INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS – CAMPUS SÃO JOÃO EVANGELISTA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DÁLISSON ANDRÉ DA SILVA ALVES

DEBI JUNIA DE PAULA

**TRABALHO PRÁTICO I**

SÃO JOÃO EVANGELISTA

2022

DÁLISSON ANDRÉ DA SILVA ALVES

DEBI JUNIA DE PAULA

**SISTEMA DE CONTROLE DE VENDAS: PAPELARIA**

SÃO JOÃO EVANGELISTA

2022

SUMÁRIO

[1. INTRODUÇÃO 4](#_Toc121949820)

[1.1. Objetivo Geral 4](#_Toc121949821)

[1.2. Objetivos Específicos 4](#_Toc121949822)

[1.3. Justificativa 5](#_Toc121949823)

[2. DESENVOLVIMENTO 5](#_Toc121949824)

[2.1. Swing 5](#_Toc121949825)

[2.2. Classe DAO 7](#_Toc121949826)

[2.3. Implementação 8](#_Toc121949827)

[3. CONCLUSÃO 10](#_Toc121949828)

# INTRODUÇÃO

Aprendemos a ler e a escrever quando entramos na escola, no jardim de infância, e a partir desse ponto inicial, aprendemos que tudo na vida existe perspectivas, tem uma razão e um motivo para sua existência.

O trabalho proposto que vai ser apresentado no decorrer deste, apresenta uma aplicação na estrutura de criação de um sistema para uma papelaria, que atenda no cadastro e lançamento de dados através das linguagens Java e SQL.

Java é uma linguagem de programação e plataforma de computação liberada pela primeira vez pela Sun Microsystems em 1995. De um início humilde, ela evoluiu para uma grande participação no mundo digital dos dias atuais, oferecendo a plataforma confiável na qual muitos serviços e aplicativos são desenvolvidos. Produtos e serviços novos e inovadores projetados para o futuro continuam a confiar no Java também.

SQL é uma linguagem padrão para trabalhar com bancos de dados relacionais. Ela é uma linguagem declarativa e que não necessita de profundos conhecimentos de programação para que alguém possa começar a escrever queries, as consultas e pedidos, que trazem resultados de acordo com o que você está buscando. SQL significa Standard Query Language, literalmente a linguagem padrão para realizar queries.

## Objetivo Geral

Este trabalho tem como objetivo geral a implementação de facilidade e o controle dos dados de compra, venda, clientes e fornecedores para uma papelaria através de um programa de aplicação Java.

## Objetivos Específicos

Esse trabalho tem como objetivos específicos:

* Controlar e manipular dados;
* Cadastrar, editar e excluir objetos;
* Fixar conceitos de interface gráfica com swing (Java).

## Justificativa

Esse trabalho é justificado devido a necessidade do uso de um sistema para o controle de vendas dentro de uma papelaria, tornando mais prático a organização e a precisão ao acesso dos dados cadastrado de todos os clientes e fornecedores, do controle de item dentro do estoque bem como o lançamento dos produtos e a venda do mesmo, sendo assim é de relevância o tema e o projeto de estudo do trabalho sob o qual foi estabelecido para esta finalidade dentro do âmbito do comércio empresarial.

# DESENVOLVIMENTO

Compreender como é feito os processos de um programa é de muita importância quando não se sabe sobre o assunto, diante disto será apresentado como funciona o processo de criação de interface gráfica e orientação objeto de maneira branda.

## Swing

O Swing é um framework que disponibiliza um conjunto de elementos gráficos para ser utilizado na plataforma Java. O Swing é compatível com o Abstract Window Toolkit (AWT), mas trabalha de forma totalmente diferente. A API Swing, diferente do AWT, não delega a tarefa de renderização ao sistema operacional, ele renderiza os elementos por conta própria. Como a AWT é uma biblioteca de baixo-nível que depende de código nativo da plataforma ela traz alguns problemas de compatibilidade entre as plataformas, fazendo com que nem sempre o programa tenha a aparência desejada em todos os sistemas operacionais. Além disso, o Swing é mais completo e os programas têm uma aparência muito parecida, independente do sistema operacional que está sendo utilizado, possui uma enorme gama de controles extras disponíveis, tais como áreas de texto que nativamente podem mostrar conteúdo como RTF ou HTML, botões com suporte a imagens, sliders, selecionadores de cores, alteração do tipo de borda para os componentes, maior controle de como desenhar os mínimos detalhes de apresentação e muito mais. No entanto, a performance é um pouco pior devido à alta abstração, consumindo assim mais memória RAM.

