MANUAL TÉCNICO

PROYECTO #1

Ángel Andrés Godoy Valdéz

Laboratorio IPC 1 Sección "B"

Carnet: 202113539

INTRODUCCIÓN

Este manual describe los procesos detrás del manejo de datos en la interfaz gráfica de la aplicación denominada “Banco Virtual” en formato java realizada así como el código fuente que fue necesario para el proyecto No. 1 para el Laboratorio del curso Introducción a la Programación de Computadoras 1.

REQUERIMIENTOS TÉCNICOS

Sistema Operativo: Windows 7/Vista/8/8.1/10/11

Aplicaciones: Java Development Kit (JDK) 12.0-actual

HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO

Java:

Java es una plataforma informática de lenguaje de programación creada por Sun Microsystems en 1995. Ha evolucionado desde sus humildes comienzos hasta impulsar una gran parte del mundo digital actual, ya que es una plataforma fiable en la que se crean muchos servicios y aplicaciones. Los nuevos e innovadores productos y servicios digitales diseñados para el futuro también siguen basándose en Java.

Aunque la mayoría de aplicaciones Java modernas combinan el tiempo de ejecución y la aplicación de Java, todavía existen algunas aplicaciones e incluso sitios web que no funcionan sin instalar Java para escritorio. El sitio web Java.com está pensado para consumidores que todavía necesitan Java en sus aplicaciones de escritorio, sobre todo las aplicaciones que tienen como destino Java 8.

Intellij IDEA:

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) para el desarrollo de programas informáticos. Es desarrollado por JetBrains (anteriormente conocido como IntelliJ), y está disponible en dos ediciones: edición para la comunidad y edición comercial.

Cada aspecto de IntelliJ IDEA ha sido diseñado para maximizar la productividad del desarrollador. Juntos, la asistencia de codificación inteligente y el diseño ergonómico hacen que el desarrollo no solo sea productivo sino también agradable.

Después de que IntelliJ IDEA haya indexado su código fuente, ofrece una experiencia increíblemente rápida e inteligente al brindar sugerencias relevantes en cada contexto: finalización de código instantánea e inteligente, análisis de código sobre la marcha y herramientas de refactorización confiables. Las herramientas de misión crítica, como los sistemas de control de versiones integrados y una amplia variedad de lenguajes y marcos compatibles, están todos a mano, sin complicaciones

de complementos incluidos. Mientras que la finalización básica sugiere nombres de clases, métodos, campos y palabras clave dentro del ámbito de visibilidad, la finalización inteligente sugiere solo los tipos que se esperan en el contexto actual.

EJECUCIÓN DE APLICACIÓN DESDE SÍMBOLO DE SISTEMA

Una vez descargado desde el repositorio de github, se debe encontrar la ruta de acceso para el jar, el cual se encuentra ubicado dentro de la carpeta .idea/artifacts.

Pasos a realizar:

1. Presionar Windows+R
2. Teclear “cmd” (esto abrira el símbolo del sistema)
3. Dentro del cmd teclear “java -jar <ruta del jar>”
4. Presionar Enter
5. ¡Listo, ya está dentro de la aplicación!

APLICACIÓN (EJECUTABLE JAR)

Clases

Cada frame que representa cada una de las pantallas es trabajado por separado debido a que así lo trabaja la interfaz drag and drop de Intellij Idea, adicional se crearon las siguientes clases auxiliares al método Main que interactúan de alguna forma con los datos ingresados.

