UNIVERSIDAD PERUANA LOS ANDES

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PRACTICA N°3

ASIGNATURA: ARQUITECTURA DE SOFTWARE

DOCENTE: FERNÁNDEZ BEJARANO RAUL ENRIQUE

INTEGRANTE: AQUINO CANO, MOISES ISRAEL

CICLO: VI

HUANCAYO-2024
PERÚ

INFORME DE TRABAJO N°3

Título: Sistema de Gestión de Ventas al Contado y a Crédito

Mi proyecto consiste en un **Sistema de Gestión de Ventas** que permite a una tienda manejar la venta de productos mediante dos modalidades: **venta al contado** y **venta a crédito**. El sistema está diseñado utilizando el patrón **Modelo-Vista-Controlador (MVC)**, separando claramente la lógica de negocio de la interfaz de usuario. A través de una interfaz gráfica amigable, el usuario puede seleccionar productos, ingresar la cantidad requerida y calcular el costo total, aplicando los descuentos correspondientes en el caso de ventas al contado o calculando los intereses y cuotas mensuales en el caso de ventas a crédito. El sistema ofrece una manera eficiente y organizada de gestionar transacciones, facilitando la administración de ventas en diferentes escenarios comerciales.

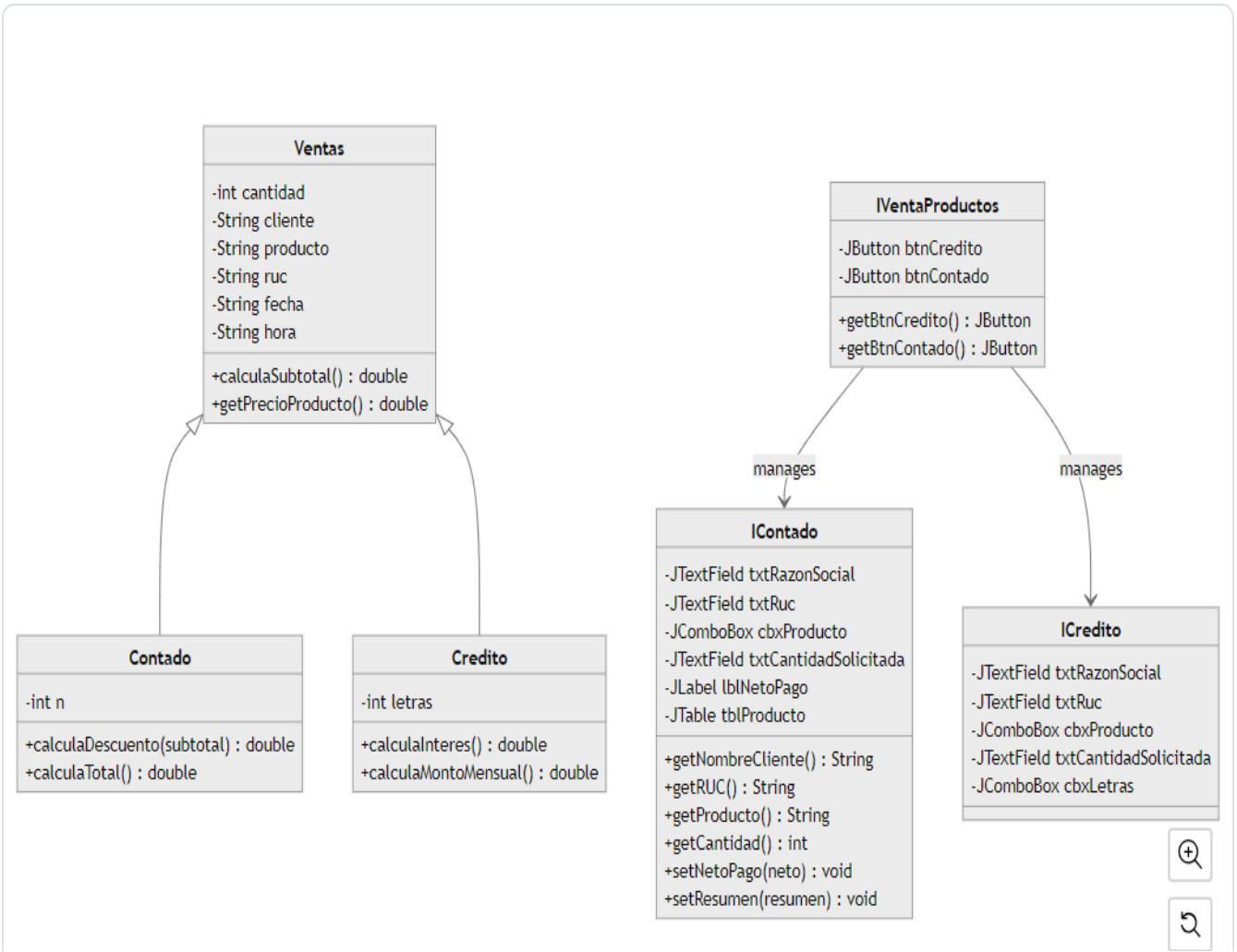
Requerimientos Funcionales:

1. **Ventas al Contado:**
    - El sistema debe permitir registrar ventas al contado, ingresando la información del cliente, el producto seleccionado y la cantidad solicitada.
    - El sistema debe calcular el subtotal de la venta, aplicar el descuento correspondiente según el monto y mostrar el total neto a pagar.
    - Debe mostrarse un resumen de la transacción, incluyendo los datos del cliente, el producto, el subtotal, el descuento y el total neto.
    - El sistema debe permitir visualizar una tabla con los productos vendidos, con columnas como descripción del producto, cantidad, precio y subtotal.
2. **Ventas a Crédito:**
    - El sistema debe permitir registrar ventas a crédito, ingresando la información del cliente, el producto seleccionado, la cantidad solicitada y el número de letras (cuotas).
    - El sistema debe calcular el subtotal de la venta y el monto mensual a pagar, basado en el número de cuotas seleccionadas.
    - Debe mostrarse un resumen de las letras, indicando el monto mensual que el cliente deberá pagar.

- El sistema debe permitir visualizar una tabla con los productos vendidos, incluyendo columnas como descripción del producto, cantidad, precio y subtotal.

3. **Interfaz de Usuario:**
   - El sistema debe ofrecer una interfaz gráfica con botones para seleccionar entre las opciones de venta al contado y venta a crédito.
   - Debe incluir campos para ingresar la información del cliente, seleccionar el producto y la cantidad, y mostrar el resumen y los detalles de la venta.
   - El sistema debe permitir al usuario regresar a la ventana principal y salir de la aplicación en cualquier momento.

DIAGRAMA DE CLASES



MODELO

CLASE VENTA
Esta clase actúa como la clase base para las ventas. Representa los atributos y métodos comunes a todas las ventas, como la cantidad de productos, nombre del cliente, producto, RUC, fecha y hora de la venta.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package Modelo;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

/**
 *
 * @author USER 17
 */
public class Ventas {
  int cantidad;
   String cliente, fecha, hora, producto, ruc;

   public Ventas() {
   }

   public Ventas(int cantidad, String cliente, String producto, String ruc) {
       this.cantidad = cantidad;
       this.cliente = cliente;
       this.producto = producto;
       this.ruc = ruc;
       // Se obtiene la fecha y hora automáticamente
       DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy");
       DateTimeFormatter htf = DateTimeFormatter.ofPattern("HH:mm:ss");
```

```
        this.fecha = dtf.format(LocalDateTime.now());
        this.hora = htf.format(LocalDateTime.now());
    }

    public int getCantidad() {
        return cantidad;
    }

    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }

    public String getCliente() {
        return cliente;
    }

    public void setCliente(String cliente) {
        this.cliente = cliente;
    }

    public String getFecha() {
        return fecha;
    }

    public String getHora() {
        return hora;
    }

    public String getProducto() {
        return producto;
    }

    public void setProducto(String producto) {
        this.producto = producto;
    }

    public String getRuc() {
        return ruc;
    }
```

```
    public void setRuc(String ruc) {
        this.ruc = ruc;
    }

    public double calculaSubtotal() throws Exception {
        return getCantidad() * getPrecioProducto();
    }

    public double getPrecioProducto() throws Exception {
        switch (getProducto().toLowerCase()) {
            case "lavadora":
                return 1500.00;
            case "refrigeradora":
                return 3500.00;
            case "licuadora":
                return 500.00;
            case "extractor":
                return 150.00;
            case "radiograbadora":
                return 750.00;
            case "dvd":
                return 100.00;
            case "blue ray":
                return 250.00;
            default:
                throw new Exception("Producto no encontrado");
        }
    }
}
```

CLASE CONTADO

Esta clase gestiona las ventas al contado y añade la capacidad de aplicar descuentos basados en el monto subtotal de la venta.

