Universidade Federal do Mato Grosso
Instituto de Computação

Documentação: Sistema de Gerenciamento de Farmácia

ANTHONY RICARDO RODRIGUES REZENDE
LETÍZIA MANUELLA SERQUEIRA EUGÊNIO
VINÍCIUS PADILHA VIEIRA

CUIABÁ, MATO GROSSO 2023

ANTHONY RICARDO RODRIGUES REZENDE
LETÍZIA MANUELLA SERQUEIRA EUGÊNIO
VINÍCIUS PADILHA VIEIRA

Documentação: Sistema de Gerenciamento de Farmácia

Documentação sobre "Sistema de Gerenciamento de Farmácia" conforme a disciplina de Algoritmos III, apresentado como requerido para a obtenção de nota na Universidade Federal de Mato Grosso - UFMT

Professor: Dr. Jivago Medeiros Ribeiro

CUIABÁ, MATO GROSSO 2023

Resumo

Documentação do Sistema de Gerenciamento de Farmácia, conforme atividade proposta como Prova de Algoritmos III, desenvolvido na Linguagem de Programação JAVA, utilizando Programação Orientada a Objetos.

1 Introdução

O Sistema de Gerenciamento de Farmácia é uma solução de *software* projetada para otimizar e gerenciar as operações diárias de uma farmácia. Ele abrange a gestão de produtos, clientes, funcionários e monitora o lucro bruto. Este documento detalha a estrutura e o *design* do sistema, com foco nos conceitos de Programação Orientada a Objetos (POO) aplicados, proporcionando uma visão aprofundada de cada componente e sua interação no sistema.

2 Descrição Geral do Sistema

O sistema é estruturado em cinco pacotes principais, cada um com responsabilidades específicas:

- **Service:** Gerencia a interação com clientes e funcionários, incluindo suas compras, detalhes e atendimentos.
- **Products:** Administra os produtos vendidos pela farmácia, incluindo detalhes como nome, preço, tipo e marca.
- **Controller:** Define interfaces para controle e monitoramento, garantindo padrões de acesso e monitoramento de lucros.

3 Detalhamento dos Pacotes e Classes

As próximas subseções apresentam o detalhamento dos pacotes e classes do sistema.

3.1 Package Service

Este pacote é responsável por gerenciar as entidades principais da farmácia.

3.1.1 Classe Cliente

Armazena informações sobre os clientes, como nome, CPF, telefone e histórico de compras. Além disso, mantém um registro do funcionário que atendeu o cliente.

```
package com.mycompany.service;
import java.util.ArrayList;
import com.mycompany.products.Produtos;

public class Cliente {

    private String nome;
    private String cpf;
    private String telefone;
    private double valorTotalCompra = 0;
    ArrayList<Produtos> compras = new ArrayList<Produtos>();
    private Funcionario atendidoPor;
    private int contaCompras = 0;

    public Cliente(String nome, String cpf, String telefone) {
        this.nome = nome;
```

```
        this.cpf = cpf;
        this.telefone = telefone;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getNome() {
        return this.nome;
    }

    public void setCPF(String cpf) {
        // Adicione validações de CPF aqui, se necessário
        this.cpf = cpf;
    }

    public String getCPF() {
        return this.cpf;
    }

    public void setTelefone(String telefone) {
        // Adicione validações de telefone aqui, se necessário
        this.telefone = telefone;
    }

    public String getTelefone() {
        return this.telefone;
    }

    public void setValorTotalCompra(double valorTotalCompra) {
        this.valorTotalCompra = this.valorTotalCompra + valorTotalCompra;
    }

    public double getValorTotalCompra() {
        return valorTotalCompra;
    }

    public void setCompras(Produtos compraCliente) {
        this.compras.add(compraCliente);
    }

    public ArrayList<Produtos> getCompras() {
        // Retornar uma cópia da lista para evitar modificações externas
        return this.compras;
    }

    public void setAtendidoPor(Funcionario input) {
        this.atendidoPor = input;
    }

    public Funcionario getAtendidoPor() {
        return this.atendidoPor;
    }

    public void setContaCompras(int input) {
        this.contaCompras += input;
    }

    public int getContaCompras() {
```

```
        return this.contaCompras;
    }
}
```

3.1.2 Classe Farmaceutico

Representa um farmacêutico, que é um tipo especializado de funcionário. Esta classe armazena informações específicas do farmacêutico e calcula seu salário com base em suas vendas e comissões.

```
package com.mycompany.service;

import com.mycompany.products.Produtos;

public class Farmaceutico extends Funcionario {

    private double salario = getSalarioBase();
    private int vendas = 0;

    public Farmaceutico(String NomeFarmacia, String CNPJ, String Endereco, String
        Telefone, String Site,
            double SalarioBase, String Nome, String CPF, String tipoFuncionario,
                int certificado) {
        super(NomeFarmacia, CNPJ, Endereco, Telefone, Site, SalarioBase, Nome, CPF
            , tipoFuncionario, certificado);
        classificandoAcesso();
    }

    public void setSalario(double input) {
        this.salario += input;
    }

    public double getSalario() {
        return this.salario;
    }

    public void setVendas(int input) {
        this.vendas += input;
        calcularComissaoMeta(this.vendas);
        super.setVendas(input);
    }

    public int getVendas() {
        return this.vendas;
    }

    @Override
    public void calcularSalarioPorProduto(Produtos produto) {
        setSalario(produto.getPreco() * 0.1);
        lucroBruto(produto.getPreco());
    }

    @Override
    public void calcularComissaoMeta(int vendas) {
        if (vendas > 6) {
            setSalario(getSalarioBase() * 0.1);
        }
    }

    @Override
```

```
    public void classificandoAcesso() {
        this.nivelAcesso = 1;
    }

    @Override
    public void lucroBruto(double valor) {
        calculandoLucroFarmacia(valor);
    }
}
```

3.1.3 Classe Farmacia

Classe abstrata que serve como base para todas as entidades relacionadas à farmácia. Define atributos comuns como nome, CNPJ, endereço, telefone e site.

```
package com.mycompany.service;

import com.mycompany.controler.painelControle;

public abstract class Farmacia implements painelControle {

    protected String Name;
    protected String CNPJ;
    protected String Endereco;
    protected String Telefone;
    protected String Site;
    protected double SalarioBase;
    protected double lucroBruto = 0;

    public Farmacia(String Name, String CNPJ, String Endereco, String Telefone,
        String Site, double SalarioBase) {
        this.Name = Name;
        this.CNPJ = CNPJ;
        this.Endereco = Endereco;
        this.Telefone = Telefone;
        this.Site = Site;
        this.SalarioBase = SalarioBase;
    }

    public String getName() {
        return Name;
    }

    public void setName(String Name) {
        this.Name = Name;
    }

    public String getEndereco() {
        return Endereco;
    }

    public void setCNPJ(String input) {
        this.CNPJ = input;
    }

    public String getCNPJ() {
        return this.CNPJ;
    }

    public void setEndereco(String Endereco) {
```

```
        this.Endereco = Endereco;
    }

    public String getTelefone() {
        return Telefone;
    }

    public void setTelefone(String Telefone) {
        this.Telefone = Telefone;
    }

    public String getSite() {
        return Site;
    }

    public void setSite(String Site) {
        this.Site = Site;
    }

    public void setSalarioBase(double input) {
        this.SalarioBase = input;
    }

    public double getSalarioBase() {
        return this.SalarioBase;
    }

    public void setLucroBruto(double input) {
        this.lucroBruto += input;
    }

    public double getLucroBruto() {
        return this.lucroBruto;
    }

}
```