Clientes

Bajo esta clase se maneja el CUI, Nombre y Apellido de los clientes.

```
34 usages  Angel Andres Godoy Valdez
public class Clientes {
    17 usages
    String CUI=" ";
    92 usages
    String Nombre=" ";
    92 usages
    String Apellido=" ";



    10 usages  Angel Andres Godoy Valdez
    public String getCUI() { return CUI; }

    5 usages  Angel Andres Godoy Valdez
    public void setCUI(String CUI) { this.CUI = CUI; }

    1 usage  Angel Andres Godoy Valdez
    public String getNombre() { return Nombre; }

    5 usages  Angel Andres Godoy Valdez
    public void  setNombre(String nombre) { Nombre = nombre; }

    Angel Andres Godoy Valdez
    public String getApellido() { return Apellido; }

    5 usages  Angel Andres Godoy Valdez
    public void setApellido(String apellido) { Apellido = apellido; }
```

CrearTransaccion

Dentro de esta clase se almacenan los métodos correspondientes para llenar las transacciones realizadas en la aplicación correspondiente a cada cliente así como su detalle y fecha para su posterior uso en el frame Historial, esto se logra a través de arreglos de una dimensión en los cuales se coloca la información necesaria y se aumenta un conteo que al llegar a las transacciones máximas ya no regresa ningún dato.

```
public void LlenarTrCliente(int op,String Detalle, double Debito, double Credito, double SaldoDisponible){
      switch (op){
          case (1):
              if(ContTr1<20){
                  Main.JtCliente1[0]=String.valueOf(GenerarID());
                  Main.JtCliente1[1]=String.valueOf(fechaStr);
                  Main.JtCliente1[2]=String.valueOf(Detalle);
                  Main.JtCliente1[3]=String.valueOf(Debito);
                  Main.JtCliente1[4]=String.valueOf(Credito);
                  Main.JtCliente1[5]=String.valueOf(SaldoDisponible);


                  Main.modeloCl1.addRow(Main.JtCliente1);


                  ContTr1++;
```

Cuentas

Parecida a la clase clientes, esta toma la función de almacenar el cliente, saldo de la cuenta y su ID único. Únicamente cuenta con un método el cual con ayuda de la librería Random es capaz de generar un ID de 6 dígitos y verificar si este se repite o no, para posteriormente asignarlo a la cuenta.

```
5 usages  Angel Andres Godoy Valdez
public String GenerarID() {
    boolean cuentaRepetida = true;
    String IdCuenta;
    do {

        int d1 = random.nextInt( bound: 10);
        int d2 = random.nextInt( bound: 10);
        int d3 = random.nextInt( bound: 10);
        int d4 = random.nextInt( bound: 10);
        int d5 = random.nextInt( bound: 10);
        int d6 = random.nextInt( bound: 10);
        int cuentasR = 0;
        IdCuenta = String.valueOf(d1) + String.valueOf(d2) + String.valueOf(d3) + String.valu
        for (int i = 0; i < Main.arrayCuentas.length; i++) {
            if (IdCuenta.equals(Main.arrayCuentas[i])) {
                cuentasR++;
            }
        }
        if (cuentasR == 0) {
            cuentaRepetida = false;
        }

    } while (cuentaRepetida);

 return IdCuenta;
}
```

Display Frames

Esta clase contiene sólo métodos que llaman a las clases que contienen la información de los distintos frames que se usan en la aplicación, para generar su “JFrame” y definir sus propiedades básicas como el título, la visibilidad, dimensión, close operation, y si se puede modificar su tamaño o no.

```
1 usage  Angel Andres Godoy Valdez
public void PantallaVerClientes(boolean mostrar) {

    JFrame frame1 = new VerClientesFrame( title: "Menu Tabla Clientes");
    ImageIcon icon = new ImageIcon( filename: "icono.png");
    frame1.setVisible(mostrar);
    frame1.setSize( width: 800,  height: 600);
    frame1.setTitle("Ver clientes");
    frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame1.setResizable(false);
    frame1.setIconImage(icon.getImage());


}
```

Main Frame

Implementa dos Jbutton con sus respectivos action listener, “aboutBtn” que al ser presionado lanza un cuadro de texto con mi nombre completo y carnet, y “loginBtn” que redirecciona a la Pantalla Login.

