Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

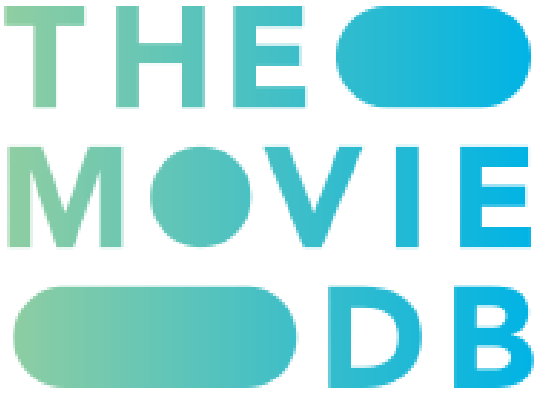
Trabalho Prático II

Regras Básicas

1. extends Trabalho Prático 01

2. Fique atento ao Charset dos arquivos de entrada e saída.

Classe + Registro



The Movie Database (O Banco de Dados de Filmes), mais conhecido pela sigla TMDb, é uma base de dados grátis e de código aberto sobre filmes e seriados (streaming, televisivas, cinematográficas), criado por Travis Bell em 2008. Atualizado constantemente através do apoio da comunidade. Era inicialmente apenas uma base de dados sobre filmes, mas em 2013 foi adicionado a seção de séries. O TMDb com o código aberto, significa que muitos sites de cinema e TV podem usar os seus dados, incluindo utilizadores individuais de Software de gerenciamento de mídia, como Plex, Kodi e outros.

Neste Trabalho Prático sua tarefa é organizar as informações dos filmes disponíveis para exibição ao usuário. Entretanto, esses dados estão espalhados em vários arquivos no formato *html*, os quais foram obtidos através de consultas à base de dados TMDb. Todos esses arquivos estão agrupados no arquivo filmes.zip, e o mesmo deve ser descompactado na pasta /tmp/filmes/. [1] Para isso, você deve ler, organizar e armazenar os dados de cada filme em memória, utilizando as estruturas de dados em aula (Lista, Pilhas e Filas). Em seguida executar as operações descritas nos arquivos de entrada. Muito cuidado ao realizar o *parser* do texto. Fique atento a descrição dos dados que serão lidos e manipulados pelo seu sistema.

1. **Classe Filme em Java**: Crie uma classe *Filme* seguindo todas as regras apresentadas no slide unidade01g_conceitosBasicos_introducaoOO.pdf. Sua classe terá os atributos privados Nome

[1] Quando reiniciamos o Linux, ele normalmente apaga os arquivos existentes na pasta /tmp/.

(String), Título Original (String), Data de Lançamento (Date), Duração (int), Genero (String), Idioma original (String), Situação (String), Orçamento (Float) [2], Palavras-Chave (Vetor de Strings). Ela terá também pelo menos dois construtores, e os métodos gets, sets, clone e imprimir e ler. O método imprimir mostra a String ‘‘`nome tituloOriginal dataLancamento duracao genero idiomaOriginal situacao orcamento [palavrasChave]`’’, contendo todos os atributos da classe. O método ler deve efetuar a leitura dos atributos de um registro. Veja que os dados estão divididos em vários arquivos.

A entrada padrão é composta por várias linhas e cada uma contém o nome de um arquivo *.html*. A última linha da entrada contém FIM. A saída padrão também contém várias linhas, uma para cada registro contido em uma linha da entrada padrão.

2. **Registro em C**: Repita a anterior criando o registro *Filme* na linguagem C.

Pesquisa

3. **Pesquisa Sequencial em Java**: Faça a inserção de alguns objetos no final de uma Lista e, em seguida, faça algumas pesquisas sequenciais. A chave primária de pesquisa será o atributo **nome**. A entrada padrão é composta por duas partes onde a primeira é igual a entrada da primeira questão 1. As demais linhas correspondem a segunda parte. A segunda parte é composta por várias linhas. Cada uma possui um elemento que deve ser pesquisado na Lista. A última linha terá a palavra FIM. A saída padrão será composta por várias linhas contendo as palavras SIM/NÃO para indicar se existe cada um dos elementos pesquisados. Além disso, crie um arquivo de log na pasta corrente com o nome matrícula_sequencial.txt com uma única linha contendo sua matrícula, tempo de execução do seu algoritmo e número de comparações. Todas as informações do arquivo de log devem ser separadas por uma tabulação ’\t’.

