
	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN/INGENIERÍA DE SISTEMAS		ASIGNATURA: PROGRAMACIÓN APLICADA	
NRO. PROYECTO:	1.1	TÍTULO PROYECTO: Prueba Practica 2 Desarrollo e implementación de un sistema de simulación de acceso y atención bancaria	
OBJETIVO: Reforzar los conocimientos adquiridos en clase sobre la programación en Hilos en un contexto real.			
INSTRUCCIONES:		1. Revisar el contenido teórico y práctico del tema	
		2. Profundizar los conocimientos revisando los libros guías, los enlaces contenidos en los objetos de aprendizaje Java y la documentación disponible en fuentes académicas en línea.	
		3. Deberá desarrollar un sistema informático para la simulación y una interfaz gráfica.	
		4. Deberá generar un informe de la práctica en formato PDF y en conjunto con el código se debe subir al GitHub personal y AVAC.	
		5. Fecha de entrega: El sistema debe ser subido al git hasta 19 de enero del 2021 – 23:55.	
ACTIVIDADES POR DESARROLLAR			

1. Enunciado:

Realizar un sistema de simulación de acceso y atención a través de colas de un banco.

Problema: Un banco necesita controlar el acceso a cuentas bancarias y para ello desea hacer un programa de prueba en Java que permita lanzar procesos que ingresen y retiren dinero a la vez y comprobar así si el resultado final es el esperado.

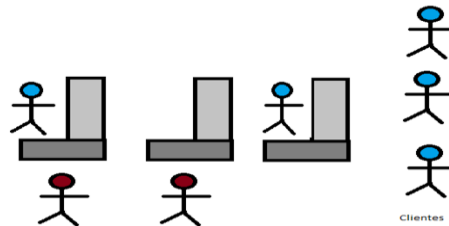
Se parte de una cuenta con 100 euros y se pueden tener procesos que ingresen 100 euros, 50 o 20. También se pueden tener procesos que retiran 100, 50 o 20 euros. Se desean tener los siguientes procesos:

- 40 procesos que ingresan 100
- 20 procesos que ingresan 50
- 60 que ingresen 20.

De la misma manera se desean lo siguientes procesos que retiran cantidades.

- 40 procesos que retiran 100
- 20 procesos que retiran 50
- 60 que retiran 20.


Ademas en el banco, existen 3 cajeros que pueden atender y hay un cola inicial de 10 clientes para ser atendidos, el proceso de atención es de 20 – 15 segundos y los clientes llegan constantemente cada 30 - 50 segundos. Ningún cajero puede atender simultáneamente, adicionalmente el tiempo de moverme de la cola al estante del cajero es de 2 - 5 segundos, esto deberán ser generados aleatoriamente entre los 100 clientes que disponen una cuenta, estos pueden volver a ingresar el numero de veces que sea necesario.



Se desea comprobar que tras la ejecución la cuenta tiene exactamente 100 euros, que era la cantidad de la que se disponía al principio. Realizar el programa Java que demuestra dicho hecho.

Calificación:

- Diagrama de Clase 10%
- MVC: 10%
- Técnicas de Programación aplicadas (Java 8, Reflexión y Programación Genérica): 10%

	Computación	Docente: Diego Quisi Peralta
	Programación Aplicada	Período Lectivo: Septiembre 2020 – Febrero 2021

- Hilos 30%
- Sincronización 10%
- Interfaz Gráfica de simulación 20%
- Informe: 10%

2. Informe de Actividades:

- Planteamiento y descripción del problema.
- Diagramas de Clases.
- Patrón de diseño aplicado
- Descripción de la solución y pasos seguidos.
 - Comprobación de las cuentas bancarias e interfaz gráfica.
- Conclusiones y recomendaciones.
- Resultados.

RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
- Los estudiantes implementan soluciones gráficas en sistemas.
- Los estudiantes están en la capacidad de implementar hilos.

RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la prueba.**

BIBLIOGRAFIA:

[1]: <https://www.ups.edu.ec/evento?calendarBookingId=98892>

Docente / Técnico Docente: Ing. Diego Quisi Peralta Msc.