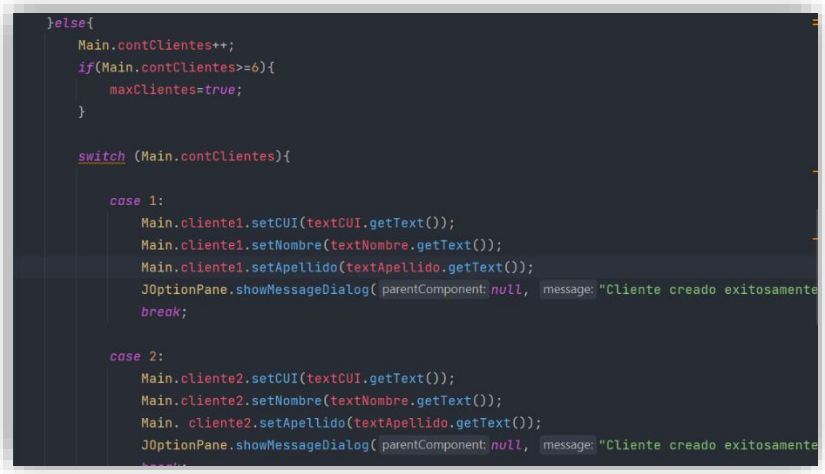
```
Angel Andres Godoy Valdez
aboutBtn.addActionListener(new ActionListener() {
    Angel Andres Godoy Valdez
    @Override
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: "Ángel Andrés Godoy Valdéz, 202113539 ",
    }
});
Angel Andres Godoy Valdez
loginBtn.addActionListener(new ActionListener() {
    Angel Andres Godoy Valdez
    @Override
    public void actionPerformed(ActionEvent e) {
        d.PantallaLogin();
        dispose();
    }
});
```

ChooseYourCharacterFrame

Dentro d esta clase se despliegan los métodos de DisplayFrame para el resto de los Frames, en esta ventana únicamente se encuentran los respectivos JFrames que en conjunto crean un menú de interacción para hacer más amigable los procesos que se llevan a cabo.

CrearFrame

En esta clase se contiene las funciones para crear un cliente, con el uso de “if” y “switch” se generan contadores para saber la cantidad de clientes existentes, al llegar a 5 el contador se detiene y así lo hace el swtich que hace posible la creación de clientes.



MenuSelClientes

Clase encargada de crear las cuentas para cada cliente, de igual forma que con los clientes la información se procesa con un switch y con contadores en la clase Main que permanecen static.

```
crearClienteBtn.addActionListener(new ActionListener() {
    Angel Andres Godoy Valdez
    @Override
    public void actionPerformed(ActionEvent e) {
        if(Main.arrayClientes[comboBoxClientes.getSelectedIndex()].getNombre()==null){
            JOptionPane.showMessageDialog( parentComponent: null,  message: "No hay ningún cliente seleccionado.",
        }else{
            switch (comboBoxClientes.getSelectedIndex()){
                case(0):
                    if(Main.contCuentas1>=5){
                        JOptionPane.showMessageDialog( parentComponent: null,  message: "No es posible crear más cu
                    }else{
                        Main.arrayCuentas[Main.contCuentas1].ID = c.GenerarID();
                        Main.contCuentas1++;
                        JOptionPane.showMessageDialog( parentComponent: null,  message: "Cuenta creada exitosamente
                    }
                    break;

                case (1):
                    if(Main.contCuentas2>=5){
                        JOptionPane.showMessageDialog( parentComponent: null,  message: "No es posible crear más cu
                    }else{
                        Main.arrayCuentas[5+Main.contCuentas2].ID = c.GenerarID();
                        Main.contCuentas2++;
                        JOptionPane.showMessageDialog( parentComponent: null,  message: "Cuenta creada exitosamente
                    }
```

