

SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO UNIVERSIDADE FEDERAL DE VIÇOSA · UFV

CAMPUS FLORESTAL

Trabalho 1 - Programação Orientada a Objetos

Heron Fillipe Silveira Santos - 4211

Henrique Alves Campos 4231

Pedro Augusto Martins Pereira 4692

Vitor Vasconcelos Lobato 4210

Vitor Vasconcelos de Melo Pontes 4255

Florestal - MG 2023

**Sumário**

[**1. Introdução 2**](#_heading=h.gjdgxs)

[**2. Desenvolvimento 3**](#_heading=h.1fob9te)

[**3. Resultados 4**](#_heading=h.tyjcwt)

[**4 Executando o Trabalho 4**](#_heading=h.9h19x11ds6y5)

[**5 Conclusão 4**](#_heading=h.3dy6vkm)

[**6 Referências 5**](#_heading=h.1t3h5sf)

# Introdução

Neste projeto, concentramos nossos esforços no desenvolvimento de um sistema destinado a um ambiente prático de supermercado, explorando os fundamentos de orientação a objetos e a linguagem de programação Java. O desafio proposto envolve não apenas a aplicação técnica dos conceitos, mas também a criação de um sistema modular e eficiente que atenda às demandas específicas de um supermercado.

# Desenvolvimento

No início do projeto, demos o pontapé inicial ao criar nosso diagrama de classes, apresentando-o na segunda entrega parcial. Nessa fase, estruturamos a base do nosso sistema, delineando as relações e responsabilidades de cada classe. Em seguida, para otimizar o desenvolvimento e assegurar uma abordagem coesa, distribuímos as classes entre os membros da equipe. Cada integrante assumiu a responsabilidade de implementar e aprimorar as funcionalidades associadas a suas classes designadas.

O Projeto ficou com as seguintes classes e organizado como ilustra imagem abaixo seguida por uma breve descrição de cada classe:

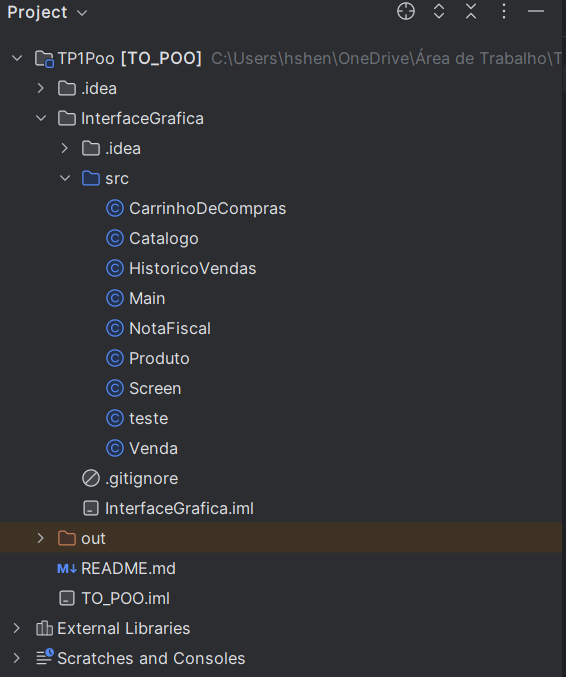


Figura 1 - Módulo do projeto

* **Carrinho de Compras**

Responsável por gerenciar os itens que o cliente comprará. Essa classe adiciona, remove e exibe os produtos presentes no carrinho, além de calcular o preço total da compra.

* **Catálogo**

Contém uma lista com os produtos disponíveis no mercado e imprime o catálogo.

* **Histórico Vendas**

Armazena o histórico de vendas, registrando informações sobre transações passadas.Essa classe adiciona e mostra a lista de todas as compras anteriores junto com algumas informações de tempo.

* **Main**

Classe principal que executa o programa.

* **Nota Fiscal**

Gera a nota fiscal da compra, mostrando cada unidade comprada, forma de pagamento e valor da compra. A classe também salva tudo em um arquivo .txt.

* **Produto**

Armazena informações sobre o produto, como nome e preço. Também controla a disponibilidade no estoque.