Firma: _____

CARRERA:

ASIGNATURA:

NRO. PRÁCTICA:

TÍTULO PRÁCTICA:

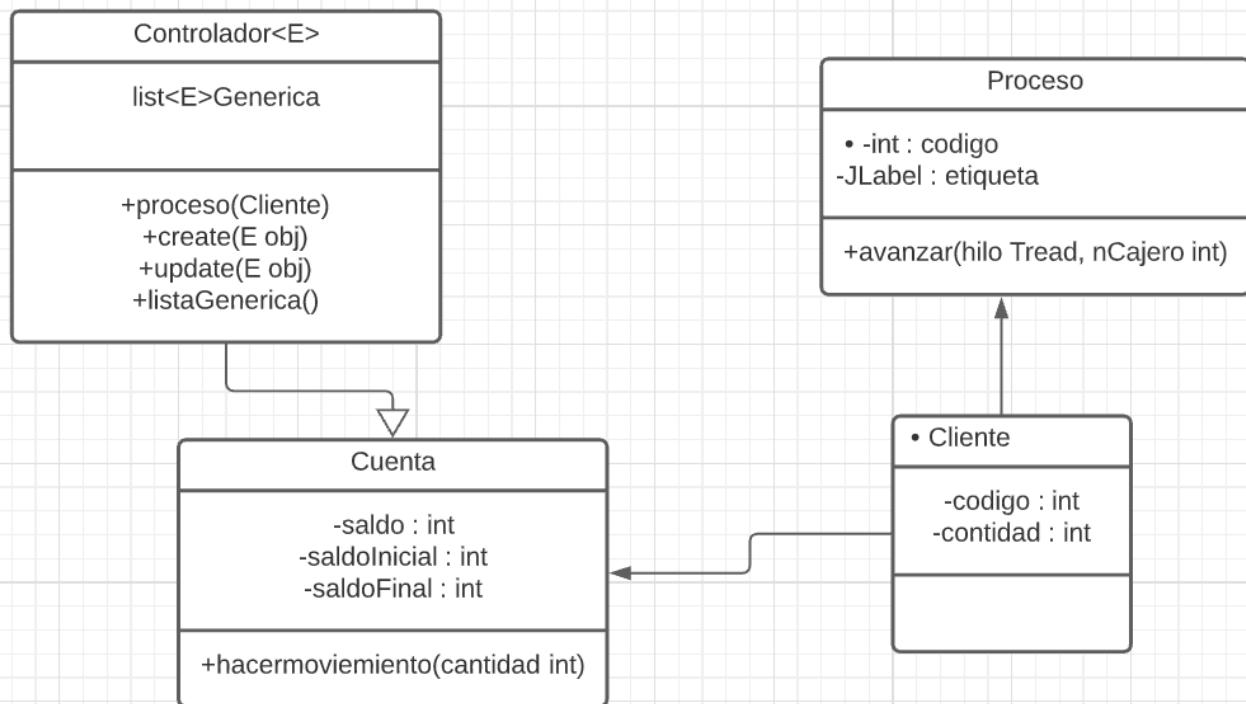
OBJETIVO ALCANZADO:

ACTIVIDADES DESARROLLADAS

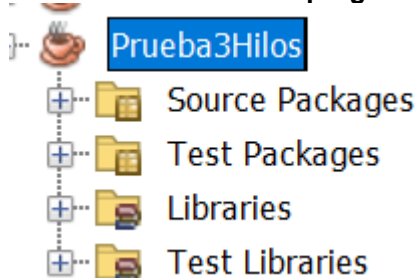
1. Planteamiento y descripción del problema.

Para el desarrollo de la práctica de hilos se procedió a plantear dos etapas lo cual consiste el diagrama de clases y la codificación en Java.

2. Diagramas de Clases.



Para el Desarrollo del programa se creo un proyecto llamado prueba 3 hilos.



