Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Introducción a la Programación y Computación 1

Sección: N

Ing. William Estuardo Escobar Argueta

Auxiliar: Javier Oswaldo Mirón Cifuentes

MANUAL TECNICO

Nombre: Andrés Alejandro Agosto Méndez

Carnet: 202113580

Manual Técnico

Clase Proyecto 1: se creó una clase con un main, para que hiciera visible la primera ventana, y se creó un arreglo global, un variable int para que puede ser utilizado en cualquier ventana.

```
package proyecto1;

public class Proyecto1 {

    //primordial

    public static Clientes TotClientes [] = new Clientes[5]; // lista de clientes global en funcion del metodo clientes
     public static int iDAutoincrementable = 1000;


    public static void main(String[] args) {
        Inicio I = new Inicio();  //declaramos el objeto en el main
        I.setVisible(true); // metodo que hace visible la ventana
    }

}
```

jFrame inicio: incluye en el constructor aparte de la parte gráfica unos métodos para centrar la ventana que vienen para todos los jFrame, y el botón redirecciona a al j Frame para autenticar el usuario, y el otro saltará un jOptionpane que informará la información del creador.

```
package proyecto1;
import javax.swing.JOptionPane;

public class Inicio extends javax.swing.JFrame { //clase de inicio el extends hereda del JFrame

    public Inicio() {  // metodo constructor
        initComponents();
           this.setTitle("Página de inicio");
            this.setLocationRelativeTo(null); //metodo de centrado de la ventana
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {...56 lines }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        JOptionPane.showMessageDialog(null, "Andrés Alejandro Agosto Méndez 202113580","About",JOptionPane.INFORMATION_MESSAGE);
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       Login l = new Login(); // CAMBIAR ESTO POR EL "ADMINISTRADOR" POR "LOGIN"
       l.setVisible(true);
       this.dispose(); // método que evita la acumulación de la ventana anterior
    }

    public static void main(String args[]) {

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Inicio().setVisible(true);
            }
        });
```

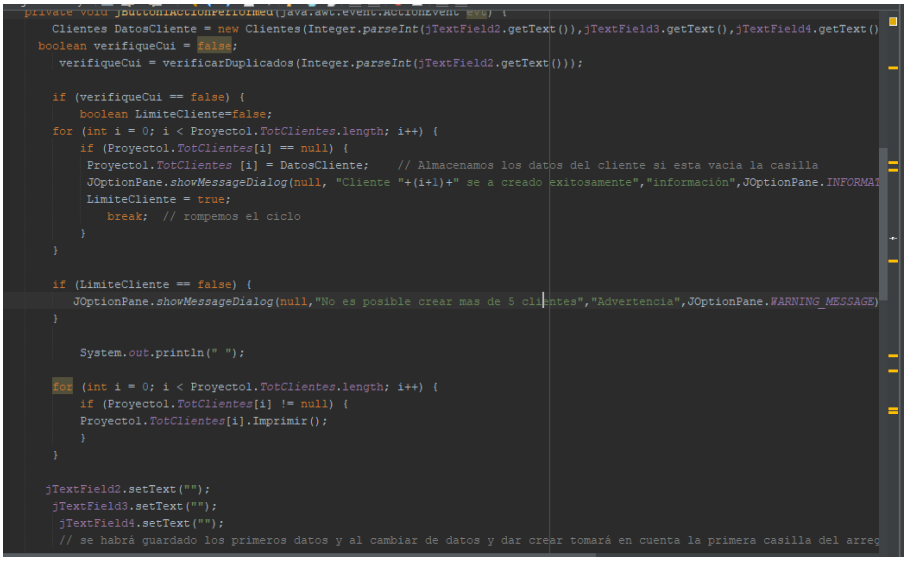
jFrame Login: este verificar con if y booleanos para que la contraseña y usuario sean los correctos, sino tira un jOptionpane que informará que algo esta incorrecto.

