**Documentación del Sistema de Punto de Venta (KFC)**

**Descripción General**

Este código implementa un sistema de punto de venta (POS) simple utilizando Java y la interfaz gráfica de usuario con Swing. Permite a los usuarios agregar productos a una tabla, realizar cálculos de totales, gestionar el pago y generar una factura. También tiene la funcionalidad de imprimir la factura.

**Clases y Componentes Principales**

1. **NewJFrame**:
   * Clase principal que extiende JFrame para la interfaz de usuario del sistema de punto de venta.
   * Contiene métodos para manejar la tabla de productos, el cálculo de totales, la generación de facturas y la impresión.
2. **jTable1**:
   * Tabla que muestra los productos, sus cantidades y precios.
   * Permite al usuario ver y editar la lista de productos seleccionados.
3. **total** y **payp**:
   * Campos de texto donde el usuario ingresa el total de la compra y el monto pagado, respectivamente.
   * Usados para calcular el saldo restante.
4. **bal**:
   * Campo de texto que muestra el saldo restante después del pago.
5. **b**:
   * Área de texto que se utiliza para mostrar la factura antes de imprimirla.

**Métodos Principales**

**1. NewJFrame() - Constructor**

public NewJFrame() {

initComponents();

this.setExtendedState(JFrame.MAXIMIZED\_BOTH);

jTable1.getColumnModel().getColumn(0).setPreferredWidth(30);

jTable1.getColumnModel().getColumn(1).setPreferredWidth(200);

}

**Funcionalidad**: Inicializa los componentes de la GUI. Configura el tamaño de la ventana y establece los anchos preferidos de las columnas de la tabla de productos (jTable1).

**2. addtable(int id, String name, int qty, double price) - Agregar Producto a la Tabla**

public void addtable(int id ,String Name,int Qty ,Double Price){

DefaultTableModel dt = (DefaultTableModel) jTable1.getModel();

DecimalFormat df = new DecimalFormat("00.00");

double totPrice = Price \* Double.valueOf(Qty);

String TotalPrice = df.format(totPrice);

for (int row = 0; row < jTable1.getRowCount(); row++) {

if (Name == jTable1.getValueAt(row, 1)) {

dt.removeRow(jTable1.convertRowIndexToModel(row));

}

}

List<Object> row = new ArrayList<>();

row.add(id);

row.add(Name);

row.add(Qty);

row.add(TotalPrice);

dt.addRow(row.toArray());

}

**Funcionalidad**:

* + Añade un nuevo producto a la tabla (jTable1).
  + Si el producto ya existe en la tabla (se identifica por el nombre), lo actualiza con la nueva cantidad y precio.
  + Calcula el precio total del producto (cantidad \* precio) y lo formatea a dos decimales antes de agregarlo a la tabla.

**3. cal() - Calcular el Total**

public void cal(){

int numOfRow = jTable1.getRowCount();

double tot = 0.0;

for (int i = 0; i < numOfRow; i++) {

double value = Double.valueOf(jTable1.getValueAt(i, 3).toString());

tot += value;

}

DecimalFormat df = new DecimalFormat("00.00");

total.setText(df.format(tot));

}

**Funcionalidad**:

* + Recorre todas las filas de la tabla jTable1 y calcula la suma total de los productos.
  + Actualiza el campo de texto total con el valor total formateado a dos decimales.