```
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package Modelo;

/**
 *
 * @author USER 17
 */
public class Contado extends Ventas{

    private int n;  // Variable para la cantidad de productos

    // Constructor que recibe la cantidad de productos y otros datos
    public Contado(int cantidad, String cliente, String producto, String ruc) {
        super(cantidad, cliente, producto, ruc);
        this.n = cantidad;  // Asignamos la cantidad de productos a la variable n
    }

    // Método getN para devolver la cantidad de productos
    public int getN() {
        return n;
    }

    // Método para calcular el descuento basado en el monto subtotal
    public double calculaDescuento(double subtotal) {
        if (subtotal < 1000) {
            return subtotal * 0.05;  // 5% de descuento
```

```
        } else if (subtotal >= 1000 && subtotal <= 3000) {
            return subtotal * 0.08;  // 8% de descuento
        } else {
            return subtotal * 0.12;  // 12% de descuento
        }
    }

    // Método para calcular el total con descuento
    public double calculaTotal() throws Exception {
        double subtotal = calculaSubtotal();  // Calculamos el subtotal usando el método de Venta
        double descuento = calculaDescuento(subtotal);  // Calculamos el descuento
        return subtotal - descuento;  // Retornamos el total a pagar con el descuento aplicado
    }
}
```

CLASE CREDITO
Esta clase gestiona las ventas a crédito, añadiendo la capacidad de calcular el interés y el monto mensual basado en la cantidad de letras (cuotas) en las que se pagará la venta.

```
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package Modelo;

/**
 *
 * @author USER 17
 */
public class Credito extends Ventas {
     private int letras;  // Cantidad de letras para el crédito
    private int x;       // Cantidad de productos (variable solicitada)

    // Constructor que inicializa la cantidad de productos, cliente, producto,
RUC y las letras
    public Credito(int cantidad, String cliente, String producto, String ruc, int
letras) {
        super(cantidad, cliente, producto, ruc);
        this.letras = letras;
        this.x = cantidad;  // Asignamos la cantidad de productos a la variable x
    }

    // Método para obtener la cantidad de productos (X)
    public int getX() {
        return x;
    }

    // Método para obtener la cantidad de letras
    public int getLetras() {
```

```
        return letras;
    }

    // Método para establecer la cantidad de letras
    public void setLetras(int letras) {
        this.letras = letras;
    }

    // Método para calcular el interés basado en la tabla proporcionada
    public double calculaInteres() throws Exception {
        double subtotal = calculaSubtotal();
        if (subtotal < 1000) {
            return subtotal * 0.03;  // Interés del 3% si es menor a 1000
        } else if (subtotal >= 1000 && subtotal <= 3000) {
            return subtotal * 0.05;  // Interés del 5% entre 1000 y 3000
        } else {
            return subtotal * 0.08;  // Interés del 8% si es mayor a 3000
        }
    }

    // Método para calcular el monto mensual del crédito basado en las letras
    public double calculaMontoMensual() throws Exception {
        double subtotal = calculaSubtotal();  // Calculamos el subtotal
        double totalConInteres = subtotal + calculaInteres();  // Total con interés
        return totalConInteres / getLetras();  // Monto mensual
    }
}
```

VISTA

JFRAME IVENTACONTADO

Esta clase es una interfaz gráfica que permite gestionar las ventas de productos, ofreciendo dos opciones principales: venta al contado y venta al crédito.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package Vista;

/**
 *
 * @author USER 17
 */
public class IVentaProductos extends javax.swing.JFrame {

    // Método para obtener el botón de ventas a crédito
    public javax.swing.JButton getBtnCredito() {
        return btnCredito;
    }

    // Método para obtener el botón de ventas al contado
    public javax.swing.JButton getBtnContado() {
        return btnContado;
    }

    /**
     * Creates new form IVenta
     */
    public IVentaProductos() {
        initComponents();
    }
```