* **Screen**

É a classe responsável pelo funcionamento da interface gráfica. Contém todas as informações da tela de exibição, formatação e métodos necessários para a execução da interface.

* **Teste**

Classe que contém algumas entradas pré definidas para agilizar o processo de testes do código.

* **Venda**

Conclui a venda, remove os itens do estoque e gera a nota fiscal.

Outro ponto a se ressaltar foi a notável dificuldade enfrentada durante a implementação da interface gráfica, que se destacou como nossa maior classe e demandou um significativo investimento de esforço e código. A complexidade inerente à criação de uma interface visualmente atraente e funcional para o sistema de gerenciamento de supermercado apresentou desafios únicos, tanto em termos de design quanto de integração com a lógica de negócios.

# Diagrama de Classes disponível em :

<https://lucid.app/lucidchart/28e7ef84-ce1b-471a-8017-410217494996/edit?invitationId=inv_352f3efd-41a9-410c-8e7d-9f7094bd238e&page=HWEp-vi-RSFO#>

# Resultados

Durante o ciclo de desenvolvimento deste projeto, alcançamos resultados significativos que refletem não apenas a aplicação prática dos conceitos aprendidos, mas também a eficácia da abordagem adotada pela equipe. A implementação do sistema de gerenciamento para o supermercado resultou em módulos funcionais e interconectados, demonstrando a eficiência da arquitetura orientada a objetos escolhida.

Além disso, a integração bem-sucedida das classes desenvolvidas pelos membros da equipe resultou em um sistema coeso e funcional. A ausência de conflitos e a compatibilidade efetiva entre os diversos módulos destacam a eficácia da colaboração e comunicação durante o processo de desenvolvimento.

Embora os resultados finais sejam evidentes na operação integrada do sistema, também valorizamos as lições aprendidas ao longo do caminho. As dificuldades encontradas e os desafios superados contribuíram para o crescimento individual e coletivo da equipe, reforçando a compreensão prática dos princípios da programação orientada a objetos.

# 4 Executando o Trabalho

## Classe Produto

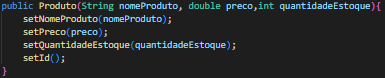
Para começar a execução do trabalho, a classe “Produto” foi desenvolvida em Java para modelar itens disponíveis em um supermercado.

Figura 2 - Construtores

Os construtores dessa classe permitem a criação de um novo item, definindo seu nome, preço e quantidade em estoque, por fim, gera identificador único para cada produto.

## Classe Catálogo

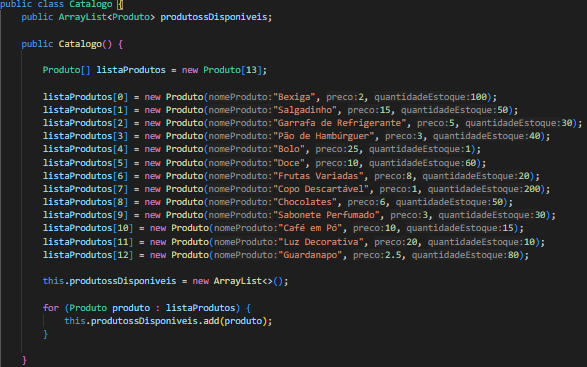
Nessa parte, definimos uma classe “Catalogo”, responsável por gerenciar uma lista de produtos em estoque em um supermercado, ou seja, ela inicia e armazena produtos a serem comprados.

Figura 3 - Lista de Produtos

## Classe Carrinho de Compras

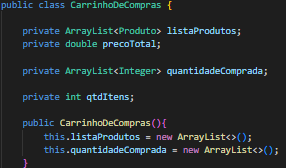
A classe “Carrinho de Compras”, como o próprio nome ja diz, simula um carrinho de compra em uma loja online ou um caixa de um supermercado, sendo ela, responsável por gerenciar os produtos selecionados pelo cliente, apresentando algumas funcionalidades, como adição e remoção de produtos, ver a lista de produtos e calcular o preço total da compra.

