Trabalho
GRAU B

ARTHUR KELLERMEN, ARTHUR RAMBOR,
EDUARDO, GIOVANI DE SOUZA, ULISSES

Proposta

Montar um programa que simule o controle de estoque de um E-commerce (Ex.: Amazon e Mercado Livre).

Parâmetros

Para implementação desse programa seguimos as regras de:

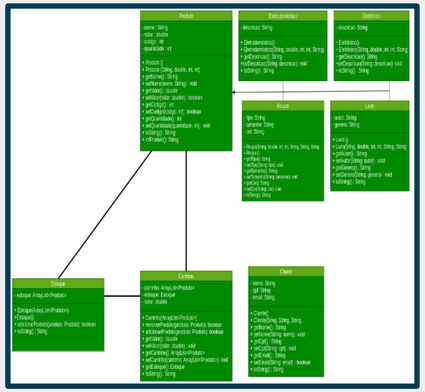
- Vender 4 tipos distintos de produtos previamente determinados (Livros, Eletrodomésticos, Roupas, Eletrônicos);
- Montar o sistema do estoque e juntamente um esquema de carrinho de compras (funcional e não complexo);
- Abranger todos os métodos implementados com testes unitários JUnit (mínimo 2 testes por método);

Tópicos do Trabalho

APRESENTAÇÃO

1	2	3	4	5
6	7	8	9	10

DIAGRAMA UML



CÓDIGO

```
try {

        String dirPath = "./Notinha";
        File diretorio = new File(dirPath);

        if (diretorio.mkdirs()) {
            System.out.println("Pasta da notinha criada em: " + diretorio.getAbsolutePath());
        }
```

Diagrama UML

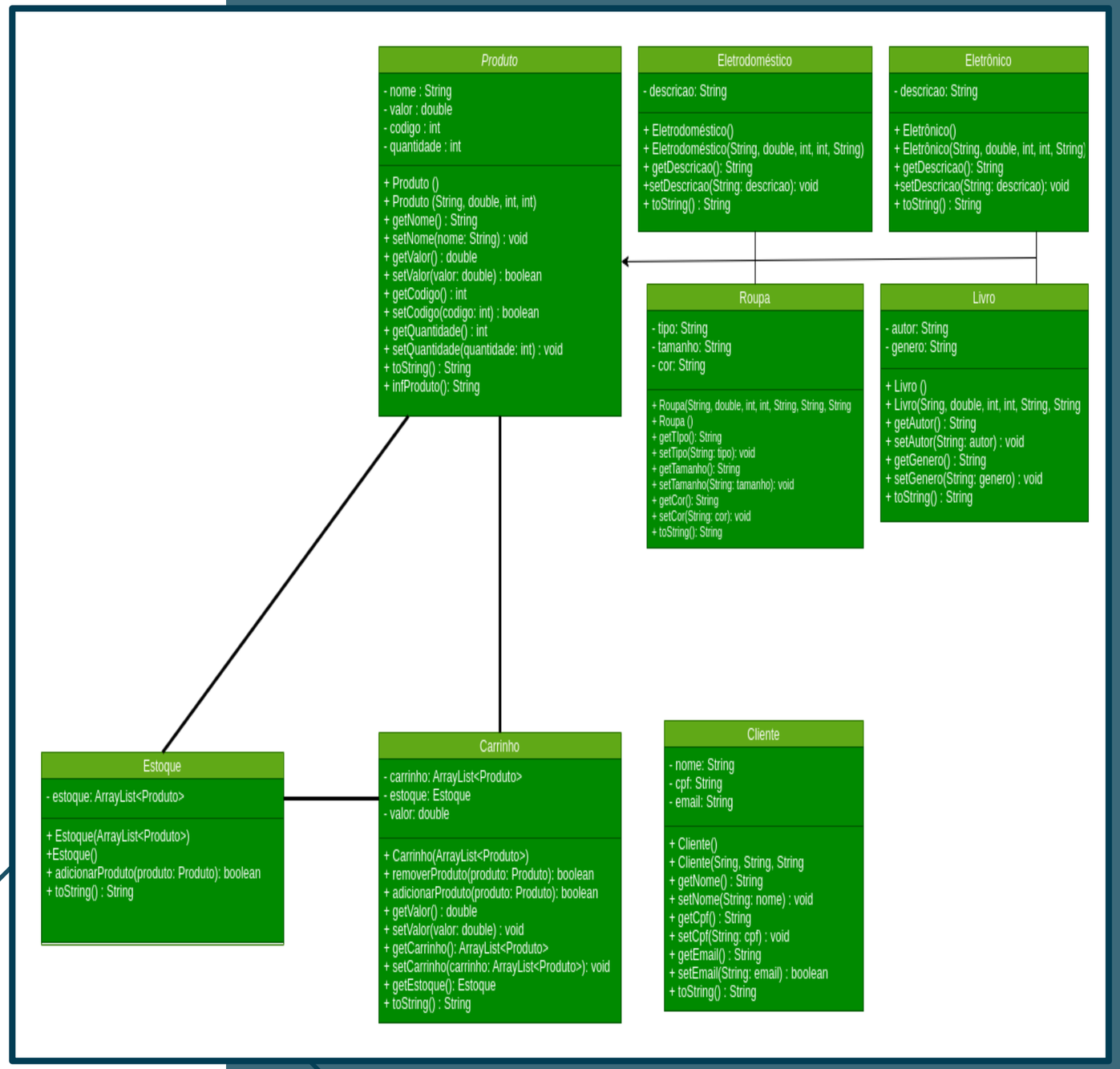
Diagrama padrão UML com classes detalhadas e indicadas as respectivas associações entre classe.