**4. Bill() - Generar la Factura**

public void Bill() {

try {

b.setText(" Código Dapp FKD \n");

b.setText(b.getText() + " 589/ King Road, \n");

b.setText(b.getText() + " Colombo, Sri lanka, \n");

b.setText(b.getText() + " +57 3006746915, \n");

b.setText(b.getText() + "-------------------------------------------------------------------------\n");

b.setText(b.getText() + " Artículo \t\tcantidad \tPrecio" +"\n");

b.setText(b.getText() + "-------------------------------------------------------------------------\n");

DefaultTableModel df = (DefaultTableModel) jTable1.getModel();

for (int i = 0; i < jTable1.getRowCount(); i++) {

String Name = df.getValueAt(i, 1).toString();

String Qty = df.getValueAt(i, 2).toString();

String Price = df.getValueAt(i, 3).toString();

b.setText(b.getText() +" "+ Name+"\t\t"+Qty +"\t"+Price + "\n");

}

b.setText(b.getText() + "-------------------------------------------------------------------------\n");

b.setText(b.getText() + "Sub Total : " + total.getText() +"\n");

b.setText(b.getText() + "Dinero : " + payp.getText() +"\n");

b.setText(b.getText() + "Balance : " + bal.getText() +"\n");

b.setText(b.getText() + "-------------------------------------------------------------------------\n");

b.setText(b.getText() + " Gracias por su compra...!"+"\n");

b.setText(b.getText() + "-------------------------------------------------------------------------\n");

b.setText(b.getText() + " Tiendas KFC/c/Dappcode"+"\n");

b.print();

} catch (Exception e) {

JOptionPane.showMessageDialog(this, e, "Error", JOptionPane.ERROR\_MESSAGE);

}

}

**Funcionalidad**:

* + Genera una factura con todos los detalles de la compra, como nombre del artículo, cantidad, precio, subtotal, dinero pagado, y saldo.
  + Muestra un mensaje de agradecimiento y los detalles de la tienda.
  + Manda la factura a la impresora.

**5. pay() - Validar y Calcular el Pago**

public void pay(){

String totalText = total.getText();

String payText = payp.getText();

if (payText == null || payText.isEmpty()) {

JOptionPane.showMessageDialog(this, "Por favor ingresa valores válidos Pagar", "Input Error", JOptionPane.ERROR\_MESSAGE);

return;

}

Bill();

}

**Funcionalidad**:

* + Valida que se haya ingresado un monto en el campo payp (monto pagado).
  + Si el monto es válido, llama al método Bill() para generar la factura.

**6. jButton1ActionPerformed() - Incrementar Cantidad del Producto**

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

int i = Integer.valueOf(q1.getText());

++i;

q1.setText(String.valueOf(i));

addtable(1, "ZRW Combo ", i, 24000.0);

cal();

}

**Funcionalidad**:

* + Incrementa la cantidad del producto en el campo q1 y actualiza la tabla con la nueva cantidad.
  + Llama al método addtable() para actualizar la tabla de productos y al método cal() para recalcular el total.

**7. jButton13ActionPerformed() - Eliminar Producto Seleccionado**

private void jButton13ActionPerformed(java.awt.event.ActionEvent evt) {

DefaultTableModel dt = (DefaultTableModel) jTable1.getModel();

int selectedRow = jTable1.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Por favor seleccione una fila", "Error", JOptionPane.ERROR\_MESSAGE);

} else {

String r = dt.getValueAt(selectedRow, 0).toString();

dt.removeRow(selectedRow);

switch (r) {

case "1": q1.setText("0"); break;

case "2": q2.setText("0"); break;

case "3": q3.setText("0"); break;

case "4": q4.setText("0"); break;

case "5": q5.setText("0"); break;

case "6": q6.setText("0"); break;

case "7": q7.setText("0"); break;

case "8": q8.setText("0"); break;

case "9": q9.setText("0"); break;

default: System.out.println("Error: No se encontró el artículo");

}

cal();

}

}

**Funcionalidad**:

* + Elimina el producto seleccionado de la tabla (jTable1).
  + Después de eliminarlo, actualiza el total llamando al método cal().

Este sistema de punto de venta (POS) permite gestionar productos, realizar cálculos de total, manejar pagos, y generar e imprimir facturas. La interfaz gráfica está implementada con Swing, y el código está estructurado para facilitar la adición de nuevos productos, el cálculo automático de los totales y la impresión de las facturas.