VerClientes

Esta clase mediante el uso de arreglos y JTables despliega en pantalla la información de los clientes existentes.

```
public VerClientesFrame (String title){
    super(title);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setContentPane(verClientesFrame);
    buscarCuentasBtn.setFocusable(false);
    buscarCuentasBtn.setBorder(BorderFactory.createMatteBorder( top: 2, left: 2, bottom: 2, right: 2, Color.black));
    volverBtn.setFocusable(false);
    volverBtn.setBorder(BorderFactory.createMatteBorder( top: 2, left: 2, bottom: 2, right: 2, Color.black));
    textCUI.setHorizontalAlignment(JTextField.CENTER);



        String[]f1 = {Main.cliente1.CUI,Main.cliente1.Nombre,Main.cliente1.Apellido};
        String[]f2 = {Main.cliente2.CUI,Main.cliente2.Nombre,Main.cliente2.Apellido};
        String[]f3 = {Main.cliente3.CUI,Main.cliente3.Nombre,Main.cliente3.Apellido};
        String[]f4 = {Main.cliente4.CUI,Main.cliente4.Nombre,Main.cliente4.Apellido};
        String[]f5 = {Main.cliente5.CUI,Main.cliente5.Nombre,Main.cliente5.Apellido};


    DefaultTableModel modelo = new DefaultTableModel();
    modelo.addColumn( columnName: "CUI");
    modelo.addColumn( columnName: "Nombre");
    modelo.addColumn( columnName: "Apellido");
    modelo.addRow(f1);
    modelo.addRow(f2);
    modelo.addRow(f3);
    modelo.addRow(f4);
    modelo.addRow(f5);
```

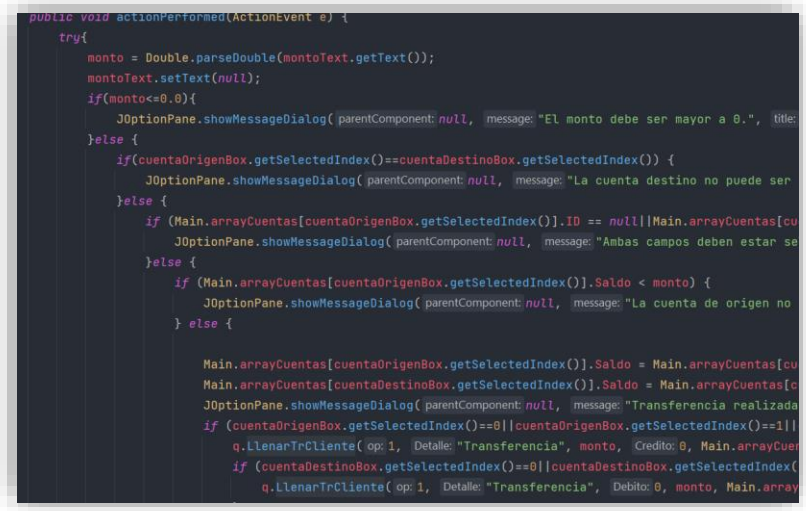
DepositoFrame

Mediante un JComboBox se despliega en pantalla la información de las cuentas creadas, junto a su ID único que las diferencia entre sí, se puede ingresar información en un textField que luego con un JButton será evaluda como válida o no, para asignar ese monto al saldo existente.

```
Angel Andres Godoy Valdez
aceptarBtn.addActionListener(new ActionListener() {
    Angel Andres Godoy Valdez
    @Override
    public void actionPerformed(ActionEvent e) {
        try{
            monto = Double.parseDouble(textMonto.getText());
            textMonto.setText(null);
            if(monto<=0.0){
                JOptionPane.showMessageDialog( parentComponent: null,  message: "El monto debe ser mayor a 0.",  ti
            }else {
                if (Main.arrayCuentas[comboBox1.getSelectedIndex()].ID == null) {
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "No hay ninguna cuenta selecci
                } else {
                    Main.arrayCuentas[comboBox1.getSelectedIndex()].Saldo = Main.arrayCuentas[comboBox1.getS
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Depósito realizado exitosamer
                    if(comboBox1.getSelectedIndex()==0||comboBox1.getSelectedIndex()==1||comboBox1.getSelect
                        c.LlenarTrCliente( op: 1, Detalle: "Depósito", Debito: 0,monto,Main.arrayCuentas[comboBox1.g
                    }
                    if(comboBox1.getSelectedIndex()==5||comboBox1.getSelectedIndex()==6||comboBox1.getSelect
                        c.LlenarTrCliente( op: 2, Detalle: "Depósito", Debito: 0,monto,Main.arrayCuentas[comboBox1.g
                    }
                    if(comboBox1.getSelectedIndex()==10||comboBox1.getSelectedIndex()==11||comboBox1.getSele
                        c.LlenarTrCliente( op: 3, Detalle: "Depósito", Debito: 0,monto,Main.arrayCuentas[comboBox1.g
```