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
                String Usuario = "administrador";
        String Contraseña = "202113580";

        String pass = new String(jPasswordField1.getPassword()); // convierte el campo de contraseña a un string

        if (jTextField1.getText().equals(Usuario)&& pass.equals(Contraseña)) {  // verifica que el usuario y contraseña sean igua
            Administrador A = new Administrador();
       A.setVisible(true);
       this.dispose(); // método que evita la acumulación de la ventana anterior

        }
        else {
            JOptionPane.showMessageDialog(null, "Usuario ó contraseña incorrectos","Advertencia",JOptionPane.WARNING_MESSAGE);
        }
    }
```

jFrame administrador: este tendrá botones que direccionará a los demás jFrame.

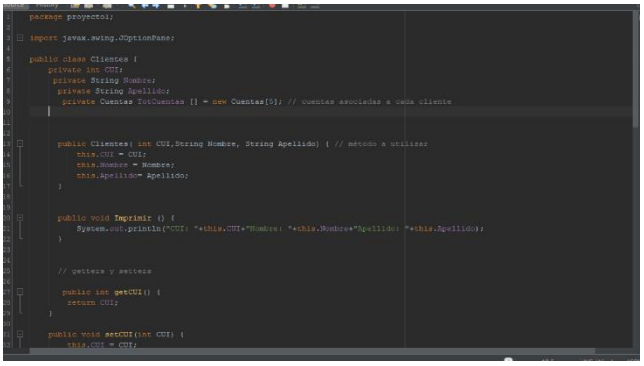
```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
            RegisClient RC = new RegisClient();
        RC.setVisible(true);
        this.dispose(); // método que evita la acumulación de la ventana anterior
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
         Login L = new Login();
        L.setVisible(true);
        this.dispose(); // método que evita la acumulación de la ventana anterior
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        CrearCuentas CC = new CrearCuentas();
        CC.setVisible(true);
        this.dispose(); // método que evita la acumulación de la ventana anterior
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
      VisualClientes V = new VisualClientes();
       V.setVisible(true);
       this.dispose(); // método que evita la acumulación de la ventana anterior
    }

    private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        Depositos D = new Depositos();
        D.setVisible(true);
        this.dispose(); // método que evita la acumulación de la ventana anterior
    }
```

jFrame regisClient : el botón verificara que se y recorrerá los ciclos y verificaciones para cumplir con las ventanas emergentes, y se imprimirá en consola para ir observando los resultados.



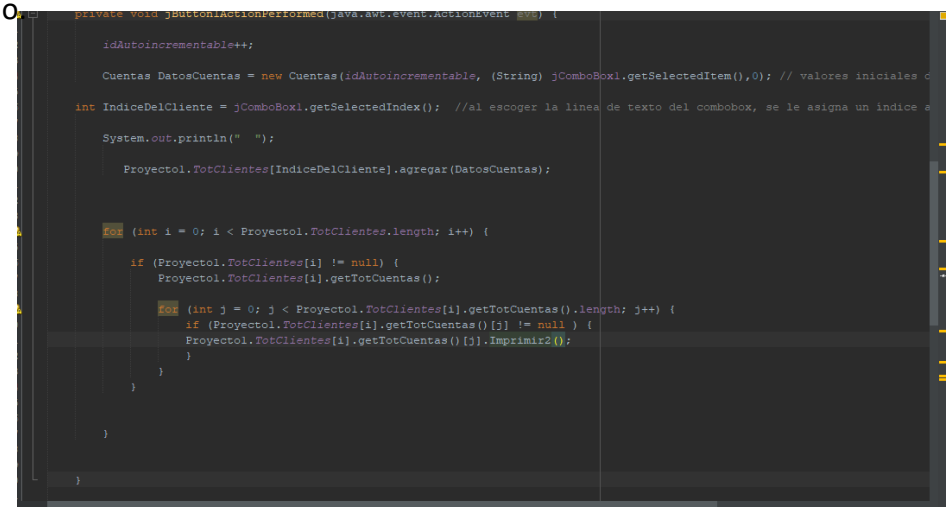
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    Clientes DatosCliente = new Clientes(Integer.parseInt(jTextField2.getText()),jTextField3.getText(),jTextField4.getText()
  boolean verifiqueCui = false;
     verifiqueCui = verificarDuplicados(Integer.parseInt(jTextField2.getText()));

    if (verifiqueCui == false) {
        boolean LimiteCliente=false;
    for (int i = 0; i < Proyectol.TotClientes.length; i++) {
        if (Proyectol.TotClientes[i] == null) {
         Proyectol.TotClientes [i] = DatosCliente;    // Almacenamos los datos del cliente si esta vacia la casilla
         JOptionPane.showMessageDialog(null, "Cliente "+(i+1)+" se a creado exitosamente","información",JOptionPane.INFORMAT
         LimiteCliente = true;
            break;  // rompemos el ciclo
        }
    }

    if (LimiteCliente == false) {
       JOptionPane.showMessageDialog(null,"No es posible crear mas de 5 clientes","Advertencia",JOptionPane.WARNING_MESSAGE)
    }

        System.out.println(" ");

    for (int i = 0; i < Proyectol.TotClientes.length; i++) {
        if (Proyectol.TotClientes[i] != null) {
        Proyectol.TotClientes[i].Imprimir();
        }
    }

   jTextField2.setText("");
    jTextField3.setText("");
     jTextField4.setText("");
     // se habrá guardado los primeros datos y al cambiar de datos y dar crear tomará en cuenta la primera casilla del arreg
```