Os principais componentes do Swing são resumidos abaixo:

**JFrame** representa a janela do programa com barra de título, ícone, botões de comando, etc. Entre os principais métodos temos o pack() que compacta a janela para o tamanho dos componentes, setSize(int, int) que define a largura e altura da janela, setLocation(int, int) que define a posição da janela na tela (x,y), setBounds(int, int, int, int) que define posição e tamanho, setVisible(boolean) que exibe a janela e setDefaultCloseOperation(int) que define o que ocorre quando o usuário tenta fechar a janela (as opções são: DO\_NOTHING\_ON\_CLOSE, HIDE\_ON\_CLOSE, DISPOSE\_ON\_CLOSE, EXIT\_ON\_CLOSE).

**JPanel** representa um tipo básico de container para inserção de componentes. Entre os principais métodos temos add(Component, int) que adiciona o componente definindo sua posição e setLayout(LayoutManagaer) que altera o tipo de layout.

**JLabel** representa um rótulo de texto. Entre os principais métodos temos o setText(String) que altera o texto e getText() que retorna o texto atual.

**JTextFiel** representa um campo de texto onde o usuário pode informar um texto em uma linha. Entre os principais métodos temos setText(String) que altera o texto e getText() que retorna o texto atual.

**JPasswordField** representa um campo de texto protegido, subclasse de JTextField. O principal método é o setEchoChar(char) que define o caractere que aparece ao digitar um texto.

**JTextArea** representa uma caixa onde o usuário pode informar várias linhas de texto. Entre os principais métodos temos o setText(String) que altera o texto, getText() que retorna o texto atual, getSelectedText() que retorna o texto selecionado pelo usuário e insert(String, int) que insere um texto na posição especificada.

**JCheckBox** representa uma caixa de seleção e permite selecionar ou não uma opção. Entre os principais métodos temos o setSelected(boolean) que altera o estado da caixa de seleção e o método isSelected() que retorna true se a caixa estiver marcada e false se não estiver marcada.

**JRadioButton** representa um componente que permite seleciona uma entre diversas opções. O JRadioButton é semelhante ao JCheckBox, inclusive com os mesmos construtores e métodos.

**JComboBox** representa uma caixa de combinação, da qual o usuário pode selecionar uma opção. Entre os principais métodos temos o addItem(Object) que adiciona um item à lista de opções, setEditable(boolean) que permite ao usuário digitar uma opção, getSelectedIndex() que retorna a posição do item atualmente selecionado, getSelectedItem() que retorna o texto do item atualmente selecionado, setSelectedIndex(int) que seleciona o item da posição especificada e setSelectedIndex(Object) que seleciona o objeto especificado na lista.

**JList** representa uma lista de opções que permite a seleção de mais de um item simultaneamente. Entre os principais métodos temos o setListData(Object[]) que preenche ou altera os itens de uma lista, getSelectedValues() que retorna um array de objetos contendo itens selecionados na lista.

**JButton** representa um botão destinado a executar uma ação. Entre os principais métodos temos o setText(String) que altera o texto do botão e setIcon(Icon) que altera o ícone do botão.

## Classe DAO

O padrão de projeto DAO surgiu com a necessidade de separarmos a lógica de negócios da lógica de persistência de dados. Este padrão permite que possamos mudar a forma de persistência sem que isso influencie em nada na lógica de negócio, além de tornar nossas classes mais legíveis.

Classes DAO são responsáveis por trocar informações com o SGBD e fornecer operações CRUD e de pesquisas, elas devem ser capazes de buscar dados no banco e transformar esses em objetos ou lista de objetos, fazendo uso de listas genéricas (BOX 3), também deverão receber os objetos, converter em instruções SQL e mandar para o banco de dados.