TransferenciaFrame

Parecido al Frame de deposito con la dificultad que por cada instancia debe ser tomada el resto de cuentas para comparar los saldos y si es posible hacer la transferencia.



```
public void actionPerformed(ActionEvent e) {
    try{
        monto = Double.parseDouble(montoText.getText());
        montoText.setText(null);
        if(monto<=0.0){
            JOptionPane.showMessageDialog( parentComponent: null,  message: "El monto debe ser mayor a 0.",  title:
        }else {
            if(cuentaOrigenBox.getSelectedIndex()==cuentaDestinoBox.getSelectedIndex()) {
                JOptionPane.showMessageDialog( parentComponent: null,  message: "La cuenta destino no puede ser
            }else {
                if (Main.arrayCuentas[cuentaOrigenBox.getSelectedIndex()].ID == null||Main.arrayCuentas[cu
                    JOptionPane.showMessageDialog( parentComponent: null,  message: "Ambas campos deben estar se
                }else {
                    if (Main.arrayCuentas[cuentaOrigenBox.getSelectedIndex()].Saldo < monto) {
                        JOptionPane.showMessageDialog( parentComponent: null,  message: "La cuenta de origen no
                    } else {

                        Main.arrayCuentas[cuentaOrigenBox.getSelectedIndex()].Saldo = Main.arrayCuentas[cu
                        Main.arrayCuentas[cuentaDestinoBox.getSelectedIndex()].Saldo = Main.arrayCuentas[c
                        JOptionPane.showMessageDialog( parentComponent: null,  message: "Transferencia realizada
                        if (cuentaOrigenBox.getSelectedIndex()==0||cuentaOrigenBox.getSelectedIndex()==1||
                            q.LlenarTrCliente( op: 1,  Detalle: "Transferencia", monto,  Credito: 0, Main.arrayCuer
                            if (cuentaDestinoBox.getSelectedIndex()==0||cuentaDestinoBox.getSelectedIndex(
                                q.LlenarTrCliente( op: 1,  Detalle: "Transferencia",  Debito: 0, monto, Main.array
```

Historial

En Historial se hace uso de los métodos que fueron creados bajo la clase CrearInstancia para capturar la información y plasmarla en un JTable comparando los ID de las cuentas.

```
Angel Andres Godoy Valdez
mostrarTrBtn.addActionListener(new ActionListener() {
    Angel Andres Godoy Valdez
    @Override
    public void actionPerformed(ActionEvent e) {
        boolean encontrado = true;
        if(textID.getText()==null){
            JOptionPane.showMessageDialog( parentComponent: null,  message: "Debe ingresar un ID válid
        }else {
            if(textID.getText().equals(Main.arrayCuentas[0].ID)||textID.getText().equals(Main.ar

                tablaHistorial.setModel(Main.modeloCl1);
                textCui.setText(Main.cliente1.CUI);
                textNombre.setText(Main.cliente1.Nombre);
                textApellido.setText(Main.cliente1.Apellido);
                encontrado=false;

            }
            if(textID.getText().equals(Main.arrayCuentas[5].ID)||textID.getText().equals(Main.ar

                tablaHistorial.setModel(Main.modeloCl2);
                textCui.setText(Main.cliente2.CUI);
                textNombre.setText(Main.cliente2.Nombre);
                textApellido.setText(Main.cliente2.Apellido);
                encontrado=false;
```