Objeto Cliente: este objeto contiene sus atributos privados y sus respectivos getters y setters para poder usarlos en los demás.

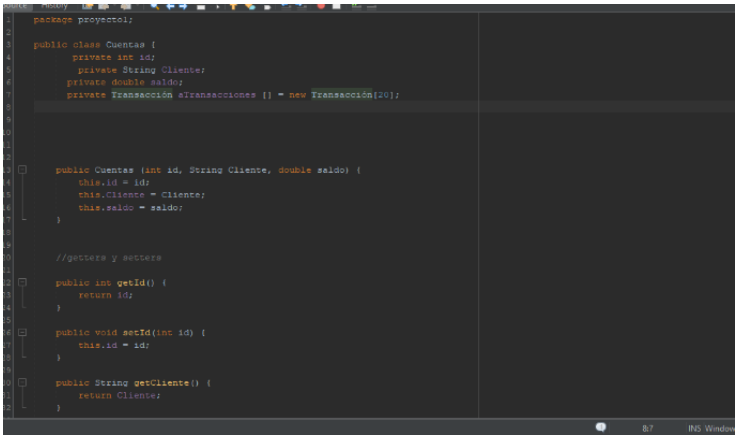


```
package proyectol;

import javax.swing.JOptionPane;

public class Clientes {
    private int CUI;
     private String Nombre;
      private String Apellido;
       private Cuentas TotCuentas [] = new Cuentas[5]; // cuentas asociadas a cada cliente


      public Clientes( int CUI,String Nombre, String Apellido) { // método a utilizar
          this.CUI = CUI;
          this.Nombre = Nombre;
          this.Apellido= Apellido;
      }


      public void Imprimir () {
          System.out.println("CUI: "+this.CUI+"Nombre: "+this.Nombre"Apellido: "+this.Apellido);
      }


      // getters y setters

       public int getCUI() {
        return CUI;
       }

    public void setCUI(int CUI) {
        this.CUI = CUI;
```