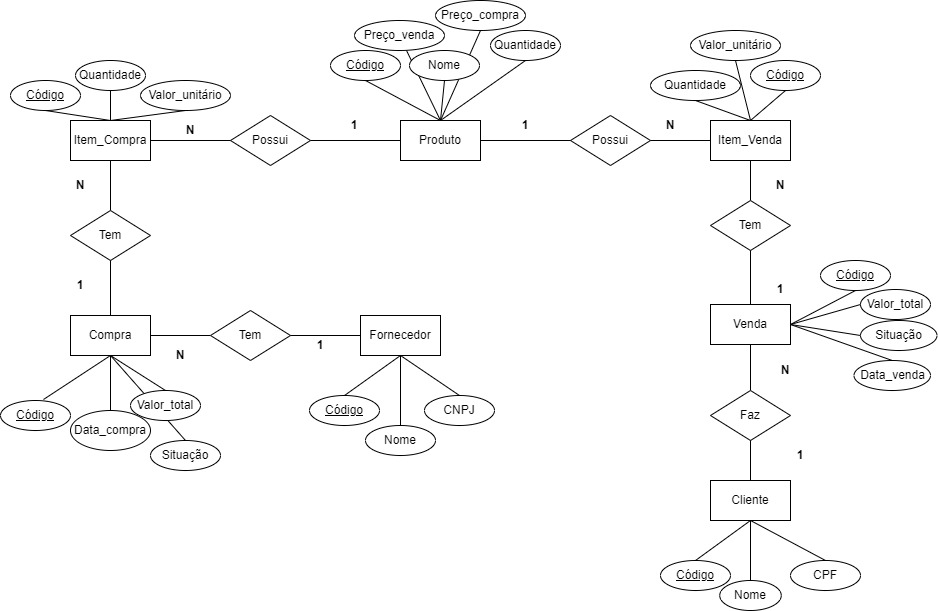
Toda interação com a base de dados se dará através destas classes, nunca das classes de negócio, muito menos de formulários.

Se aplicarmos este padrão corretamente ele vai abstrair completamente o modo de busca e gravação dos dados, tornando isso transparente para aplicação e facilitando muito na hora de fazermos manutenção na aplicação ou migrarmos de banco de dados.

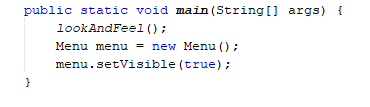
Também conseguimos centralizar a troca de dados com o SGBD (Sistema Gerenciador de Banco de Dados), teremos um ponto único de acesso a dados, tendo assim nossa aplicação um ótimo design orientado a objeto.

## Implementação

Foi implementado o diagrama de entidade e relacionamento DER com as tabelas do banco de dados e seus relacionamentos conforme imagem abaixo:

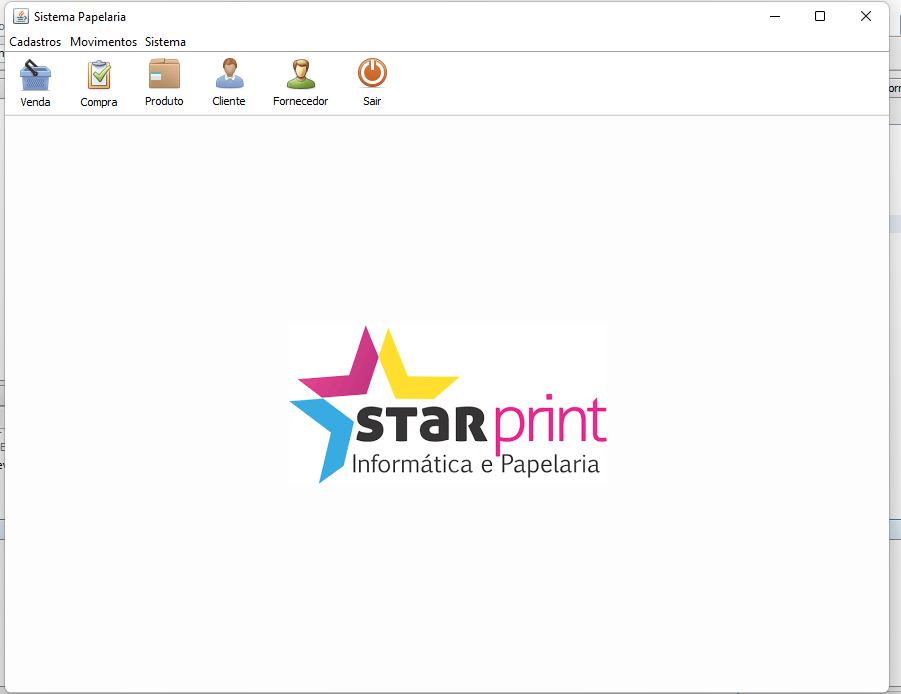


Na classe Principal para chamar os menus é utilizado o Look and Feel que age conforme mudamos de SO ou de ambiente, o look and feel se adapta de acordo com a máquina que executamos.



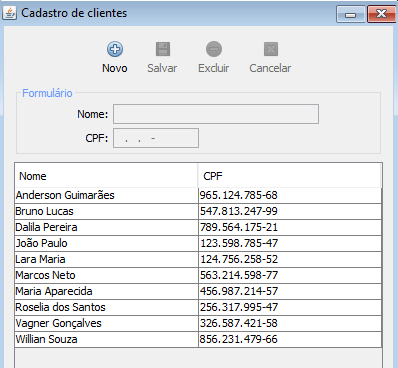
**Figura 1 – print da classe que chama o menu principal**

A tela inicial de menu, contem botões que após o clique direciona para a opção desejada. Conforme figura abaixo da representação gráfica do menu:



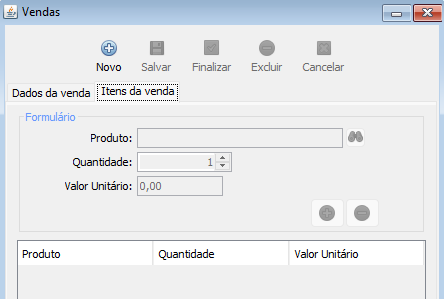
**Figura 2 – tela menu Inicial**

Selecionando as partes mais importantes, temos abaixo a opção após clique em CLIENTE na tela de menu principal é direcionado para o menu de cadastro, nesta tela temos o botão novo, salvar, excluir e cancelar. Selecionando a opção Novo, é aberto os campos para preenchimento dos dados do nome e CPF de cada cliente que se cadastre.



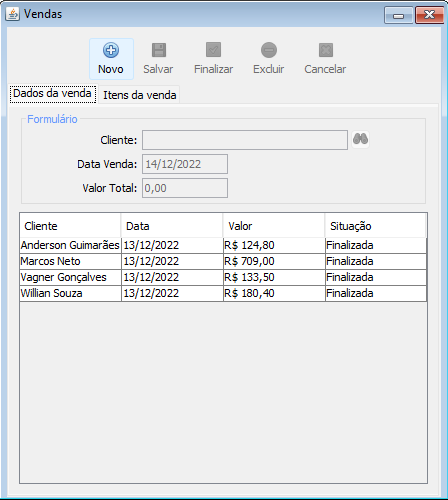
**Figura 3 – tela cadastro de clientes**

Escolhendo a opção Venda, na tela é permitido a inserção de dados do item da venda, a mesma funciona através da busca de um produto já cadastrado no sistema, após o retorno deste objeto, o programa solicita a entrada da quantidade do mesmo e o valor da unidade.



**Figura 4 – cadastro de vendas Aba Itens Venda**

Após a inserção do produto na aba item venda o próximo passo está na aba Dados da venda, onde o sistema permite a busca do cliente de acordo com o banco de dados da listagem de nomes cadastrado e realiza o cálculo automático do valor total da compra de acordo com a quantidade de produtos informado, se a compra estiver tudo certo, o botão finalizar permite o lançamento desta venda dentro do banco de dados.



**Figura 5 – tela cadastro venda aba Dados Venda**

# CONCLUSÃO

Após os conceitos apresentados e observados, é possível concluir que quando se trata da criação de interfaces gráficas, a linguagem Java possui significativa importância. Os diversos componentes gráficos e a diversidades de gerenciadores de layout tornam a linguagem Java a opção ideal para a criação de interfaces gráficas em projetos.