4. **Pesquisa Binária em Java**: Repita a questão anterior, contudo, usando a Pesquisa Binária. A entrada e a saída padrão serão iguais às da questão anterior. O nome do arquivo de log será matrícula_binaria.txt.

Estrutras Sequenciais

5. **Lista com Alocação Sequencial em Java**: Crie uma Lista de Filmes baseada na lista de inteiros vista na sala de aula. Sua lista deve conter todos os atributos e métodos existentes na lista de inteiros, contudo, adaptados para a classe *Filme*. De toda forma, lembre-se que, na verdade, temos uma lista de ponteiros e cada um deles aponta para um objeto *Filme*. Neste exercício, faremos inserções, remoções e mostraremos os elementos de nossa lista.

[2] Em caso de orçamento inexistente, favor colocar o valor 0,00.

Os métodos de inserir e remover devem operar conforme descrito a seguir, respeitando parâmetros e retornos. Primeiro, o *void inserirInicio(Filme filme)* insere um objeto na primeira posição da Lista e remaneja os demais. Segundo, o *void inserir(Filme filme), int posição)* insere um objeto na posição p da Lista, onde $p < n$ e n é o número de objetos cadastrados. Em seguida, esse método remaneja os demais objetos. O *void inserirFim(Filme filme))* insere um objeto na última posição da Lista. O *Filme removerInicio()* remove e retorna o primeiro objeto cadastrado na Lista e remaneja os demais. O *Filme remover(int posição)* remove e retorna o objeto cadastrado na p-ésima posição da Lista e remaneja os demais. O *Filme removerFim()* remove e retorna o último objeto cadastrado na Lista.

A entrada padrão é composta por duas partes. A primeira é igual a entrada da primeira questão. As demais linhas correspondem a segunda parte. A primeira linha da segunda parte tem um número inteiro n indicando a quantidade de objetos a serem inseridos/removidos. Nas próximas n linhas, tem-se n comandos de inserção/remoção a serem processados neste exercício. Cada uma dessas linhas tem uma palavra de comando: II inserir no início, I* inserir em qualquer posição, IF inserir no fim, RI remover no início, R* remover em qualquer posição e RF remover no fim. No caso dos comandos de inserir, temos também o nome do arquivo *html* com os dados do registro a ser inserido. No caso dos comandos de "em qualquer posição", temos também a posição. No Inserir, a posição fica imediatamente após a palavra de comando. A saída padrão tem uma linha para cada objeto removido sendo que essa informação será constituída pela palavra "(R)" e o atributo **nome**. No final, a saída mostra os atributos relativos a cada objeto cadastrado na lista após as operações de inserção e remoção.

6. **Pilha com Alocação Sequencial em Java**: Crie uma Pilha de Filmes baseada na pilha de inteiros vista na sala de aula. Neste exercício, faremos inserções, remoções e mostraremos os elementos de nossa pilha. A entrada e a saída padrão serão como as da questão anterior, contudo, teremos apenas os comandos I para inserir na pilha (empilhar) e R para remover (desempilhar).

7. **Fila Circular com Alocação Sequencial em Java**: Crie uma classe *Fila Circular* de *Filme*. Essa fila deve ter tamanho cinco. Em seguida, faça um programa que leia vários registros e insira seus atributos na fila. Quando o programa tiver que inserir um objeto e a fila estiver cheia, antes, ele deve fazer uma remoção. A entrada padrão será igual à da questão anterior. A saída padrão será um número inteiro para cada registro inserido na fila. Esse número corresponde à média **arredondada** do atributo `duracao` dos registros contidos na fila após cada inserção. O final da saída padrão mostra os registros existentes na fila seguindo o padrão da questão anterior.

8. **Lista com Alocação Sequencial em C**: Refaça a questão 5 na linguagem C.