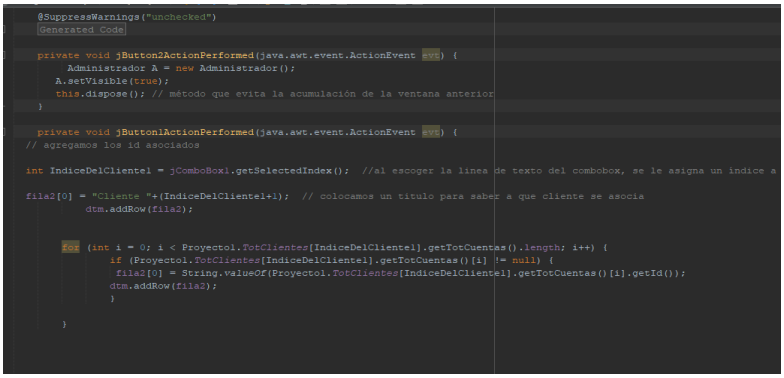
jFrame CrearCuentas: El id se irá incrementando cada vez que se presione el botón crear e ira almacenando los datos a través de un método llamado agregar que recorre las cuentas y verifica si están vacíos los espacios y mediante un objeto de tipo cuenta se colocan los atributos, con el IndiceDelCliente se busca identificar rápido la posición sin la necesidad de usar un ciclo.



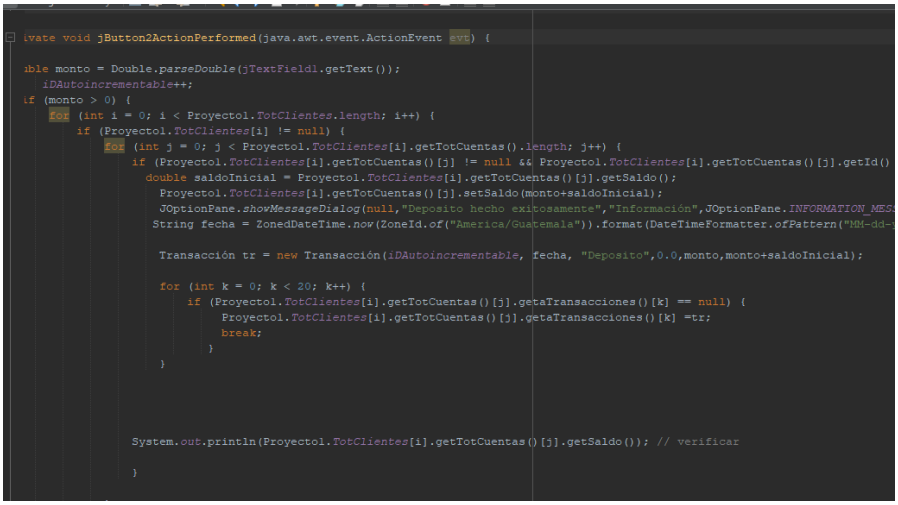
Objeto Cuenta: este objeto contiene sus atributos privados y sus respectivos getters y setters para poder usarlos en los demás



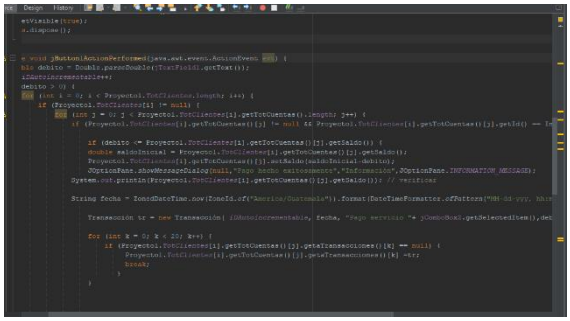
jFrame VisualClientes: el botón de buscar cuentas asociadas va añadiendo a la tabla a través de un for las filas respectivas.



jFrame Depositos: primero se verifica que el monto sea mayor a 0 sino no se podrá, luego recorre el arreglo clientes, verifica que tenga algo, recorre el arreglo de cuentas y verifica que teng algo y que el id sea igual que al del jComboBox y a través de un setter agrega el monto al ultimo saldo actualizado y con el objeto tr de tipo transacción irá guardando los datos para la tabla historial de transacciones.