Este trabalho abre a possibilidade para se entender sobre o gerenciamento de dados através da programação em Java com a conexão ao banco de dados. Onde permite para o usuário uma manipulação precisa dos dados em sua aplicação.

REFERÊNCIAS

Classe DAO. Disponível em: <https://www.devmedia.com.br/dao-pattern-persistenciadedadosutilizandoopadraodao/30999#:~:text=Classes%20DAO%20s%C3%A3o%20respons%C3%A1vis%20por,em%20instru%C3%A7%C3%B5es%20SQL%20e%20mandar>. Acessado em: 14/12/2022.

Java. Disponível em:< https://www.java.com/ptBR/download/help/whatis\_java.htm l# :~:text=Java%20%C3%A9%20uma%20linguagem%20de,servi%C3%A7os%20e%20aplicativos%20s%C3%A3o%20desenvolvidosJ>. Acessado em 14/02/2022.

Java Swing: Disponível em:<https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981 >. Acessado em 14/02/2022.

SQL. Disponível em: < https://www.alura.com.br/artigos/o-que-e-sql#:~ :text =SQL % 20%C3%A9%20uma%20linguagem%20padr%C3%A3o,o%20que%20voc%C3%AA%20est%C3%A1%20buscando>. Acessado em 14/02/2022.

APÊNDICE A – (Nome do apêndice)

***Classe Principal***

public class Principal {

/\*\*

\* Responsável por iniciar a aplicação

\*

\* @param args lista de parâmetros do main

\*/

public static void main(String[] args) {

lookAndFeel();

Menu menu = new Menu();

menu.setVisible(true);

}

private static void lookAndFeel() {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

JFrame.setDefaultLookAndFeelDecorated(true);

JDialog.setDefaultLookAndFeelDecorated(true);

} catch (ClassNotFoundException | InstantiationException | IllegalAccessException | UnsupportedLookAndFeelException e) {

JOptionPane.showMessageDialog(null, e, "Erro", JOptionPane.ERROR\_MESSAGE);

}

}

}

***Classe Cliente***

package com.jdenner.to;

import java.util.Date;

/\*\*

\* Classe contendo os dados do cliente

\*/

public class Cliente {

private int codigo;

private String nome;

private String cpf;

public Cliente() {

this.codigo = 0;

this.nome = "";

this.cpf = "";

}

public Cliente(int codigo) {

this.codigo = codigo;

this.nome = "";

this.cpf = "";

}

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public String getCpf() {

return cpf;

}

public void setCpf(String cpf) {

this.cpf = cpf;

}

@Override

public String toString() {

return getNome();

}

@Override

public boolean equals(Object o) {

if (o instanceof Cliente) {

Cliente c = (Cliente) o;

if (c.getCodigo() == this.getCodigo()) {

return true;

}

}

return false;

}

}

***Classe Compra***

package com.jdenner.to;

import com.jdenner.to.enums.Situacao;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

/\*\*

\* Classe contendo os dados da compra

\*/

public class Compra {

private int codigo;

private Fornecedor fornecedor;

private Date dataCompra;

private Double valorTotal;

private Situacao situacao;

private List<ItemCompra> itens;

private List<ItemCompra> itensRemover;

public Compra() {

this.codigo = 0;

this.fornecedor = new Fornecedor();

this.dataCompra = new Date();

this.valorTotal = 0.0;

this.itens = new ArrayList<>();

this.itensRemover = new ArrayList<>();

}

public Compra(int codigo) {

this.codigo = codigo;

this.fornecedor = new Fornecedor();

this.dataCompra = new Date();

this.valorTotal = 0.0;

this.itens = new ArrayList<>();

this.itensRemover = new ArrayList<>();

}

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public Fornecedor getFornecedor() {

return fornecedor;

}

public void setFornecedor(Fornecedor fornecedor) {

this.fornecedor = fornecedor;

}

public Date getDataCompra() {

return dataCompra;

}

public void setDataCompra(Date dataCompra) {

this.dataCompra = dataCompra;

}

public Double getValorTotal() {

double total = 0;

for (ItemCompra iv : itens) {

total += (iv.getValorUnitario() \* iv.getQuantidade());

}

return total;

}

public Situacao getSituacao() {

return situacao;

}

public void setSituacao(Situacao situacao) {

this.situacao = situacao;