Superclasse: Produto

- Subclasses: Roupa, Eletrodoméstico, Livro e Eletrônico;

Classes compostas:

- Carrinho (contendo um ArrayList de produtos);
- Estoque (contendo um ArrayList de produtos);



Código

Produto

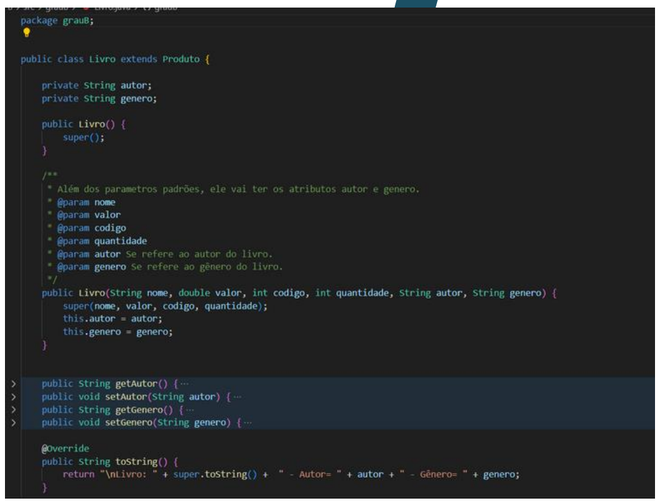
Como já mencionado, a classe produto é uma superclasse generalista que vai conter informações necessárias para as subcategorias de produto.
Nessa classe, além de possuir os métodos tradicionais (getters/setters) foi necessário realizar um filtro no momento de configurar o código e o valor, ou seja, proibindo valores abaixo de zero.
Principal método desta classe é o "public String infProduto()" que passa as informações de maneira simplificada sobre o produto, este método é utilizado no "main" para informar as características dos produtos selecionados (carrinho) de maneira resumida, outra utilização desse resumo de informações é no momento de imprimir uma nota fiscal.

```java
package grauB;


public abstract class Produto {

    private String nome;
    private double valor;
    public int codigo;
    public int quantidade;

    /**
     * Produto é qualquer produto presente na loja.
     * @param nome É o nome de determinado produto.
     * @param valor O valor de determinado produto.
     * @param codigo Cada produto terá seu código específico.
     * @param quantidade Se refere a quantidade desse produto presente no estoque da empresa.
     */

    public Produto(String nome, double valor, int codigo, int quantidade) {
        this.nome = nome;
        this.valor = valor;
        this.codigo = codigo;
        this.quantidade = quantidade;
    }

    public Produto() {
    }

    public String getNome() {···

    public void setNome(String nome) {···

    public double getValor() {···

    public boolean setValor(double valor) {···

    public int getCodigo() {···
```