3.1.4 Classe Funcionario

Classe abstrata que representa um funcionário genérico da farmácia, contendo informações como nome, CPF, tipo de funcionário e certificações.

```
package com.mycompany.service;

//Importando  a interface: painelControle + classe mãe: Produtos
import com.mycompany.products.Produtos;

public abstract class Funcionario extends Farmacia {

    protected String Nome;
    protected String CPF;
    protected String tipoFuncionario;
    protected double salario;
    protected int vendas = 0;
    protected int certificado = 0;
    public int nivelAcesso;

    public Funcionario(String NomeFarmacia, String CNPJ,
    String Endereco, String Telefone, String Site, double SalarioBase,
    String Nome, String CPF, String tipoFuncionario, int certificado) {
```

```
        super(NomeFarmacia, CNPJ, Endereco, Telefone, Site, SalarioBase);
        this.Nome = Nome;
        this.CPF = CPF;
        this.tipoFuncionario = tipoFuncionario;
        this.certificado = certificado;
    }

    public void setNome(String input) {
        this.Nome = input;
    }

    public String getNome() {
        return this.Nome;
    }

    public void setCPF(String input) {
        this.CPF = input;
    }

    public String getCPF() {
        return this.CPF;
    }

    public void setTipoFuncionario(String input) {
        this.tipoFuncionario = input;
    }

    public String getTipoFuncionario() {
        return this.tipoFuncionario;
    }

    public void setSalario(double input) {
        this.salario = input;
    }

    public double getSalario() {
        return this.salario;
    }

    public void setCertificado(int input) {
        this.certificado = input;
    }

    public int getCertificado() {
        return this.certificado;
    }

    public void calculandoLucroFarmacia(double valor) {
        super.setLucroBruto(valor);
    }

    public void setVendas(int inputVendas) {
        this.vendas += inputVendas;
    }

    public int getVendas() {
        return this.vendas;
    }

    public abstract void calcularSalarioPorProduto(Produtos input);
```

```
    public abstract void calcularComissaoMeta(int input);
}
```

3.1.5 Classe Vendedor

Representa um vendedor, que é um tipo de funcionário. Esta classe armazena informações específicas do vendedor e calcula seu salário com base em suas vendas e comissões.

```
package com.mycompany.service;

import com.mycompany.products.Produtos;

public class Vendedor extends Funcionario {

    private int vendas = 0;
    private double salario = getSalarioBase();

    public Vendedor(String NomeFarmacia, String CNPJ, String Endereco, String
        Telefone, String Site, double SalarioBase,
            String Nome, String CPF, String tipoFuncionario, int certificado) {
        super(NomeFarmacia, CNPJ, Endereco, Telefone, Site, SalarioBase, Nome, CPF
            , tipoFuncionario, certificado);
        classificandoAcesso();
    }

    public void setVendas(int input) {
        this.vendas += input;
        calcularComissaoMeta(this.vendas);
    }

    public int getVendas() {
        return this.vendas;
    }

    public void setSalario(double input) {
        this.salario += input;
    }

    public double getSalario() {
        return this.salario;
    }

    @Override
    public void calcularSalarioPorProduto(Produtos produto) {
        setSalario(produto.getPreco() * 0.1);
        lucroBruto(produto.getPreco());
    }

    @Override
    public void calcularComissaoMeta(int vendas) {
        if (vendas > 6) {
            setSalario(getSalarioBase() * 0.1);
        }
    }

    @Override
    public void classificandoAcesso() {
        this.nivelAcesso = 0;
```

```
    }

    @Override
    public void lucroBruto(double valor) {
        calculandoLucroFarmacia(valor);
    }
}
```

3.2 Package Products

Este pacote gerencia os produtos disponíveis para venda na farmácia.

3.2.1 Classe NRemedio

Representa os produtos que não são remédios, contendo informações como nome, fabricante, seção e marca.

```
package com.mycompany.products;

public class NRemedio extends Produtos {

    // Criando todas as características da classe NRemedio
    private String nomeProduto;
    private String FabricanteProduto;
    private String SecaoProduto;
    private String MarcaProduto;

    public NRemedio(String Nome, double Preco, String Tipo, String Marca,
            String FabricanteProduto, String SecaoProduto) {
        super(Nome, Preco, Tipo, Marca);
        this.nomeProduto = Nome;
        this.FabricanteProduto = FabricanteProduto;
        this.SecaoProduto = SecaoProduto;
        this.MarcaProduto = Marca;
    }

    public String getNomeProduto() {
        return this.nomeProduto;
    }

    public void setNomeProduto(String nomeProduto) {
        this.nomeProduto = nomeProduto;
    }

    public String getFabricanteProduto() {
        return this.FabricanteProduto;
    }

    public void setFabricanteProduto(String FabricanteProduto) {
        this.FabricanteProduto = FabricanteProduto;
    }

    public String getSecaoProduto() {
        return this.SecaoProduto;
    }

    public void setSecaoProduto(String SecaoProduto) {
        this.SecaoProduto = SecaoProduto;
    }

    public String getMarcaProduto() {
```

```
        return this.MarcaProduto;
    }

    public void setMarcaProduto(String MarcaProduto) {
        this.MarcaProduto = MarcaProduto;
    }

    @Override
    public double PrecoProduto() {
        return this.getPreco();
    }
}
```

3.2.2 Classe Produtos

Classe abstrata que serve como base para todos os produtos vendidos pela farmácia, definindo atributos comuns como nome, preço, tipo e marca.

```
package com.mycompany.products;

public abstract class Produtos {

    protected String Nome;
    protected double Preco;
    protected String Tipo;
    protected String Marca;

    public Produtos(String Nome, double Preco, String Tipo, String Marca) {
        this.Nome = Nome;
        this.Preco = Preco;
        this.Tipo = Tipo;
        this.Marca = Marca;
    }

    public String getNome() {
        return this.Nome;
    }

    public double getPreco() {
        return this.Preco;
    }

    public String getTipo() {
        return this.Tipo;
    }

    public String getMarca() {
        return this.Marca;
    }

    public void setNome(String Nome) {
        this.Nome = Nome;
    }

    public void setPreco(double Preco) {
        this.Preco = Preco;
    }

    public void setTipo(String Tipo) {
```

```
        this.Tipo = Tipo;
    }

    public void setMarca(String Marca) {
        this.Marca = Marca;
    }

    public abstract double PrecoProduto();
}
```

3.2.3 Classe Remedio

Representa os remédios, contendo informações adicionais como se é necessário receita, tipo de remédio, quantidade e seção que ele se encontra.

```
package com.mycompany.products;

public class Remedio extends Produtos {

    private String nomeRemedio;
    private boolean ReceitaRemedio;
    private String TipoRemedio;
    private int Quantidade;
    private String FarmaceuticaRemedio;
    private String MarcaRemedio;
    private String secaoRemedio;

    public Remedio(String Nome, double Preco, String Tipo, String Marca, boolean
        ReceitaRemedio,
            String TipoRemedio, int Quantidade, String FarmaceuticaRemedio, String
                secaoRemedio) {
        super(Nome, Preco, Tipo, Marca);
        this.nomeRemedio = Nome;
        this.ReceitaRemedio = ReceitaRemedio;
        this.TipoRemedio = TipoRemedio;
        this.Quantidade = Quantidade;
        this.FarmaceuticaRemedio = FarmaceuticaRemedio;
        this.MarcaRemedio = Marca;
        this.secaoRemedio = secaoRemedio;
    }

    public String getNomeRemedio() {
        return this.nomeRemedio;
    }

    public void setNomeRemedio(String nomeRemedio) {
        this.nomeRemedio = nomeRemedio;
    }

    public boolean getReceitaRemedio() {
        return this.ReceitaRemedio;
    }

    public void setReceitaRemedio(boolean ReceitaRemedio) {
        this.ReceitaRemedio = ReceitaRemedio;
    }

    public String getTipoRemedio() {
        return TipoRemedio;
```

```
    }

    public void setTipoRemedio(String TipoRemedio) {
        this.TipoRemedio = TipoRemedio;
    }

    public int getQuantidade() {
        return this.Quantidade;
    }

    public void setQuantidade(int Quantidade) {
        this.Quantidade = Quantidade;
    }

    public String getFarmaceuticaRemedio() {
        return FarmaceuticaRemedio;
    }

    public void setFarmaceuticaRemedio(String FarmaceuticaRemedio) {
        this.FarmaceuticaRemedio = FarmaceuticaRemedio;
    }

    public String getMarcaRemedio(String MarcaRemedio) {
        return this.MarcaRemedio;
    }

    public void setMarcaRemedio(String MarcaRemedio) {
        this.MarcaRemedio = MarcaRemedio;
    }

    public void setSecaoRemedio(String secaoRemedio) {
        this.secaoRemedio = secaoRemedio;
    }

    public String getSecaoString() {
        return this.secaoRemedio;
    }

    @Override
    public double PrecoProduto() {
        return this.getPreco();
    }

}
```

3.3 Package Controler

Este pacote define padrões de controle e monitoramento.

