PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Trabalho Prático 3

Curso : *Sistemas de Informação*
Disciplina : *Algoritmos e Estruturas de Dados*
Professora : *Eveline Alonso Veloso*

Regras Básicas:

1. Desenvolva esse trabalho prático a partir do que já foi implementado no Trabalho prático 2.
2. Estude bastante cada par de entrada/saída.
3. Todos os programas deverão ser desenvolvidos na linguagem de programação Java.
4. Esse trabalho prático poderá ser desenvolvido em grupos de, no máximo, três integrantes.
5. Cópias de trabalho, se existirem, serão encaminhadas ao colegiado de coordenação didática do curso.
6. Fique atento ao *charset* dos arquivos de entrada e saída. Recomenda-se a utilização dos métodos da classe MyIO.java para leitura de dados do teclado.
7. Para cada exercício, vocês devem submeter apenas um arquivo (.java) por grupo. Essa regra será necessária para a submissão de trabalhos no VERDE e no identificador de plágios utilizado na disciplina.
8. A resolução (código) de cada exercício deverá ser submetida ao VERDE.
9. A correção será realizada automaticamente pelo VERDE e validada por meio de apresentações durante as aulas práticas da disciplina.

Base de Dados:

Spotify é um serviço de *streaming* de música, *podcast* e vídeo lançado oficialmente em 7 de outubro de 2008. É atualmente o serviço de *streaming* mais popular e usado do mundo. Desenvolvido pela *startup* Spotify AB (Estocolmo, Suécia), fornece conteúdo protegido de restrições relacionadas a direitos digitais de gravadoras e empresas de mídia. O Spotify é um serviço *freemium*; com recursos básicos gratuitos, com propagandas ou limitações, enquanto recursos adicionais, como qualidade de transmissão aprimorada e *downloads* de música, oferecidos para assinaturas pagas.

Spotify está disponível na maior parte da Europa, parte da América, Austrália, Nova Zelândia e parte da Ásia. Disponível para a maioria dos dispositivos modernos, incluindo computadores Windows, macOS e Linux, bem como *smartphones* e *tablets* com iOS, Windows Phone e Android. Pode-se encontrar as músicas desejadas por navegação ou pesquisas referentes a artista, álbum, gênero, lista de reprodução ou

gravadora. Seus usuários podem criar, editar ou compartilhar *playlists*, compartilhar faixas em redes sociais ou fazer *playlists* com outros usuários. Fornece acesso a mais de 30 milhões de músicas e, em julho de 2019, contava com mais de 232 milhões de usuários ativos, incluindo 108 milhões de assinantes pagantes.

O arquivo dataAEDs.csv contém um conjunto de dados relacionados a mais de 175.000 músicas coletadas da plataforma Spotify Web API, que podem ser agrupadas por artista, ano ou gênero. Tal arquivo deve ser copiado para a pasta /tmp.

Exercícios:

Estruturas de dados implementadas por meio de vetores:

1. **Pilha implementada por meio de vetor em Java**
   Crie uma pilha, implementada por meio de vetor, de objetos da classe *Musica*. Lembre-se que, na verdade, temos um vetor de referências para objetos do tipo *Musica*.
   Neste exercício, faremos inserções e remoções de itens na pilha e, após o processamento de todas as operações, mostraremos seus elementos.

   Os métodos de sua pilha devem operar conforme descrito a seguir, respeitando-se parâmetros e tipos de retorno:
   - Sua classe *Pilha* deverá ter dois construtores.
   - *void empilhar*(*Musica musica*): empilha um objeto do tipo *Musica*.
   - *Musica desempilhar* (): desempilha e retorna a *Musica* do topo da pilha.
   - *void mostrar*(): a partir do fundo da pilha, para todos os objetos do tipo *Musica* presentes na pilha, exibe a posição do objeto na pilha seguida dos valores de seus atributos (observe o formato de cada linha da saída esperada).

   A entrada padrão é dividida em duas partes. A primeira contém, em cada linha, uma *string* indicando o *id* da música que deve ser inicialmente inserida na pilha de músicas, na ordem em que são apresentadas. Após a palavra FIM, inicia-se a segunda parte da entrada padrão.
   A primeira linha dessa segunda parte da entrada padrão apresenta um número inteiro *n* indicando a quantidade de músicas que serão em seguida empilhadas ou desempilhadas. Nas próximas *n* linhas, tem-se *n* comandos de inserção ou remoção que devem ser processados neste exercício. Cada uma dessas linhas tem uma palavra de comando, conforme descrito a seguir:
   - I: empilhar;
   - R: desempilhar.

   No caso dos comandos de inserção, temos também uma *string* indicando o *id* da música que deve empilhada na pilha de músicas.

   A saída padrão apresenta uma linha para cada música desempilhada, sendo que essa informação será constituída pela *string* "(R)" seguida do atributo *nome* da

música retirada da pilha. Em seguida, teremos, ainda na saída padrão, os atributos relativos às músicas armazenadas na pilha após o processamento de todas as operações de inserção e remoção (observe o formato de cada linha da saída esperada).

2. **Fila circular implementada por meio de vetor em Java**

Crie uma fila circular, implementada por meio de vetor, de objetos da classe *Musica*. Essa fila deve conseguir armazenar simultaneamente até cinco músicas. Neste exercício, faremos inserções e remoções de itens na fila.

