
	<b>Computación</b>	<b>Docente: Diego Quisi Peralta</b>
	Programación Aplicada	<b>Período Lectivo:</b> Septiembre 2020 – Febrero 2021

			<b>FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES</b>		
<b>CARRERA:</b> COMPUTACIÓN/INGENIERÍA DE SISTEMAS			<b>ASIGNATURA:</b> PROGRAMACIÓN APLICADA		
<b>NRO. PROYECTO:</b>	1.1	<b>TÍTULO PROYECTO:</b> Prueba Practica 2 Desarrollo e implementación de un sistema de simulación de acceso y atención bancaria			
<b>OBJETIVO:</b> Reforzar los conocimientos adquiridos en clase sobre la programación en Hilos en un contexto real.					
<b>INSTRUCCIONES:</b>		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. <b>Fecha de entrega:</b> El sistema debe ser subido al git hasta <b>17 de enero del 2021 – 23:55.</b>			
<b>ACTIVIDADES POR DESARROLLAR</b>					

## 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.

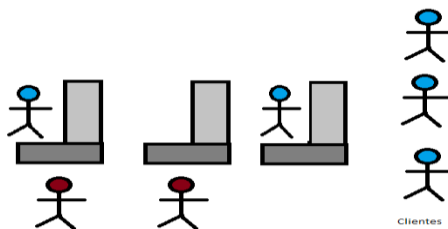
Se parte de una cuenta con 100 euros y se pueden tener procesos que ingresen 100 euros, 50 o 20. También se pueden tener procesos que retiran 100, 50 o 20 euros. Se desean tener los siguientes procesos:

- 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%

	<b>Computación</b>	<b>Docente: Diego Quisi Peralta</b>
	Programación Aplicada	<b>Período Lectivo:</b> 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 grafica.
- 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:**

1.  
1

**TÍTULO PRÁCTICA:** Prueba Practica 2

**OBJETIVO ALCANZADO:**

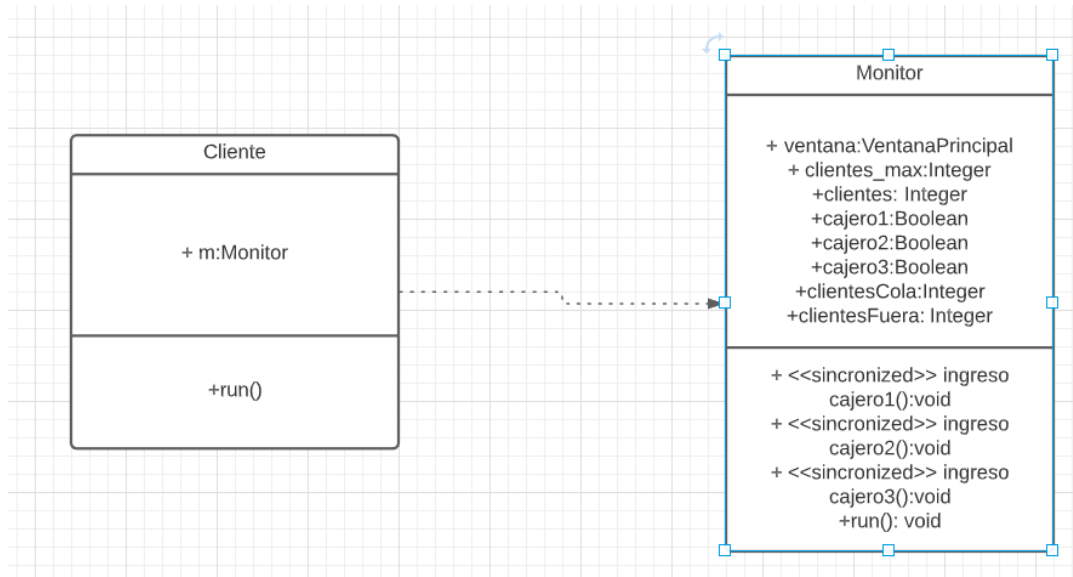
Reforzar los conocimientos adquiridos en clase sobre la programación en Hilos en un contexto real.

**ACTIVIDADES DESARROLLADAS**

En Java un hilo es un objeto con capacidad de correr en forma concurrente el método run (). En cierta manera es como tener dos "program counters" para un mismo código.

El patrón de diseño aplicado fue el Iterator el cual sirvió para guardar en arrays una lista de JLabels y ser mostrado en la interfaz grafica concorde a la sincronización de los hilos del programa.

