PUC Minas

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Lista 07

Professor: *Marta Noronha*

Disciplina: *Laboratório de Algoritmos e Estrutura de Dados II*
Data de entrega: *23/04/2024*

Requisitos

1. Todos os programas deverão ser desenvolvidos na linguagem de programação Java.
2. Essas práticas poderão ser desenvolvidas em grupos de, no máximo, dois integrantes.
3. Cópias, se existirem, serão encaminhadas ao colegiado de coordenação didática do curso.
4. 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. É necessário definir o charset a ser utilizado antes de começar a leitura de dados do teclado, da seguinte forma: `MyIO.setCharset("UTF-8")`.
5. As saídas esperadas, cadastradas no VERDE pelo professor, foram geradas empregando-se: `System.out.println()`.
6. Em cada submissão, enviar apenas um arquivo (.java). A regra será necessária para a submissão de exercícios no VERDE e no identificador de plágios utilizado na disciplina.
7. A resolução (código) de cada exercício deverá ser submetida ao VERDE.
8. A execução do código submetido será realizada automaticamente pelo VERDE, mas o código será analisado e validado pelo professor.
9. Se for necessário ler o arquivo *"pub.in"*, o mesmo será disponibilizado para *download* no VERDE juntamente com o arquivo *"pub.out"*.

Criação da classe Jogos

O conjunto de dados utilizado neste exercício foi disponibilizado em Kaggle.
Crie uma classe Jogo com os seguintes atributos declarados com modo de acesso privado:

- `rank` (int): classificação (posição) do jogo em relação aos demais.
- `nome do jogo` (String): o nome do jogo.
- `plataforma` (String): Onde o jogo pode ser executado (Wii, DS, entre outros).
- `ano` (int): ano de lançamento.
- `gênero` (String): esportes, corrida, entre outros.
- `editora` (String): empresa que disponibiliza o jogo (produtora).
- `NA_Vendas` (double): Vendas do jogo na América do Norte.

- `EU_Vendas` (double): Vendas do jogo na Europa.
- `JP_Vendas` (double): Vendas do jogo no Japão.
- `Outras_Vendas` (double): Vendas do jogo em outros lugares.
- `Vendas_Global` (double): Vendas global do jogo.

A classe Jogo também deve conter, obrigatoriamente, ao menos, dois construtores (*1. padrão (default); 2. nome_do_jogo/plataforma/ano*), e os métodos *getters* e *setters* de cada atributo, além dos métodos `clone()`, `ler()`, `imprimir()`, `toString()` e `MaisVendido()`.

O método `clone()` deve retornar um objeto da mesma classe e contendo os mesmos valores de atributos do atual objeto analisado (**vide exemplo do método clone no slide "Unidade 0 - Nivelamento - Ponteiros (C/C++) e Referências (Java)**" postado na disciplina teórica). **Não é permitido usar a interface Cloneable para esta finalidade.**

O método `ler()` deve receber cada linha do arquivo como parâmetro e armazenar os valores contidos em cada linha nos atributos de cada objeto que foi instanciado.

O método `imprimir()` exibe os valores dos atributos do objeto Jogo, conforme o modelo indicado no fim deste documento, conforme mostrado AQUI.

O método `toString()` deve ser criado para permitir a impressão dos atributos da classe, sem necessidade de invocação do método `imprimir()`.

O método `MaisVendido()` deve ser implementado para informar em que região ou país o jogo foi mais vendido (NA_vendas, EU_Vendas, JP_Vendas ou Outras_Vendas). Caso mais de uma região ou país tenha tido o mesmo número de vendas, informar somente o primeiro encontrado com a maior venda. A *String* informando o nome da região ou país deve ser retornada.

> Os métodos que não obedecem a forma como devem ser implementados, conforme descrição acima, serão penalizados ainda que o envio no Verde contabilize 100% de acerto. Isso será feito porque o desenvolvedor deve seguir atentamente as regras de implementação de um método, quando houver, considerando os parâmetros e o tipo de retorno que um método deve possuir quando estes são solicitados. Em caso de dúvidas, consulte a Professora.

Após a criação da classe, deve ser criado um mecanismo para processamento de uma entrada de dados. A entrada de dados é dividida em 2 partes:

- Parte 1: Armazenamento de informações em vetor;
- Parte 2: Pesquisa de informações armazenadas no vetor criado na parte 1.
- Parte 3: Enfileiramentos e desenfileiramentos na fila CIRCULAR criada na Parte 2.

Parte 1 (Leitura de Arquivo): Armazenamento de informações contidos no arquivo jogos.txt em vetor

Seu programa deve ler um arquivo-texto chamado **"jogos.txt"** que, no VERDE, localiza-se na pasta **"/tmp"**.

O aluno(a) deve preencher um vetor de objetos da classe Jogo com os dados dos diversos jogos informados no arquivo "jogos.txt". Considere que o vetor de jogos pode ser preenchido com até 1500 jogos, ainda que algumas posições fiquem vazias após a inserção dos jogos neste vetor.