Posteriormente se desarrollo el MVC con las clases detalladas a continuación

En el paquete modelo se considero pertinente crear 3 clases:

Cliente:

```
/*  
 * To change this license header, choose License Headers in Project Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */  
  
package ec.edu.ups.modelo;  
  
  
import ec.edu.ups.vista.VentanaPrincipal;  
import java.util.Random;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
  
/**  
 *  
 * @author ASUS  
 */  
public class Cliente implements Runnable{  
    private int id;  
    private Cuenta cuenta;  
    int cantidad;  
  
    public Cliente(int id,Cuenta cuenta, int cantidad) {  
        this.id=id;  
        this.cuenta = cuenta;  
        this.cantidad = cantidad;  
    }  
  
    public Cliente(Cuenta cuenta, int cantidad) {  
        this.cuenta = cuenta;  
        this.cantidad = cantidad;  
    }  
}
```

```
public Cliente(int id) {  
    this.id = id;  
}  
  
public int getId() {  
    return id;  
}  
  
public void setId(int id) {  
    this.id = id;  
}  
  
public Cuenta getCuenta() {  
    return cuenta;  
}  
  
public void setCuenta(Cuenta cuenta) {  
    this.cuenta = cuenta;  
}  
  
public int getCantidad() {  
    return cantidad;  
}  
  
public void setCantidad(int cantidad) {  
    this.cantidad = cantidad;  
}  
  
@Override  
public int hashCode() {  
    int hash = 7;  
    hash = 13 * hash + this.id;  
}
```

```
return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Cliente other = (Cliente) obj;
    if (this.id != other.id) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "Cliente{" + "id=" + id + ", cuenta=" + cuenta + ", cantidad=" + cantidad + "}";
}

@Override
public void run() {
    cuenta.hacerMovimiento(cantidad);
}
}
Cuenta
```

```
/*  
 * To change this license header, choose License Headers in Project Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */
```

```
package ec.edu.ups.modelo;
```

```
import ec.edu.ups.vista.VentanaPrincipal;
```

```
/**  
 *  
 * @author ASUS  
 */
```

```
public class Cuenta {
```

```
    private int saldo;
```

```
    private int saldoInicial;
```

```
    private VentanaPrincipal ventanaPrincipal;
```

```
    private int saldoFinal;
```

```
    public int getSaldoInicial() {
```

```
        return saldoInicial;
```

```
    }
```

```
    public void setSaldoInicial(int saldoInicial) {
```

```
        this.saldoInicial = saldoInicial;
```

```
    }
```

```
    public VentanaPrincipal getVentanaPrincipal() {
```

```
        return ventanaPrincipal;
```

```
    }
```

```
    public void setVentanaPrincipal(VentanaPrincipal ventanaPrincipal) {
```



```
this.ventanaPrincipal = ventanaPrincipal;
}

public int getSaldoFinal() {
    return saldoFinal;
}

public void setSaldoFinal(int saldoFinal) {
    this.saldoFinal = saldoFinal;
}

public Cuenta(int saldo){
    this.saldoInicial=saldo;
    this.saldo=saldo;
}

public synchronized void hacerMovimiento(int cantidad){
    this.saldo = this.saldo+cantidad;
}

public boolean esSimulacionCorrecta(){
    if (this.saldo==this.saldoInicial) return true;
    return false;
}

public int getSaldo(){
    return this.saldo;
}

@Override
public String toString() {
    return "Cuenta{" + "saldo=" + saldo + ", saldoInicial=" + saldoInicial
        + ", ventanaPrincipal=" + ventanaPrincipal + ", saldoFinal=" + saldoFinal + '}';
}
```

```
}
```

