Lista Prática Sobre: Maven, Versionamento com Git, Arquivos e Bibliotecas Externas

Orientações:
- Data de Entrega 10/12/2021 via Tarefa do Microsoft Teams
- Entregar apenas o link de um repositório git (GitHub/GitLab)

Objetivos da Lista:
- Aprimorar conhecimentos de versionamento com git
- Criar um repositório no GitHub/GitLab
- Familiarizar com o processo de automatização da *build* e gerência de dependências com Maven
- Familiarizar com recursos de *streams* do Java
- Conhecer as bibliotecas: OpenCSV e Lombok

Questão 1 Arquivos CSV (Comma-separated values) são arquivos muito utilizados para facilitar o envio, recebimento e armazenamento de um conjunto de dados. Nesses arquivos os dados ficam separados por "," (por isso o nome). Várias bibliotecas que fazem análise estatística, usam arquivos CSV. Os arquivos CSV, normalmente, possuem a primeira linha como um cabeçalho dos dados armazenados.

Considere o exemplo de um CSV apresentado na Figura 1. É um arquivo com informações de alunos. A primeira linha é o cabeçalho com quais dados estão armazenados.

```
1 Nome,Idade,Curso
2 Capiroto,22,Eng Computacao
3 Capirota,23,Eng Software
4 Apolonia,18, Eng Producao
5 Solaire,26, Eng Biomedica
6 Astora,35, Eng Telecom
```

Figura 1:

Para converter esse arquivo em um objeto Java, podemos usar a biblioteca OpenCSV, que faz o mapeamento. Para isso precisamos criar uma classe modelo com os campos que desejamos fazer a leitura. A classe na Figura 2 é capaz de mapear o CSV mostrado na Figura 1.

Observe que existem membros da classe com o mesmo nome do cabeçalho do arquivo CSV. É necessário também criarmos *getters* e *setters*, mas a biblioteca Lombok cuida disso. Veja a anotação `@Data` na linha 5. Essa biblioteca precisa ser colocada como dependência para o Maven.

Para fazer a conversão usando o OpenCSV podemos usar o exemplo apresentado na Figura 3.

Para usar os recursos da biblioteca OpenCSV é necessário colocar como dependência para o Maven. Após a execução da linha 12 uma `List` será criada com objetos do tipo `Pessoa` com cada linha do CSV mapeado para um objeto. Com isso podemos manipular esses objetos e executar operações.

```java
package br.inatel.cdg.model;

import lombok.Data;

@Data
public class Pessoa {

    private String nome;
    private int idade;
    private String curso;

    public Pessoa(String nome, int idade, String curso) {
        this.nome = nome;
        this.idade = idade;
        this.curso = curso;
    }
}
```

Figura 2: Classe modelo mapear um CSV

```java
List<Pessoa> listaPessoas = new ArrayList<>();

//csvFilePath e o caminho onde se encontra o arquivo
//nesse exemplo e um instancia de Path do pacote java.nio
try {
Reader reader = Files.newBufferedReader(csvFilePath);
CsvToBean<Pessoa> csvToBean = new CsvToBeanBuilder(reader)
        .withType(Pessoa.class)
        .withIgnoreLeadingWhiteSpace(true)
        .build();

listaPessoas = csvToBean.parse();
} catch (IOException e) {
e.printStackTrace();
}
return listaPessoas;
```

Figura 3: Exemplo de Conversão de um CSV para um Java Pojo (Plain old Java Objects).

Tarefa:
Você deverá criar um programa em Java capaz de ler um arquivo CSV (vendas−games.csv, incluído na tarefa do teams) com informações de vendas de jogos digitais. E criar as seguintes funcionalidades:

- Filtrar e criar um *ArrayList* (ou coleção equivalente) buscando jogos de plataformas específicas. Exemplo: Filtrar no CSV jogos para PS4, ou para PC.
- Filtrar e cirar um *ArrayList* (ou coleção equivalente) buscando jogos de *publishers* específicas. Exemplo: Filtrar no CSV jogos publicados pela Activision, ou Sony.

Dicas:

- Pesquisar sobre as bibliotecas Lombok e OpenCSV
- Estudar a função *filter()* do Java
- Verificar como carregar arquivos da pasta *resources* presente no Maven (facilita empacotar arquivos de configuração)
- Criar enums para os tipos de Plataformas e Publishers