Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Engenharia de *Software* – *Campus* Lourdes
Algoritmos e Estruturas de Dados II (Laboratório)
Profs. Eveline Alonso Veloso e **João Caram – Semestre 1/2025**
Oficina: Revisão e Nivelamento (Dados em Arquivos-texto)

Este documento é um guia da atividade desenvolvida em conjunto no laboratório de AEDs II. Nesta atividade utilizamos as classes previamente desenvolvidas, na atividade prática anterior, para carregar e armazenar dados de forma simples em um arquivo-texto semiestruturado. A atividade é individual e seu conteúdo servirá como base para atividades avaliativas futuras. Portanto, atenção ao que será desenvolvido em aula e procure fazer sua parte na tarefa, enviando as atualizações para o repositório da atividade no GitHub.

Tema: Armazenando dados de Produtos

Já desenvolvemos as classes básicas para que nosso cliente possa registrar vendas de produtos perecíveis e não perecíveis. Porém, ainda está nos faltando o desenvolvimento do sistema principal. Além disso, nos falta também uma maneira simples de ***persistir*** *os dados dos produtos, ou seja, fazer que eles sejam armazenados antes da finalização do sistema e carregados no momento de sua inicialização. Foi definido que os arquivos de dados para esse* software *terão a seguinte estrutura:*
N
tipo; descrição;preçoDeCusto;margemDeLucro;[dataDeValidade]

A primeira linha contém um número inteiro N > 0, indicando quantos produtos vêm a seguir. Na sequência, uma linha com a informação de cada produto. O tipo é "1" para produtos não perecíveis e "2" para perecíveis. A informação referente à data de validade só existe para produtos perecíveis e estará no formato dd/mm/aaaa.

Preparação:
Na classe abstrata **Produto**, sobrescreva o método ***equals*** que verifica se dois produtos são iguais por meio de suas descrições. Considere que a comparação das descrições dos produtos não é sensível ao caso.

Tarefa 2 (em aula):
Criar métodos nas classes já existentes que permitam instanciar objetos a partir das linhas de dados especificadas, bem como gerar linhas de dados a partir de um objeto.

Para realizar esta tarefa, devem ser acrescentados à classe abstrata **Produto** mais dois métodos, além dos já existentes, desenvolvidos na prática anterior.

O primeiro método consiste apenas na assinatura do método abstrato abaixo:

```
/**
* Gera uma linha de texto a partir dos dados do produto.
* @return Uma string no formato "tipo;descrição;preçoDeCusto;margemDeLucro;[dataDeValidade]"
*/
public abstract String gerarDadosTexto();
```

Já o segundo método que deve ser acrescentado à classe abstrata **Produto** deverá ter a seguinte assinatura:

```
/**
* Cria um produto a partir de uma linha de dados em formato texto.
* A linha de dados deve estar de acordo com a formatação
* "tipo;descrição;preçoDeCusto;margemDeLucro;[dataDeValidade]"
* ou o funcionamento não será garantido. Os tipos são 1, para produto não perecível; e 2, para perecível.
* @param linha Linha com os dados do produto a ser criado.
* @return Um produto com os dados recebidos
*/
static Produto criarDoTexto(String linha) {

        /* A implementação deste método deve separar os atributos existentes na String linha, verificar se o
produto é do tipo 1 ou 2, e instanciar o objeto adequado, com os dados fornecidos e de acordo com seu tipo. O
objeto instanciado é retornado pelo método. */

}
```

Adicionalmente, será necessário implementar o método ***gerarDadosTexto()*** nas duas subclasses, **ProdutoNaoPerecivel** e **ProdutoPerecivel**, obedecendo-se a estrutura abaixo:

Classe **ProdutoNaoPerecivel**:

```
/**
* Gera uma linha de texto a partir dos dados do produto.
* Preço e margem de lucro são formatados com 2 casas decimais.
* @return Uma string no formato "1;descrição;preçoDeCusto;margemDeLucro"
*/
@Override
public String gerarDadosTexto() {

}
```

Classe **ProdutoPerecivel**:

```
/**
* Gera uma linha de texto a partir dos dados do produto.
* Preço e margem de lucro são formatados com 2 casas decimais.
* Data de validade é formatada no formato dd/mm/aaaa
* @return Uma string no formato "2;descrição;preçoDeCusto;margemDeLucro;dataDeValidade"
*/
@Override
public String gerarDadosTexto() {

}
```

Tarefa 3 (em aula):
Criar um programa principal que consiga carregar vários Produtos (de qualquer tipo) para um vetor. O programa deve permitir também localizar produtos e armazenar os produtos no arquivo ao final de sua execução.

Assim, implemente em sua classe principal os seguintes métodos:

```
/**
* Lê os dados de um arquivo-texto e retorna um vetor de produtos. Arquivo-texto no formato:
* N  (quantidade de produtos) <br/>
* tipo;descrição;preçoDeCusto;margemDeLucro;[dataDeValidade] <br/>
* Deve haver uma linha para cada um dos produtos.
* Retorna um vetor vazio em caso de problemas com a leitura do arquivo.
* @param nomeArquivoDados Nome do arquivo de dados a ser aberto.
* @return Um vetor com os produtos carregados, ou vazio em caso de problemas de leitura.
*/
static Produto[] lerProdutos(String nomeArquivoDados) {

}

/** Localiza um produto no vetor de produtos cadastrados, a partir do nome de produto informado pelo usuário,
*  e imprime seus dados.
*  A busca não é sensível ao caso. No caso de não encontrar o produto, imprime uma mensagem padrão */
static void localizarProdutos() {

}

/**
* Salva os dados dos produtos cadastrados no arquivo csv informado. Sobrescreve todo o conteúdo do arquivo.
* @param nomeArquivo Nome do arquivo a ser gravado.
*/
public static void salvarProdutos(String nomeArquivo) {

}
```

Tarefa 4 (a ser concluída até o dia 23/02):
Incluir, no programa principal, as opções de listar todos os produtos cadastrados e cadastrar novos produtos.

Dessa forma, acrescente à sua classe principal os seguintes métodos:

```
/** Lista todos os produtos cadastrados, numerados, um por linha */
static void listarTodosOsProdutos() {

}

 /**
 * Rotina para cadastro de um novo produto: pergunta ao usuário o tipo do produto, lê os dados correspondentes,
 * cria o objeto adequado de acordo com seu tipo, e inclui o produto no vetor.
 */
 static void cadastrarProduto() {

 }
```

Instruções e observações:

- O projeto deve estar hospedado na tarefa correspondente do GitHub Classroom. Endereço para aceitar a tarefa: https://classroom.github.com/a/pEVEnAgS
- As atividades pontuadas da disciplina podem depender direta ou indiretamente dos códigos desenvolvidos nas aulas. Portanto, é essencial o comprometimento no acompanhamento das atividades semanais.
- Para a correção das atividades pontuadas, serão considerados todos os *commits/pushes* realizados ao longo das semanas, não somente o último com a resposta final do exercício.