Proceso:

```
/*
```

```
* To change this license header, choose License Headers in Project Properties.
```

```
* To change this template file, choose Tools | Templates
```

```
* and open the template in the editor.
```

```
*/
```

```
package ec.edu.ups.modelo;
```

```
import java.util.Random;
```

```
import java.util.logging.Level;
```

```
import java.util.logging.Logger;
```

```
import javax.swing.JLabel;
```

```
/**
```

```
*
```

```
* @author ASUS
```

```
*/
```

```
public class Proceso implements Runnable {
```

```
    int id;
```

```
    JLabel label;
```

```
    boolean[] cajeros = new boolean[3];
```

```
    private Random random = new Random();
```

```
    private int valor;
```

```
    public Proceso(int valor) {
```

```
        this.valor = valor;
```

```
        this.random = new Random();
```

```
    }
```

```
public int getValor() {  
    return valor;  
}  
  
public void setValor(int valor) {  
    this.valor = valor;  
}  
  
public Proceso() {  
}  
  
public Proceso(int id, JLabel label, boolean[] cajeros) {  
    this.id = id;  
    this.label = label;  
    this.cajeros = cajeros;  
}  
  
public void avanzar(Thread hilo, int numeroCajero) {  
  
    int tiempoCajero = (int) (Math.random() * (20 - 15 + 1) + 15);  
    try {  
  
        Thread.sleep(tiempoCajero * 100);  
        cajeros[numeroCajero] = false;  
    } catch (InterruptedException ex) {  
        Logger.getLogger(Proceso.class.getName()).log(Level.SEVERE, null, ex);  
    }  
  
}  
  
public synchronized void salir(Thread hilo) {
```

```
for (int i = 0; i < 13; i++) {  
    notifyAll();  
    label.setLocation(label.getLocation().x, label.getLocation().y + 30);  
    try {  
        Thread.sleep(100);  
    } catch (InterruptedException ex) {  
        Logger.getLogger(Proceso.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}  
}
```

```
public synchronized void comenzar() {  
    int numeroCajero = 0;  
    boolean ocupados = false;  
    int ubicacionX = label.getLocation().x;  
    int ubicacionY = label.getLocation().y;  
    try {  
        for (int i = 0; i < 3; i++) {  
            if (!cajeros[i]) {  
                numeroCajero = i;  
                ocupados = false;  
                cajeros[i] = true;  
                break;  
            } else {  
                ocupados = true;  
            }  
        }  
        if (ocupados) {  
            this.wait();  
        }  
        int tiempoLlegarCajero = (int) (Math.random() * (9 - 7 + 1) + 7);
```

```
switch (this.id) {  
    case 14 -> {  
        switch (numeroCajero) {  
            case 0 -> {  
                int movimiento = 190 / tiempoLlegarCajero;  
                for (int i = 0; i < tiempoLlegarCajero; i++) {  
                    label.setLocation(label.getLocation().x  
                        + movimiento, label.getLocation().y);  
                    Thread.sleep(308);  
                }  
            }  
        }  
        case 1 -> {  
            int movimiento2 = 320 / tiempoLlegarCajero;  
            for (int i = 0; i < tiempoLlegarCajero; i++) {  
                label.setLocation(label.getLocation().x  
                    + movimiento2, label.getLocation().y);  
                Thread.sleep(300);  
            }  
        }  
        case 2 -> {  
            int movimiento3 = 450 / tiempoLlegarCajero;  
            for (int i = 0; i < tiempoLlegarCajero; i++) {  
                label.setLocation(label.getLocation().x  
                    + movimiento3, label.getLocation().y);  
                Thread.sleep(290);  
            }  
        }  
    }  
    case 5 -> {  
        switch (numeroCajero) {  
            case 0 -> {
```

```

        int movimiento = 240 / tiempoLlegarCajero;
        for (int i = 0; i < tiempoLlegarCajero; i++) {
            label.setLocation(label.getLocation().x
                + movimiento, label.getLocation().y);
            Thread.sleep(310);
        }
    }

    case 1 -> {
        int movimiento2 = 370 / tiempoLlegarCajero;
        for (int i = 0; i < tiempoLlegarCajero; i++) {
            label.setLocation(label.getLocation().x + movimiento2, label.getLocation().y);
            Thread.sleep(300);
        }
    }

    case 2 -> {
        int movimiento3 = 500 / tiempoLlegarCajero;
        for (int i = 0; i < tiempoLlegarCajero; i++) {
            label.setLocation(label.getLocation().x + movimiento3, label.getLocation().y);
            Thread.sleep(290);
        }
    }

    }

    default -> {
        switch (numeroCajero) {
            case 0 -> {
                int movimiento = 290 / tiempoLlegarCajero;
                for (int i = 0; i < tiempoLlegarCajero; i++) {
                    label.setLocation(label.getLocation().x
                        + movimiento, label.getLocation().y);
                    Thread.sleep(310);
                }
            }
        }
    }
}

