**Programação de dispositivos móveis**

**T1 – Modelagem e implementação gráfica de um caixa de supermercado**

**Luis Felipe Chary de Lima**

**12.01994-0**

**6º CA noturno**

**Introdução**

Este trabalho tem como objetivo implementar a modelagem de um sistema de caixa de marcado através de um programa escrito em linguagem JAVA.

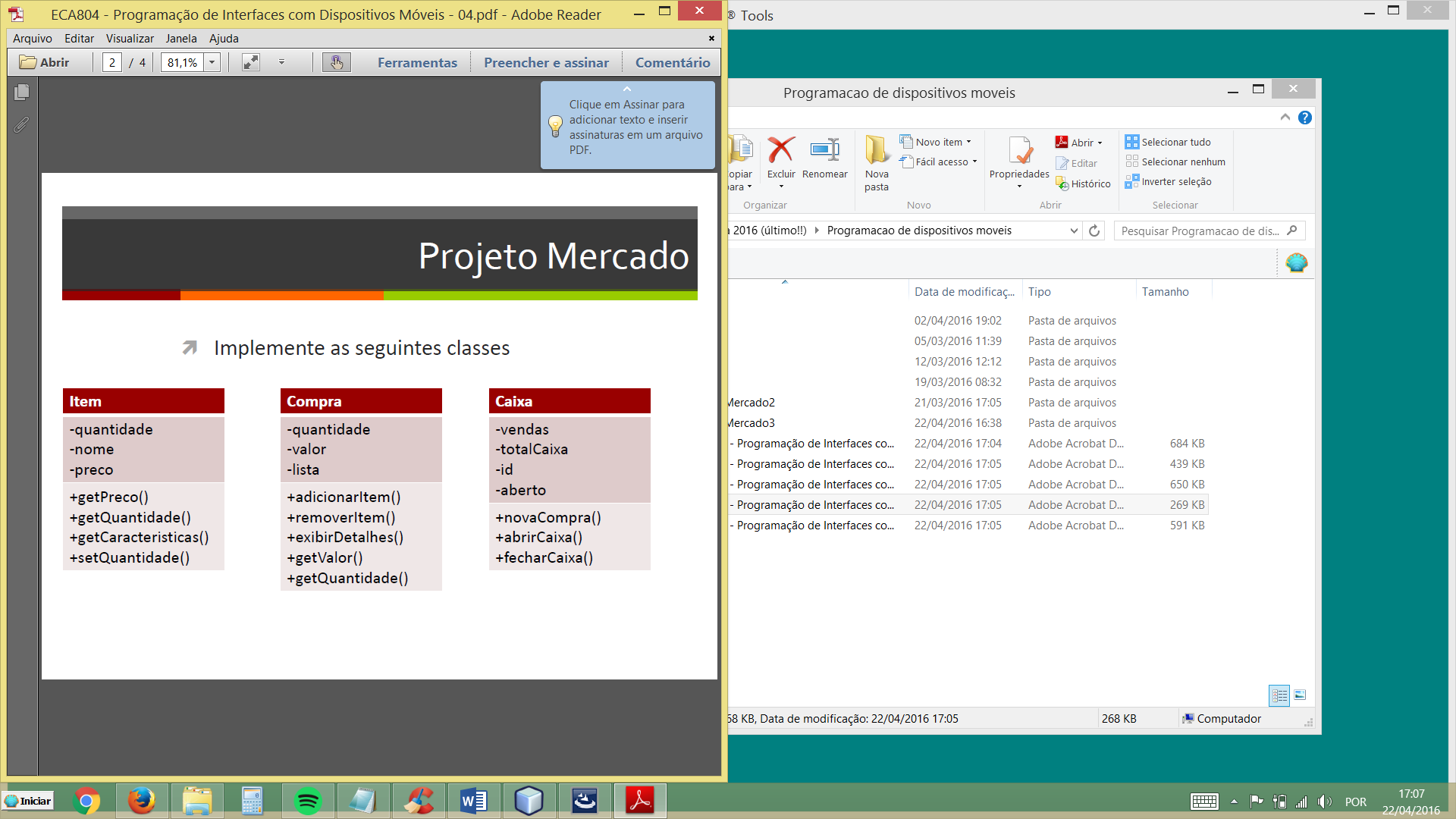
**Conceito e classes**

Neste trabalho foram utilizadas 4 classes: Caixa, Item, Compra e Janela.

* A classe **Item**:

Esta classe modela o comportamento e os atributos de um item ou produto que pode ser comprado em um mercado comum.

