Proyecto 1
Segundo semestre 2022

Lab IPC1

Manual de Usuario

Creado por: Juan Pablo Samayoa Ruiz

202109705

- **Creacion de la clase clientes**

Creacion de las variables a utilizar en la clase clientes:

```
int CUI;
String nombre;
String apellido;
```

```
public String StringCli;
static int correlativo=0;
public static Clientes[] clientesRegistrados = new Clientes[5];
```

Clientes[]: Esta es la matriz en el cual se almacenaran los datos ingresados.

Creacion de metodos para registrar al cliente:

```
public static void AgregarCliente(Clientes nuevoCliente){
    for (Clientes clientesRegistrado : clientesRegistrados) {
        if(clientesRegistrado != null){
            if(clientesRegistrado.CUI == nuevoCliente.CUI){
                JOptionPane.showMessageDialog(null, "CUI REPETIDO");
                return;
            }
        }
    }
    for(int i=0; i<clientesRegistrados.length; i++){
        if(clientesRegistrados[i] == null){
            clientesRegistrados[i] = nuevoCliente;
            JOptionPane.showMessageDialog(null, "Cliente creado exitosamente");
            System.out.println("Cui: " + String.valueOf(nuevoCliente.getCUI()) + " nombre: " + nuevoCliente.getNombre()
            return;
        }
    }
    JOptionPane.showMessageDialog(null, "No se pueden crear más clientes");
}
```

Este metodo lo que realiza es un validacion y exploracion de los datos que se van a ingresar por medio de una interfaz gráfica, de forma que pormedio de un for que verifica los CUI de clientes ingresados para que estos no repitan y un segundo for el cual verifica que se guarden los datos dentro del vector, y al momento de que este se llene mostrará un mensaje emergente indicando que ya se llenó por completo

Creacion del metodo para buscar clientes:

```
public static Clientes buscarClientesCui(String CUI){
    int CuiU = Integer.parseInt(CUI);
    for (Clientes clientesRegistrado : clientesRegistrados) {
        if(clientesRegistrado != null){
            if(clientesRegistrado.CUI == CuiU){
                return clientesRegistrado;
            }
        }
    }
    JOptionPane.showMessageDialog(null, "El cliente no existe");
    return null;
}
```

El siguiente metodo utiliza como base el CUI registrado por el administrador y como este ya se encuentra asociado con el nombre, apellido y numero de cuenta solamente es necesario retornar una variable con los datos seleccionados

Declaración de objetos:

```
public Clientes(){
    CUI = -1;
    nombre = "";
    apellido = "";
}

public Clientes(int CUI, String nombre, String apellido){
    this.CUI = CUI;
    this.nombre = nombre;
    this.apellido = apellido;
}
```

Valores iniciales de las variables: En esta parte estaremos asignando el valor de las variables iniciales de la clase y un metodo el cual declara los objetos a utililizar, en este caso el CUI, nombre apellido y el cliente

Get y set de las variables:

```
public int getCUI() {
    return CUI;
}

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

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}
```

Desde esta parte se le declara un get y set para las variables, de esta forma obtenemos los datos que se ingresan y esos mismos datos se almacenan en la variable que se solicite.

- ## Creacion de la clase Cuentas

Import Clase Clientes: En este caso se hace un import de la clase clientes, ya que se estaran utilizando elementos de la misma para poder crear las cuentas necesarias.

```
import proyecto.pkg1.ipc.pkg2s.Clientes;
```

Creacion de las variables a utilizar:

```
public int id_cuenta;
public Clientes ClienteAsociado;
public double monto;
```

En esta parte se crean las variables principales a utilizar las cuales son el id de la cuenta, el cliente (El cual se creó en la clase Clientes) y el monto de la cuenta que inicialmente debe ser 0

Creacion del almacenamiento de cuentas:

```
public static Cuentas[] cuentasRegistradas = new Cuentas[25];
```

Esta matriz almacenará el numero total de cuentas creadas teniendo como limete la creacion de 25 cuentas

Metodo cuentas clientes:

```
private static int cuentasClientes(Clientes cliente){
    int NUMcuentas = 0;
    for(Cuentas cuenta : cuentasRegistradas){
        if(cuenta != null){
            if(cuenta.ClienteAsociado.equals(cliente)){
                NUMcuentas += 1;
            }
        }
    }
    return NUMcuentas;
}
```

Este metodo lo que hace es asociar a un cliente que se tenga almazenado y detectar las cuentas que este tenga creadas dandole un maximo de 5 cuentas.