JFRAME ICONTADO

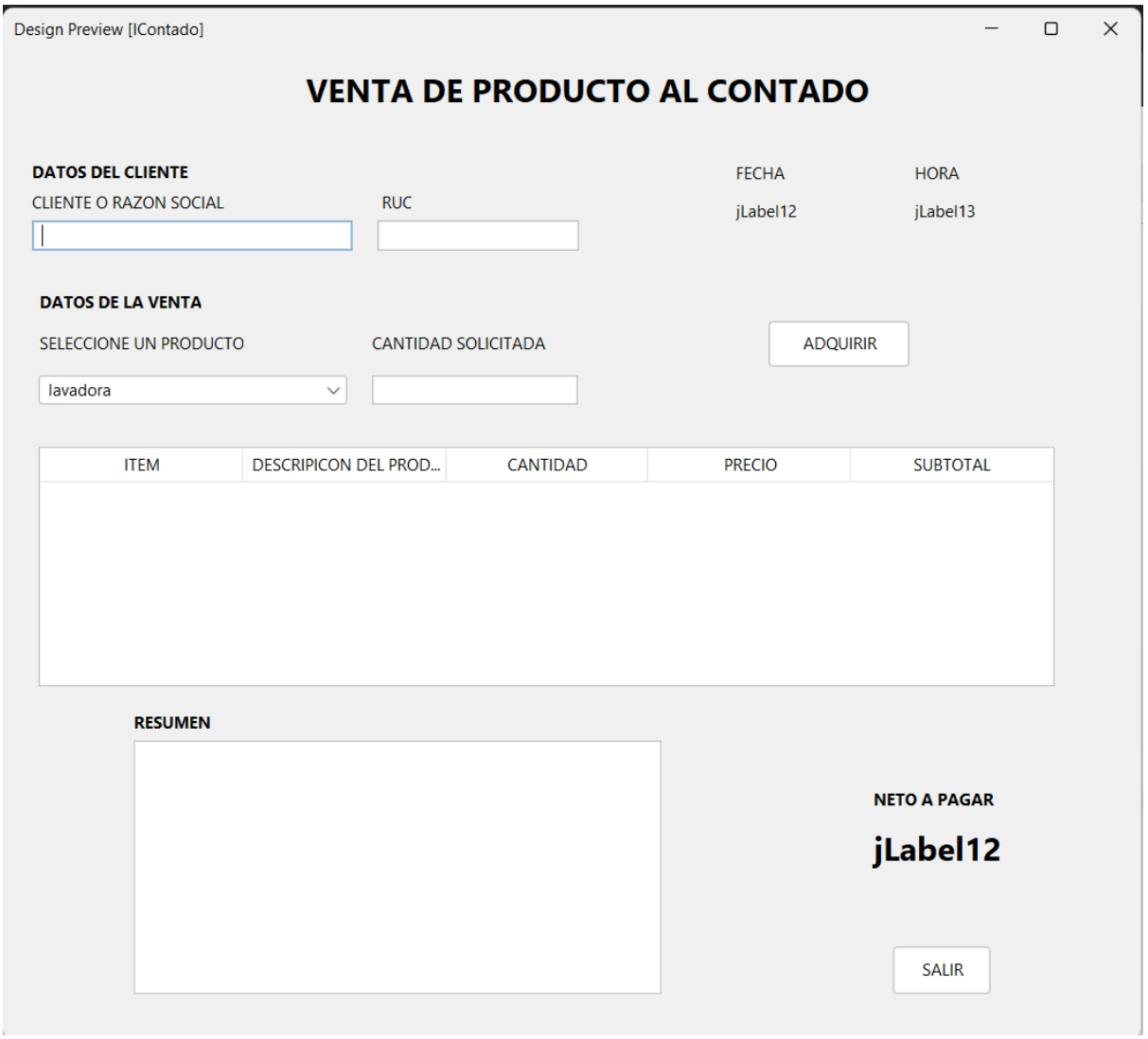
Esta clase es la interfaz gráfica que gestiona las ventas al contado. Proporciona campos y componentes necesarios para registrar una venta y calcular el monto neto a pagar.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package Vista;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

/**
 *
 * @author USER 17
 */
public class IContado extends javax.swing.JFrame {

    // Método para obtener el nombre del cliente o razón social
    public String getNombreCliente() {
        return txtRazonSocial.getText();
```

```
}

// Método para obtener el RUC del cliente
public String getRUC() {
    return txtRuc.getText();
}

// Método para obtener el producto seleccionado en el combo box
public String getProducto() {
    return cbxProducto.getSelectedItem().toString();
}

// Método para obtener la cantidad solicitada
public int getCantidad() {
    return Integer.parseInt(txtCantidadSolicitada.getText());
}

// Método para obtener el neto a pagar
public double getNetoPago() {
    return Double.parseDouble(lblNetoPago.getText());
}

// Método para establecer el valor del neto a pagar
public void setNetoPago(double neto) {
    lblNetoPago.setText(String.valueOf(neto));
}

// Método para establecer el texto del resumen de la venta
public void setResumen(String resumen) {
    txpResumen.setText(resumen);
}

// Método para obtener la fecha
public javax.swing.JLabel getLblFecha() {
    return lblFecha;
}

// Método para obtener la hora
public javax.swing.JLabel getLblHora() {
    return lblHora;
```

