Centro de Informática
Universidade Federal da Paraíba

Programação Orientada a Objetos 25.1
Projeto Final
(Controle de Lista de Compras)

Miracle Temitope Hazzan
20240144610

Setembro, 2025.

1. Introdução:

Este projeto tem como objetivo criar um sistema simples de gerenciamento de lista de compras usando a linguagem Java e os conceitos de Programação Orientada a Objetos (POO).

O sistema desenvolvido ajuda a resolver problema de organizar produtos, planejar quanto gastar e acompanhar os preços nos supermercados de forma prática, permitindo cadastrar produtos, registrar compras e gerar informações úteis para análise.

Com ele, o usuário pode:

- adicionar produtos com nome, unidade de medida, quantidade planejada e preço planejado;
- registrar compras feitas em diferentes meses e supermercados, com quantidade e preço unitário;
- calcular quanto foi gasto em um mês específico;
- descobrir o menor e o maior preço de um produto e em qual supermercado foi encontrado;
- comparar se os gastos reais estão próximos do que foi planejado;
- identificar qual supermercado foi o mais barato em determinado mês, explicando o motivo.

O sistema foi construído passo a passo, testado a cada etapa, com código organizado, comentado e fácil de entender. Dessa forma, ele atende aos requisitos do projeto da disciplina, mas também mostra como a POO pode ser aplicada em um problema real e útil do dia a dia.

2. Ferramentas Utilizadas:

Para o desenvolvimento do projeto foi utilizada a IDE IntelliJ IDEA Community Edition, escolhida por ser uma ferramenta fácil de usar. A linguagem de programação adotada foi o Java, que permitiu aplicar de forma prática os conceitos de Programação Orientada a Objetos.

Não foram utilizados frameworks externos. Foram usadas apenas bibliotecas nativas do Java, como:

- java.util (para uso de listas, mapas e scanner),
- java.lang (para exceções e operações básicas).

3. Desenvolvimento:

O sistema foi desenvolvido em Java, aplicando os conceitos de Programação Orientada a Objetos (POO), como encapsulamento, abstração e tratamento de exceções. O trabalho foi feito de forma passo a passo, sempre testando e validando cada parte antes de avançar, o que garantiu um código funcional e de fácil entendimento.

O projeto foi organizado em pacotes e dividido em quatro classes principais: Produto, Compra, GerenciadorListaCompras e Main. Também foi criada uma classe de exceção personalizada chamada ProductNotFoundException.

A. Classe Produto:

A classe Produto representa os itens que o usuário deseja cadastrar na lista de compras.
Ela possui atributos como nome, unidade de medida, quantidade planejada e preço planejado.
Com essa estrutura, é possível não apenas registrar o que precisa ser comprado, mas também planejar quanto se pretende gastar em cada produto.

Essa classe conta com construtor, getters e setters e o método toString, que facilita a exibição organizada dos dados.

B. Classe Compra:

A classe Compra representa uma compra realizada pelo usuário.
Seus atributos são: mês, nome do produto, quantidade comprada, preço unitário e supermercado.

Com essas informações, o sistema consegue calcular quanto foi gasto em um mês, comparar preços entre supermercados e identificar o menor e o maior preço de cada produto.
Assim como em Produto, a classe possui construtor, getters, setters e o método toString para exibir as informações de maneira clara.

C. Classe GerenciadorListaCompras

A classe GerenciadorListaCompras é o coração do sistema, pois faz o gerenciamento dos produtos e das compras registradas.
Ela utiliza duas listas (ArrayList): uma para armazenar os produtos e outra para armazenar as compras.

Principais métodos implementados:

- adicionarProduto: adiciona novos produtos à lista.

- registrarCompra: registra uma nova compra, mas verifica se o produto já foi cadastrado. Caso contrário, lança a exceção ProductNotFoundException.

- calcularTotalMes: calcula o total gasto em um mês específico.

- getMenorPrecoProduto: retorna o menor preço encontrado para determinado produto.

- getMaiorPrecoProduto: retorna o maior preço encontrado para determinado produto.

- supermercadoMaisBarato: identifica qual supermercado foi o mais barato em determinado mês e explica o motivo da vitória (qual produto teve o menor preço).

Essa classe concentra toda a lógica do sistema, mantendo o código organizado e separado por responsabilidades.

D. Classe Main:

A classe Main é responsável pela interação com o usuário.
Nela foi implementado um menu interativo usando Scanner, que permite:

1. Adicionar produtos (com nome, unidade, quantidade e preço planejado).
2. Registrar compras mensais.
3. Listar produtos cadastrados.
4. Listar compras realizadas.
5. Calcular o total gasto em determinado mês.
6. Consultar o menor preço de um produto.
7. Consultar o maior preço de um produto.
8. Identificar o supermercado mais barato em um mês.
9. Atualizar preço planejado de produto
10. Consultar supermercado mais caro do mês

