primeira música da lista duplamente encadeada circular.

4.2.4 `getNome() e setNome(String nome)`

- **Descrição**: Definição e retorno do nome da playlist
- **Retorno**:No caso do get ele retorna exatamente o atributo(nome da playlist) ao qual faz referência.
- **Parâmetros (setNome())**:
    - `String nome:` Nome que deseja para a playlist.

4.2.5 `public Playlist(String nome)`

- **Descrição**: Construtor do objeto. Quando instanciado dessa maneira, cria uma playlist, configurando a cabeça(Classe Musica) como null e a cabecaPlaylist como null e os atributos proximo e anterior(Classe Playlist) apontando para si mesmos.
- **Parâmetros**:
    - `String nome:` Nome da playlist.

4.2.6 `public boolean estaVaziaPlaylist()`

- **Descrição**: Este método verifica se a lista de playlists está vazia.
- **Retorno**: Retorna true se não houver nenhuma playlist na lista.

4.2.7 `public boolean estaVazia()`

- **Descrição**: Este método verifica se a lista de músicas está vazia.
- **Retorno**: Retorna true se não houver nenhuma música na lista.

4.2.8 `public void inserirMusica(Musica novaMusica)`

- **Descrição**: Este método insere uma nova música ao final da playlist a qual a música está ligada, verifica se a lista de músicas está vazia, se tiver seta a nova música como cabeça, caso contrário usa a lógica de inserir um nó(música)no final da lista, criando um objeto do tipo música para guardar a referência do último elemento da

lista e ,assim, conseguir realizar as novas ligações com os setProximo e setAnterior, garantindo a ciclicidade da lista.

- **Parâmetros**:
    - `Musica novaMusica`: A nova música a ser adicionada..

4.2.9 `public void inserirPlaylist(Playlist novaPlaylist)`

- **Descrição**: Este método insere uma nova playlist ao final da lista de playlists, verifica se a lista de playlists está vazia, se tiver seta a nova playlist como cabeça, caso contrário usa a lógica de inserir um nó(playlist)no final da lista, criando um objeto do tipo playlist para guardar a referência do último elemento da lista e ,assim, conseguir realizar as novas ligações com os setProximo e setAnterior, garantindo a ciclicidade da lista.
- **Parâmetros**:
    - `Playlist novaPlaylist:` A nova playlist a ser adicionada.

4.2.10 `public Musica buscarMusica(String nome)`

- **Descrição**: Esse método busca uma música na playlist através do nome, que é passado como parâmetro na chamada do mesmo. Ele percorre todos os nós(músicas) da lista comparando o nome de cada uma, caso seja o mesmo retorna a música comparada.
- **Parâmetros**:
    - `String nome:` O nome da música que deseja procurar na playlist.
- **Retorno**: A referência do objeto da classe `Musica` encontrada. Caso não encontre ou a lista for vazia, retorna `null`.

4.2.11 `public Playlist buscarPlaylist(String nome)`

- **Descrição**: Esse método busca uma playlist por meio do nome, usando algumas verificações, inicialmente verifica se a lista de playlists está vazia, retornando null caso seja verdade. Caso contrário cria um objeto temporário do tipo playlist que armazena a referência da cabeça da lista de playlists e roda toda essa lista com base nessa playlist temporária fazendo comparações por meio do getNome, caso a verificação seja verdadeira retorna a referência dessa playlist, pois se trata da playlist desejada, caso não entre nessa verificação retorna null, se referindo a não ter achado nenhuma playlist com o nome passado por parâmetro.
- **Parâmetros**:
    - `String nome:` A nome da playlist a ser buscada.
- **Retorno**: A referência do objeto da classe `Playlist` encontrada através do nome da mesma. Caso não encontre ou a lista for vazia, retorna `null`.

4.2.12 `public Musica buscarMusica(int numero)`

- **Descrição**: Este método busca uma música com base na lista de músicas da playlist, ordenadas no menu no console, esse número representará em qual posição a música se encontra na lista em que o usuário visualizou, caso a playlist não esteja vazia, haverá uma procura por toda a lista enquanto ela não acabar e enquanto o número não for encontrado.
- **Parâmetros**:
    - `Int numero:` O número da música a ser buscado na playlist.
- **Retorno**: A música caso encontre, ou null.