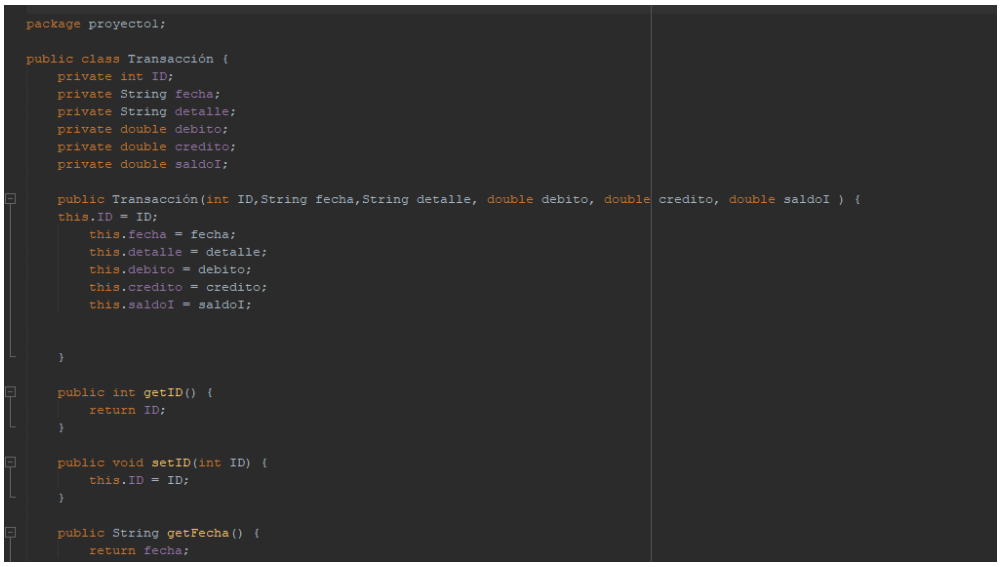


jFrame pago de servicios: esto es exactamente lo mismo que con el jFrame Depositos, la diferencia aquí es que en el setter se le resta al saldo el monto.



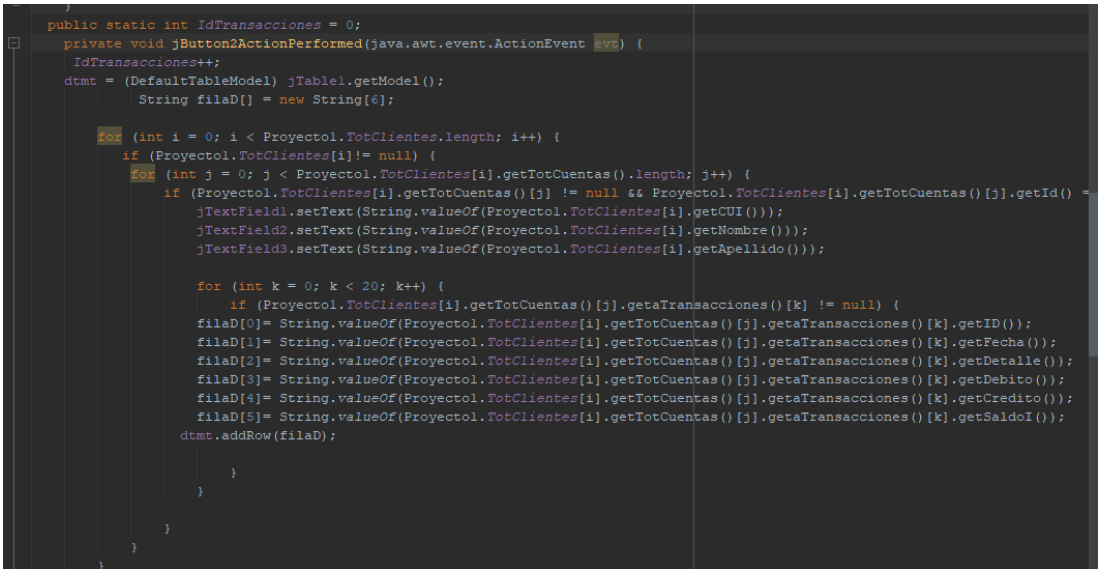
Transferencia: esta es la unión de pago de servicios y depósitos, a la cuenta de origen se le resta el monto en el setter, y la de origen se le suma el monto en el setter luego se obtiene en un getter.

