PUC Minas

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Lista 03

Professora: *Marta Noronha*

Disciplina: *Laboratório de Algoritmos e Estrutura de Dados II*
Data de entrega: *22/03/2024*

Requisitos

1. Todos os programas deverão ser desenvolvidos na linguagem de programação Java.
2. Cópias, se existirem, serão encaminhadas ao colegiado de coordenação didática do curso.
3. 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("ISO-8859-1").`
4. As saídas esperadas, cadastradas no VERDEpelo professor, foram geradas empregando-se: `System.out.println().`
5. Os arquivos *pub.in* e *pub.out* estarão disponíveis no VERDEse forem necessários para a execução dessa atividade.
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.

Criação da classe Jogos

O conjunto de dados utilizado neste exercício foi disponibilizado em Kaggle.
Implemente 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 também deve conter, obrigatoriamente, ao menos, dois construtores (*1. padrão (default); 2. (nome do jogo)/plataforma/ano*), e os métodos *gets*, *sets*, métodos `clone()`, `ler()`, `imprimir()` e `toString()`.

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***. Cada linha possui os dados de atributos de cada objeto do tipo Jogo. O método deve 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 da classe, sem necessidade de invocação do método `imprimir()`.

> Em Java, existe a classe **Object**, que define os comportamentos mínimos que todos os objetos possuem. Um desses métodos é o **toString()**, cujo propósito é fornecer uma representação textual do objeto. Por padrão, a implementação de toString() na classe Object **retorna uma string** contendo o nome da classe seguido de um código hash, mas é comum que os desenvolvedores sobrescrevam esse método para fornecer uma representação mais significativa, geralmente exibindo os valores dos atributos do objeto.
>
> Todas as classes em Java, inclusive a que contém o método main, herdam da classe Object. Dessa forma, ao sobrescrever o método toString(), o desenvolvedor redefine a forma como o objeto é convertido em string, permitindo uma representação customizada e mais informativa.

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 1: Armazenamento de informações em vetores: um aleatório e outro ordenado

A classe ArrayList NÃO poderá ser usada para esta atividade.

O aluno(a) deve preencher dois vetores de objetos da classe Jogo com os dados dos diversos jogos informados na entrada padrão (teclado - *pub.in*). A primeira linha contém a quantidade de jogos que devem ser armazenados em cada vetor de jogos, ou seja, os vetores possuem o mesmo tamanho.

Cada uma das linhas seguintes apresenta os dados de um jogo, separados pelo símbolo "|". 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);

Importante: O caractere "|" é utilizado para expressões regulares. Para dividir uma palavra por "|", utilize a seguinte sequência: "\\|".

Na primeira parte do *pub.in*, a qual é encerrada com a palavra FIM, estão os dados dos jogos que devem ser armazenados no primeiro vetor. Este vetor de jogos não deve estar ordenado.

Na segunda parte do *pub.in*, a qual é encerrada com a palavra FIM, estão os dados dos jogos que devem ser armazenados no segundo vetor. Este vetor de jogos está ordenado pelo nome do jogo.

Parte 2: Pesquisa de informações armazenadas nos vetores por pesquisa sequencial e pesquisa binária.

Antes de realizar as pesquisas nos vetores:

- Implemente no programa principal (*main*) um procedimento para a **pesquisa sequencial** que receba o **vetor de jogos não ordenados** e a **string** contendo o nome do jogo a ser buscado. Caso existam mais de um jogo com o mesmo nome (por exemplo, um jogo que foi atualizado), somente deve ser impressa a informação do primeiro jogo encontrado, conforme o exemplo apresentado AQUI.
- Implemente no programa principal (*main*) um procedimento para a **pesquisa binária** que receba o **vetor de jogos ordenados** e a **string** contendo o nome do jogo a ser buscado. Caso existam mais de um jogo com o mesmo título (por exemplo, um jogo que foi atualizado), somente deve ser impressa a informação do primeiro jogo encontrado, conforme o exemplo apresentado AQUI.
- Ambos procedimentos devem contabilizar o total de comparações realizados. *Obs: Modifique o algoritmo de pesquisa binária, visto em sala, para que a comparação entre o jogo no vetor e o jogo buscado via string seja realizada somente uma única vez.*

Após o armazenamento dos dados nos dois vetores, o programa deve processar a terceira parte da entrada padrão (a qual também termina com a palavra FIM). Cada linha da terceira parte contém somente o nome do jogo que deverá ser buscado em ambos os vetores de jogos.

Para cada linha lida, use a pesquisa sequencial para contabilizar a quantidade de comparações, imprimindo a saída padrão. Em seguida, pesquise o jogo contido nessa linha pela pesquisa binária. Novamente contabilize as comparações e imprima a saída padrão solicitada.

A saída padrão deve obedecer o seguinte formato (*não incluir as chaves*): *{nome do jogo}. {editora}. {ano}. Vendas global: {Vendas_Global}. Sequencial: {valor}, binaria: {valor}*

Exemplo:

Namco Bandai Games. Kelly Slater's Pro Surfer. 2007. Vendas global: 0.19. Sequencial: 657, binaria: 9

Sabendo que a pesquisa sequencial é $\Theta(n)$ e a pesquisa binária é $\Theta(log(n))$, compare o total de operações em cada pesquisa com a complexidade vista na aula prática.