Figura 4 - Classe Carrinho de Compra

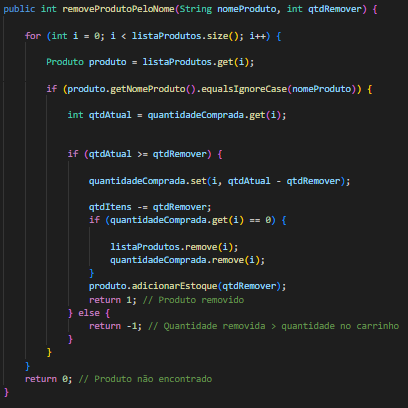
Como exemplo, temos a função de remoção de produtos, onde ele recebe o nome e quantidade de produtos que serão excluídos do carrinho de compras.

Figura 5 - Remoção de Produtos

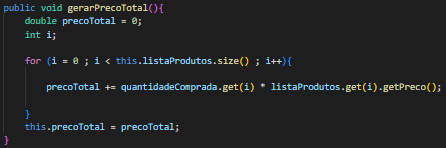
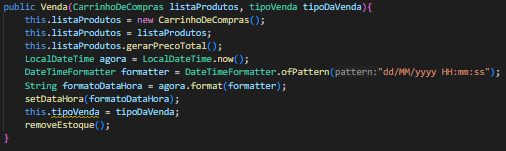
Outra função presente nessa classe é a que gera a valor total da compra, onde ela pega o valor unitário de cada produto e multiplica pela quantidade a ser comprada e por fim, soma o valor total de cada produto para encontrar o valor final.

Figura 6 - Gerar Preço Total

## Classe Venda

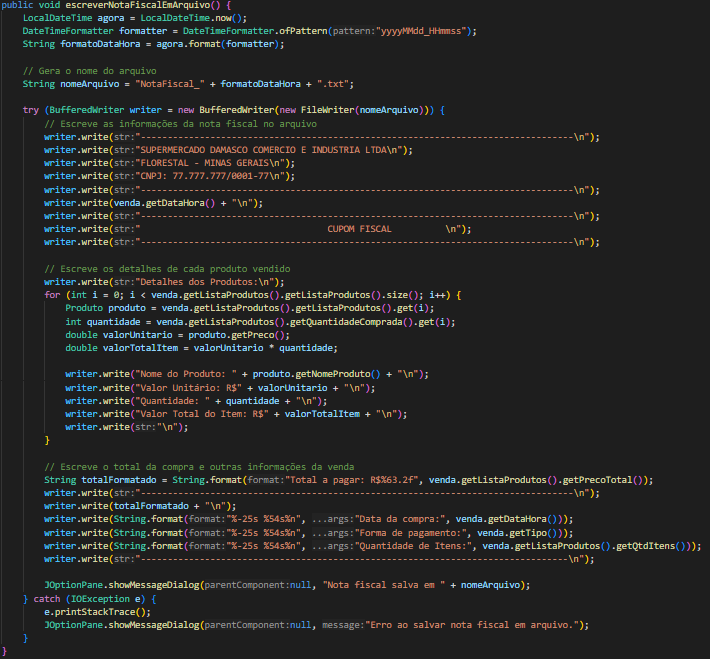
No início do projeto, decidimos que essa classe só entraria em “ação” quando o cliente decidisse finalizar a compra, sendo assim, quando o cliente confirmar a compra ela funciona, ou seja, ela é responsável pela conclusão de um transação de compra.

Figura 7 - Construtor 

Como pode ser visto na figura acima, essa função cria uma nova venda baseando-se no carrinho de compras fornecido, calcula seu valor total, registra a data e hora da venda, define o tipo de venda e por fim atualiza o estoque do supermercado.

## Classe Nota Fiscal

A classe ‘NotaFiscal’ é responsável por gerar e emitir a nota fiscal de cada venda, utilizamos a classe ‘JOptionPane’ para exibir mensagens de erro ou confirmação ao cliente sobre a geração da nota fiscal.

Figura 8 - Nota Fiscal

O método acima, gera um arquivo de texto contendo todas as informações da nota fiscal da venda realizada.