```

```
}  
  
case 1 -> {  
    int movimiento2 = 420 / tiempoLlegarCajero;  
    for (int i = 0; i < tiempoLlegarCajero; i++) {  
        label.setLocation(label.getLocation().x + movimiento2, label.getLocation().y);  
        Thread.sleep(300);  
    }  
}  
  
case 2 -> {  
    int movimiento3 = 550 / tiempoLlegarCajero;  
    for (int i = 0; i < tiempoLlegarCajero; i++) {  
        label.setLocation(label.getLocation().x  
            + movimiento3, label.getLocation().y);  
        Thread.sleep(290);  
    }  
}  
}  
}  
  
this.avanzar(new Thread(this), numeroCajero);  
this.salir(new Thread(this));  
notifyAll();  
label.setLocation(ubicacionX, ubicacionY);  
} catch (InterruptedException e) {  
    e.printStackTrace();  
}  
}  
  
@Override  
public synchronized void run() {  
    boolean b = false;  
    while (b == false) {
```

```
comenzar();
```

```
}
```

```
}
```

```
}
```

A continuacion de detellaran las clases del controlador que en esta ocasión solo se creo dos llamados.

Controlador Generico

```
/*
```

```
* To change this license header, choose License Headers in Project Properties.
```

```
* To change this template file, choose Tools | Templates
```

```
* and open the template in the editor.
```

```
*/
```

```
package ec.edu.ups.controlador;
```

```
import ec.edu.ups.modelo.Cliente;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
/**
```

```
*
```

```
* @author ASUS
```

```
*/
```

```
public abstract class ControladorGenerico <T>{
```

```
    private List<T> listaGenerica;
```

```
    /*
```

```
        private List<Cliente> clientes;
```

```
        private List<Cliente> enLaFila;
```

```
        private List<Cliente> fueraDeFila;
```

```
        private Random random;
```

```
        private ControladorGenerico() {
```

```
            clientes = new ArrayList<>();
```



```
enLaFila = new ArrayList<>();
fueraDeFila = new ArrayList<>();
random = new Random();
}

public static ControladorGenerico getInstance() {
    return ControladorHolder.INSTANCE;
}

private static class ControladorHolder {
    private static final ControladorGenerico INSTANCE = new ControladorGenerico();
}

*/

public ControladorGenerico() {
    listaGenerica = new ArrayList<>();
}

public void create(T objeto) {
    listaGenerica.add(objeto);
}

public List<T> getListaGenerica() {
    return listaGenerica;
}

public void update(T objetoActualizado) {
    for (T objeto : listaGenerica) {
        if (objeto.equals(objetoActualizado)) {
            listaGenerica.set(listaGenerica.indexOf(objeto), objetoActualizado);
        }
    }
}
```

```

    }

}

Y controlador cliente que esta clase no es abstracta a diferencia de la clase controlador generico
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ec.edu.ups.controlador;

import ec.edu.ups.modelo.Cliente;

/**
 *
 * @author ASUS
 */
public class ControladorCliente extends ControladorGenerico<Cliente>{

    private List<Cliente> clientes;
    private List<Cliente> enLaFila;
    private List<Cliente> fueraDeFila;
    private Random random;

    public ControladorCliente(List<Cliente> clientes) {
        this.clientes = clientes;
        random = new Random();
    }

    public void ingresarEnFila() {

    }
}

```

```
public void procesosCleinte(Cliente cliente) {
```

```
}
```

}
Ahora en el problema fue planteado un requerimiento donde se pueda observar el procesos que esta haciendo el cliente en el cajero para ello en el paquete vista se desarrollo una clase llamada ventana mostrar datos donde se podrá apreciar en una tabla dichos procesos.

```
Vtnmostrardatos  
/*
```

```
* To change this license header, choose License Headers in Project Properties.
```

```
* To change this template file, choose Tools | Templates
```

```
* and open the template in the editor.
```

```
*/
```

```
package ec.edu.ups.vista;
```

```
import ec.edu.ups.controlador.ControladorCliente;
```

```
import ec.edu.ups.modelo.Cliente;
```

```
import java.util.List;
```

```
import javax.swing.table.DefaultTableModel;
```

```
/**
```

```
*
```

```
* @author ASUS
```

```
*/
```

```
public class vtnMostrarDatos extends javax.swing.JFrame {
```

```
    private ControladorCliente controlador;
```

```
    List<Cliente> listaCliente;
```

```
/**
```

```
* Creates new form vtnMostrarDatos
```

```
*/
```

```
public vtnMostrarDatos(List<Cliente> listaClientes) {
```

```

initComponents();

this.listaCliente = listaClientes;

cargarDatos();

}

public void cargarDatos() {
    DefaultTableModel datos = (DefaultTableModel) tblDatos.getModel();

    datos.setRowCount(0);
    for (Cliente c : listaCliente) {
        Object[] rowData = {c.getId(),c.getCantidad(),c.getCuenta().getSaldo()};
        datos.addRow(rowData);
    }
    tblDatos.setModel(datos);
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jScrollPane1 = new javax.swing.JScrollPane();
    tblDatos = new javax.swing.JTable();
    jLabel1 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