3.3.1 Interface painelControle

Define métodos padrão para classificar o acesso e monitorar o lucro bruto da farmácia.

```
//Painel de Controle de Monitoramente de LucroBruto e acessos da Farmácia LAV
package com.mycompany.controler;

public interface painelControle {

   public void classificandoAcesso();
```

```
    public void lucroBruto(double input);

}
```

4 Aplicação Profunda dos Conceitos de POO

Nesta seção são apresentados os principais conceitos de Programação Orientada a Objetos, aplicadas neste sistema.

4.1 Encapsulamento

O encapsulamento é evidente em todas as classes, onde os atributos são declarados como privados, garantindo que o acesso direto a esses atributos seja restrito. Os métodos públicos *get* e *set* são usados para acessar e modificar esses atributos, garantindo que os dados sejam acessados e modificados de maneira controlada e segura.

4.2 Relação de Herança

A herança é usada para criar uma relação hierárquica entre classes semelhantes, permitindo a reutilização de código e a extensão de funcionalidades. A estrutura de herança do sistema é claramente definida, com classes como Farmaceutico e Vendedor herdados de Funcionario, que por sua vez herda de Farmacia.

4.3 Classe Abstrata

As classes abstratas Farmacia, Funcionario e Produtos servem como “espinha dorsal"do sistema, fornecendo uma estrutura comum para as subclasses. Elas definem atributos e métodos que são essenciais para suas subclasses, mas não podem ser instanciadas diretamente, garantindo que apenas subclasses concretas sejam criadas.

4.4 Sobrescrita/Polimorfismo

O polimorfismo é evidenciado pela sobrescrita de métodos em subclasses. Por exemplo, as classes Farmaceutico e Vendedor sobrescrevem métodos da classe Funcionario, permitindo que cada tipo de funcionário tenha comportamentos específicos para esses métodos.

4.5 Uso da Interface

A interface painelControle é crucial para garantir padrões de controle e monitoramento. Ela define métodos que todas as classes que a implementam devem ter, garantindo consistência e padrões em todo o sistema.

4.6 Package Main

- `cadastrarFuncionarios(ArrayList<Funcionario> funcionariosFarmacia, double SalarioBase)`: Este método é usado para cadastrar os funcionários únicos da farmácia LAV. Recebe um ArrayList de objetos Funcionario e um valor double SalarioBase como parâmetros. Ele cria três objetos Funcionario (Anthony, Letízia e Vinícius) e os adiciona ao ArrayList.
- `verificandoCliente(ArrayList<Cliente> clienteFarmacia, String CPF)`: Este método recebe um ArrayList de objetos Cliente e uma string CPF como parâmetros. Ele procura por um objeto Cliente com o CPF fornecido no ArrayList e retorna verdadeiro se encontrar um, falso caso contrário.
- `cadastrandoProdutos(ArrayList<Produtos> todosProdutos)`: Este método recebe um ArrayList de objetos Produtos como parâmetro. Ele adiciona vários objetos NRemedio e Remedio ao ArrayList e o retorna.

- `Buscar(ArrayList<Cliente> clienteFarmacia, String CPF)`: Este método recebe um ArrayList de objetos Cliente e uma string CPF como parâmetros. Ele procura por um objeto Cliente com o CPF fornecido no ArrayList e retorna o seu índice se encontrar um, -1 caso contrário.
- `formatarNumeroComDuasCasasDecimais(double numero)`: Este método recebe um número double como parâmetro. Ele formata o número para ter duas casas decimais e o retorna como uma string.
- `cadastrando(Cliente clienteNovo, ArrayList<Cliente> clienteFarmacia, Object selecionarOpcao)`: Este método recebe um objeto clienteNovo do tipo Cliente, um ArrayList de objetos Cliente e um objeto selecionarOpcao como parâmetros. Ele solicita ao usuário que insira as informações do cliente e adiciona o objeto Cliente ao ArrayList. Ele retorna o objeto selecionarOpcao.
- `calculandoTotalCarrinho(Cliente clienteAtual)`: Este método recebe um objeto clienteAtual do tipo Cliente como parâmetro. Ele calcula o preço total das compras do cliente e o retorna como um valor double.
- `comprasAtualCliente(Cliente clienteAtual)`: Este método recebe um objeto clienteAtual do tipo Cliente como parâmetro. Ele retorna uma string contendo as compras do cliente.
- `buscandoNRemedio(ArrayList<Produtos> listaNRemedios, String compraNRemedio)`: Este método recebe um ArrayList de objetos Produtos e uma string compraNRemedio como parâmetros. Ele procura por um objeto NRemedio com o nome fornecido no ArrayList e o retorna.
- `buscandoRemedio(ArrayList<Produtos> listaRemedios, String comprasRemedio)`: Este método recebe um ArrayList de objetos Produtos e uma string compraRemedio como parâmetros. Ele procura por um objeto Remedio com o nome fornecido no ArrayList e o retorna.
- `main(String[] args)`: Este é o método principal do programa. Ele cria vários objetos e ArrayLists e os utiliza para simular um sistema de farmácia com uma interface de linha de comando. Ele contém várias instruções if-else que lidam com a entrada do usuário e executam diferentes ações com base nela.