**Classe Histórico de Vendas**

A classe ‘HistoricoVendas’ representa, como o próprio nome já diz, um histórico de vendas, mantendo uma lista de vendas (‘Venda’ objetos), A seguir temos uma visão geral sobre.

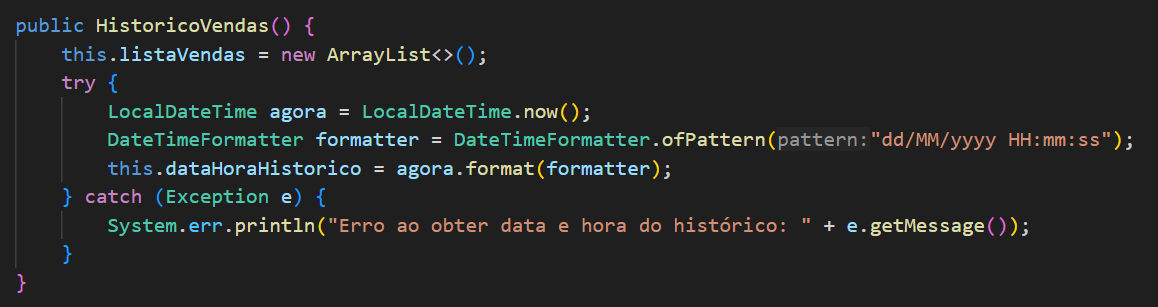
A figura abaixo descreve o construtor da classe, que inicializa uma lista de vendas(‘listaVendas’) e captura a data e hora atuais do sistema quando o objeto é criado.  


Figura 9 - Construtor da classe Historico Vendas

Nela, possuímos vários métodos que possibilitam o funcionamento correto. A seguir temos alguns deles.