```
    }

    // Método para obtener el JLabel lblNetoPago
    public javax.swing.JLabel getLblNetoPago() {
        return lblNetoPago;
    }

    public javax.swing.JButton getBtnSalir(){
        return btnSalir;
    }

    // Método para obtener el modelo de la tabla
    public javax.swing.JTable getTblProducto() {
        return tblProducto;
    }

    // Método para limpiar el formulario después de una operación
    public void limpiarFormulario() {
        txtRazonSocial.setText("");
        txtRuc.setText("");
        txtCantidadSolicitada.setText("");
        lblNetoPago.setText("0.00");
        txpResumen.setText("");
    }

    // Método para mostrar mensajes al usuario (por ejemplo, una
confirmación)
    public void mostrarMensaje(String mensaje) {
        javax.swing.JOptionPane.showMessageDialog(this, mensaje);
    }

    // Método para obtener el botón "Adquirir"
    public javax.swing.JButton getBtnAdquirir() {
        return btnAdquirir;
    }



    /**
     * Creates new form IContado
```

```
     */
    public IContado() {
        initComponents();
         // Inicializar fecha y hora actual
        DateTimeFormatter dtfFecha =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
        DateTimeFormatter dtfHora =
DateTimeFormatter.ofPattern("HH:mm:ss");
        lblFecha.setText(dtfFecha.format(LocalDateTime.now()));
        lblHora.setText(dtfHora.format(LocalDateTime.now()));
    }
```



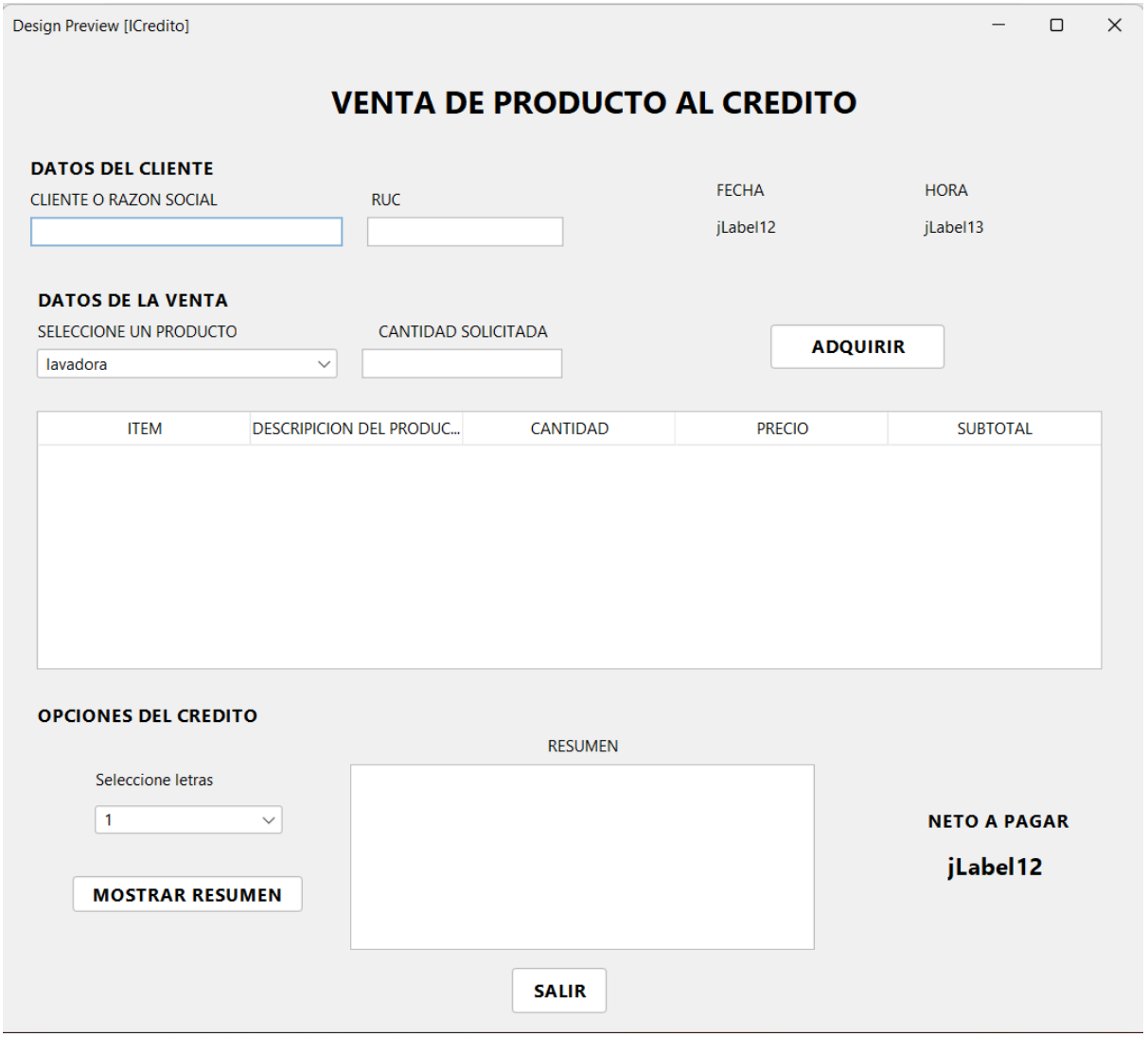
JFRAME ICREDITO

Esta clase es la interfaz gráfica para las ventas a crédito. Proporciona componentes necesarios para registrar la información del cliente y la venta, además de calcular el total con el interés.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package Vista;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import javax.swing.JOptionPane;

/**
 *
 * @author USER 17
 */
    public class ICredito extends javax.swing.JFrame {

    // Método para mostrar mensajes al usuario (por ejemplo, error o confirmación)
    public void mostrarMensaje(String mensaje) {