4.6.1 Classe Main

```
package com.mycompany.main;
//Importando classes services
import com.mycompany.service.Funcionario;
import com.mycompany.service.Vendedor;
import com.mycompany.service.Farmaceutico;
import com.mycompany.service.Cliente;
import com.mycompany.products.NRemedio;
import com.mycompany.products.Produtos;
import com.mycompany.products.Remedio;
//Importando bibliotecas úteis
import javax.swing.JOptionPane;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Random;
public class main {
   // Método para cadastrar os funcionários únicos da Farmácia LAV
   public static void cadastrarFuncionarios(ArrayList < Funcionario >
      funcionariosFarmacia, double SalarioBase) {
      Funcionario funcionarioAnthony = new Vendedor("LAV",
         "48. 048. 138/0001-79", "Boa Esperança, Rua 8, N. 254",
         "(66) 99233-7652", "www.farmaciaLav.com.br",
         SalarioBase, "Anthony Ricardo Rodrigues Rezende",
         "072.417.431-02", "Vendedor", 0);
```

```
    funcionariosFarmacia.add(funcionarioAnthony);
    Funcionario funcionarioLetizia = new Farmaceutico("LAV",
       "48. 048. 138/0001-79", "Boa Esperança, Rua 8, N. 254",
       "(69) 98417-7172", "www.farmaciaLav.com.br",
       SalarioBase * 2, "Letízia Manuella Serqueira Eugênio",
       "012.237.441-02", "Farmacêutico", 1);
    funcionariosFarmacia.add(funcionarioLetizia);
    Funcionario funcionarioVinicius = new Vendedor("LAV",
       "48. 048. 138/0001-79", "Boa Esperança, Rua 8, N. 254",
       "(66) 99233-7652", "www.farmaciaLav.com.br",
       SalarioBase, "Vinicius Padilha Vieira",
       "022.407.431-24", "Vendedor", 0);
    funcionariosFarmacia.add(funcionarioVinicius);
}
// Funcao que busca reconhecer se há algum cpf já registrado retornando true,
    se
// verdadeiro e false se o contrário
public static boolean verificandoCliente(ArrayList < Cliente >
    clienteFarmacia, String CPF) {
    for (int i = 0; i < clienteFarmacia.size(); i++) {
       String cpfBuscado = clienteFarmacia.get(i).getCPF();
       if (cpfBuscado.equals(CPF)) {
          return true;
       }
    }
    return false;
}
// Método feito para adicionar todos os produtos do estoque da Farmácia LAV
public static ArrayList < Produtos > cadastrandoProdutos(
    ArrayList < Produtos > todosProdutos) {
    // Todos os produtos da Farmácia que se classificam como Remédios
    Produtos addProduto = new NRemedio("Fralda Supreme Care P",
       49.90, "Outros", "Huggies", "Kimberly-Clark", "Higiene"
       );
    todosProdutos.add(addProduto);
    addProduto = new NRemedio(
       "Sabonete Líquido Relaxante Hora Do Sono", 26.99,
       "Outros", "Johnson's",
       "Johnson & Johnson Industrial Ltda.", "Higiene");
    todosProdutos.add(addProduto);
    addProduto = new NRemedio(
       "Fio Dental Johnson's Reach Essencial Sabor Menta",
       12.19, "Outros", "Johnson's", "Johnson & Johnson",
       "Higiene");
    todosProdutos.add(addProduto);
    addProduto = new NRemedio(
       "Absorvente Externo Noturno Plus Noite e Dia Suave 32un",
       26.72, "Outros", "Sempre Livre",
       "Johnson & Johnson Industrial Ltda.", "Higiene");
    todosProdutos.add(addProduto);
    addProduto = new NRemedio(
       "Gatorade Frutas Cítricas Isotônico com 500ml", 9.25,
       "Outros", "Gatorade", "X3 Brasil Soluções", "Bebidas");
    todosProdutos.add(addProduto);
    addProduto = new NRemedio(
       "Sorvete Kibon Cremosíssimo Napolitano 1,5 Litro",
       29.90, "Outros", "Kibon", "Kibon", "Alimentos");
    todosProdutos.add(addProduto);
    addProduto = new NRemedio(
       "Escova de dente portátil para viagem", 3.99, "Outros",
```

```
      "Needs", "Needs", "Higiene");
    todosProdutos.add(addProduto);
    addProduto = new NRemedio(
      "Protetor Solar Facial Sun Fresh Derm Care FPS 70 Pele Mista a Oleosa",
      52.59, "Outros", "Neutrogena", "Johnson & Johnson",
      "Cosméticos");
    todosProdutos.add(addProduto);
    addProduto = new NRemedio("Hidratante Facial Vita C 50g",
      27.99, "Outros", "Needs", "Needs", "Cosméticos");
    todosProdutos.add(addProduto);
    addProduto = new NRemedio(
      "Desodorante Rexona Men Antibacterial Aerossol Antitranspirante 150ml",
      16.49, "Outros", "Rexona", "Rexona", "Higiene");
    todosProdutos.add(addProduto);
    // Todos os produtos da Farmácia que se classificam como N. Remédios
    addProduto = new Remedio("Benzetacil", 17.99, "Remédio",
      "Schering-Plough", true, "Injetável", 4,
      "Schering-Plough", "Antibióticos");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Dipirona", 5.99, "Remédio",
      "EMS S/A", false, "Comprimidos", 10, "EMS S/A",
      "Analgésicos");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Neosoro", 6.29, "Remédio",
      "Neo Química", false, "Solução Gotas", 30,
      "Neo Química", "Descongestionantes");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Rivotril", 27.01, "Remédio",
      "Roche", true, "Comprimidos", 30, "Roche",
      "Ansiolíticos");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Torsilax", 17.69, "Remédio",
      "Neo Química", false, "Comprimidos", 30, "Neo Química",
      "Analgésicos");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Isotretinoína", 82.75, "Remédio",
      "Eurofarma", true, "Cápsulas", 30, "Eurofarma",
      "Antiacne");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Ivermectina", 18.69, "Remédio",
      "Germad", false, "Comprimidos", 4, "EMS S/A",
      "Antiparasitários");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Sal de Fruta", 4.48, "Remédio",
      "GSK", false, "Em pó", 10, "GSK-Matriz", "Antiácidos");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Ciclo 21", 6.38, "Remédio",
      "União Química", false, "Comprimidos", 21,
      "União Química", "Anticoncepcionais");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Ácido Mefenâmico", 7.61, "Remédio",
      "Medley", false, "Comprimidos", 12, "Medley",
      "Anti-inflamatórios");
    todosProdutos.add(addProduto);
    addProduto = new Remedio("Resfenol", 11.35, "Remédio",
      "Kleyhertz", false, "Cápsulas", 10, "Kleyhertz",
      "Antigripais");
    return todosProdutos;
}
// Funcao para buscar e retornar um produto do tipo Remedio na lista do
```

```
    produtos
// Remédios
public static Produtos buscandoRemedio(ArrayList < Produtos >
   remedios, String nomeRemedio) {
   for (int i = 0; i < remedios.size(); i++) {
      if (remedios.get(i).getNome().equals(nomeRemedio)) {
         Produtos resultado = remedios.get(i);
         return resultado;
      }
   }
   return null;
}
// Funcao para buscar e retornar um produto do tipo N. Remedio na lista dos
// produtos N. Remedios
public static Produtos buscandoNRemedio(ArrayList < Produtos >
   Nremedios, String nomeNRemedio) {
   for (int i = 0; i < Nremedios.size(); i++) {
      if (Nremedios.get(i).getNome().equals(nomeNRemedio)) {
         Produtos resultado = Nremedios.get(i);
         return resultado;
      }
   }
   return null;
}
// Funcao para concatenar e retornar a String com os nomes de todos os produtos
// selecionados para comprar pelo cliente
public static String comprasAtualCliente(Cliente cliente) {
   ArrayList < Produtos > produtosCliente = cliente
.getCompras();
   String listaStringProdutos = "";
   for (int i = cliente.getContaCompras(); i < produtosCliente
      .size(); i++) {
      listaStringProdutos += "-> " + produtosCliente.get(i)
         .getNome() + "\n";
   }
   return listaStringProdutos;
}
// Função para calcular e retornar a soma de todos os produtos que estão no
// carrinho do cliente
public static double calculandoTotalCarrinho(Cliente cliente) {
   ArrayList < Produtos > produtosCliente = cliente
.getCompras();
   double valorTotal = 0;
   for (int i = cliente.getContaCompras(); i < produtosCliente
      .size(); i++) {
      valorTotal += produtosCliente.get(i).getPreco();
   }
   return valorTotal;
}
// Calculando o salário dos Funcionários da Farmácia LAV
public static void calculaSalarioFuncionario(
   Funcionario funcionario, ArrayList < Funcionario >
   farmaciaFuncionarios, Cliente cliente, int flag) {
   ArrayList < Produtos > todosProdutos = cliente.getCompras();
   int i;
   for (i = 0; i < farmaciaFuncionarios.size(); i++) {
      if (funcionario.getCPF().equals(farmaciaFuncionarios.get(
            i).getCPF())) {
         break;
      }
```

```
    }
    if (flag == 0) {
       for (int j = cliente.getContaCompras(); j < todosProdutos
          .size(); j++) {
          farmaciaFuncionarios.get(i).calcularSalarioPorProduto(
             todosProdutos.get(j));
       }
    } else {
       if (funcionario.getTipoFuncionario().equals(
             "Farmacêutico")) {
          for (int j = cliente.getContaCompras(); j <
             todosProdutos.size(); j++) {
             if (todosProdutos.get(j).getTipo().equals(
                "Remédio")) {
                if (((Remedio) todosProdutos.get(j))
                   .getTipoRemedio().equals("Injetável")) {
                   farmaciaFuncionarios.get(1)
                      .calcularSalarioPorProduto(todosProdutos
                         .get(j));
                }
             }
          }
       } else {
          for (int j = cliente.getContaCompras(); j <
             todosProdutos.size(); j++) {
             if (todosProdutos.get(j).getTipo().equals(
                "Remédio")) {
                if (((Remedio) todosProdutos.get(j))
                   .getTipoRemedio() != "Injetável") {
                   farmaciaFuncionarios.get(i)
                      .calcularSalarioPorProduto(todosProdutos
                         .get(j));
                }
             } else if ((todosProdutos.get(j).getTipo().equals(
                   "Outros"))) {
                farmaciaFuncionarios.get(i)
                   .calcularSalarioPorProduto(todosProdutos.get(
                      j));
             }
          }
       }
    }
 }
 // Inserindo o número de vendas
 public static void inserirVendas(Funcionario atendente, ArrayList <
    Funcionario > farmaciaFuncionarios, int vendas) {
    int i;
    for (i = 0; i < farmaciaFuncionarios.size(); i++) {
       if (atendente.getNome().equals(farmaciaFuncionarios.get(i)
             .getNome())) {
          break;
       }
    }
    if (farmaciaFuncionarios.get(i).getTipoFuncionario().equals(
          "Vendedor")) {
       ((Vendedor) farmaciaFuncionarios.get(i)).setVendas(
       vendas);
    } else if (farmaciaFuncionarios.get(i).getTipoFuncionario()
       .equals("Farmacêutico")) {
       ((Farmaceutico) farmaciaFuncionarios.get(i)).setVendas(
```

```
            vendas);
      }
   }
   public static void alterarCarrinho(String NomeProduto, ArrayList <
      Produtos > produtosCliente) {
      for (int i = 0; i < produtosCliente.size(); i++) {
         if (produtosCliente.get(i).getTipo().equals("Remédio")) {
            if (((Remedio) produtosCliente.get(i)).getNome()
               .equals(NomeProduto)) {
               produtosCliente.remove(i);
               break;
            }
         } else {
            if (((NRemedio) produtosCliente.get(i)).getNome()
               .equals(NomeProduto)) {
               produtosCliente.remove(i);
               break;
            }
         }
      }
   }
   // Criando o relatório de recibo por cliente, no qual está persistindo no
   // arquivo relatorioClienteRecibo.txt
   public static String relatorioPorCliente(Cliente cliente,
      ArrayList < Produtos > produtosCliente) {
      String relatorioCliente = "Cliente \nCPF: " + cliente
      .getCPF() + "\n\n** Produtos Comprados **\n\n";
      for (int i = 0; i < produtosCliente.size(); i++) {
         relatorioCliente += "Nome: " + produtosCliente.get(i)
            .getNome() + "\nTipo: " + produtosCliente.get(i)
            .getTipo() + "\nMarca: " + produtosCliente.get(i)
            .getMarca() + "\nPreço: " + produtosCliente.get(i)
            .getPreco() + "\n\n";
      }
      relatorioCliente += "Valor Total da Compra: " +
         formatarNumeroComDuasCasasDecimais(cliente
            .getValorTotalCompra()) +
         "\n\nObrigado pela Preferência\nVolte sempre, atenciosamente Farmácia LAV
             ";
      return relatorioCliente;
   }
   // Funcao para retornar relatório do tipo String já adequadamente formatado
   public static String relatorioFinalDiaFarmacia(ArrayList <
      Cliente > clienteFarmacia, ArrayList < Funcionario >
      Funcionarios) {
      double valorTotalDesconhecidos = 0;
      ////////////////////////// Relatório Funcionários //////////////
      String relatorio = "** Relatório de Funcionários **\n\n";
      double lucroBruto = 0;
      for (int i = 0; i < Funcionarios.size(); i++) {
         relatorio = relatorio + "Nome do funcionário: " +
            Funcionarios.get(i).getNome() + "\n" + "Cargo: " +
            Funcionarios.get(i).getTipoFuncionario() + "\n";
         if (Funcionarios.get(i).getTipoFuncionario().equals(
               "Vendedor")) {
            relatorio = relatorio + "Número de Vendas: " + ((
                  Vendedor) Funcionarios.get(i)).getVendas() +
               "\n" + "Taxa de Comissão (%) por Produto: " + "10" +
               "\n" +
               "Taxa de Comissão por meta (%) sob Salário Base: " +
```

```
                ”10” + ”\n” + ”Salário (R$): ” +
                formatarNumeroComDuasCasasDecimais(Funcionarios.get(
                    i).getSalario()) + ”\n”;
            } else if (Funcionarios.get(i).getTipoFuncionario()
                .equals(”Farmacêutico”)) {
                relatorio = relatorio + ”Número de Vendas: ” + ((
                      Farmaceutico) Funcionarios.get(i)).getVendas() +
                    ”\n” + ”Taxa de Comissão (%) por Produto: ” + ”10” +
                    ”\n” +
                    ”Taxa de Comissão por meta (%) sob Salário Base: ” +
                    ”10” + ”\n” + ”Salário (R$): ” +
                    formatarNumeroComDuasCasasDecimais(Funcionarios.get(
                        i).getSalario()) + ”\n”;
            }
            // Calculando o lucroBruto
            lucroBruto += Funcionarios.get(i).getLucroBruto();
            relatorio = relatorio + ”\n”;
        }
        ////////////////////////// Relatório clientes
            ////////////////////////////////
        relatorio = relatorio + ”\n** Relatório de Clientes **\n\n”;
        for (Cliente C: clienteFarmacia) {
            ArrayList < Produtos > todasCompras = C.getCompras();
            if (C.getNome() != ”Desconhecido”) {
                relatorio = relatorio + ”Nome: ” + C.getNome() + ”\n” +
                    ”CPF: ” + C.getCPF() + ”\n” +
                    ”Quantidade de compras: ” + todasCompras.size() +
                    ”\n” + ”Valor total de compras (R$): ” +
                    formatarNumeroComDuasCasasDecimais(C
                        .getValorTotalCompra()) + ”\n\n”;
            } else {
                valorTotalDesconhecidos += C.getValorTotalCompra();
            }
        }
        relatorio = relatorio + ”Valor total de não cadastrados: ” +
            formatarNumeroComDuasCasasDecimais(
                valorTotalDesconhecidos);
        relatorio +=
            ”\n\n** Lucro Bruto Farmácia LAV **\n\nValor Total: ” +
            formatarNumeroComDuasCasasDecimais(lucroBruto);
        return relatorio;
    }
    // Função para persistir as informações do relatorio em um arquivo ->
    // ”relatorio.txt”
    public static void escrevendoArquivoFarmacia(
        String relatorioFarmacia) {
        try {
            BufferedWriter writer = new BufferedWriter(new FileWriter(
                ”projetoFarmacia/src/main/java/com/mycompany/arquivo/relatorioFarmacia
                    .txt”
                ));
            writer.write(relatorioFarmacia);
            writer.close();
            System.out.println(
                ”String relatorioFarmacia exportada com sucesso para o arquivo.”
                );
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