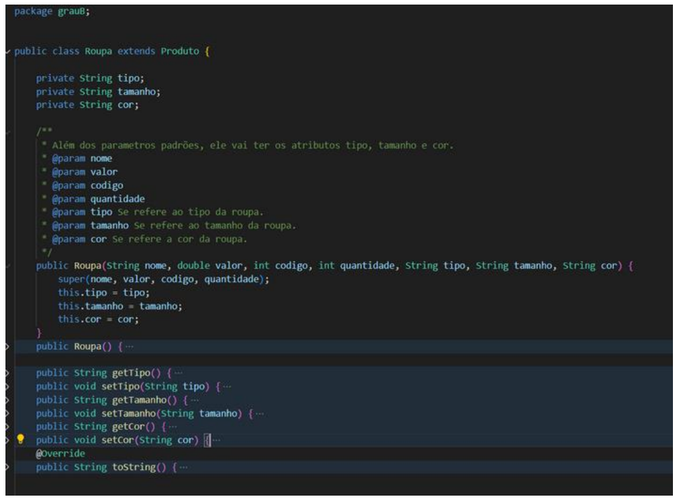
Código

Subclasses

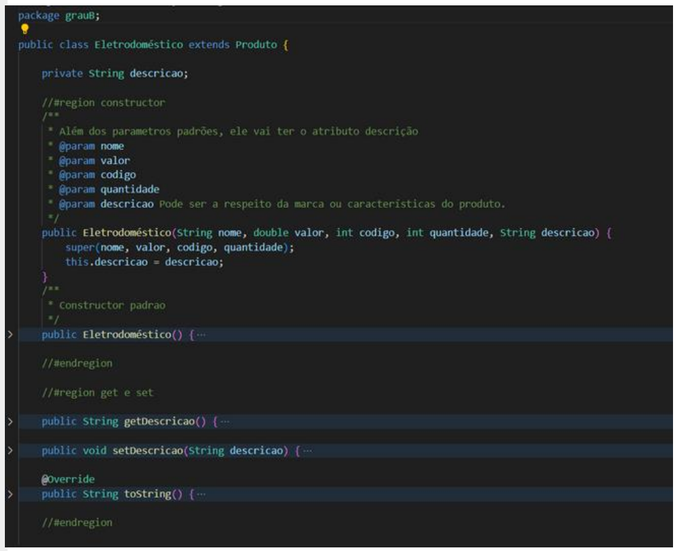
Classes derivadas da classe mãe Produto estendendo todos as características com o método "super" e criando novos conforme necessidade individual (Ex.: Livro: String gênero).



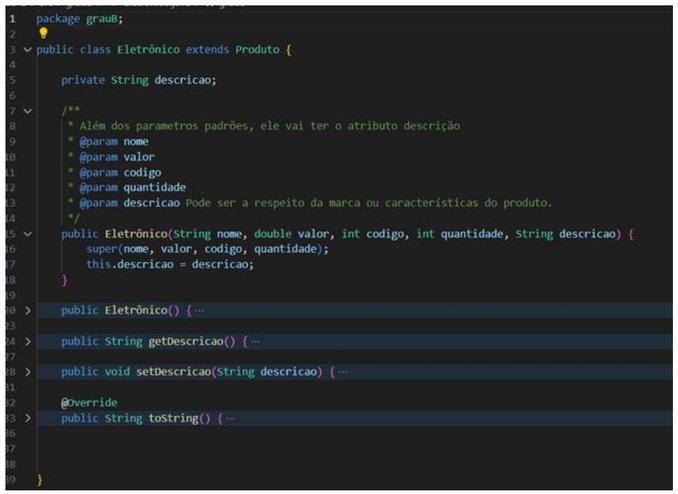
```java
package grauB;

public class Livro extends Produto {

    private String autor;
    private String genero;

    public Livro() {
        super();
    }

    /**
     * Além dos parametros padrões, ele vai ter os atributos autor e genero.
     * @param nome
     * @param valor
     * @param codigo
     * @param quantidade
     * @param autor Se refere ao autor do livro.
     * @param genero Se refere ao gênero do livro.
     */
    public Livro(String nome, double valor, int codigo, int quantidade, String autor, String genero) {
        super(nome, valor, codigo, quantidade);
        this.autor = autor;
        this.genero = genero;
    }

    public String getAutor() {...
    public void setAutor(String autor) {...
    public String getGenero() {...
    public void setGenero(String genero) {...

    @Override
    public String toString() {
        return "\nLivro: " + super.toString() +  " - Autor= " + autor + " - Gênero= " + genero;
    }
```



```java
package grauB;

public class Roupa extends Produto {

    private String tipo;
    private String tamanho;
    private String cor;

    /**
     * Além dos parametros padrões, ele vai ter os atributos tipo, tamanho e cor.
     * @param nome
     * @param valor
     * @param codigo
     * @param quantidade
     * @param tipo Se refere ao tipo da roupa.
     * @param tamanho Se refere ao tamanho da roupa.
     * @param cor Se refere a cor da roupa.
     */
    public Roupa(String nome, double valor, int codigo, int quantidade, String tipo, String tamanho, String cor) {
        super(nome, valor, codigo, quantidade);
        this.tipo = tipo;
        this.tamanho = tamanho;
        this.cor = cor;
    }
    public Roupa() {...

    public String getTipo() {...
    public void setTipo(String tipo) {...
    public String getTamanho() {...
    public void setTamanho(String tamanho) {...
    public String getCor() {...
    public void setCor(String cor) {...
    @Override
    public String toString() {...
```



```java
package grauB;

public class Eletrodoméstico extends Produto {

    private String descricao;

    //#region constructor
    /**
     * Além dos parametros padrões, ele vai ter o atributo descrição
     * @param nome
     * @param valor
     * @param codigo
     * @param quantidade
     * @param descricao Pode ser a respeito da marca ou caracteristicas do produto.
     */
    public Eletrodoméstico(String nome, double valor, int codigo, int quantidade, String descricao) {
        super(nome, valor, codigo, quantidade);
        this.descricao = descricao;
    }
    /**
     * Constructor padrao
     */
    public Eletrodoméstico() {...

    //#endregion

    //#region get e set

    public String getDescricao() {...

    public void setDescricao(String descricao) {...

    @Override
    public String toString() {...

    //#endregion
```



```java
package grauB;

public class Eletrônico extends Produto {

    private String descricao;

    /**
     * Além dos parametros padrões, ele vai ter o atributo descrição
     * @param nome
     * @param valor
     * @param codigo
     * @param quantidade
     * @param descricao Pode ser a respeito da marca ou caracteristicas do produto.
     */
    public Eletrônico(String nome, double valor, int codigo, int quantidade, String descricao) {
        super(nome, valor, codigo, quantidade);
        this.descricao = descricao;
    }

    public Eletrônico() {...

    public String getDescricao() {...

    public void setDescricao(String descricao) {...

    @Override
    public String toString() {...


}
```