Metodo de crear cuentas bancarias:

```
public static void CrearCuentaBanc(Cuentas cuenta){
    if(cuentasClientes(cuenta.ClienteAsociado)==5){
        JOptionPane.showMessageDialog(null, "Ya no se pueden crear más cuentas");
        return;
    }
    for(int i=0; i<cuentasRegistradas.length; i++){
        if(cuentasRegistradas[i] == null){
            cuenta.id_cuenta = i+1;
            cuentasRegistradas[i] = cuenta;
            JOptionPane.showMessageDialog(null, "Cuenta creada exitosamente");
            System.out.println("Cuenta creada: " + cuenta.id_cuenta + " " +cuenta.ClienteAsociado.CUI);
            return;
        }
    }
}
```

En este metodo lo que se hace es que se toma en cuenta las cuentas creadas que tiene el cliente, en el caso de que lleguen a ser 5 este tirará un mensaje el cual mostrará que se alcanzó el maximo de cuentas.

El for que se encuentra despues del if para verificar la cantidad de cuentas es el que realmente las creará, de esta forma recorrerá los clientes registrados y les asignará una cuenta y un ID de cuenta el cual se autoincrementará y al momento de qu la cuenta se cree y se asocie a un cliente se mostrará un mensaje para indicando que la cuenta se a creado exitosamente.

Metodo par asiganrle valor a las variables iniciales de la clase:

```
public Cuentas(Clientes cliente){
    id_cuenta = -1;
    ClienteAsociado = cliente;
    monto = 0;
}
```

Este metodo contiene de parametros los clientes registrados de la clase Clientes para, de esta forma obtendrá los clientes registrados en la matriz y se las asiganará como un valor string a la variable Clientes asociados.

Metodo para realizar el deposito de clientes:

```
public void DepMon(double monto){
    if(monto <= 0){
        JOptionPane.showMessageDialog(null, "El monto debe ser mayor 0");
        return;
    }
    this.monto += monto;
    JOptionPane.showMessageDialog(null, "Deposito realizado");
    System.out.println("Monto realizado: " + monto + " Nuevo Saldo: " +this.monto);
}
```

Este metodo lo que hace es realizar la suma al valor inicial de la variable monto, de forma de que si el monto ingresado es menor o igual a 0, y si este es mayor a 0 se agregará el

valor ingresado al monto almacenado para despues mostrar un mensaje de que se realizó correctamente

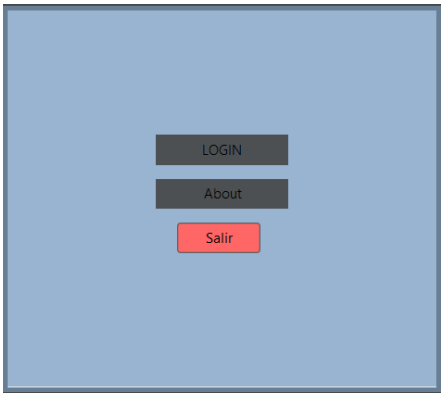
Metodo para buscar cuentas por medio del CUI:

```
public static Cuentas[] BuscarCuentasCUI(String CUI){
    Clientes clienteA = Clientes.buscarClientesCui(CUI);
    if(clienteA == null){
        return null;
    }
    int NumCuenta = cuentasClientes(clienteA);
    Cuentas[] CuentasA = new Cuentas[NumCuenta];
    int cont =0;
    for(Cuentas cuenta : cuentasRegistradas){
        if(cuenta != null){
            if(cuenta.ClienteAsociado.equals(clienteA)){
                CuentasA[cont] = cuenta;
                cont +=1;
            }
        }
    }
    return CuentasA;
}
```

Este metodo lo que hará es examinar la matriz que almacena las cuentas registradas y asociadas a los clientesm de forma que empezará a buscar las cuentas que estén asociadas.