```
public static void escrevendoReciboCliente(
   String relatorioClienteRecibo) {
   try {
      BufferedWriter writer = new BufferedWriter(new FileWriter(
         ”projetoFarmacia/src/main/java/com/mycompany/arquivo/
             relatorioClienteRecibo.txt”
         ));
      writer.write(relatorioClienteRecibo);
      writer.close();
      System.out.println(
         ”String relatorioClienteRecibo exportada com sucesso para o arquivo.”
         );
   } catch (IOException e) {
      e.printStackTrace();
   }
}
// Função de Busca da posição dos clientes em função de seu respectivo CPF
public static int Buscar(ArrayList < Cliente > clienteFarmacia,
   String CPF) {
   for (int i = 0; i < clienteFarmacia.size(); i++) {
      if (clienteFarmacia.get(i).getCPF().equals(CPF)) {
         return i;
      }
   }
   return -1;
}
// Funcao para formatar os valores vindos de cálculo de salário e comissões em
// até 2 casas decimais
public static String formatarNumeroComDuasCasasDecimais(
   double numero) {
   DecimalFormat formato = new DecimalFormat(”0.00”);
   return formato.format(numero);
}
public static Object cadastrando(Cliente clienteNovo, ArrayList <
   Cliente > clienteFarmacia, Object selecionarOpcao) {
   // Cadastrando os Clientes
   JOptionPane.showMessageDialog(null,
      ”** Bem-Vindo ** \n\n Precisamos de algumas Informações \n\n CPF:\n Nome
          :\n Telefone:”
      );
   String cpfCliente = JOptionPane.showInputDialog(
      ”Digite o seu CPF: ”);
   // Verificando se o cpf do cliente atual já foi cadastrado
   if (verificandoCliente(clienteFarmacia, cpfCliente) &&
      clienteFarmacia.get(Buscar(clienteFarmacia, cpfCliente))
      .getNome() != ”Desconhecido”) {
      JOptionPane.showMessageDialog(null,
         ”Bem-vindo novamente !!”);
   } else {
      // Caso o cliente não esteja cadastrado...
      String nomeCliente = JOptionPane.showInputDialog(
         ”Digite o seu Nome: ”);
      String telCliente = JOptionPane.showInputDialog(
         ”Digite o seu Telefone: ”);
      JOptionPane.showMessageDialog(null,
         ”** Informações Passadas **\n\nCPF: ” + cpfCliente +
         ”\nNome: ” + nomeCliente + ”\nTelefone: ” +
         telCliente);
      int resposta = JOptionPane.showConfirmDialog(null,
         ”Você deseja alterar os dados ?”, ”Confirmação”,
```