Código

Estoque

Estoque foi definido como um ArrayList de produtos.
Principal método: adicionarProduto.
Método de adição de produtos, nele contém a verificação de quantidade de produtos, permitindo somente adicionar valores superiores a zero.

```java
package grauB;
import java.util.ArrayList;

public class Estoque {
    private ArrayList<Produto> estoque = new ArrayList<Produto>();

    public Estoque() {...
    /**
     * O estoque é composto de um ArrayList de produtos
     * @param estoque
     */
    public Estoque(ArrayList<Produto> estoque) {
        this.estoque = new ArrayList<Produto>();
    }
    /**
     * É um método que adiciona produtos no estoque
     * @param produto Ele confere se a quantidade do produto adicionado é maior que 0, e em seguida adiciona ao estoque
     * @return
     */
    public boolean adicionarProduto(Produto produto) {
        try {
        if (produto.getQuantidade() <= 0) {
            return false;
        }
            return estoque.add(produto);
        } catch (Exception e) {
            return false;
        }
    }

    @Override
    public String toString() {
        return "Estoque:" + estoque;
    }

}
```

Código

Carrinho

Carrinho foi definido como um ArrayList de produtos e tem como atributo principal o valor total do carrinho.
Principais métodos: removerProduto e o adicionarProduto.
Em suma, utilizam métodos presentes na classe ArrayList, ou seja, .ad e .remove. Unico trabalho que foi feito para diferenciar e ter uma finalidade diferente da tradicional, foi adicionar o produto de volta ao estoque quando ele é retirado do carrinho e quando ele é adicionado no carrinho ele desconta do estoque.

```java
package grauB;


public abstract class Produto {

    private String nome;
    private double valor;
    public int codigo;
    public int quantidade;

    /**
     * Produto é qualquer produto presente na loja.
     * @param nome É o nome de determinado produto.
     * @param valor O valor de determinado produto.
     * @param codigo Cada produto terá seu código específico.
     * @param quantidade Se refere a quantidade desse produto presente no estoque da empresa.
     */

    public Produto(String nome, double valor, int codigo, int quantidade) {
        this.nome = nome;
        this.valor = valor;
        this.codigo = codigo;
        this.quantidade = quantidade;
    }

    public Produto() {
    }

    public String getNome() {···

    public void setNome(String nome) {···

    public double getValor() {···

    public boolean setValor(double valor) {···

    public int getCodigo() {···
```

Código

Main

Além do "public static void main(String[] args)", existe o "public static boolean notinha".
No método principal é realizado um menu com interações com o cliente, onde serão selecionados por ele quais produtos deseja do estoque, adicionando ou não no carrinho e, no momento do pagamento é chamado o método "notinha".
Nesse método, é criado um arquivo .txt com informações da compra realizada pelo cliente, simulando um nota fiscal.

```java
package grauB;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    /**
     * Função para imprimir nota(arquivo txt) na pasta local do projeto
     * @param cpfNaNota 1 = sim, 0 = não
     * @param cpf recebe o cpf do cliente
     * @param carrinho recebe o carrinho com a ArrayList de produtos
     * @return true para certo e false para errado
     */
    public static boolean notinha(int cpfNaNota, String cpf, Carrinho carrinho) {
        try {
            String dirPath = "./Notinha";
            File diretorio = new File(dirPath);
            if (diretorio.mkdirs()) {
                System.out.println("Pasta da notinha criada em: " + diretorio.getAbsolutePath());
            }
            String path = diretorio.getAbsolutePath() + "\\notinha.txt";
            try (BufferedWriter bw = new BufferedWriter(new FileWriter(path))) {

                System.out.println(x:"Notinha impressa com sucesso!");
                if (cpfNaNota == 1) {
                    bw.write("Cliente de CPF: " + cpf);
                    bw.write(" | Produtos comprados: " + "\n");
                    for (Produto produto : carrinho.getCarrinho()) {
                        if (produto instanceof Produto) {
                            Produto product = (Produto) produto;
                            bw.write(product.infProduto() + "\n");
                        }
                    }
                    bw.write("Valor total: R$ " + String.format(format:"%.2f", carrinho.getValor()));
```