Segue o diagrama UML simplificado que representa esta classe:



* A classe **Compra**:

Esta classe modela os atributos e comportamentos básicos de uma “compra.

Segue o diagrama UML simplificado que representa esta classe:



* A classe **Caixa**:

Esta classe modela os atributos e comportamentos básicos de um caixa se supermercado.

Segue o diagrama UML simplificado que representa esta classe:



* A classe **Janela**:

Esta classe herda a classe Caixa (e, portanto, possui todos os seus atributos e métodos) e também implementa a interface gráfica que interage com o usuário.

Segue o diagrama UML simplificado que representa esta classe:

Janela

-frame

-travaTudo()

-destravaTudo()

-pegaNovoItem()

-zeraCampos()

-gbc

-compraAtual

-scroll

-labelValor

-labelQuantidade

-labelNome

-labelTotal

-memo

-botaoAbrirFechar

-botaoRemoverItem

-botaoCancelarCompra

-botaoFinalizarCompra

-botaoNovoItem

-textoValor

-textoQuantidade

-textoNome

-painelPrincipal

**Definições e funcionalidades**

Para a organização dos componentes dentro da janela, foi utilizado o layout manager GridBagLayout() por proporcionar uma possibilidade de organização mais ampla e intuitiva.

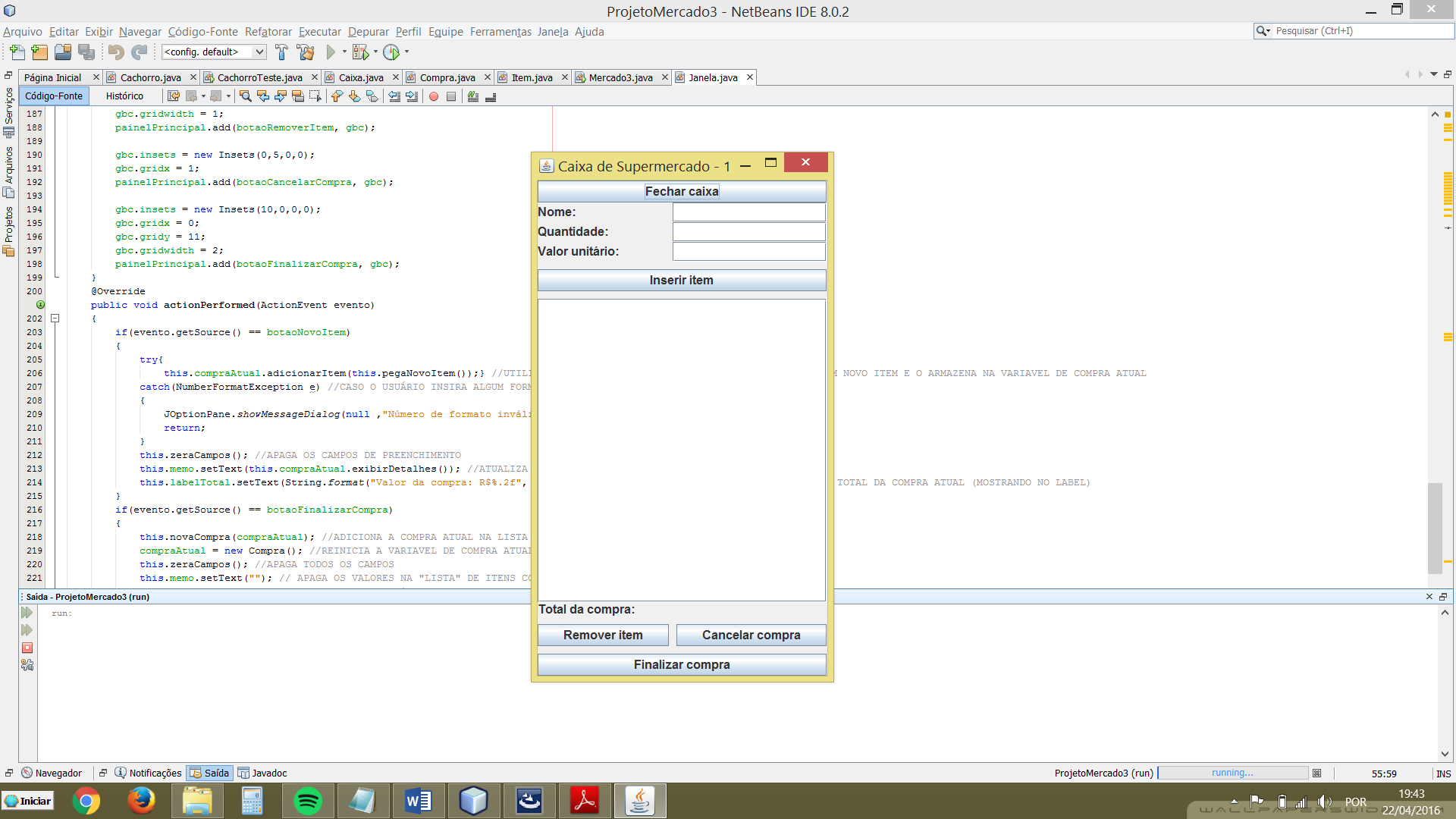
Foram usados 4 campos de texto:

* textoNome -> Campo de texto aonde será inserido o nome do item a ser adicionado a compra.
* textoQuantidade -> Campo de texto aonde será inserida a quantidade comprada do próximo item a ser inserido na compra.
* textoValor -> Campo de texto aonde será inserido o valor unitário do item a ser inserido na compra.
* Memo -> Campo de texto aonde constam todos os itens que estão sendo comprados assim como seus preços unitários e quantidades.

Dentro da classe temos uma sobreposição do método ActionPerformed(ActionEvent e) da classe ActionListener a qual define os eventos para cada um dos botões que são possíveis de se clicar (mais detalhes no código).

As ações desempenhadas pelos botões são:

* botaoAbrirFechar -> Botão responsável pelas ações de abrir ou fechar o caixa. Quando o caixa está aberto o botão executa a tarefa de fecha-lo, quando o caixa está fechado o botão executa a tarefa de abri-lo. Toda vez que o caixa é fechado o programa emite um aviso com o valor total contido no caixa.
* botaoNovoItem -> Botão responsável pela ação de adicionar o item descrito pelos campos de nome, quantidade e preço unitário do produto a compra atual que está ocorrendo no caixa.
* botaoRemoverItem -> Botão responsável pela ação de remover um item da lista de compras.
* botaoCancelarCompra -> Botão responsável pela ação de cancelar a compra atual.
* botaoFinalizarCompra -> Botão responsável pela ação de finalizar a compra atual (contabiliza-la no total do caixa) e iniciar uma nova compra. Toda vez que uma compra é finalizada é emitido um viso da quantidade total de dinheiro contido no caixa.



labelTotal

labelQuantidade

labelValor

labelNome

botaoRemoverItem

botaoFinalizarCompra

botaoCancelarCompra

memo

botaoNovoItem

textoValor

textoQuantidade

textoNome

botaoAbrirFechar

**Código**

* Classe Item:

public class Item {

private int quantidade;

private double preco;

private String nome;

public double getPreco()

{

return this.preco; //valor unitário

}

public int getQuantidade()

{

return this.quantidade;

}

public void setQuantidade(int quantidade)

{

this.quantidade = quantidade;

}

public String getCaracteristicas()

{

String s;

s = "Nome: " + this.nome;

s = s + "\nPreço unitário: R$" + String.format("%.2f",this.preco);

s = s + "\nQuantidade: " + this.quantidade;

return s;

}

public Item(String nome, double preco)

{

this.nome = nome;

this.preco = preco;

this.quantidade = 1;

}

public Item(String nome, double preco, int quantidade)

{

this.nome = nome;

this.preco = preco;

this.quantidade = quantidade;

}

}

* Classe Compra:

public class Compra {

private int quantidade;

private double valor;

private List<Item> lista;

public void adicionarItem(Item x)

{

this.quantidade++;

this.lista.add(x);

this.valor = this.valor + x.getPreco() \* x.getQuantidade();

}

public int getQuantidade()

{

return this.quantidade;

}

public double getValor()

{

return this.valor;

}

public boolean removerItem(int i)

{

if(this.lista.contains(this.lista.get(i)))

{

this.quantidade--;

this.valor = this.valor - this.lista.get(i).getPreco() \* this.lista.get(i).getQuantidade();

this.lista.remove(i);

return true;

}

return false;

}

public String exibirDetalhes()

{

String s = "";

for(int i = 0; i < this.lista.size(); i++)

{

s = s + "\nItem: " + i + "\n" + this.lista.get(i).getCaracteristicas() + "\n";

}

return s;

}

public Compra()

{

this.quantidade = 0;

this.valor = 0;

this.lista = new ArrayList<Item>();