```
            JOptionPane.YES_NO_OPTION);
         if (resposta == JOptionPane.YES_OPTION) {
            Object[] opcaoCliente = {
               "CPF",
               "Nome",
               "Telefone",
               "Tudo"
            };
            // Perguntando se o cliente-usuário deseja alterar algo
            Object selecionaOpcaoCliente = JOptionPane
               .showInputDialog(null, "** Alteração de Dados **",
                  "Opção", JOptionPane.INFORMATION_MESSAGE, null,
                  opcaoCliente, null);
            // Todas as opções de alteração de Dados
            if (selecionaOpcaoCliente.equals("CPF")) {
               cpfCliente = JOptionPane.showInputDialog(
                  "Digite o seu CPF: ");
            } else if (selecionaOpcaoCliente.equals("Nome")) {
               nomeCliente = JOptionPane.showInputDialog(
                  "Digite o seu Nome: ");
            } else if (selecionaOpcaoCliente.equals("Telefone")) {
               telCliente = JOptionPane.showInputDialog(
                  "Digite o seu Telefone: ");
            } else if (selecionaOpcaoCliente.equals("Tudo")) {
               cpfCliente = JOptionPane.showInputDialog(
                  "Digite o seu CPF: ");
               nomeCliente = JOptionPane.showInputDialog(
                  "Digite o seu Nome: ");
               telCliente = JOptionPane.showInputDialog(
                  "Digite o seu Telefone: ");
            }
         }
         // Inserindo todos as informações na classe cliente instanciado como
             clienteNovo
         clienteNovo.setCPF(cpfCliente);
         clienteNovo.setNome(nomeCliente);
         clienteNovo.setTelefone(telCliente);
         // Verificando se o CPF já foi cadastrado
         if (verificandoCliente(clienteFarmacia, cpfCliente)) {
            int posCliente = Buscar(clienteFarmacia, cpfCliente);
            clienteFarmacia.set(posCliente, clienteNovo);
         } else {
            clienteFarmacia.add(clienteNovo);
         }
      }
      return selecionarOpcao = "Comprar";
      /////////// Caso o usuário clique em "Comprar" vai entrar aqui ///////////
   }
   public static void comprando(Funcionario atendente,
      Funcionario auxAtendente, ArrayList < Funcionario >
      farmaciaFuncionarios, ArrayList < Cliente > clienteFarmacia,
      int comprasVendedor, int comprasFarmaceutico,
      Cliente clienteNovo, ArrayList < Produtos > produtosFarmacia,
      int respostaFecharCarrinhoCliente, int flag, Random random) {
      // Mostrando que o atendimento foi direcionado a um dos funcionários da farm
          ácia
      // LAV
      atendente = farmaciaFuncionarios.get(random.nextInt(3));
      JOptionPane.showMessageDialog(null,
         "Atendimento com:\n\n Nome: " + atendente.getNome() +
```

```
   "\n Cargo: " + atendente.getTipoFuncionario());
// Pedindo CPF do cliente-usuário, para verificar se já tem cadastro ou não
String cpf = JOptionPane.showInputDialog("Passe o seu CPF:");
if (!verificandoCliente(clienteFarmacia, cpf)) {
   comprasVendedor = 0;
   comprasFarmaceutico = 0;
}
while (true) {
   boolean receitaCliente = true;
   // Mostrando as opções Remédio e não remédio
   if (Buscar(clienteFarmacia, cpf) == -1) {
      clienteNovo.setCPF(cpf);
      clienteNovo.setNome("Desconhecido");
      clienteFarmacia.add(clienteNovo);
   }
   String[] opcaoCompra = {
      "Remédios",
      "Outros"
   };
   String compraSelecionada = (String) JOptionPane
      .showInputDialog(null, "Escolha uma opção:",
         "Menu Compras - LAV", JOptionPane.QUESTION_MESSAGE,
         null, opcaoCompra, opcaoCompra[0]);
   /////////// Caso o usuário clique em Remédios //////////
   if (compraSelecionada.equals("Remédios")) {
      ArrayList < Produtos > listaRemedios = new ArrayList <
         Produtos > ();
      int countRemedios = 0;
      // Coletando Produtos do Tipo Remédio
      for (int i = 0; i < produtosFarmacia.size(); i++) {
         if (produtosFarmacia.get(i).getTipo().equals(
               "Remédio")) {
            listaRemedios.add(((Remedio) produtosFarmacia
               .get(i)));
            countRemedios++;
         }
      }
      Object[] nomesRemedios = new Object[countRemedios];
      for (int i = 0; i < countRemedios; i++) {
         nomesRemedios[i] = ((Remedio) listaRemedios.get(i))
            .getNomeRemedio();
      }
      Produtos produtoCliente = null;
      // Interface para selecionar o produtos, dentro do estoque da Farmácia
      // LAV
      String compraRemedio = (String) JOptionPane
         .showInputDialog(null, "Escolha uma opção:",
            "Menu Compras - LAV", JOptionPane
            .QUESTION_MESSAGE, null, nomesRemedios,
            nomesRemedios[0]);
      // Caso algum produto, seja selecionado seguimos...
      if (compraRemedio != "") {
         produtoCliente = buscandoRemedio(listaRemedios,
            (String) compraRemedio);
         boolean receita = ((Remedio) produtoCliente)
            .getReceitaRemedio();
         if (receita) {
            int resposta = JOptionPane.showConfirmDialog(
               null, "Você possui Receita ?",
               "Confirmação", JOptionPane.YES_NO_OPTION);
```

```
            if (resposta != JOptionPane.YES_OPTION) {
                JOptionPane.showMessageDialog(null,
                    "Para comprar esse remédio,\n você precisa de receita"
                    );
                receitaCliente = false;
            }
        }
        if (receitaCliente) {
            // Usando o atributo nível de acesso que vem do implements
            // painel de controle
            // para classificar se o atendente tem acesso para aplicar
            // injetáveis ou não
            if (((Remedio) produtoCliente).getTipoRemedio()
                .equals("Injetável")) {
                if (atendente.nivelAcesso == 0) {
                    auxAtendente = farmaciaFuncionarios.get(1);
                    JOptionPane.showMessageDialog(null,
                        "Vamos te encaminhar para a farmacêutica\npara realizar
                             a aplicação do injetável\n\nFarmacêutica: " +
                        auxAtendente.getNome());
                    flag = 1;
                }
            }
            if (atendente.getNome().equals(
                    "Letízia Manuella Serqueira Eugênio")) {
                comprasFarmaceutico++;
            } else {
                comprasVendedor++;
            }
        }
    }
    if (receitaCliente) {
        // Interface, perguntando ao usuário cliente se ele desej a
            adicionar ao
        // carrinho ou não
        int respostaClienteProduto = JOptionPane
            .showConfirmDialog(null,
                "Deseja colocar no carrinho ?\n\nNome: " +
                produtoCliente.getNome() + "\nMarca: " +
                produtoCliente.getMarca() + "\nPreço: " +
                produtoCliente.getPreco() + "\n",
                "Sobre o Produto", JOptionPane.YES_NO_OPTION);
        // Caso deseje adicionar, prosseguimos por aqui
        if (respostaClienteProduto == JOptionPane
            .YES_OPTION) {
            clienteFarmacia.get(Buscar(clienteFarmacia, cpf))
                .setCompras(produtoCliente);
            double valorTotal = calculandoTotalCarrinho(
                clienteFarmacia.get(Buscar(clienteFarmacia,
                    cpf)));
            String comprasAtualCliente = comprasAtualCliente(
                clienteFarmacia.get(Buscar(clienteFarmacia,
                    cpf)));
            JOptionPane.showMessageDialog(null,
                "** Seu carrinho **\nValor Total: " +
                formatarNumeroComDuasCasasDecimais(
                    valorTotal) + "\n\n" +
                comprasAtualCliente);
        }
    }
```