        JOptionPane.showMessageDialog(this, mensaje);
    }

    // Método para obtener el valor de lblNetoPago
    public javax.swing.JLabel getLblNetoPago() {
        return lblNetoPago;  // Retorna el JLabel que muestra el neto a pagar
    }

        // Métodos Getters y Setters
    public String getTxtRazonSocial() {
        return txtRazonSocial.getText();
    }
```

```
public String getTxtRuc() {
    return txtRuc.getText();
}

public String getCbxProducto() {
    return cbxProducto.getSelectedItem().toString();
}

public int getCantidad() {
    return Integer.parseInt(txtCantidadSolicitada.getText());
}

public int getCbxLetras() {
    return Integer.parseInt(cbxLetras.getSelectedItem().toString());
}

public javax.swing.JTable getTblProducto() {
    return tblProducto;
}

public javax.swing.JTextPane getTxpResumenCredito() {
    return txpResumenCredito;  // Manteniendo txpResumenCredito
}


public void setLblNetoPago(double neto) {
    lblNetoPago.setText(String.valueOf(neto));
}

public javax.swing.JLabel getLblFecha() {
    return lblFechaActual;
}

public javax.swing.JLabel getLblHora() {
    return lblHoraActual;
}

public javax.swing.JButton getBtnAdquirir() {
    return btnAdquirir;
```

```
    }

    public javax.swing.JButton getBtnMostrarResumen() {
        return btnMostrarResumen;
    }

    public void setResumen(String texto) {
        txpResumenCredito.setText(texto);  // txpResumenCredito es el
JTextPane donde se muestra el resumen
    }


    public javax.swing.JButton getBtnSalir() {
        return btnSalir;
    }

    /**
     * Creates new form ICredito
     */
    public ICredito() {
        initComponents();
        // Inicializar fecha y hora actual
        DateTimeFormatter dtfFecha =
DateTimeFormatter.ofPattern("dd/MM/yyyy");
        DateTimeFormatter dtfHora =
DateTimeFormatter.ofPattern("HH:mm:ss");
        lblFechaActual.setText(dtfFecha.format(LocalDateTime.now()));
        lblHoraActual.setText(dtfHora.format(LocalDateTime.now()));
    }
```