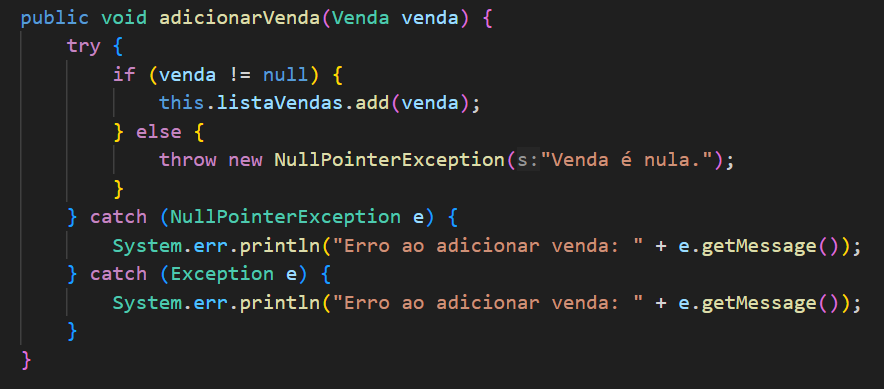
* adicionarVenda  
  

Figura 10 - Método da classe

* listarVendas

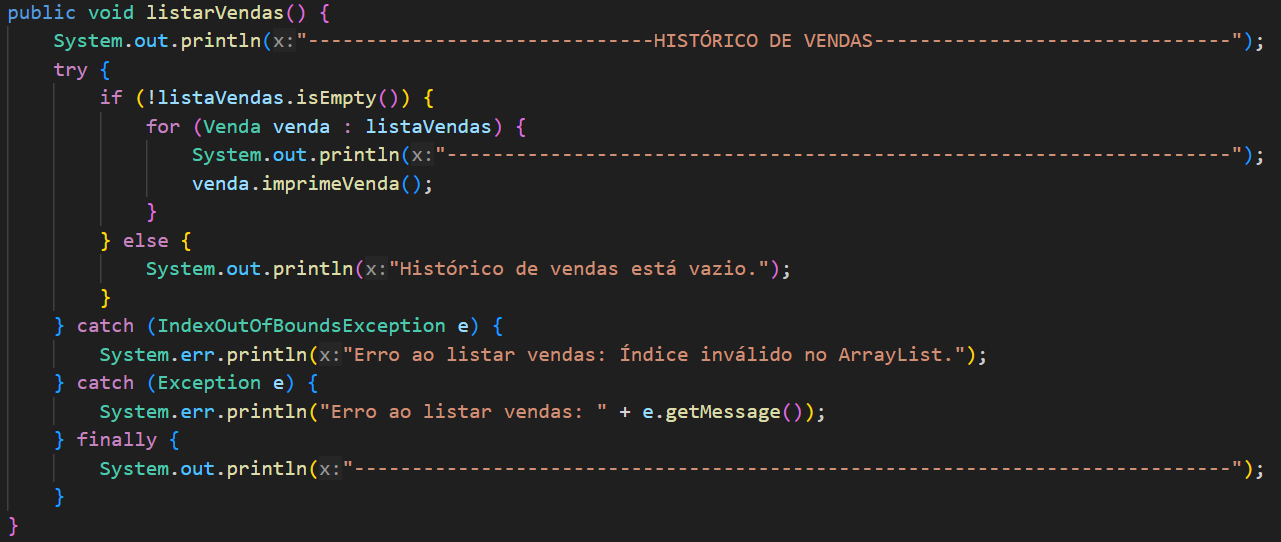


Figura 11 - Método da classe

* getDataHoraHistorico
* setDataHoraHistorico

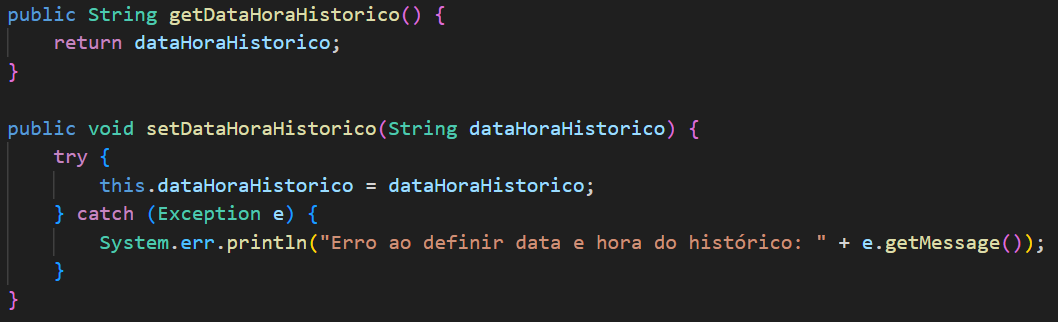


Figura 12 - Método da classe

É importante salientar que o tratamento de exceções foi implementado para lidar com possíveis erros durante a execução, garantindo uma experiência de uso mais robusta.

## Classe Screen

Este arquivo representa o desenvolvimento da interface gráfica de um sistema de supermercado, realizado por meio da utilização da biblioteca Swing em Java. A biblioteca Swing é conhecida por facilitar a criação de interfaces intuitivas e interativas.

Neste código, estão implementadas diversas funcionalidades que permitem aos usuários interagir com o sistema de supermercado. Isso inclui a visualização do catálogo de produtos disponíveis, a adição de itens ao carrinho de compras, a remoção de produtos do carrinho e a conclusão da compra.



Figura - Interface Gráfica

A imagem acima, mostra como ficou a interface gráfica criada no projeto, nela podemos fazer a adição e remoção de produtos baseados no nome dele.