}

public void setSituacao(int situacao) {

if (situacao == Situacao.ABERTA.getId()) {

setSituacao(Situacao.ABERTA);

} else if (situacao == Situacao.FINALIZADA.getId()) {

setSituacao(Situacao.FINALIZADA);

} else if (situacao == Situacao.CANCELADA.getId()) {

setSituacao(Situacao.CANCELADA);

}

}

public List<ItemCompra> getItens() {

return itens;

}

public List<ItemCompra> getItensRemover() {

return itensRemover;

}

public void addItem(ItemCompra itemCompra) {

itens.add(itemCompra);

}

public void removeItem(ItemCompra itemCompra) {

itens.remove(itemCompra);

if (itemCompra.getCodigo() != 0) {

itensRemover.add(itemCompra);

}

}

public int quantidadeItens() {

return itens.size();

}

@Override

public String toString() {

return String.valueOf(this.codigo);

}

}

***Classe Fornecedor***

package com.jdenner.to;

/\*\*

\* Classe contendo os dados do fornecedor

\*/

public class Fornecedor {

private int codigo;

private String nome;

private String cnpj;

public Fornecedor() {

this.codigo = 0;

this.nome = "";

this.cnpj = "";

}

public Fornecedor(int codigo) {

this.codigo = codigo;

this.nome = "";

this.cnpj = "";

}

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public String getCnpj() {

return cnpj;

}

public void setCnpj(String cnpj) {

this.cnpj = cnpj;

}

@Override

public String toString() {

return getNome();

}

@Override

public boolean equals(Object o) {

if (o instanceof Fornecedor) {

Fornecedor c = (Fornecedor) o;

if (c.getCodigo() == this.getCodigo()) {

return true;

}

}

return false;

}

}

***classe ItemCompra***

package com.jdenner.to;

/\*\*

\* Classe de relação entre a compra e o produto

\*/

public class ItemCompra {

private int codigo;

private Compra compra;

private Produto produto;

private int quantidade;

private Double valorUnitario;

public ItemCompra() {

this.codigo = 0;

this.compra = new Compra();

this.produto = new Produto();

this.quantidade = 0;

this.valorUnitario = 0.0;

}

public ItemCompra(int codigo) {

this.codigo = codigo;

this.compra = new Compra();

this.produto = new Produto();

this.quantidade = 0;

this.valorUnitario = 0.0;

}

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public Compra getCompra() {

return compra;

}

public void setCompra(Compra compra) {

this.compra = compra;

}

public Produto getProduto() {

return produto;

}

public void setProduto(Produto produto) {

this.produto = produto;

}

public int getQuantidade() {

return quantidade;

}

public void setQuantidade(int quantidade) {

this.quantidade = quantidade;

}

public Double getValorUnitario() {

return valorUnitario;

}

public void setValorUnitario(Double valorUnitario) {

this.valorUnitario = valorUnitario;

}

@Override

public String toString() {

return getProduto().getNome();

}

@Override

public boolean equals(Object o) {

if (o instanceof ItemCompra) {

ItemCompra iv = (ItemCompra) o;

if (iv.getCodigo() == this.getCodigo()) {

return true;

}

}

return false;

}

}

***Classe ItemVenda***

package com.jdenner.to;

/\*\*

\* Classe de relação entre a venda e o produto

\*/

public class ItemVenda {

private int codigo;

private Venda venda;

private Produto produto;

private int quantidade;

private Double valorUnitario;

public ItemVenda() {

this.codigo = 0;

this.venda = new Venda();

this.produto = new Produto();

this.quantidade = 0;

this.valorUnitario = 0.0;

}

public ItemVenda(int codigo) {

this.codigo = codigo;

this.venda = new Venda();

this.produto = new Produto();

this.quantidade = 0;

this.valorUnitario = 0.0;

}

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public Venda getVenda() {

return venda;

}

public void setVenda(Venda venda) {

this.venda = venda;

}

public Produto getProduto() {

return produto;

}

public void setProduto(Produto produto) {

this.produto = produto;

}

public int getQuantidade() {

return quantidade;

}

public void setQuantidade(int quantidade) {

this.quantidade = quantidade;