```
      ////////// Caso o usuário selecione Outros vai entrar aqui
          //////////
   } else if (compraSelecionada.equals("Outros")) {
      if (atendente.getNome().equals(
            "Letízia Manuella Serqueira Eugênio")) {
         comprasFarmaceutico++;
      } else {
         comprasVendedor++;
      }
      // Coletando os produtos do tipo "Outros" em um arrayList
      ArrayList < Produtos > listaNRemedios = new ArrayList <
         Produtos > ();
      int countNRemedios = 0;
      for (int i = 0; i < produtosFarmacia.size(); i++) {
         if (produtosFarmacia.get(i).getTipo().equals(
               "Outros")) {
            listaNRemedios.add(
               ((NRemedio) produtosFarmacia.get(i)));
            countNRemedios++;
         }
      }
      // Preparando o object, no qual a interface vai precisar para mostrar
          o
      // leque de
      // produtos do tipo "Outros"
      Object[] nomesNRemedios = new Object[countNRemedios];
      for (int i = 0; i < countNRemedios; i++) {
         nomesNRemedios[i] = ((NRemedio) listaNRemedios.get(
            i)).getNomeProduto();
      }
      Produtos produtoCliente = null;
      // Mostrando a interface com os produtos do tipo "Outros"
      String compraNRemedio = (String) JOptionPane
         .showInputDialog(null, "Escolha uma opção:",
            "Menu Compras - LAV", JOptionPane
            .QUESTION_MESSAGE, null, nomesNRemedios,
            nomesNRemedios[0]);
      // Caso algum produto seja selecionado, prosseguimos...
      if (compraNRemedio != "") {
         produtoCliente = buscandoNRemedio(listaNRemedios,
            (String) compraNRemedio);
      }
      // Mostrando as caracteríticas do produto do tipo "Outros" que foi
      // selecionado + perguntando se deseja adicioná-lo no carrinho
      int respostaClienteProduto = JOptionPane
         .showConfirmDialog(null,
            "Deseja colocar no carrinho ?\n\nNome: " +
            produtoCliente.getNome() + "\nMarca: " +
            produtoCliente.getMarca() + "\nPreço: " +
            produtoCliente.getPreco() + "\n",
            "Sobre o Produto", JOptionPane.YES_NO_OPTION);
      // Caso o usuário-cliente desejar, prosseguimos
      if (respostaClienteProduto == JOptionPane.YES_OPTION) {
         clienteFarmacia.get(Buscar(clienteFarmacia, cpf))
            .setCompras(produtoCliente);
         double valorTotal = calculandoTotalCarrinho(
            clienteFarmacia.get(Buscar(clienteFarmacia,
               cpf)));
         String comprasAtualCliente = comprasAtualCliente(
            clienteFarmacia.get(Buscar(clienteFarmacia,
```

```
            cpf)));
         JOptionPane.showMessageDialog(null,
            "** Seu carrinho **\nValor Total: " +
            formatarNumeroComDuasCasasDecimais(
            valorTotal) + "\n\n" + comprasAtualCliente);
      }
   }
   // Perguntando ao cliente-usuário se deseja continuar comprando
   int resposta = JOptionPane.showConfirmDialog(null,
      "Deseja continuar comprando ?", "Confirmação",
      JOptionPane.YES_NO_OPTION);
   // Caso sim, prosseguimos
   if (resposta != JOptionPane.YES_OPTION) {
      if (clienteNovo.getCompras() != null) {
         String comprasAtualCliente = "";
         if (Buscar(clienteFarmacia, cpf) != -1) {
            comprasAtualCliente = comprasAtualCliente(
               clienteFarmacia.get(Buscar(clienteFarmacia,
                  cpf)));
         } else {
            comprasAtualCliente = comprasAtualCliente(
               clienteNovo);
         }
         // Mostrando tudo que há no carrinho do usuário-cliente até o
         // momento
         int respostaFecharCarrinho = JOptionPane
            .showConfirmDialog(null,
               "** No seu carrinho tem: **\n\n" +
               comprasAtualCliente +
               "\n\n Deseja fechar o carrinho ?",
               "Confirmação", JOptionPane.YES_NO_OPTION);
         if (respostaFecharCarrinho != JOptionPane
            .YES_OPTION) {
            respostaFecharCarrinhoCliente = 0;
         } else {
            respostaFecharCarrinhoCliente = 1;
         }
         // Se o cliente-usuário clicar em sim...
         if (respostaFecharCarrinho == JOptionPane
            .YES_OPTION) {
            // Analisando se u usuário-cliente quer alterar o
            // carrinho ?
            int editarCarrinho = JOptionPane
               .showConfirmDialog(null,
                  "Antes de fechar, deseja editar o carrinho ?",
                  "Confirmação", JOptionPane.YES_NO_OPTION);
            while (true) {
               if (editarCarrinho == JOptionPane
                  .YES_OPTION) {
                  int tamanhoCarrinho = clienteNovo
                     .getCompras().size();
                  String[] nomesProdutosCarrinho =
                     new String[tamanhoCarrinho];
                  for (int i = 0; i < tamanhoCarrinho; i++) {
                     Produtos produtoAtual = clienteNovo
                        .getCompras().get(i);
                     nomesProdutosCarrinho[i] = produtoAtual
                        .getNome();
                  }
                  // Mostrando a interface com os produtos
```

```
            // do tipo
            // "Outros"
            String selecionaProdutoCarrinho = (
               String) JOptionPane.showInputDialog(
                  null,
                  "Escolha um produto para retirar:",
                  "Menu de Alteração de Carrinho",
                  JOptionPane.QUESTION_MESSAGE, null,
                  nomesProdutosCarrinho,
                  nomesProdutosCarrinho[0]);
            if (selecionaProdutoCarrinho != "") {
               alterarCarrinho(
                  selecionaProdutoCarrinho,
                  clienteNovo.getCompras());
            }
            // Retirando uma venda do funcionário
            // que atendeu o cliente
            if (auxAtendente != null) {
               if (selecionaProdutoCarrinho.equals(
                     "Benzetacil")) {
                  auxAtendente.setVendas(auxAtendente
                     .getVendas() - 1);
               } else {
                  atendente.setVendas(atendente
                     .getVendas() - 1);
               }
            } else {
               atendente.setVendas(atendente
               .getVendas() - 1);
            }
            // Perguntando ao cliente-usuário se ele
            // deseja continuar a alteração
            int continuarEditando = JOptionPane
               .showConfirmDialog(null,
                  "Deseja continuar alterando ?",
                  "Confirmação", JOptionPane
                  .YES_NO_OPTION);
            if (continuarEditando != JOptionPane
               .YES_OPTION) {
               break;
            }
         } else {
            break;
         }
      }
      double valorTotal = calculandoTotalCarrinho(
         clienteFarmacia.get(Buscar(clienteFarmacia,
            cpf)));
      if (verificandoCliente(clienteFarmacia, cpf) &&
         clienteFarmacia.get(Buscar(clienteFarmacia,
            cpf)).getNome() != "Desconhecido") {
         // Caso o usuário-cliente tenha cadastro, vamos
         // mostrar essa tela
         JOptionPane.showMessageDialog(null,
            "[Pagamento] - Você tem Cadastro.\n\nValor Total: R$" +
            formatarNumeroComDuasCasasDecimais(
               valorTotal) +
            "\nCom desconto de 10%: R$" +
            formatarNumeroComDuasCasasDecimais(
               valorTotal * 0.9));
```