El diagrama de clases el cual se siguió fue el siguiente:



El cual la clase monitor monitorea las acciones del cliente

La arquitectura utilizada es la de la modelo vista controlador, la cual en el modelo se encuentran las clases del diagrama mostrado anteriormente

**Clase 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.logging.Level;
import java.util.logging.Logger;

/**

```

```
* @author user
*/
public class Cliente implements Runnable {

    Monitor m;

    public Cliente(Monitor m) {

        this.m = m;
        m.clientes++;

    }

    public Cliente() {
    }

    @Override
    public void run() {
        while (true) {
            if (Thread.currentThread().isAlive()) {
                if (!m.cajero1) {
                    m.ingresoCajero1();
                    break;
                } else if (!m.cajero2) {
                    m.ingresoCajero2();
                    break;
                } else if (!m.cajero3) {
                    m.ingresoCajero3();
                    break;
                }
            }
        }
    }
}

}
```

### Clase Monitor

```
/*
 * 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.upse.modelo;

import ec.edu.upse.controlador.controladorCliente;
import ec.edu.upse.vista.VentanaPrincipal;
import javax.swing.JLabel;

/**
 *
 * @author user
 */
public class Monitor implements Runnable {
    VentanaPrincipal ventana ;

    int clientes_max;
    int clientes;

    boolean cajero1;
```

```

boolean cajero2;

boolean cajero3;

int clientesCola;

int clientesFuera;

public Monitor(VentanaPrincipal ventana) {

    cajero1= false;
    cajero2= false;
    cajero3=false;
    clientes=0;
    clientesCola=10;
    clientes_max=90;
    clientesFuera=0;

    this.ventana=ventana;
;
}

public synchronized void ingresoCajero1(){
    JLabel c =new javax.swing.JLabel();
    cajero1=true;
    clientesCola--;
    int r = (int) (Math.random() * 3) + 1;
    if(r==1){
        ingresaDinero();
    }else if(r==2){
        retiraDinero();
        cajero1=false;
        clientesFuera++;

        c.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Image/iconsP.png")));
        c.setBounds(-40, -40, 60,60);
        ventana.getPanelCola().add(c);

    }

    try{
        Thread.sleep(4000);
    }catch(InterruptedException e){
    }
    cajero1=false;
    ventana.getPanelCola().remove(c);
}

public synchronized void ingresoCajero2(){
    JLabel c =new javax.swing.JLabel();

    cajero2=true;
    clientesCola--;
    int r = (int) (Math.random() * 3) + 1;
    if(r==1){
        ingresaDinero();
    }else if(r==2){
        retiraDinero();
    }
}

```

```

        try{
            Thread.sleep(4000);
        }catch(InterruptedException e){
        }
        cajero2=false;
        clientesFuera++;
        c.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Im-
age/iconsP.png")));
        c.setBounds(-40, -40, 60,60);
        ventana.getPanelCola().add(c);

    }

    public synchronized void ingresoCajero3(){
        JLabel c =new javax.swing.JLabel();
        cajero3=true;
        clientesCola--;
        int r = (int) (Math.random() * 3) + 1;
        if(r==1){
            ingresaDinero();
        }else if(r==2){
            retiraDinero();
        }
        try{
            Thread.sleep(4000);
        }catch(InterruptedException e){
        }
        cajero3=false;
        clientesFuera++;
        c.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Im-
age/iconsP.png")));
        c.setBounds(-40, -40, 60,60);
        ventana.getPanelCola().add(c);

    }

    public void ingresaDinero(){

        int randomNumber = (int) (Math.random() * 4) + 1;
        switch (randomNumber) {
            case 1:
                ventana.getjTextArea1().setText("Cliente "+clientes+" ingresa 100
euros");
                break;
            case 2:
                ventana.getjTextArea1().setText("Cliente "+clientes+"ingresa 50 eu-
ros");
                break;
            case 3:
                ventana.getjTextArea1().setText("Cliente "+clientes+"ingresa 20 eu-
ros");
                break;
            default:
                break;
        }
        try{
            Thread.sleep(3000);
        }catch(InterruptedException e){

        }
    }
}

```

```

public void retiraDinero() {
    int randomNumber = (int) (Math.random() * 4) + 1;
    switch (randomNumber) {
        case 1:
            ventana.getjTextArea1().setText("Cliente "+clientes+" retira 100 eu-
ros");
            break;
        case 2:
            ventana.getjTextArea1().setText("Cliente "+clientes+"retira 50 eu-
ros");
            break;
        case 3:
            ventana.getjTextArea1().setText("Cliente "+clientes+"retira 20 eu-
ros");
            break;
        default:
            break;
    }
    try{
        Thread.sleep(3000);
    }catch(InterruptedException e){