> **NÃO DEVE SER USADO O ARRAYLIST PARA ARMAZENAR OS OBJETOS DESTA TAREFA. O USO DO ARRAYLIST PARA RESOLUÇÃO DESTA LISTA IMPLICA NO RECEBIMENTO DA NOTA ZERO.**

Cada uma das linhas seguintes apresenta os dados de um jogo, separados pelo símbolo "|" (Use " \\|" para fazer o split do vetor). Os dados possuem, em ordem, as seguintes informações:

- `rank` (int);
- `nome do jogo` (String);
- `plataforma` (String);
- `ano` (int);
- `gênero` (String);
- `editora` (String);
- `NA_Vendas` (double);
- `EU_Vendas` (double);
- `JP_Vendas` (double);
- `Outras_Vendas` (double);
- `Vendas_Global` (double);

A última linha do arquivo é vazia.

Parte 2: Pesquisa de informações armazenadas no vetor criado na Parte 1.

Após o processamento da primeira parte da entrada de dados, o programa deve processar a entrada contida no arquivo *pub.in*. Cada linha da segunda parte contém, em ordem, as seguintes informações:

- nome do jogo;
- ano;
- editora;
- plataforma.

A última linha do arquivo *pub.in*, usada para a parte 2 desta lista, contém a palavra FIM.

As entradas devem ser pesquisadas no vetor de jogos, criado na parte 1. Os registros encontrados na pesquisa devem ser armazenados em uma **FILA CIRCULAR, capaz de armazenar até 60 jogos simultaneamente**, que será utilizado na Parte 3 desta prática.

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 (Jogo jogo)`: enfileira um objeto do tipo Jogo;
- `Jogo desenfileirar ()`: desenfileira e retorna o jogo da frente da fila;
- `void mostrar ()`: para todos os objetos do tipo Jogo 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);
- `obterSomaVendasGlobal()`: deve atualizar o total de vendas global a cada jogo incluído ou removido da fila circular. O resultado deve ser arredondado para ser retornado como um inteiro.

A execução desta parte do trabalho deve ser finalizada após a leitura da palavra FIM.

O tamanho da fila deve ser atualizado a cada operação de inserção ou remoção.

A cada operação de inserção na fila, deve ser impresso a soma atualizada das vendas globais.

Se, no momento de execução da operação enfileirar, a fila estiver cheia, antes de enfileirar um jogo será necessário desenfileirar outro.

O programa deve contabilizar a quantidade de jogos lidos na entrada padrão, neste caso *pub.in*, que **foram** encontrados no arquivo *jogos.txt*.

Ao final da segunda parte, após ler os registros que foram pesquisados no *pub.in*, deverão ser impressas as informações:

Quantidade de jogos encontrados: {total de jogos}
Tamanho da fila: {tamanho da fila}
(Obs: as chaves não devem ser impressas)

Parte 3: Enfileiramentos e desenfileiramentos na fila criada na Parte 2.

A primeira linha da segunda parte da entrada padrão (*pub.in*) apresenta um número inteiro *n* indicando a quantidade de jogos que serão enfileirados ou desenfileirados. Nas próximas *n* linhas, tem-se *n* comandos de enfileiramento ou desenfileiramento, que devem ser processados. Cada uma dessas linhas tem uma palavra de comando, conforme descrito a seguir:

- E: enfileirar;
- D: desenfileirar.

No caso dos comandos de enfileiramento, temos também uma string contendo as seguintes informações, que devem ser pesquisadas no vetor de jogos da Parte 1:

- nome do jogo;
- ano;
- editora;
- plataforma.

Após encontrar o registro no vetor de jogos, adicioná-lo à fila. **Se, no momento de execução da operação enfileirar, a fila estiver cheia, antes de enfileirar um jogo será necessário desenfileirar outro.**

A saída padrão será composta por:

- um número inteiro correspondente ao total de vendas global de todos os jogos contidos na fila, após cada enfileiramento.
- Uma linha para cada jogo desenfileirado, sendo que essa informação será constituída pela string "(D)" seguida dos atributos desse jogo.
- Ao final da execução, devem ser impressas as informações dos jogos armazenados na fila (observe o formato de cada linha da saída esperada).

Para o caso do desenfileiramento deve ser impresso:
(D) [*Nome do jogo*] [*Plataforma*] [*Vendas global*] {*Rank*}. {*Gênero*}. {*Editora*}. Mais vendido:{ *País ou região mais vendido*}.

Exemplo da saída após desenfileirar:

(D) [*Pro Yaky? Spirits 5*] [*DS*] [*0.08*] *11614. Adventure. Square Enix. Mais vendido: JP_Vendas.*

Para o caso da impressão dos jogos que estão na fila, deve ser impresso:
[posição na fila] [*Nome do jogo*] [*Plataforma*] [*Vendas global*] {*Rank*}. {*Gênero*}. {*Editora*}. Mais vendido:{ *País ou região mais vendido*}.

Exemplo da saída:

[*4*][*Skate 3*] [*Wii*] [*28.62*] *9. Platform. Nintendo. Mais vendido: NA_Vendas.*

Exercício 02 - Fila implementada com alocação dinâmica

Refaça o exercício anterior usando alocação dinâmica de memória.

Neste exercício, sua classe Fila deverá ter apenas um construtor e não deve conter um limite máximo de registros.