Código

Main

Parte onde implementou-se o menu de escolha para o cliente e foram definidas as variáveis Produto para preenchimento do Estoque da loja fictícia.

```
                bw.write("Valor total: R$ " + String.format("%.2f", carrinho.getValor()));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return true;
    } catch (Exception e) {
        System.out.println(e.toString());
        return false;
    }
}
Run | Debug
public static void main(String[] args) {
    Scanner in = new Scanner(System.in);

    System.out.println("--CADASTRO--");
    System.out.print("Digite seu nome: ");
    String nome = in.nextLine();
    System.out.print("Digite seu CPF: ");
    String cpf = in.nextLine();
    System.out.print("Digite seu email: ");
    String email = in.nextLine();

    Cliente cliente = new Cliente(nome, cpf, email);

    Produto camisaG = new Roupa("Camisa G", 60, 101, 4, "Tecido", "G", "Preta");
    Produto camisaM = new Roupa("Camisa M", 60, 102, 16, "Tecido", "M", "Preta");
    Produto camisaP = new Roupa("Camisa P", 60, 103, 10, "Tecido", "P", "Preta");
    Produto pp = new Livro("Pequeno Principe", 40, 201, 2, "Antoine de Saint-Exupéry", genero
```

Código

Main

Definição do restante de variáveis componentes do Estoque. Com continuidade do menu de escolha e impressão das informações de Estoque.

```
Produto crepusculo = new Livro(nome:"Crepúsculo", valor:50, codigo:202, quantidade:5, autor:"Stephenie Meyer", genero:"Romance");
Produto dk = new Livro(nome:"Dom Quixote", valor:30, codigo:203, quantidade:3, autor:"Miguel de Cervantes", genero:"Aventura");
Produto PC = new Eletrônico(nome:"PC Gamer", valor:3300, codigo:301, quantidade:3, descricao:"AMD Ryzen 5 4600G, 16GB DDR4, SSD 4
Produto iphone = new Eletrônico(nome:"Iphone 14", valor:8000, codigo:302, quantidade:10, descricao:"Apple iPhone14 Pro Max 128 GB
Produto monitor = new Eletrônico(nome:"Monitor 23,5'", valor:1400, codigo:303, quantidade:5, descricao:"Samsung");
Produto freezer = new Eletrodoméstico(nome:"Freezer", valor:3500, codigo:401, quantidade:4, descricao:"Horizontal Consul 2 portas
Produto geladeira = new Eletrodoméstico(nome:"Geladeira", valor:2800, codigo:402, quantidade:8, descricao:"Electrolux");
Produto fogao = new Eletrodoméstico(nome:"Fogão", valor:2000, codigo:403, quantidade:8, descricao:"Brastemp 4 bocas");

Estoque empresa = new Estoque(new ArrayList());

empresa.adicionarProduto(camisaG);
empresa.adicionarProduto(camisaM);
empresa.adicionarProduto(camisaP);
empresa.adicionarProduto(pp);
empresa.adicionarProduto(crepusculo);
empresa.adicionarProduto(dk);
empresa.adicionarProduto(PC);
empresa.adicionarProduto(iphone);
empresa.adicionarProduto(monitor);
empresa.adicionarProduto(freezer);
empresa.adicionarProduto(fogao);
empresa.adicionarProduto(geladeira);

System.out.println();
System.out.println(x:"Bem vindo a Loja Grau B, segue abaixo os produtos disponíveis em nosso estoque: ");

System.out.println(empresa);

Carrinho usuario = new Carrinho(new ArrayList<>());

System.out.println(x:" ");
System.out.println("Carrinho do " + cliente.getNome());

int opcao = 1000;
do {
    System.out.print(
```