4.2.13 `public Playlist buscarPlaylist(int numero)`

- **Descrição**: Esse método busca uma playlist por meio de um número, usando algumas verificações, inicialmente verifica se a lista de playlists está vazia, retornando null caso seja verdade. Caso contrário cria um objeto temporário do tipo playlist que armazena a referência da cabeça da lista de playlists e uma variável(procura) para representar a posição de cada nó playlist na lista, como se fosse a posição dos valores em um vetor, e com isso, roda toda essa lista com base nessa playlist temporária fazendo comparações se o valor de procura, que a cada ciclo é incrementado é igual ao valor passado por parâmetro , caso a verificação seja verdadeira retorna a playlist temporária que está naquele ciclo, caso não entre nessa verificação retorna null, se referindo a não ter achado nenhuma playlist com o nome passado por parâmetro.
-
- **Parâmetros**:
    - `Int numero:` O número da playlist a ser buscado na lista de playlist.
- **Retorno**: A referência do objeto da classe `Playlist` encontrada através do número da mesma. Caso não encontre ou a lista for vazia, retorna `null`.

4.2.14 `public boolean removerPlaylist(String nome)`

- **Descrição**: Esse método criará uma variável para armazenar a playlist(caso exista) retornada pelo método buscarPlaylist(String nome), se a variável receber null, significa que a playlist não foi encontrada e por isso não poderá ser removida, caso uma playlist seja retornada, haverá a verificação se ela se encontra na cabeça, se ela se encontrar na cabeça, há a verificação se o próximo dela é ela mesma, o que significaria que só existe essa playlist na lista e para ser removida, basta desreferencia-la, caso existam mais playlists na lista, a cabeça passa a ser a próxima playlist da antiga cabeça e, a última playlist passa a apontar para a nova cabeça, enquanto para remover no meio da lista ou no final, a playlist anterior terá como a

próxima playlist, a próxima playlist da playlist a ser removida e a próxima playlist da playlist a ser removida terá como anterior a playlist anterior da playlist a ser removida, após isso, a playlist a ser removida é referenciada para `null`.

- **Parâmetros**:
  - `String nome:` O nome da playlist a ser removida na lista de playlists.
- **Retorno**: True caso encontre a playlist e consiga removê-la ou False se não encontrar a playlist que o usuário deseja remover.

4.2.15 `public boolean removerMusica(String nome)`

- **Descrição**:: Esse método criará um objeto do tipo música(remMus) para armazenar a referência da cabeça da playlist ao qual faz parte, e com isso começa as verificações passando inicialmente pelo uma verificação(caso ela não esteja vazia) se a música armazenada no objeto temporário, usando o getNome() é igual ao nome passado por parâmetro, se sim verifica se só tem um único elemento e com isso seta a cabeça como null, removendo a música, caso tenha mais de um elemento segue na lógica de a música buscada ser o elemento da cabeça, logo remove-se o primeiro elemento, criando outro objeto do tipo música para armazenar a referência do último elemento e assim fazendo as novas ligações eliminando no final a cabeça, setando null por meio do setProximo e setAnterior, caso não seja o mesmo nome da cabeça, usa o método buscarMusica para achar a referência da música na lista a qual o nome passado por parâmetro é igual, faz mais algumas verificações se o retorno for diferente de null, corresponde a ter achado alguma música com aquele nome e verifica se ela se encontra pelo meio da lista e aplica a lógica de remoção quando está no meio de elementos na lista e se não estive pelo meio, só resta estar no final e assim usa a lógica de remoção no final em que o penúltimo elemento se liga com a cabeça e a cabeça se liga com o penúltimo elemento e finalizando com o nó musica a ser removido sendo setado para null, representado que ele foi removido, nos casos em que remMus tem seus atributos proximo e anterior apontados para null o método retorna true, caso contrário retorna false.
- **Parâmetros**:
  - `String nome:` O nome da música a ser removida na lista de playlist.
- **Retorno**: True se a música foi removida com sucesso ou false se não achou a música ou se não removeu corretamente

---

4.2.16 `public void imprimeFila(Musica musica)`

- **Descrição**: Esse método tem a função de mostrar as músicas a serem tocadas na playlist caso a música passada como parâmetro exista, se ela tiver ponteiros apontando para outras músicas, essas também serão mostradas no console e serão enumeradas pela ordem de precedência na hora de exibir.
- **Parâmetros**:
  - `Musica musica:` Objeto música para impressão de fila.