}

public Double getValorUnitario() {

return valorUnitario;

}

public void setValorUnitario(Double valorUnitario) {

this.valorUnitario = valorUnitario;

}

@Override

public String toString() {

return getProduto().getNome();

}

@Override

public boolean equals(Object o) {

if (o instanceof ItemVenda) {

ItemVenda iv = (ItemVenda) o;

if (iv.getCodigo() == this.getCodigo()) {

return true;

}

}

return false;

}

}

***classe Produto***

package com.jdenner.to;

/\*\*

\* Classe contendo os dados do produto

\*/

public class Produto {

private int codigo;

private String nome;

private Double precoCompra;

private Double precoVenda;

private int quantidade;

public Produto() {

this.codigo = 0;

this.nome = "";

this.precoCompra = 0.0;

this.precoVenda = 0.0;

this.quantidade = 0;

}

public Produto(int codProduto) {

this.codigo = codProduto;

this.nome = "";

this.precoCompra = 0.0;

this.precoVenda = 0.0;

this.quantidade = 0;

}

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public String getNome() {

return nome;

}

public void setNome(String nome) {

this.nome = nome;

}

public Double getPrecoCompra() {

return precoCompra;

}

public void setPrecoCompra(Double precoCompra) {

this.precoCompra = precoCompra;

}

public Double getPrecoVenda() {

return precoVenda;

}

public void setPrecoVenda(Double precoVenda) {

this.precoVenda = precoVenda;

}

public int getQuantidade() {

return quantidade;

}

public void setQuantidade(int quantidade) {

this.quantidade = quantidade;

}

@Override

public String toString() {

return getNome();

}

@Override

public boolean equals(Object o) {

if (o instanceof Produto) {

Produto p = (Produto) o;

if (p.getCodigo() == this.getCodigo()) {

return true;

}

}

return false;

}

}

***classe Venda***

package com.jdenner.to;

import com.jdenner.to.enums.Situacao;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

/\*\*

\* Classe contendo os dados da venda

\*/

public class Venda {

private int codigo;

private Cliente cliente;

private Date dataVenda;

private Double valorTotal;

private Situacao situacao;

private List<ItemVenda> itens;

private List<ItemVenda> itensRemover;

public Venda() {

this.codigo = 0;

this.cliente = new Cliente();

this.dataVenda = new Date();

this.valorTotal = 0.0;

this.itens = new ArrayList<>();

this.itensRemover = new ArrayList<>();

}

public Venda(int codigo) {

this.codigo = codigo;

this.cliente = new Cliente();

this.dataVenda = new Date();

this.valorTotal = 0.0;

this.itens = new ArrayList<>();

this.itensRemover = new ArrayList<>();

}

public int getCodigo() {

return codigo;

}

public void setCodigo(int codigo) {

this.codigo = codigo;

}

public Cliente getCliente() {

return cliente;

}

public void setCliente(Cliente cliente) {

this.cliente = cliente;

}

public Date getDataVenda() {

return dataVenda;

}

public void setDataVenda(Date dataVenda) {

this.dataVenda = dataVenda;

}

public Double getValorTotal() {

double total = 0;

for (ItemVenda iv : itens) {

total += (iv.getValorUnitario() \* iv.getQuantidade());

}

return total;

}

public Situacao getSituacao() {

return situacao;

}

public void setSituacao(Situacao situacao) {

this.situacao = situacao;

}

public void setSituacao(int situacao) {

if (situacao == Situacao.ABERTA.getId()) {

setSituacao(Situacao.ABERTA);

} else if (situacao == Situacao.FINALIZADA.getId()) {

setSituacao(Situacao.FINALIZADA);

} else if (situacao == Situacao.CANCELADA.getId()) {

setSituacao(Situacao.CANCELADA);

}

}

public List<ItemVenda> getItens() {

return itens;

}

public List<ItemVenda> getItensRemover() {

return itensRemover;

}

public void addItem(ItemVenda itemVenda) {

itens.add(itemVenda);

}

public void removeItem(ItemVenda itemVenda) {

itens.remove(itemVenda);

if (itemVenda.getCodigo() != 0) {

itensRemover.add(itemVenda);

}

}

public int quantidadeItens() {

return itens.size();

}

}