Ln 67, Col 1 Tab Size: 4 UTF-8 CRLF

Código

Main

Switch case onde adiciona o produto desejado ao carrinho, adição feita com base na escolha do cliente mediante ao código do produto.

```
        s:"Digite o código do produto que você quer adicionar ou digite 0 para finalizar o carrinho: ");
opcao = in.nextInt();

switch (opcao) {
case 101:
    usuario.adicionarProduto(camisaG);
    break;
case 102:
    usuario.adicionarProduto(camisaM);
    break;
case 103:
    usuario.adicionarProduto(camisaP);
    break;
case 201:
    usuario.adicionarProduto(pp);
    break;
case 202:
    usuario.adicionarProduto(crepusculo);
    break;
case 203:
    usuario.adicionarProduto(dk);
    break;
case 301:
    usuario.adicionarProduto(PC);
    break;
case 302:
    usuario.adicionarProduto(iphone);
    break;
case 303:
    usuario.adicionarProduto(monitor);
    break;
case 401:
    usuario.adicionarProduto(freezer);
    break;
case 402:
    usuario.adicionarProduto(fogao);
    break;
```

Código

Cliente

Classe criada para armazenar informações sobre o cliente.
Vale ressaltar que as informações do cliente são solicitadas via teclado no início do "main" e depois são impressas na nota.
Além dos getters e setters presentes na classe, acontece uma verificação se o email contém "@" no setEmail.

```
        case 402:
            usuario.adicionarProduto(fogao);
            break;
        case 403:
            usuario.adicionarProduto(geladeira);
            break;
        case 0:
            System.out.println(x:"Carrinho finalizado!");
            break;
        default:
            System.out.println(x:"Código Inválido!");
            break;
        }
    } while (opcao != 0);

    System.out.println(x:"\nProdutos selecionados:");
    for (Produto produto : usuario.getCarrinho()) {
        if (produto instanceof Produto) {
            Produto product = (Produto) produto;
            System.out.println(product.infProduto());
        }
    }
    System.out.println("Valor Total: R$ " + String.format(format:"%.2f", usuario.getValor()));

    System.out.println(x:"CPF na nota? (s/n)");
    char cpfNaNota = in.next().charAt(index:0);

    if (cpfNaNota == 's' || cpfNaNota == 'S') {
        notinha(cpfNaNota:1, cliente.getCpf(), usuario);
    } else {
        notinha(cpfNaNota:0, cpf:null, usuario);
    }

    System.out.println("Obrigado pela confiança, " + cliente.getNome());
```

Código

Testes JUnit

Testes dos métodos de classe da classe Produto.
Testes confirmando a validade dos códigos do produto somente quando código for positivo. E outros testes com intenção de confirmar que o setValor aceita apenas números positivos.

```java
/**
 * verifica se a condicao esta certa inserindo um valor negativo e depois um positivo
 * @author ulisses953
 */
@Test
void testSetCodigo() {
    var livro = new Livro();

    assertFalse(livro.setCodigo(-10));

    assertTrue(livro.setCodigo(codigo:10));

}
/**
 * teste referente ao valor do produto
 * @author ulisses953
 */
@Test
void testGetValor(){
    var livro = new Livro();

    assertFalse(livro.setValor(-10));

    assertTrue(livro.setValor(valor:10));
}
```

Código

Testes JUnit

Testes dos métodos de classe da classe Estoque.
Testes confirmando que ao adicionar um produto válido ao Estoque, o método retorna "true" e, caso contrário, retorna "false".

```
void testAdiciona1() {
    Estoque est = new Estoque();
    Livro liv = new Livro(nome:"Crepúsculo", valor:30.00d, codigo:201, quantidade:4, autor:"Stephanie Meyer", genero:"Romance");
    assertTrue(est.adicionarProduto(liv));
}

/**
 * Teste que adiciona produto ao estoque e confirma retorno false ao não adicionar
 */
@Test
void testAdiciona2() {
    Estoque est = new Estoque();
    Livro liv = new Livro(nome:"Crepúsculo", valor:30.00d, codigo:201, quantidade:0, autor:"Stephanie Meyer", genero:"Romance");

    assertFalse(est.adicionarProduto(liv));
}
```