4.2.17 `public String toString()`

- **Descrição**: Constrói uma representação textual da lista de músicas que pertence ao objeto Playlist.
- **Retorno**: Uma `String` formatada com os nomes das músicas.

4.2.18 `public String toStringPlaylist()`

- **Descrição**: Constrói uma representação textual das playlists contidas na lista duplamente encadeada circular de playlists.
- **Retorno**: Uma `String` formatada com os nomes das playlists.

4.3. Classe Menu

4.3.1 `public int leituraTeclado(String texto)`

- **Descrição**: Mostra ao usuário um menu em forma de texto para que o usuário possa escolher um número que esteja numerado nas opções mostradas para executar determinada ação no programa.
- **Parâmetros**:
  - `String texto:` Mensagem que mostrará o menu para o usuário.
- **Retorno**: O número inteiro que representa a opção que o usuário quer executar.

4.3.2 `public String leituraTecladoChar(String texto)`

- **Descrição**: Mostra ao usuário uma mensagem final para que ele pressione qualquer letra para prosseguir no programa.
- **Parâmetros**:
  - `String texto:` Mensagem que mostrará a mensagem de finalização de uma ação para que o usuário prossiga no programa.
- **Retorno**: A string digitada pelo usuário para que passe para um próximo case, caso exista.

4.3.3 `public String leituraTecladoNome(String texto)`

- **Descrição**: Mostra ao usuário uma mensagem para que ele insira o nome da playlist a ser criada e armazena-o em uma variável.
- **Parâmetros**:

    - `String texto:` Mensagem que mostrará a mensagem para o usuário informar um nome para a playlist a ser criada.
- **Retorno**: Nome da playlist.

4.3.4 `public Musica menuAdicionarMusica()`

- **Descrição**: Cria variáveis para possibilitar a criação de uma música válida, então, pede ao usuário o nome do artista, nome da música, nome do álbum, nome do gênero e a duração da música.
- **Retorno**: O objeto musica criado pelo usuário.

4.3.5 `public Static void menuPlaylist()`

- **Descrição**: Mostra ao usuário todo o menu com todas as opções que podem ser executadas com a lista de playlists, seja criar uma playlist, visualizar o conteúdo dela, adicionar uma música, remover alguma playlist, remover uma música de uma determinada playlist, escutar a música ou sair do programa.

4.3.6 `public void consoleEspaco()`

- **Descrição**: Executa 25 quebras de linhas sucessivas para limpar o console.

4.3.7 `public void criarPlaylist(Playlist Playlists)`

- **Descrição**: Pede para o usuário inserir um nome para a playlist que deseja ser criada e então insere na lista de playlists uma nova playlist com o nome informado pelo usuário.
- **Parâmetros**:
    - `Playlist Playlists:` Toda a lista de playlists já existente.

4.3.8 `public void mostrarPlaylist(int op,Playlist Playlists)`

- **Descrição**: Busca a playlist escolhida pelo usuário para mostrar seu conteúdo, se essa playlist existir, todas as músicas dentro dela são exibidas para o usuário.
- **Parâmetros**:
    - Int op: A posição da playlist quando é exibida para o usuário.

- `Playlist Playlists:` Toda a lista de playlists já existente.

4.3.9 `public void mostrarPlaylists(Playlist Playlists)`

- **Descrição**: Mostra todas as playlists, de forma enumerada,que já existem para o usuário.
- **Parâmetros**:
  - `Playlist Playlists:` Toda a lista de playlists já existente.

