Atividade Prática: Gerenciador de Playlist com Listas Encadeadas

1. Introdução

Na última aula, aprendemos os fundamentos das listas simplesmente encadeadas, implementando as operações essenciais de inserção, remoção e busca. Nesta atividade, daremos um passo adiante para aplicar esse conhecimento em um cenário prático e criativo: a construção de um gerenciador de playlists de música.

O foco agora não é apenas construir a lista, mas sim **manipular sua estrutura** de formas mais complexas, um desafio que testa o verdadeiro entendimento sobre o gerenciamento de nós e ponteiros (ou referências).

2. Objetivos de Aprendizagem

Ao final desta atividade, você será capaz de:

- Aplicar o conceito de listas encadeadas em um problema prático.
- Implementar algoritmos de manipulação de nós (travessia, remoção e inserção no meio).
- Gerenciar referências (ou ponteiros) para alterar a estrutura da lista de forma eficiente.
- Praticar a identificação e o tratamento de casos de borda (lista vazia, primeiro/último elemento).

3. O Cenário

Você receberá um código-fonte inicial (em Java) que contém a estrutura de uma `Playlist`. Nesta estrutura:

- Cada **nó** da lista representa uma `Musica`, contendo informações como título, artista e duração.
- A classe `Playlist` já possui os métodos básicos para `adicionarMusica`, `removerMusica` e `exibirPlaylist`.

Sua tarefa é se concentrar na implementação das funcionalidades avançadas descritas abaixo.

4. Instruções e Desafios

Abra o arquivo fonte fornecido e navegue até a seção **"DESAFIO: IMPLEMENTAR ESTES MÉTODOS"**. Você deverá implementar a lógica para as seguintes funções:

Desafio 1: Simulando um Player - `tocarProxima()`

Esta função simula o comportamento de um player de música.

- Na primeira vez que for chamada, deve exibir os dados da primeira música da playlist.
- A cada chamada subsequente, deve avançar e exibir a próxima música.
- Caso chegue ao final da playlist, ela deve "dar a volta" e, na próxima chamada, tocar a primeira música novamente (comportamento de "repeat").
- **Dica:** Utilize a variável `musicaAtual` presente na classe para guardar o estado de qual música está "tocando".

Desafio 2: Reorganizando a Playlist - `moverMusica(String titulo, int novaPosicao)`

Esta é a tarefa principal e mais desafiadora. A função deve encontrar uma música pelo `titulo` e movê-la para uma `novaPosicao` na lista (onde 1 é o início da playlist).

- **Exemplo:** Se a playlist é `A → B → C → D` e você chama `moverMusica("C", 2)`, a lista resultante deve ser `A → C → B → D`.
- Você deve manipular as referências (`proximo`) dos nós para "desconectar" a música de sua posição original e "reconectá-la" na nova posição.
- **Atenção aos casos de borda:**
  - O que acontece se a música a ser movida já for a primeira?
  - O que acontece ao mover uma música para a posição 1?
  - E ao mover para a última posição da lista?
  - E se a música não for encontrada ou a `novaPosicao` for inválida (ex: 0, negativa ou maior que o tamanho da lista)?

Desafio 3: O Desafio Final - `inverterPlaylist()`

Esta função deve inverter completamente a ordem dos elementos da playlist.

- O que era o último nó deve se tornar o `head`, e todas as referências `proximo` devem ser reajustadas.
- O grande desafio é fazer essa inversão **in-place**, ou seja, sem criar uma segunda lista ou nós adicionais. Você deve apenas reatribuir as referências dos nós já existentes.
- **Dica:** Normalmente, este algoritmo é resolvido com o auxílio de três referências/ponteiros: `anterior`, `atual` e `proximo`.