Design Preview [ICredito]
VENTA DE PRODUCTO AL CREDITO
DATOS DEL CLIENTE
CLIENTE O RAZON SOCIAL
RUC
FECHA
HORA
jLabel12
jLabel13
DATOS DE LA VENTA
SELECCIONE UN PRODUCTO
lavadora
CANTIDAD SOLICITADA
ADQUIRIR
ITEM
DESCRIPICION DEL PRODUC...
CANTIDAD
PRECIO
SUBTOTAL
OPCIONES DEL CREDITO
RESUMEN
Seleccione letras
1
MOSTRAR RESUMEN
NETO A PAGAR
jLabel12
SALIR

CONTROLADOR

CONTROLADOR
Esta clase gestiona la lógica del flujo de trabajo entre la vista (interfaz gráfica) y el modelo (datos de la aplicación) en el patrón MVC. El controlador maneja los eventos de la interfaz de usuario y realiza las acciones necesarias para procesar la información.

```
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package Controlador;

import Modelo.Contado;
import Modelo.Credito;
import Vista.IVentaProductos;
import Vista.ICredito;
import Vista.IContado;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import javax.swing.table.DefaultTableModel;

/**
 *
 * @author USER 17
 */
public class Controlador {
    private IVentaProductos vistaPrincipal;

    public Controlador(IVentaProductos vista) {
        this.vistaPrincipal = vista;

        // Asignar eventos a los botones de la vista principal
        this.vistaPrincipal.getBtnCredito().addActionListener(e -> abrirCredito());
        this.vistaPrincipal.getBtnContado().addActionListener(e ->
abrirContado());
```

```
    }

    // Método para abrir la ventana de ventas a crédito
    private void abrirCredito() {
        ICredito creditoVista = new ICredito();  // Instanciamos la ventana de
crédito
        creditoVista.setVisible(true);          // Mostramos la ventana de crédito
        vistaPrincipal.setVisible(false);       // Ocultamos la ventana actual

        // Asignar eventos a los botones de la vista de crédito
        creditoVista.getBtnAdquirir().addActionListener(e ->
procesarVentaCredito(creditoVista));
        creditoVista.getBtnMostrarResumen().addActionListener(e ->
mostrarResumenCredito(creditoVista));
        creditoVista.getBtnSalir().addActionListener(e -> salir(creditoVista));  //
Llamamos al método salir y pasamos la ventana actual
    }

    // Método para abrir la ventana de ventas al contado
    private void abrirContado() {
        IContado contadoVista = new IContado();  // Instanciamos la ventana de
contado
        contadoVista.setVisible(true);          // Mostramos la ventana de contado
        vistaPrincipal.setVisible(false);       // Ocultamos la ventana actual

        // Asignar eventos a los botones de la vista de contado
        contadoVista.getBtnAdquirir().addActionListener(e ->
procesarVenta(contadoVista));
        contadoVista.getBtnSalir().addActionListener(e -> salir(contadoVista));  //
Llamamos al método salir y pasamos la ventana actual
    }

    // Método actualizado para manejar el evento de salir en ICredito y
IContado
    private void salir(javax.swing.JFrame ventanaActual) {
        ventanaActual.setVisible(false); // Ocultar la ventana actual
        vistaPrincipal.setVisible(true); // Mostrar la ventana principal
(IVentaProductos)
    }
```

```
    // Método que maneja la lógica del botón "Adquirir" para ventas al contado
    private void procesarVenta(IContado vistaContado) {
        try {
            // Configurar el formato para mostrar el símbolo de moneda y dos
decimales
            DecimalFormatSymbols simbolos = new DecimalFormatSymbols();
            simbolos.setDecimalSeparator('.');
            simbolos.setGroupingSeparator(',');
            DecimalFormat df = new DecimalFormat("$ #,##0.00", simbolos);

            // 1. Obtener los datos desde la vista
            String cliente = vistaContado.getNombreCliente();  // Obtiene el
nombre del cliente
            String ruc = vistaContado.getRUC();  // Obtiene el RUC del cliente
            String producto = vistaContado.getProducto();  // Obtiene el producto
seleccionado
            int cantidad = vistaContado.getCantidad();  // Obtiene la cantidad
seleccionada

            // 2. Crear una instancia de la clase Contado (Modelo)
            Contado venta = new Contado(cantidad, cliente, producto, ruc);

            // 3. Calcular el subtotal y el total con descuento
            double subtotal = venta.calculaSubtotal();
            double descuento = venta.calculaDescuento(subtotal);
            double total = venta.calculaTotal();

            // 4. Añadir el producto a la tabla en la vista
            DefaultTableModel model = (DefaultTableModel)
vistaContado.getTblProducto().getModel();
            model.addRow(new Object[]{
                model.getRowCount() + 1,  // Item (número de la fila)
                producto,  // Descripción del producto
                cantidad,  // Cantidad seleccionada
                df.format(venta.getPrecioProducto()),  // Precio formateado
                df.format(subtotal)  // Subtotal formateado
            });

            // 5. Actualizar el resumen de la venta en la vista
            vistaContado.setResumen("** RESUMEN DE VENTA **\n"
```

```
                + "------------------------------\n"
                + "CLIENTE: " + cliente + "\n"