4.3.10 `public Musica adicionarMusica()`

- **Descrição**: Utiliza de variáveis já criadas para possibilitar a criação de uma música válida, então, pede ao usuário nome da música, nome do cantor(a), nome do gênero, o álbum do qual a música faz parte e a duração da música.
- **Retorno**: O objeto musicaNova criado pelo usuário.

4.3.11 `public void imprimeTodasMusicasPlaylist(Playlist p)`

- **Descrição**: Mostrará ao usuário todas as músicas da playlist escolhida por ele.
- **Parâmetros**:
  - `Playlist p:` objeto playlist escolhida pelo usuário.

4.3.12 `public void imprimeTodasMusicas(ArrayList<Musica> todasMusicas)`

- **Descrição**: Mostrará ao usuário todas as músicas já existentes e armazenadas em um arquivo.txt.
- **Parâmetros**:
  - `ArrayList<Musica> todasMusicas:` Arraylist com todas as músicas lidas do arquivo salvo.

4.3.13 `public void tocarMusica(Playlist Playlists, Musica musica, int posPlaylist)`

- **Descrição**: Buscará a playlist com base na posição dela, com isso, exibirá a música que está tocando e um pequeno menu de interação para que o usuário consiga pausar, ir para próxima música, voltar uma música ou sair da playlist.
- **Parâmetros**:
    - `Playlist Playlists:` Lista com todas as playlists salvas.
    - Musica musica: Objeto musica para ser tocada, tendo uma próxima e uma anterior a ela, caso ela seja a única música da playlist, a anterior e a próxima também será ela, pois se trata de uma lista duplamente encadeada circular.
    - Int posPlaylist: Posição em que a playlist se encontra na lista de playlists.

4.4. Classe ArquivoTexto

4.4.1 `public ArrayList<Musica> carregarArquivo(String nomeArquivo) throws FileNotFoundException`

- **Descrição**: Esse método passa para o programa, todas as músicas já salvas em um arquivo, caso ele exista, através de uma varredura completa do arquivo, armazenando todas as músicas numa lista de músicas.
- **Parâmetros**:
    - `String nomeArquivo:` Nome do arquivo para que possa ser feito a leitura dos dados do arquivo.
- **Retorno**: Uma lista de músicas composta por todas as músicas já salvas no arquivo se o arquivo existir, senão, retorna null.

4.4.1 `public void salvarArquivo(ArrayList<Musica> musicas) throws IOException`

- **Descrição**: Esse método salva os novos dados em um arquivo, adicionando mais músicas e suas informações ao arquivo.
- **Parâmetros**:
    - `ArrayList<Musica> musicas:` Arraylist com todas as músicas já registradas.

4.5. Classe ArquivoObjeto

4.5.1 `public ArquivoObjeto(String nomeArquivo, Object o) throws IOException`

- **Descrição**: Esse construtor cria o arquivo a partir da primeira execução do programa e a partir de sua inicialização, salva também os dados inseridos.
- **Parâmetros**:
    - `String nomeArquivo:` É o nome do arquivo a ser criado.

- Object o: é um objeto genérico, passado para o construtor para que esse, possa salvar o arquivo na primeira execução.

4.5.2 `public void salvarArquivo(Object o)`

- **Descrição**: Este método tem a função de escrever no arquivo, tudo que for salvo pelo programa ao executar determinados comandos e solicitar este método.
- **Parâmetros**:
  - `Object o:` Objeto genérico para que possa ser feita a escrita dos objetos no arquivo.

4.5.2 `public Object carregarArquivo()`

- **Descrição**: Este método do tipo objeto tem a função de ler tudo que estiver no arquivo e passar para um objeto genérico.
- **Retorno**: retorna o objeto genérico que salvou tudo o que foi lido, ou retorna null, se ocorrerem todos os erros possíveis durante a leitura do arquivo.

4.6. Classe Main

4.6.1 `public static void main(String[] args)`

- **Descrição**: Este método é o método principal, ele que chamará os outros métodos das outras classes, para manipulação das listas e dos arquivos, utilizando dos métodos das classes já explicadas para fazer a comunicação entre o programa e o usuário.