- ## Creacion de interfaz grafica

Interfaz grafica del inicio



-1 Panel
-2 Botones

Interfaz grafica del login

-2 labels
-2 textBox
-2 botones
-1 Panel

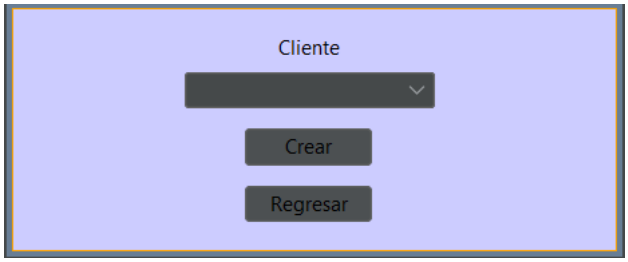
Interfaz grafica menú

-1 Panel
-8 botones

Interfaz grafica registro de cuentas

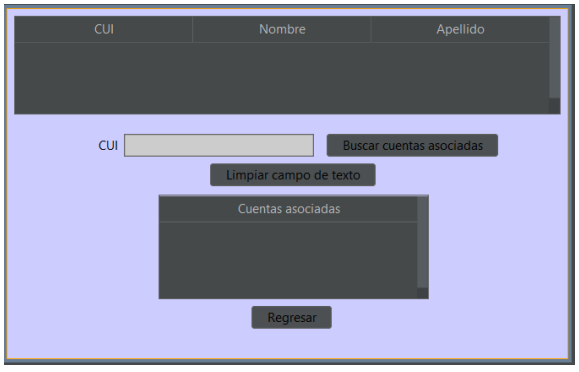
-1 Panel
-3 label
-3 TextBox
-3 botones

Interfaz grafica crear cuentas



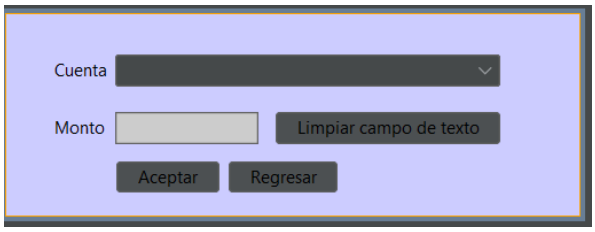
-1 Panel
-1 label
-1 comboBox
-2 botones

Interfaz grafica de info. Cliente



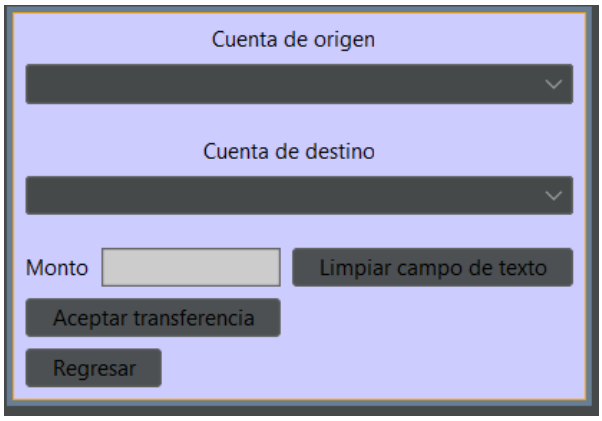
-1 Panel
-1 label
-1 textBox
-2 tables
-3 botones

Interfaz grafica Dep. Cliente



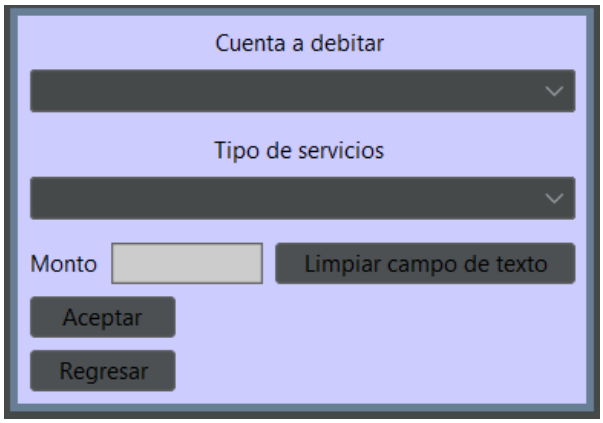
-1 comboBox
-1 TextBox
-2 label
-3 botones

Interfaz grafica Realizar transferencia



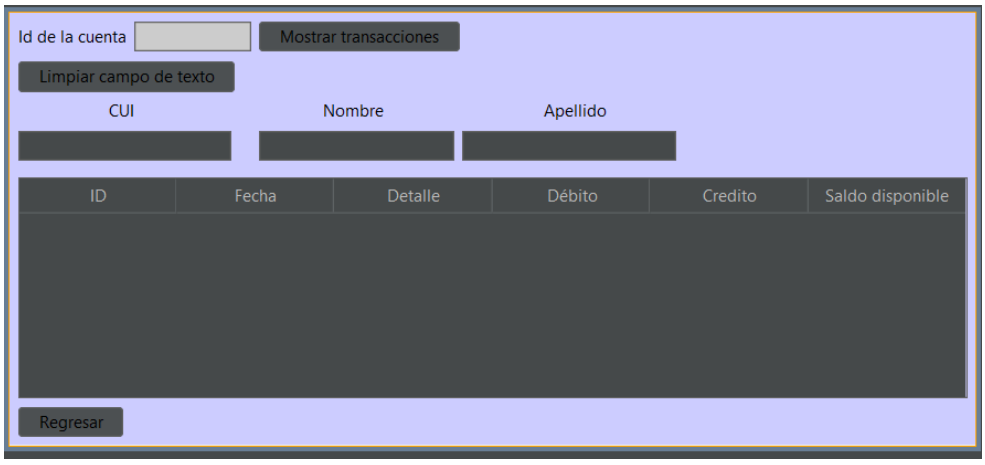
-1 comboBox
-1 TextBox
-2 label
-3 botones

Interfaz grafica Pago de servicios



-1 comboBox
-1 TextBox
-2 label
-3 botones

Interfaz grafica historial



-1 Table
-3 Botones
-4 Campos de Texto
-4 labels

- **Anexos**

Video de youtube: https://youtu.be/402zc0ylMGo