}

}

* Classe Caixa:

public class Caixa {

private List<Compra> vendas; //lista com as compras efetuadas no caixa

private double totalCaixa; //valor total arrecadado pelo caixa

private int id; //identificação do caixa

private boolean aberto; //variável que sinaliza o estado do caixa, aberto = true /fechado = false

public double getTotalCaixa()

{

return this.totalCaixa;

}

public void abrirCaixa()

{

this.aberto = true;

}

public void fecharCaixa()

{

this.aberto = false;

}

public void novaCompra(Compra c) //adiciona uma nova compra a lista de compras do caixa e atualiza o valor total do caixa

{

if(this.aberto){

totalCaixa = totalCaixa + c.getValor();

this.vendas.add(c);

}

}

//sobrecarga de construtores

public Caixa(int id, double total, boolean aberto)

{

this.id = id;

this.totalCaixa = total;

this.aberto = aberto;

this.vendas = new ArrayList<Compra>();

}

public Caixa(int id)

{

this.id = id;

this.totalCaixa = 0;

this.aberto = false;

this.vendas = new ArrayList<Compra>();

}

public Caixa(int id, double total)

{

this.aberto = false;

this.id = id;

this.totalCaixa = total;

this.vendas = new ArrayList<Compra>();

}

}

* Classe Janela:

public class Janela extends Caixa implements ActionListener {

private JFrame frame; //janela

private JPanel painelPrincipal;

private JTextField textoNome;

private JTextField textoQuantidade;

private JTextField textoValor;

private JButton botaoNovoItem;

private JButton botaoFinalizarCompra;

private JButton botaoCancelarCompra;

private JButton botaoRemoverItem;

private JButton botaoAbrirFechar;

private JTextArea memo;

private JLabel labelTotal;

private JLabel labelNome;

private JLabel labelQuantidade;

private JLabel labelValor;

private JScrollPane scroll;

private Compra compraAtual;

private GridBagConstraints gbc;

public Janela(int id)

{

super(id); //configura o id do caixa atual

gbc = new GridBagConstraints(); //usado para configurar o layout

frame = new JFrame("Caixa de Supermercado - " + id);

painelPrincipal = new JPanel(new GridBagLayout());

textoNome = new JTextField();

textoQuantidade = new JTextField();

textoValor = new JTextField();

botaoNovoItem = new JButton("Inserir item");

botaoFinalizarCompra = new JButton("Finalizar compra");

botaoCancelarCompra = new JButton("Cancelar compra");

botaoRemoverItem = new JButton("Remover item");

botaoAbrirFechar = new JButton("Abrir caixa");

memo = new JTextArea();

labelTotal = new JLabel("Total da compra:");

labelNome = new JLabel("Nome:");

labelQuantidade = new JLabel("Quantidade:");

labelValor = new JLabel("Valor unitário:");

compraAtual = new Compra(); //inicializa variável de compra

this.travaTudo();// caixa fechado

memo.setEditable(false); //trava entradas no campo de texto

scroll = new JScrollPane(memo); //scroll usado no textArea para nao limitar a visao

//scroll.setViewportView(memo);

//frame.add(scroll);

//painel4.add(scroll, BorderLayout.CENTER);

//CONFIGURAÇÕES DE TEXTO (FONTE)

memo.setFont(new Font("Arial", Font.BOLD, 14));

labelTotal.setFont(new Font("Arial", Font.BOLD, 16));

labelNome.setFont(labelTotal.getFont());

labelQuantidade.setFont(labelNome.getFont());

labelValor.setFont(labelNome.getFont());

botaoNovoItem.setFont(labelValor.getFont());

botaoFinalizarCompra.setFont(botaoNovoItem.getFont());

botaoCancelarCompra.setFont(botaoNovoItem.getFont());

botaoRemoverItem.setFont(botaoNovoItem.getFont());

botaoAbrirFechar.setFont(botaoNovoItem.getFont());

textoNome.setFont(new Font("Arial", Font.PLAIN, 14));

textoQuantidade.setFont(textoNome.getFont());

textoValor.setFont(textoNome.getFont());

////////////////////////////////////////////////

//CORES

painelPrincipal.setBackground(new Color(240, 240, 240));

////////////////////////////////

//CONFIGURAÇÕES DA JANELA

frame.getContentPane().add(painelPrincipal);

frame.setSize(400, 700);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLocation(700, 200);

frame.setVisible(true);

/////////////////////////

//associa o action listener aos botoes

this.botaoNovoItem.addActionListener(this);

this.botaoFinalizarCompra.addActionListener(this);

this.botaoAbrirFechar.addActionListener(this);

this.botaoCancelarCompra.addActionListener(this);

this.botaoRemoverItem.addActionListener(this);

//LAYOUT!!