Objeto Transacciones: este objeto tiene sus atributos privados y sus respectivos getters y setters, estos servirán para el historial de transacciones y serán las filas que se han estado almacenando en otro arreglo de transacciones.



```
package proyectol;

public class Transacción {
    private int ID;
    private String fecha;
    private String detalle;
    private double debito;
    private double credito;
    private double saldoI;

    public Transacción(int ID,String fecha,String detalle, double debito, double credito, double saldoI ) {
    this.ID = ID;
        this.fecha = fecha;
        this.detalle = detalle;
        this.debito = debito;
        this.credito = credito;
        this.saldoI = saldoI;


    }

    public int getID() {
        return ID;
    }

    public void setID(int ID) {
        this.ID = ID;
    }

    public String getFecha() {
        return fecha;
```

jFrame historial de transacciones: luego se hace lo mismo con el de visualizar clientes, solo que se pedirá el arreglo de transacciones para cada cuenta, y se hará un getter para lo se necesita poner en cada columna de la fila, se recorre el arreglo clientes, verifica que haya algo, recorre el arreglo cuentas, verifica que haya algo y obtiene los cui nombre y apellido, luego en otro for recorre el arreglo de las transacciones y obtiene lo que se necesita.



```
public static int IdTransacciones = 0;
  private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
   IdTransacciones++;
  dtmt = (DefaultTableModel) jTable1.getModel();
          String filaD[] = new String[6];

      for (int i = 0; i < Proyectol.TotClientes.length; i++) {
        if (Proyectol.TotClientes[i]!= null) {
         for (int j = 0; j < Proyectol.TotClientes[i].getTotCuentas().length; j++) {
             if (Proyectol.TotClientes[i].getTotCuentas()[j] != null && Proyectol.TotClientes[i].getTotCuentas()[j].getId() =
                 jTextField1.setText(String.valueOf(Proyectol.TotClientes[i].getCUI()));
                 jTextField2.setText(String.valueOf(Proyectol.TotClientes[i].getNombre()));
                 jTextField3.setText(String.valueOf(Proyectol.TotClientes[i].getApellido()));

                 for (int k = 0; k < 20; k++) {
                     if (Proyectol.TotClientes[i].getTotCuentas()[j].getaTransacciones()[k] != null) {
                 filaD[0]= String.valueOf(Proyectol.TotClientes[i].getTotCuentas()[j].getaTransacciones()[k].getID());
                 filaD[1]= String.valueOf(Proyectol.TotClientes[i].getTotCuentas()[j].getaTransacciones()[k].getFecha());
                 filaD[2]= String.valueOf(Proyectol.TotClientes[i].getTotCuentas()[j].getaTransacciones()[k].getDetalle());
                 filaD[3]= String.valueOf(Proyectol.TotClientes[i].getTotCuentas()[j].getaTransacciones()[k].getDebito());
                 filaD[4]= String.valueOf(Proyectol.TotClientes[i].getTotCuentas()[j].getaTransacciones()[k].getCredito());
                 filaD[5]= String.valueOf(Proyectol.TotClientes[i].getTotCuentas()[j].getaTransacciones()[k].getSaldoI());
               dtmt.addRow(filaD);

                     }
                 }

             }
         }
```