Código

Testes JUnit

Testes dos métodos de classe da classe Carrinho.
Testes confirmando que ao adicionar um produto válido ao Carrinho, o método retorna "true" e, caso contrário, retorna "false".

```java
void testAdiciona(){
    Estoque est = new Estoque();
    Livro liv = new Livro(nome:"Crepúsculo", valor:30.00d, codigo:201, quantidade:1, autor:"Stephanie Meyer", genero:"Romance");
    est.adicionarProduto(liv);

    Carrinho carrinho = new Carrinho(new ArrayList<>());
    assertTrue(carrinho.adicionarProduto(liv));
    assertFalse(carrinho.adicionarProduto(liv));
}

/**
 * Teste que remove produto ao carrinho e confirma retorno true ao remover
 * Teste que remove produto ao carrinho e confirma retorno false ao não remover
 */
@Test
void testRemove() {

    Estoque est = new Estoque();
    Livro liv = new Livro(nome:"Crepúsculo", valor:30.00d, codigo:201, quantidade:1, autor:"Stephanie Meyer", genero:"Romance");
    est.adicionarProduto(liv);

    Carrinho carrinho = new Carrinho(new ArrayList<>());
    carrinho.adicionarProduto(liv);
    assertTrue(carrinho.removerProduto(liv));
    assertFalse(carrinho.removerProduto(liv));
}
```

Código

Testes JUnit

Testes dos métodos de classe da classe Cliente.
Testes confirmando que ao adicionar um e-mail válido ao perfil de cliente, o método retorna "true" e, caso contrário, retorna "false".

```java
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import org.junit.jupiter.api.Test;

public class ClienteTest {
    /**
     *
     * @autor ulisses953
     */
    @Test
    void testSetEmail() {
        Cliente a = new Cliente();
        assertFalse(a.setEmail(email:"ulisseskranzdamotta"));
        assertTrue(a.setEmail(email:"ulisseskranzdamotta@gmail.com"));
    }
}
```