//gbc.gridx = 0; //"POSICIONA" OBJETO NO PAINEL

//gbc.gridy = 0;

//gbc.weightx = 0.5; //DEFINE UMA PRIORIDADE DE TAMANHO

//gbc.fill = GridBagConstraints.HORIZONTAL; //DIZ QUE O OBJETO PREENCHERÁ TODO ESPAÇO HORIZONTAL QUE FOI ESTABELECIDO PARA ELE

//gbc.insets = new Insets(0, 10, 0, 0);

//painelPrincipal.add(botaoAberto, gbc); //ADICIONA O OBJETO AO PAINEL CONSIDERANDO OS PARÂMETROS CONFIGURADOS ANTERIORMENTE

gbc.gridx = 0;

gbc.gridy = 0;

gbc.weightx = 0.5;

gbc.insets = new Insets(0,0,0,0);

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.HORIZONTAL;

painelPrincipal.add(botaoAbrirFechar, gbc);

//gbc.insets = new Insets(10, 20, 0, 0);

gbc.gridx = 0;

gbc.gridy = 1;

painelPrincipal.add(labelNome, gbc);

gbc.gridx = 1;

gbc.ipady = 5;

gbc.insets = new Insets(0,0,0,0);

painelPrincipal.add(textoNome, gbc);

gbc.ipady = 0;

gbc.gridy = 2;

gbc.gridx = 0;

//gbc.insets = new Insets(10, 20, 0, 0);

painelPrincipal.add(labelQuantidade, gbc);

gbc.gridx = 1;

gbc.ipady = 5;

gbc.insets = new Insets(0,0,0,0);

painelPrincipal.add(textoQuantidade, gbc);

gbc.ipady = 0;

gbc.gridy = 3;

gbc.gridx = 0;

//gbc.insets = new Insets(10,20,0,0);

painelPrincipal.add(labelValor, gbc);

gbc.gridx = 1;

gbc.ipady = 5;

gbc.insets = new Insets(0,0,0,0);

gbc.fill = GridBagConstraints.HORIZONTAL;

painelPrincipal.add(textoValor, gbc);

gbc.ipady = 0;

gbc.gridwidth = 2;

gbc.gridy = 4;

gbc.gridx = 0;

gbc.insets = new Insets(10,0,0,0);

gbc.fill = GridBagConstraints.HORIZONTAL;

painelPrincipal.add(botaoNovoItem, gbc);

gbc.gridheight = 4; //DEFINE UM ESPAÇO MAIOR (VERTICAL)

gbc.gridy = 5;

gbc.weighty = 0.6;

gbc.fill = GridBagConstraints.BOTH; //PREENCHE OS ESPAÇOS VERTICAIS E HORIZONTAIS (POSSIVEIS)

painelPrincipal.add(scroll, gbc); //ADICIONO O SCROLL E NAO O TEXTAREA

gbc.gridy = 9;

gbc.gridheight = 1;

gbc.weighty = 0;

gbc.gridwidth = 2;

gbc.insets = new Insets(0,0,10,0);

gbc.fill = GridBagConstraints.HORIZONTAL;

painelPrincipal.add(labelTotal, gbc);

gbc.fill = GridBagConstraints.BOTH;

gbc.insets = new Insets(0, 0, 0, 5);

gbc.gridy = 10;

gbc.gridwidth = 1;

painelPrincipal.add(botaoRemoverItem, gbc);

gbc.insets = new Insets(0,5,0,0);

gbc.gridx = 1;

painelPrincipal.add(botaoCancelarCompra, gbc);

gbc.insets = new Insets(10,0,0,0);

gbc.gridx = 0;

gbc.gridy = 11;

gbc.gridwidth = 2;

painelPrincipal.add(botaoFinalizarCompra, gbc);

}

@Override

public void actionPerformed(ActionEvent evento)

{

if(evento.getSource() == botaoNovoItem)

{

try{

this.compraAtual.adicionarItem(this.pegaNovoItem());} //UTILIZA FUNÇÃO QUE CONVERTE OS VALORES DOS CAMPOS EM UM NOVO ITEM E O ARMAZENA NA VARIAVEL DE COMPRA ATUAL

catch(NumberFormatException e) //CASO O USUÁRIO INSIRA ALGUM FORMATO INVÁLIDO NOS CAMPOS QUE DEVEM TER NÚMEROS