Os métodos de sua fila circular devem operar conforme descrito a seguir, respeitando-se parâmetros e tipos de retorno:

- Sua classe *Fila* deverá ter dois construtores.
- *void enfileirar* (*Musica musica*): enfileira um objeto do tipo *Musica*.
- *Musica desenfileirar* (): desenfileira e retorna a *Musica* da frente da fila.
- *void mostrar*(): para todos os objetos do tipo *Musica* presentes na fila, exibe a posição do objeto na fila seguida dos valores de seus atributos (observe o formato de cada linha da saída esperada).
- *double obterMediaDuration* (): calcula e retorna a média das durações das músicas presentes na fila.

A entrada padrão será como a da questão **Pilha implementada por meio de vetor em Java**, contudo, o comando I será utilizado para inserir na fila (enfileirar); e R, para remover da fila (desenfileirar).
Observe que, quando, no momento de execução da operação enfileirar, a fila estiver cheia, antes de enfileirar uma música será necessário desenfileirar outra.

A saída padrão será um número **inteiro** corresponde à média **arredondada** das durações das músicas contidas na fila, após cada inserção.
Além disso, a saída padrão também apresenta uma linha para cada música desenfileirada, sendo que essa informação será constituída pela *string* "(R)" seguida do atributo *nome* dessa música. Em seguida, teremos, ainda na saída padrão, os atributos relativos às músicas armazenadas na fila após o processamento de todas as operações de inserção e remoção (observe o formato de cada linha da saída esperada).

3. **Lista implementada por meio de vetor em Java**

Crie uma lista, implementada por meio de vetor, de objetos da classe *Musica*.
Neste exercício, faremos inserções e remoções de itens na lista e, após o processamento de todas as operações, mostraremos seus elementos.

Os métodos de sua lista devem operar conforme descrito a seguir, respeitando-se parâmetros e tipos de retorno:

- Sua classe *Lista* deverá ter dois construtores.
- *void inserirInicio* (*Musica musica*): insere um objeto do tipo *Musica* na primeira posição da lista, necessitando remanejar todos os demais.
- *void inserir* (*Musica musica*, *int posicao*): insere uma música na posição da lista indicada pelo parâmetro *posicao*, desse método; onde 0 <=

posicao <= *n*, sendo *n* o número de músicas já inseridas na estrutura. Esse método também remaneja os demais objetos da lista.
- *void inserirFim* (*Musica musica*): insere um objeto da classe *Musica* na última posição da lista.
- *Musica removerInicio* (): remove e retorna a primeira música da lista, remanejando as demais.
- *Musica remover*(*int posicao*): remove e retorna o objeto *Musica* armazenado na posição da lista indicada pelo parâmetro *posicao*, desse método; necessitando remanejar os demais.
- *Musica removerFim*(): remove e retorna a última *Musica* da lista.
- *void mostrar*(): para todos os objetos do tipo *Musica* presentes na lista, exibe a posição do objeto na lista seguida dos valores de seus atributos (observe o formato de cada linha da saída esperada).

A entrada padrão é dividida em duas partes, conforme a entrada padrão do exercício **Pilha implementada por meio de vetor em Java**. A primeira contém, em cada linha, uma *string* indicando o *id* da música que deve ser inicialmente inserida na lista de músicas, na ordem em que são apresentadas. Após a palavra FIM, inicia-se a segunda parte da entrada padrão.
A primeira linha dessa segunda parte da entrada padrão apresenta um número inteiro *n* indicando a quantidade de músicas que serão em seguida inseridas ou removidas da lista. Nas próximas *n* linhas, tem-se *n* comandos de inserção ou remoção que devem ser processados neste exercício. Cada uma dessas linhas tem uma palavra de comando, conforme descrito a seguir:
- II: inserir no início;
- I* inserir em uma determinada posição;
- IF: inserir no final;
- RI: remover do início;
- R*: remover de uma determinada posição; e
- RF: remover do final.

No caso dos comandos de inserção, temos também uma *string* indicando o *id* da música que deve inserida na lista de músicas.
No caso dos comandos de inserção e remoção "em uma determinada posição", temos também um inteiro indicando essa posição. No comando de inserção, a posição fica imediatamente após a palavra de comando. Lembre-se que o primeiro item da lista encontra-se na posição 0.

A saída padrão deve ser como a da questão **Pilha implementada por meio de vetor em Java**.

Estruturas de dados implementadas por meio de células auto-referenciadas:

4. **Pilha com alocação dinâmica de memória em Java**
   Refaça o exercício **Pilha implementada por meio de vetor em Java** usando alocação dinâmica de memória.
   Neste exercício, sua classe *Pilha* deverá ter apenas um construtor.

5. **Fila com alocação dinâmica de memória em Java**
   Refaça o exercício **Fila circular implementada por meio de vetor em Java** usando alocação dinâmica de memória.
   Neste exercício, sua classe *Fila* deverá ter apenas um construtor.
   Lembre-se que essa fila deve conseguir armazenar simultaneamente, no máximo, cinco músicas.

6. **Lista encadeada em Java**
   Refaça o exercício **Lista implementada por meio de vetor em Java** usando lista encadeada.
   Neste exercício, sua classe *Lista* deverá ter apenas um construtor.

7. ***Quicksort* com lista duplamente encadeada em Java**
   Refaça o exercício ***Quicksort*** do **Trabalho prático 2** usando lista duplamente encadeada.
   O nome do arquivo de *log* dessa questão será matrícula_quicksort2.txt.