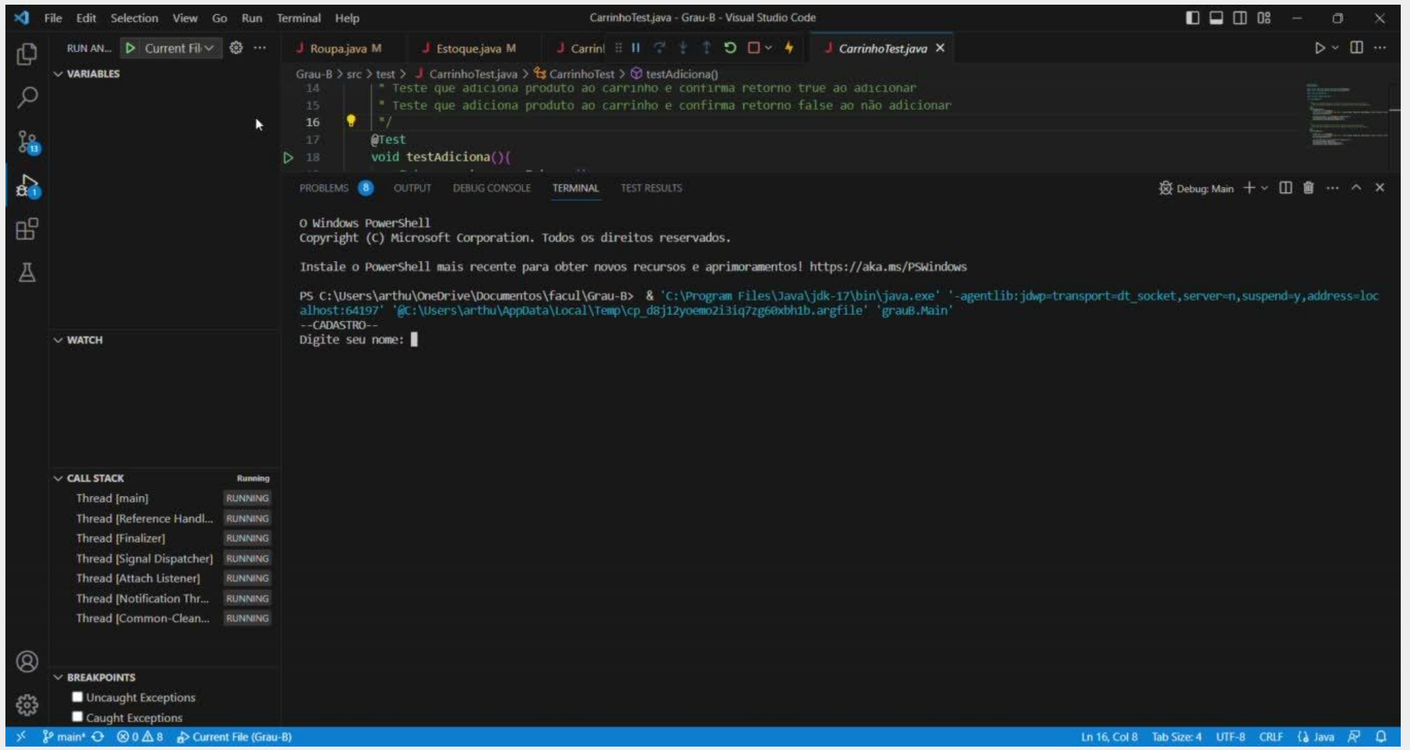
Código

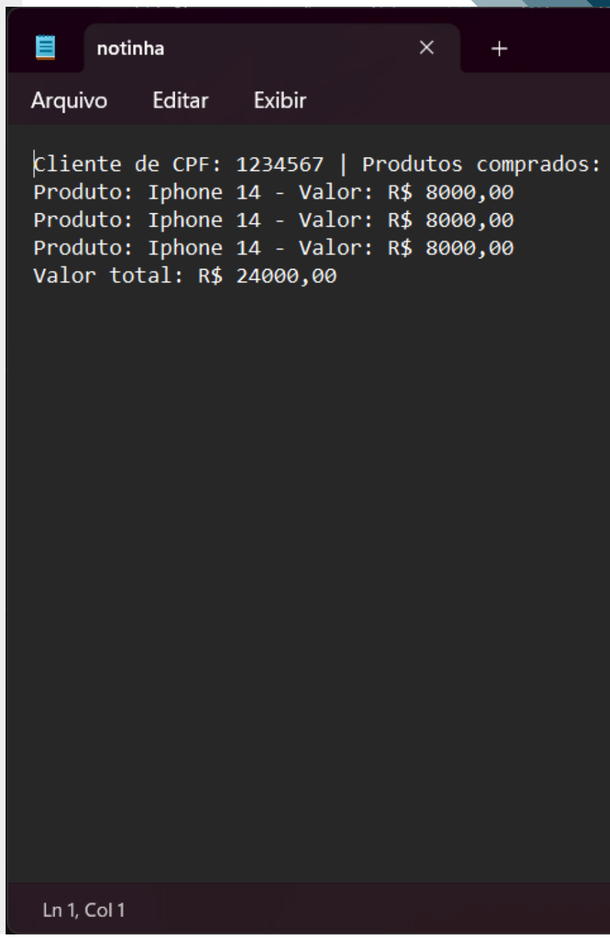
Testes JUnit

Testes dos métodos de classe da classe Main.
Testes confirmando que o método de impressão da notinha só é executado corretamente quando cliente opta por isso (retorno true) não executando quando cliente não desejar (retorno false).

```
    @Test
    void testNotinha1() {
        Main main = new Main();
        int cpfNota = 1;
        String cpf = "12345";
        Carrinho car = new Carrinho();
        assertTrue(main.notinha(cpfNota, cpf, car));
    }
    /**
     * Teste para validar retorno falso ao adicionar produto inválido na nota fiscal com método
     */
    @Test
    void testNotinha2() {
        Main main = new Main();
        int cpfNota = 4;
        String cpf = "12345";
        Carrinho car = new Carrinho();
        assertFalse(main.notinha(cpfNota, cpf, car));
    }

}
```

Exemplo:


File Edit Selection View Go Run Terminal Help
CarrinhoTest.java - Grau-B - Visual Studio Code
RUN AN... Current Fil
VARIABLES
Roupa.java M
Estoque.java M
Carrin
CarrinhoTest.java
Grau-B > src > test > CarrinhoTest.java > CarrinhoTest > testAdiciona()
14 * Teste que adiciona produto ao carrinho e confirma retorno true ao adicionar
15 * Teste que adiciona produto ao carrinho e confirma retorno false ao não adicionar
16 */
17 @Test
18 void testAdiciona(){
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL TEST RESULTS
Debug: Main
O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.
Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows
PS C:\Users\arthu\OneDrive\Documentos\facul\Grau-B> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:64197' '@C:\Users\arthu\AppData\Local\Temp\cp_d8j12yoemo2i3iq7zg60xbh1b.argfile' 'grau8.Main'
--CADASTRO--
Digite seu nome:
WATCH
CALL STACK Running
Thread [main] RUNNING
Thread [Reference Handl... RUNNING
Thread [Finalizer] RUNNING
Thread [Signal Dispatcher] RUNNING
Thread [Attach Listener] RUNNING
Thread [Notification Thr... RUNNING
Thread [Common-Clean... RUNNING
BREAKPOINTS
Uncaught Exceptions
Caught Exceptions
main* 0 8 Current File (Grau-B)
Ln 16, Col 8 Tab Size: 4 UTF-8 CRLF Java


notinha
Arquivo Editar Exibir
Cliente de CPF: 1234567 | Produtos comprados:
Produto: Iphone 14 - Valor: R$ 8000,00
Produto: Iphone 14 - Valor: R$ 8000,00
Produto: Iphone 14 - Valor: R$ 8000,00
Valor total: R$ 24000,00
Ln 1, Col 1