0. Encerrar o programa.

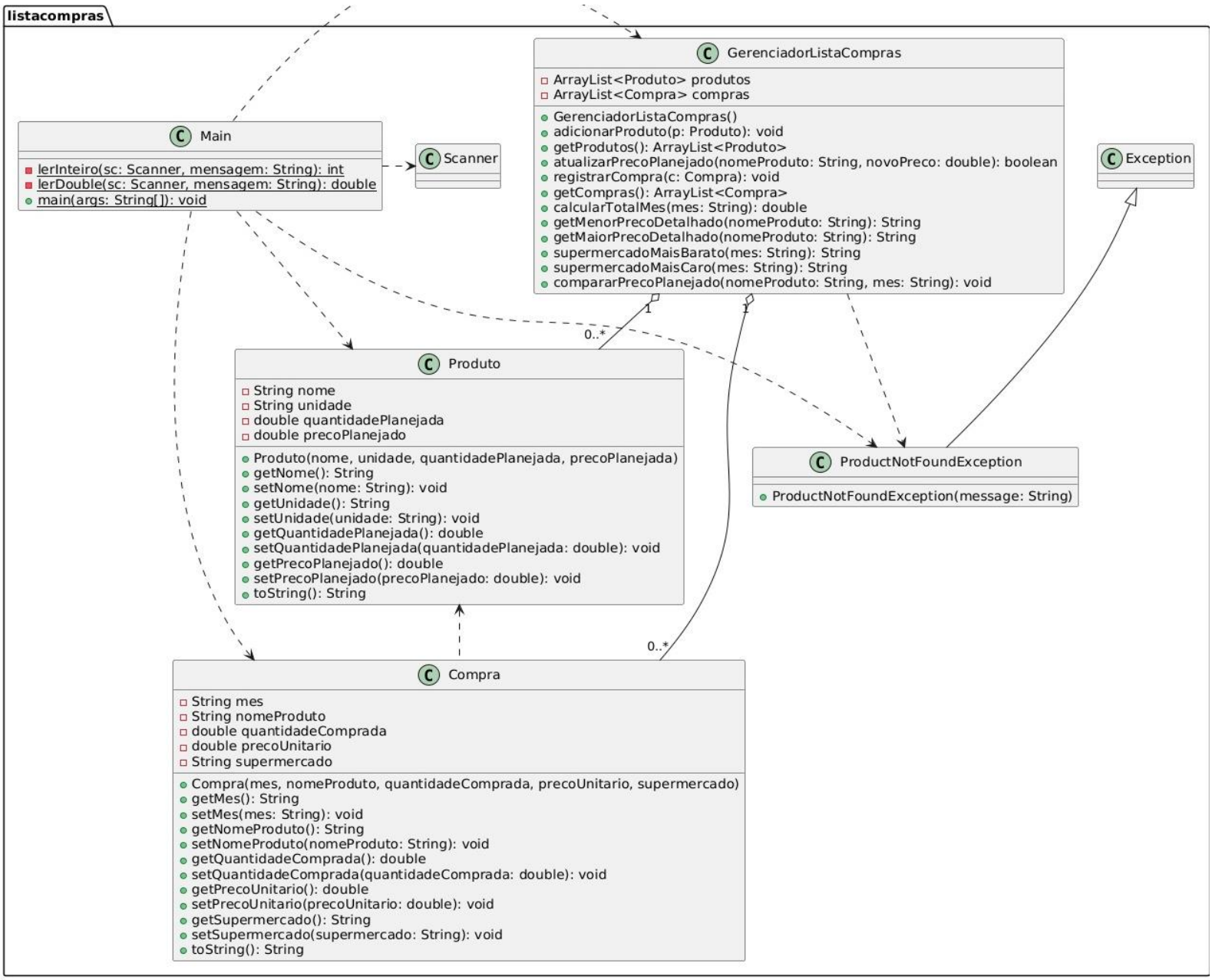
Além disso, foram incluídos tratamentos de erro para garantir que o programa não quebre quando o usuário digita valores incorretos (como letras no lugar de números).

5. Classe ProductNotFoundException:

Essa classe foi criada para tratar situações em que o usuário tenta registrar uma compra de um produto que não foi previamente cadastrado.
Ela torna o sistema mais confiável e evita erros lógicos.

4. Modelagem do problema(UML):

Para a criação do diagrama UML, foi utilizada a ferramenta PlantText, que é uma interface online para a linguagem de descrição PlantUML. Nela, é possível escrever a descrição das classes e seus relacionamentos em texto simples, e a ferramenta gera automaticamente a imagem do diagrama correspondente.



5. Dificuldades Encontradas:

Durante o desenvolvimento do projeto, uma das maiores dificuldades foi o **tratamento de exceções**. No começo, o programa estava funcionando bem, mas bastava eu digitar um número no lugar de uma palavra, ou uma palavra no lugar de um número, e tudo quebrava. O programa simplesmente fechava com um erro gigante na tela.

Nesses momentos eu precisava voltar ao código, procurar onde estava o problema e ajustar não só no **Main**, mas também em outras classes que tinham relação com ele. Isso exigiu bastante paciência, porque cada correção puxava outra. Passei horas testando e ajustando para que o sistema ficasse estável e fácil de usar, sem travar quando alguém cometesse um erro na entrada.

Eu confesso que quase chorei quando percebi que, se eu fosse **tratar todos os possíveis erros**, o código ia ficar enorme e complicado demais. Então, decidi focar em tratar apenas os erros que realmente poderiam dar dor de cabeça para o professor (ou para qualquer pessoa que fosse usar o programa). Isso deixou o sistema mais simples e, ao mesmo tempo, confiável.

No fim, esses momentos de teste e correção foram os mais longos, mas também os que mais me ensinaram sobre a importância de pensar na experiência de quem vai usar o sistema.

6. Resumo:

O desenvolvimento do sistema foi feito de forma gradual e estruturada.
Cada classe possui uma responsabilidade bem definida, o que facilita a manutenção e a compreensão do código.
Com isso, o programa atende aos requisitos do projeto: é simples, correto, legível e funcional, além de mostrar claramente a aplicação da Programação Orientada a Objetos em um problema real do dia a dia. Acesse o codigo. Você será direcionado ao Github.