```

```
setName("Procesos de las personas"); // NOI18N

tblDatos.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null},
        {null, null, null}
    },
    new String [] {
        "Numero Cliente", "Estado", "Cantidad Final"
    }
) {
    boolean[] canEdit = new boolean [] {
        false, true, true
    };
}
```

```

    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
    }
});

jScrollPane1.setViewportView(tblDatos);

jLabel1.setText("Procesos de las personas en ell banco");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(53, 53, 53)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 538,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(37, 37, 37)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel1)
                    .addGap(218, 218, 218))))
        );
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(25, 25, 25)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 14,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 276,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(27, 27, 27))
        );

```

);

pack();

}// </editor-fold>

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

private javax.swing.JScrollPane jScrollPane1;

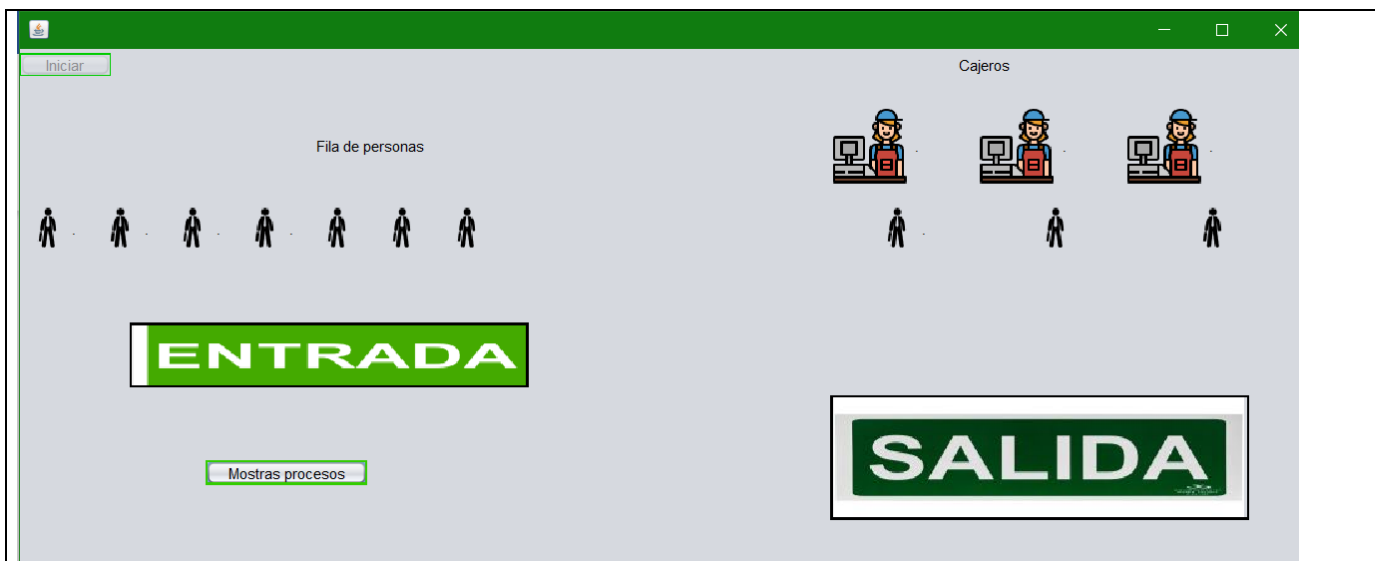
private javax.swing.JTable tblDatos;