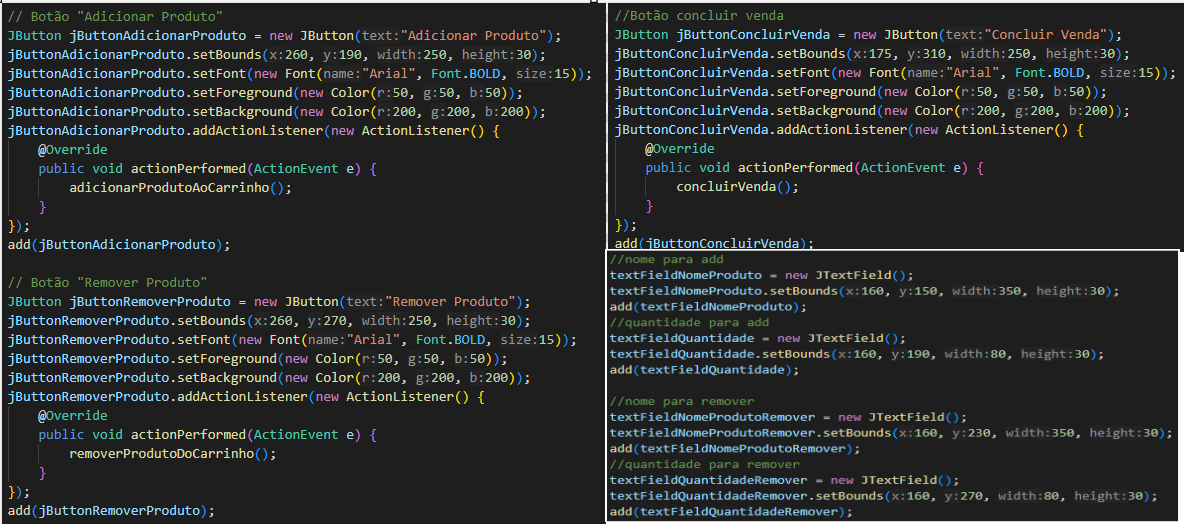


Figura - Botões criados para interface.

Como mostra a imagem acima, esses foram os botões criados na interface para realizar as ações do carrinho de compras, como adição ou remoção de produtos, ver a lista de produtos e finalizar a compra.

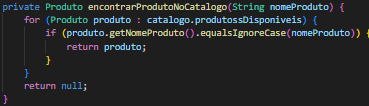


Figura - Função Encontrar Produto

A figura mostra a função utilizada tanto na função de adição quanto na de remoção do produto, pois, cabe a ela buscar no catálogo o produto, utilizando o nome dele como parâmetro e percorrendo a lista comparando os nomes dos produtos.

## .JAR

Primeiramente, verifique se o Java Development Kit (JDK) está devidamente instalado em seu sistema. Antes de executar o arquivo .jar, é crucial atribuir permissões de execução. Execute o seguinte comando no terminal ou prompt de comando:

## chmod +x TP1Poo.jar

1. Abra o terminal ou prompt de comando.
2. Navegue até o diretório onde o arquivo TP1Poo.jar está armazenado.
3. Utilize o comando a seguir para iniciar o trabalho: **java -jar TP1Poo.jar**

Essas etapas permitirão a execução adequada da aplicação e interação com a classe principal, Screen.java.

Além da execução via linha de comando com permissões concedidas ao arquivo .jar, é possível executar o trabalho utilizando uma IDE. Isso envolve a abertura do projeto na IDE, compilação da classe principal (Screen.java) .

# 5 Conclusão

Em conclusão, o desenvolvimento deste projeto representou uma jornada enriquecedora e desafiadora para nossa equipe. Ao abordar a criação de um sistema de gerenciamento para um supermercado, aplicamos de maneira prática os paradigmas da programação orientada a objetos e os conceitos fundamentais da linguagem Java.

Ao longo das etapas, desde a elaboração do diagrama de classes até a implementação e integração efetiva, cada membro da equipe desempenhou um papel crucial. A distribuição de responsabilidades e a colaboração em reuniões permitiram que cada parte do sistema fosse tratada com especialização, enquanto as interações regulares garantiram a coesão global do projeto.

# 6 Referências

Para particionar o projeto utilizamos o GitHub [1].  
<https://github.com/alveshenriique/TP1Poo>

Para criação do UML utilizamos o Lucidchart [2].

[1] Google Docs. Disponível em: <<https://docs.google.com>> Último acesso em: 11 de dezembro de 2023.

[2] GitHub. Disponível em: <<https://github.com/alveshenriique/TP1Poo>> Último acesso em: 11 de dezembro de 2023.

[3] Lucidchart. Disponível em: <<https://www.lucidchart.com/pages/>> Último acesso em: 11 de dezembro de 2023.