```
            clienteFarmacia.get(Buscar(clienteFarmacia,
               cpf)).setValorTotalCompra(valorTotal *
               0.9);;
         } else {
            // Caso o usuário-cliente não tenha cadastro,
            // vamos mostrar essa tela
            JOptionPane.showMessageDialog(null,
               "[Pagamento] - Você não tem Cadastro.\n\nValor Total: R$"
                  +
               formatarNumeroComDuasCasasDecimais(
                  valorTotal));
            clienteFarmacia.get(Buscar(clienteFarmacia,
               cpf)).setValorTotalCompra(valorTotal);
         }
         if (clienteNovo.getCompras().size() > 0 &&
            clienteNovo.getValorTotalCompra() > 0) {
            // Mostrando o recibo de compra e pagamento do cliente, de
            // acordo com o
            // seu carrinho
            String relatorioCliente = relatorioPorCliente(
               clienteNovo, clienteNovo.getCompras());
            JOptionPane.showMessageDialog(null,
               relatorioCliente);
            escrevendoReciboCliente(relatorioCliente);
            // Caso o atendimento de um vendedor precisou passar
            // alguma aplicação para
            // a farmacêutica Letizia flag será 1, logo prosseguimos
            // por aqui... e isso mudará o funcionamento da Função
            // calculaSalarioFuncionario
            // Inserindo vendas + calculando salário
            if ((flag == 1) && (atendente
                  .getTipoFuncionario() != "Farmacêutico"
                  )) {
               comprasFarmaceutico++;
               comprasVendedor--;
               inserirVendas(auxAtendente,
                  farmaciaFuncionarios,
                  comprasFarmaceutico);
               calculaSalarioFuncionario(auxAtendente,
                  farmaciaFuncionarios, clienteFarmacia
                  .get(Buscar(clienteFarmacia, cpf)),
                  flag);
            }
            // Inserindo vendas + calculando salário
            if (atendente.getTipoFuncionario().equals(
                  "Vendedor")) {
               inserirVendas(atendente,
                  farmaciaFuncionarios, comprasVendedor
                  );
            } else if (atendente.getTipoFuncionario()
               .equals("Farmacêutico")) {
               inserirVendas(atendente,
                  farmaciaFuncionarios,
                  comprasFarmaceutico);
            }
            calculaSalarioFuncionario(atendente,
               farmaciaFuncionarios, clienteFarmacia
               .get(Buscar(clienteFarmacia, cpf)), flag
               );
            // Abaixo estamos "setando" o número de compras
```

```
                    // realizadas com os funcionários, seja do
                    // tipo vendedor ou do tipo farmacêutico
                    if (atendente.getTipoFuncionario().equals(
                          "Vendedor")) {
                       clienteFarmacia.get(Buscar(clienteFarmacia,
                          cpf)).setContaCompras(
                          comprasVendedor);
                    } else {
                       clienteFarmacia.get(Buscar(clienteFarmacia,
                          cpf)).setContaCompras(
                          comprasFarmaceutico);
                    }
                 }
              }
           }
           if (respostaFecharCarrinhoCliente == 1) {
              break;
           }
        }
     }
  }
  public static void main(String[] args) {
     // Criando a flag para validacao de casos de aplicação de injetáveis
     int flag = 0;
     // Lista para incluir Todos os produtos
     ArrayList < Produtos > produtosFarmacia = new ArrayList <
        Produtos > ();
     // Lista para incluir todos os clientes atendidos na farmácia
     ArrayList < Cliente > clienteFarmacia = new ArrayList <
        Cliente > ();
     // Coletando informações sobre a Farmacia
     String NomeFarmacia = "LAV";
     String CNPJ = "48. 048. 138/0001-79";
     String Telefone = "(66) 99233-7652";
     String Endereco = "Boa Esperança, Rua 8, N. 254";
     String Site = "www.farmacialav.com.br";
     double SalarioBase = 1300;
     // Lista para incluir os funcionários da Farmácia
     ArrayList < Funcionario > farmaciaFuncionarios =
        new ArrayList < Funcionario > ();
     cadastrarFuncionarios(farmaciaFuncionarios, SalarioBase);
     // Cadastrando Todos os Produtos - 10 Remédios & 10 N. Remédios
     produtosFarmacia = cadastrandoProdutos(produtosFarmacia);
     while (true) {
        // Armazennando a flag para permitir continuidade do carrinho
        int respostaFecharCarrinhoCliente = 0;
        // Armazenar a quantidade de compras efetuadas pelo cliente
        int comprasVendedor = 0;
        int comprasFarmaceutico = 0;
        // Atualizando o flag para 0 novamente, pois receberemos um novo cliente;
        flag = 0;
        // Instanciano uma var random, pois vamos preciÇsar para randomizar o
        // atendimento do cliente, de acordo com os funcionários fixos
        Random random = new Random();
        Funcionario auxAtendente = null;
        Funcionario atendente = null;
        // Instanciando Cliente + Definição e Informações
        Cliente clienteNovo = new Cliente("", "", "");
        JOptionPane.showMessageDialog(null,
           "** Farmácia LAV **\n\n Nome: " + NomeFarmacia +
```

```
            "\n CNPJ: " + CNPJ + "\n Telefone: " + Telefone +
            "\n Endereço: " + Endereco + "\n Site: " + Site +
            "\n\n [ FARMÁCIA ABERTA ]");
        JOptionPane.showMessageDialog(null,
            "[AVISO] Para garantir desconto de 10%, realize o seu cadastro."
            );
        Object[] opcao = {
            "Cadastrar",
            "Comprar",
            "Sair"
        };
        // Mostrando o *Menu LAV*, onde o cliente-usuário decide o que fazer (
            comprar,
        // cadastrar, sair)
        Object selecionarOpcao = JOptionPane.showInputDialog(null,
            "** Menu LAV **", "Opção", JOptionPane
            .INFORMATION_MESSAGE, null, opcao, null);
        // Apresentando as Informções que os Clientes devem inserir
        if (selecionarOpcao.equals("Cadastrar")) {
            // retorno porque ao cadastrar quero encaminha o usuário-cliente para
                as
            // compras, logo
            // devo alterar a variavell selecionarOpcao
            selecionarOpcao = cadastrando(clienteNovo,
                clienteFarmacia, selecionarOpcao);
        }
        if (selecionarOpcao.equals("Comprar")) {
            comprando(atendente, auxAtendente,
                farmaciaFuncionarios, clienteFarmacia,
                comprasVendedor, comprasFarmaceutico,
                clienteNovo, produtosFarmacia,
                respostaFecharCarrinhoCliente, flag, random);
        } else if (selecionarOpcao.equals("Sair")) {
            // relatorio Farmacia [relatório cliente + relatório funcionário +
                lucroBruto]
            String relatorioFarmacia = relatorioFinalDiaFarmacia(
                clienteFarmacia, farmaciaFuncionarios);
            // Adicionando relatório no arquivo, a informação vai persistir em
            // "relatorio.txt", caso haja algo escrito ele limpa o arquivo e
                escreve
            // novamente
            escrevendoArquivoFarmacia(relatorioFarmacia);
            // Toto o relatório geral da Farmácia LAV foi encaminhado para o
                arquivo
            // relatorioFarmacia
            // O break abaixo é usado para parar o programa totalmente;
            break;
        }
    }
  }
}
```

5 Resumo dos Recursos Utilizados

Recursos

O programa também inclui várias outras características, tais como:

- Uma interface de linha de comando com diversas opções, tais como o registro de clientes, adição de produtos ao carrinho e cálculo do preço total das compras;

- Utilização de *ArrayLists* para armazenar objetos de diferentes classes;
- Utilização de estruturas de controle *if-else* para lidar com a entrada do usuário e realizar diferentes ações com base nela;
- Utilização de vários métodos para realizar diferentes tarefas, como o registro de funcionários e clientes, adição de produtos ao carrinho e cálculo do preço total das compras;
- Utilização de diversas classes que seguem os princípios da Programação Orientada a Objetos (POO), como herança, polimorfismo e encapsulamento.

6 Conclusão:

O Sistema de Gerenciamento de Farmácia é um programa criado usando conceitos da Programação Orientada a Objetos (POO) aprendidos no decorrer da disciplina de Algoritmos III. Tais conceitos foram utilizados para tornar o sistema forte, organizado e fácil de expandir.