// End of variables declaration

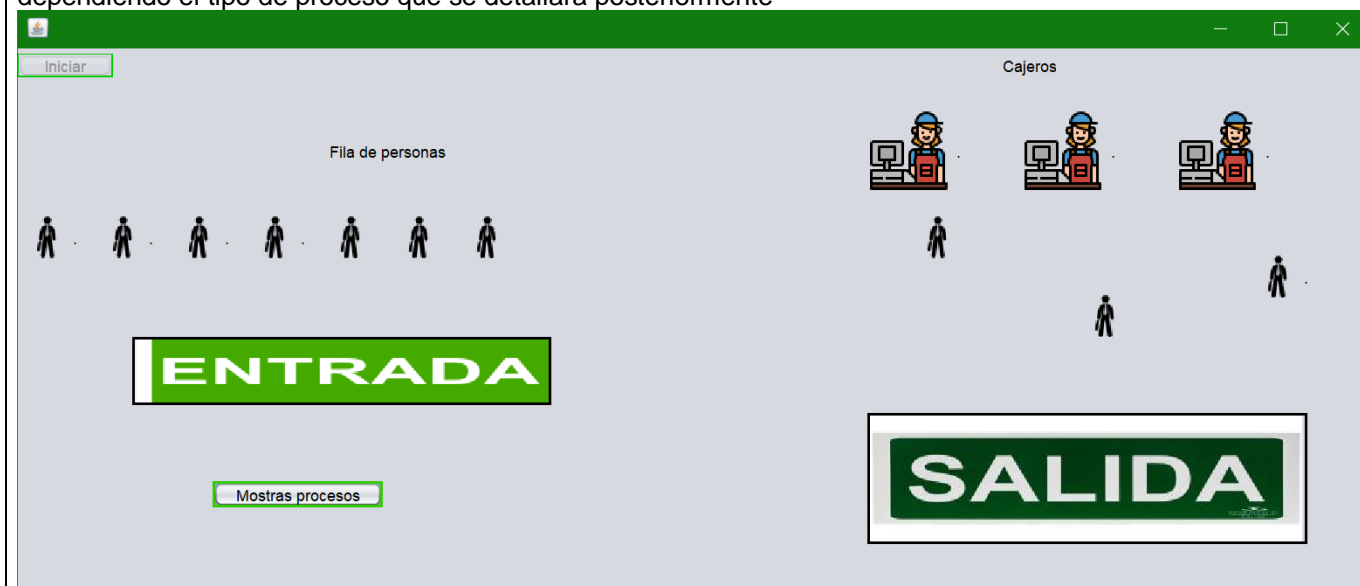
}
En el desarrollo de las interfaces es muy simple la GUI ya que solo tendremos dos opciones en este caso el botón iniciar y el botón mostrar procesos..



Posteriormente se considera unas imágenes donde se apreciar el punto de partida o entrada y el punto de salida o final además de ellos el usuario deberá presionar en el botón iniciar teniendo lo siguiente.



podemos apreciar que las personas están en cada caja correspondiente y luego ellos avanzan hacia la salida dependiendo el tipo de proceso que se detallara posteriormente



En la siguiente ventana resultado de mostrar los procesos podemos apreciar el id del cliente el estado o el proceso ejecutándose y la cantidad final de su cuenta

th

ir

Procesos de las personas en el banco

Numero Cliente	Estado	Cantidad Final
34	-100	100
35	-100	100
36	-100	100
37	-100	100
38	-100	100
39	-100	100
40	-20	100
41	-20	100
42	-20	100
43	-20	100
44	-20	100
45	-20	100
46	-20	100
47	-20	100
48	-20	100
49	-20	100

1 :

RESULTADO(S) OBTENIDO(S):

- Interpreta de forma correcta los algoritmos de programación y su aplicabilidad.
- Identifica correctamente qué herramientas de programación se pueden aplicar.

CONCLUSIONES:

- Los estudiantes identifican las principales estructuras para la creación de sistemas informáticos.
 - Los estudiantes implementan soluciones gráficas en sistemas.
- Los estudiantes están en la capacidad de implementar hilos.

RECOMENDACIONES:

- Revisar la información proporcionada por el docente previo a la práctica.
- Haber asistido a las sesiones de clase.
- **Consultar con el docente las dudas que puedan surgir al momento de realizar la prueba.**

Nombre de estudiante: John Farez

Firma de estudiante:

Diego