    }
    }
    @Override
    public void run() {

    }

    public int getClientes_max() {
        return clientes_max;
    }

    public void setClientes_max(int clientes_max) {
        this.clientes_max = clientes_max;
    }

    public int getClientes() {
        return clientes;
    }

    public void setClientes(int clientes) {
        this.clientes = clientes;
    }

    public int getClientesCola() {
        return clientesCola;
    }

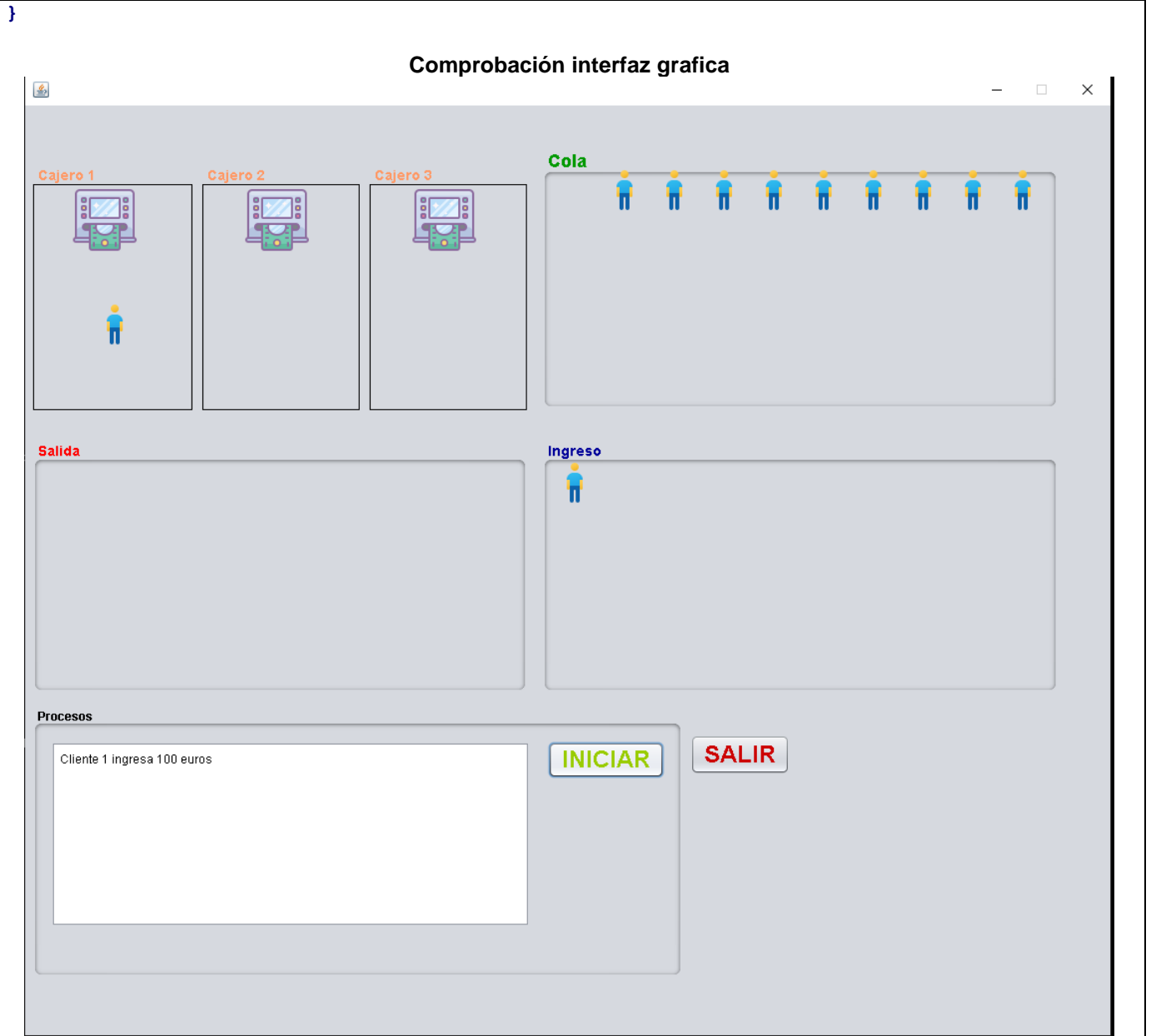
    public void setClientesCola(int clientesCola) {
        this.clientesCola = clientesCola;
    }

    public int getClientesFuera() {
        return clientesFuera;
    }

    public void setClientesFuera(int clientesFuera) {
        this.clientesFuera = clientesFuera;
    }
}

```





### RESULTADO(S) OBTENIDO(S):

Se logro aplicar conocimientos previos acerca del tema de hilos y sincronización en java relacionándolo así en un contexto real en un sistema de procesos bancarios

### CONCLUSIONES:

Le logro identificar las estructuras que se deben tomar para realizar un sistema informático, con el uso de diagrama de clases, arquitectura MVC y el tema de hilos y sincronización en java

**RECOMENDACIONES:**

El método iterator fue de mucha ayuda ya que se lograron generar JLabels automáticamente concorde al numero de objetos que se requieren

***Nombre de estudiante:*** JUAN JOSE CORDOVA CALLE

***Firma de estudiante:***

A handwritten signature in dark ink, appearing to be 'JJC', is shown on a light gray background.