{

JOptionPane.showMessageDialog(null ,"Número de formato inválido!");

return;

}

this.zeraCampos(); //APAGA OS CAMPOS DE PREENCHIMENTO

this.memo.setText(this.compraAtual.exibirDetalhes()); //ATUALIZA A "LISTA" DE ITENS COMPRADOS

this.labelTotal.setText(String.format("Valor da compra: R$%.2f", this.compraAtual.getValor())); //ATUALIZA O VALOR TOTAL DA COMPRA ATUAL (MOSTRANDO NO LABEL)

}

if(evento.getSource() == botaoFinalizarCompra)

{

this.novaCompra(compraAtual); //ADICIONA A COMPRA ATUAL NA LISTA DE COMPRAS DO CAIXA

compraAtual = new Compra(); //REINICIA A VARIAVEL DE COMPRA ATUAL (ZERA)

this.zeraCampos(); //APAGA TODOS OS CAMPOS

this.memo.setText(""); // APAGA OS VALORES NA "LISTA" DE ITENS COMPRADOS

this.labelTotal.setText("Valor da compra: R$0,00");

JOptionPane.showMessageDialog(null, String.format("Total do caixa: R$%.2f", this.getTotalCaixa())); //MOSTRA O TOTAL ARRECADADO PELO CAIXA ATÉ O MOMENTO

}

if(evento.getSource() == botaoAbrirFechar)

{

if(botaoAbrirFechar.getText() == "Abrir caixa"){ //ABRE O CAIXA

this.destravaTudo(); // LIBERA OS CAMPOS PARA SEREM PREENCHIDOS E BOTOES PARA SEREM CLICADOS

this.abrirCaixa(); //"ABRE" O CAIXA

this.botaoAbrirFechar.setText("Fechar caixa");

}

else{

this.travaTudo(); //TRAVA CAMPOS E BOTOES

this.fecharCaixa(); //"FECHA" O CAIXA

this.botaoAbrirFechar.setText("Abrir caixa");

JOptionPane.showMessageDialog(null, String.format("Total do caixa: R$%.2f", this.getTotalCaixa())); //MOSTRA O TOTAL ARRECADADO PELO CAIXA ATÉ O MOMENTO

}

}

if(evento.getSource() == botaoRemoverItem){

int numero = -1;

while(numero <0){

try{

numero = Integer.parseInt(JOptionPane.showInputDialog("Digite o número do item a ser removido"));

}catch(NumberFormatException e){

JOptionPane.showMessageDialog(null, "Formato de número incorreto!");

}

}

this.compraAtual.removerItem(numero);

this.memo.setText(this.compraAtual.exibirDetalhes());

this.labelTotal.setText(String.format("Valor da compra: R$%.2f", this.compraAtual.getValor()));

}

if(evento.getSource() == botaoCancelarCompra){

this.compraAtual = new Compra();

this.zeraCampos();

this.memo.setText("");

this.labelTotal.setText(String.format("Valor da compra: R$%.2f", this.compraAtual.getValor()));

}

}

private void travaTudo() //TRAVA CAMPOS E BOTOES

{

this.botaoNovoItem.setEnabled(false);

this.botaoFinalizarCompra.setEnabled(false);

this.textoNome.setEnabled(false);

this.textoQuantidade.setEnabled(false);

this.textoValor.setEnabled(false);

this.botaoCancelarCompra.setEnabled(false);

this.botaoRemoverItem.setEnabled(false);

}

private void destravaTudo() //LIBERA CAMPOS E BOTOES

{

this.botaoNovoItem.setEnabled(true);

this.botaoFinalizarCompra.setEnabled(true);

this.textoNome.setEnabled(true);

this.textoQuantidade.setEnabled(true);

this.textoValor.setEnabled(true);

this.botaoCancelarCompra.setEnabled(true);

this.botaoRemoverItem.setEnabled(true);

}

private Item pegaNovoItem() //COLHE OS VALORES NOS CAMPOS DE TEXTO E OS UTILIZA PARA CARACTERIZAR UM NOVO ITEM

{

String nome = textoNome.getText();

int quantidade = Integer.parseInt(textoQuantidade.getText());

double valor = Double.parseDouble(textoValor.getText());

Item i = new Item(nome, valor, quantidade);

return i;

}

private void zeraCampos() //LIMPA CAMPOS

{

this.textoNome.setText("");

this.textoQuantidade.setText("");

this.textoValor.setText("");

}

}