                + "RUC: " + ruc + "\n"
                + "FECHA: " + vistaContado.getLblFecha().getText() + "\n"
                + "HORA: " + vistaContado.getLblHora().getText() + "\n"
                + "------------------------------\n"
                + "SUBTOTAL: " + df.format(subtotal) + "\n"
                + "DESCUENTO: " + df.format(descuento) + "\n"
                + "NETO: " + df.format(total));

            // 6. Actualizar el valor del neto a pagar en el label "lblNetoPago"
            vistaContado.getLblNetoPago().setText(df.format(total));

        } catch (Exception ex) {
            // Mostrar mensaje de error en caso de excepción
            vistaContado.mostrarMensaje("Error al procesar la venta: " +
ex.getMessage());
        }
    }

    // Método que maneja la lógica del botón "Adquirir" en ICredito
private void procesarVentaCredito(ICredito vistaCredito) {
    try {
        // Configurar el formato para mostrar el símbolo de moneda y dos
decimales
        DecimalFormatSymbols simbolos = new DecimalFormatSymbols();
        simbolos.setDecimalSeparator('.');
        simbolos.setGroupingSeparator(',');
        DecimalFormat df = new DecimalFormat("$ #,##0.00", simbolos);

        // 1. Obtener los datos desde la vista
        String cliente = vistaCredito.getTxtRazonSocial();  // Obtener nombre del
cliente
        String ruc = vistaCredito.getTxtRuc();              // Obtener RUC del cliente
        String producto = vistaCredito.getCbxProducto();    // Obtener el producto
seleccionado
        int cantidad = vistaCredito.getCantidad();          // Obtener la cantidad
seleccionada
        int letras = vistaCredito.getCbxLetras();           // Obtener el número de
letras de pago
```

```
        // 2. Crear una instancia de la clase Credito (Modelo)
        Credito venta = new Credito(cantidad, cliente, producto, ruc, letras);

        // 3. Calcular el subtotal y el monto mensual
        double subtotal = venta.calculaSubtotal();
        double totalMensual = venta.calculaMontoMensual();

        // 4. Añadir el producto a la tabla en la vista
        DefaultTableModel model = (DefaultTableModel)
vistaCredito.getTblProducto().getModel();
        model.addRow(new Object[]{
            model.getRowCount() + 1,  // Item (número de la fila)
            producto,                 // Descripción del producto
            cantidad,                 // Cantidad seleccionada
            df.format(venta.getPrecioProducto()),  // Precio formateado
            df.format(subtotal)       // Subtotal formateado
        });

        // 5. Actualizar el valor del neto a pagar en el label "lblNetoPago"
        vistaCredito.getLblNetoPago().setText(df.format(subtotal));  // Mostrar el
subtotal como neto a pagar al inicio

    } catch (Exception ex) {
        vistaCredito.mostrarMensaje("Error al procesar la venta: " +
ex.getMessage());
    }
}

// Método para mostrar el resumen de las letras de pago
private void mostrarResumenCredito(ICredito vistaCredito) {
    try {
        // Configurar el formato de moneda
        DecimalFormatSymbols simbolos = new DecimalFormatSymbols();
        simbolos.setDecimalSeparator('.');
        simbolos.setGroupingSeparator(',');
        DecimalFormat df = new DecimalFormat("$ #,##0.00", simbolos);

        // Obtener la cantidad de letras y el subtotal
        int letras = vistaCredito.getCbxLetras();
```

```
        double total =
Double.parseDouble(vistaCredito.getLblNetoPago().getText().replace("$",
"").replace(",", ""));  // Obtener el total del neto

        // Calcular el monto mensual de acuerdo con las letras
        double montoMensual = total / letras;

        // Construir el resumen de letras en el JTextPane
        StringBuilder resumen = new StringBuilder();
        resumen.append("** RESUMEN DE PAGOS **\n")
                .append("--------------------------\n");

        // Mostrar la distribución de letras
        for (int i = 1; i <= letras; i++) {
            resumen.append("Letra ").append(i).append(":
").append(df.format(montoMensual)).append("\n");
        }

        // Mostrar el resumen en la vista
        vistaCredito.getTxpResumenCredito().setText(resumen.toString());

    } catch (Exception ex) {
        vistaCredito.mostrarMensaje("Error al mostrar el resumen: " +
ex.getMessage());
    }
}


    // Método para manejar el evento de salir
    private void salir() {
        System.exit(0);  // Cierra la aplicación
    }
}
```

CLASE PRINCIPAL

Esta clase es el punto de entrada principal de la aplicación. Se encarga de inicializar la vista principal y asociarla con el controlador.

```
/*
 * Click
nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 */

package com.mycompany.ejemplo04;

import Controlador.Controlador;
import Vista.IVentaProductos;

/**
 *
 * @author USER 17
 */
public class Ejemplo04 {

    public static void main(String[] args) {
        IVentaProductos vista = new IVentaProductos();  // Instanciamos la vista
principal
        Controlador controlador = new Controlador(vista);  // Controlador para
manejar la navegación
        vista.setVisible(true);  // Mostrar la vista principal
    }
}
```