

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

			PRÁCTICA DE LABORATORIO		
CARRERA: COMPUTACIÓN			ASIGNATURA: PROGRAMACIÓN APLICADA		
NRO. PRÁCTICA:	1.1	TÍTULO PRÁCTICA: Prueba Practica 2			
OBJETIVO ALCANZADO: <ul style="list-style-type: none"> Reforzar los conocimientos adquiridos en clase sobre la programación en Hilos en un contexto real. 					
ACTIVIDADES DESARROLLADAS					
1. Revisar el contenido teórico y práctico del tema <p>Hilos en Java</p> <p>A veces necesitamos que nuestro programa Java realice varias cosas simultáneamente. Otras veces tiene que realizar una tarea muy pesada, por ejemplo, consultar en el listín telefónico todos los nombres de chica que tengan la letra n, que tarda mucho y no deseamos que todo se quede parado mientras se realiza dicha tarea. Para conseguir que Java haga varias cosas a la vez o que el programa no se quede parado mientras realiza una tarea compleja, tenemos los hilos (Threads).</p>					
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. <p>En Java los hilos están encapsulados en la clase Thread. Para crear un hilo tenemos dos posibilidades:</p> <ul style="list-style-type: none"> Heredar de Thread redefiniendo el método run(). Crear una clase que implemente la interfaz Runnable que nos obliga a definir el método run(). <p>En ambos casos debemos definir un método run() que será el que contenga el código del hilo. Desde dentro de este método podremos llamar a cualquier otro método de cualquier objeto, pero este método run() será el método que se invoque cuando iniciemos la ejecución de un hilo. El hilo terminará su ejecución cuando termine de ejecutarse este método run().</p>					
3. Deberá desarrollar un sistema informático para la simulación y una interfaz gráfica.					

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

• Planteamiento y descripción del problema.

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:

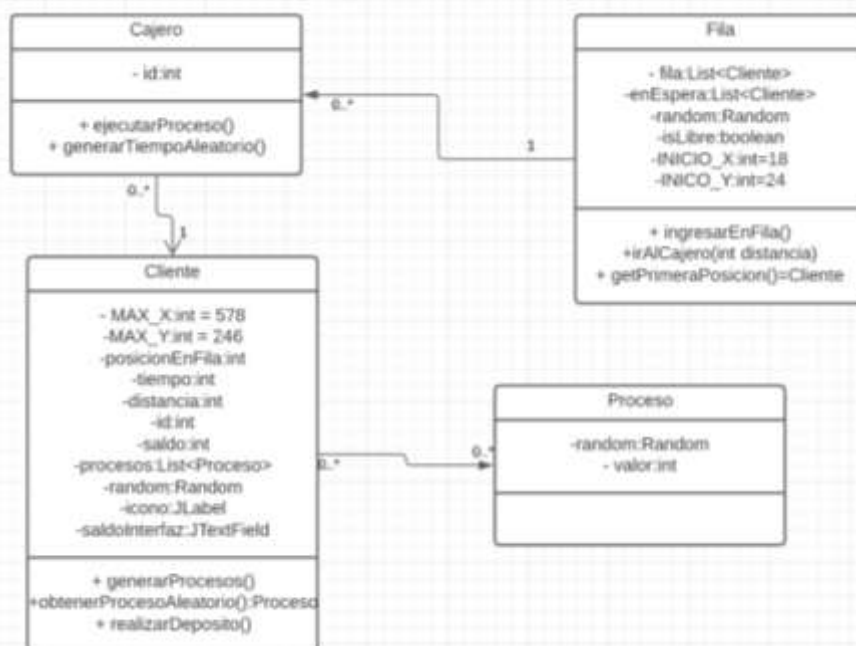
- 40 procesos que ingresan 100
- 20 procesos que ingresan 50
- 60 que ingresen 20.

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.

Además, en el banco, existen 3 cajeros que pueden atender y hay una 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 número de veces que sea necesario.

• Diagramas de Clases.




- **Patrón de diseño aplicado**

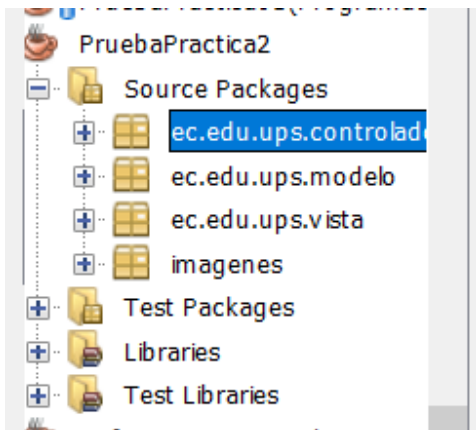
Singelton:

```
public class Controlador {  
  
    private List<Cliente> clientes;  
    private List<Cliente> enLaFila;  
    private List<Cliente> fueraDeFila;  
    private Random random;  
  
    private Controlador() {  
        clientes = new ArrayList<>();  
        enLaFila = new ArrayList<>();  
        fueraDeFila = new ArrayList<>();  
        random = new Random();  
    }  
  
    public static Controlador getInstance() {  
        return ControladorHolder.INSTANCE;  
    }  
  
    private static class ControladorHolder {  
        private static final Controlador INSTANCE = new Controlador();  
    }  
  
}
```

- **Descripción de la solución y pasos seguidos.**

1. Se creo un proyecto en java con el nombre de PruebaPractica2 en el cual se implementó el MVC así como también un paquete llamado imágenes en donde se almacenaran todas las imágenes del sistema

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		



2. En el paquete modelo se desarrollaron las clases del diagrama de clases antes presentado

```

public class Cliente extends Thread{

    private final int MAX_X = 578;
    private final int MAX_Y = 246;
    private int posicionEnFila;
    private int tiempo;
    private int distancia;
    private int id;
    private int saldo;
    private List<Proceso> procesos;
    private Random random;
    private JLabel icono;
    private JTextField saldoInterfaz;

    public Cliente(int id, JLabel icono, JTextField saldoInterfaz, int posicionEnFila) {
        this.id = id;
        this.saldo = 100;
        this.procesos = new ArrayList<>();
        this.random = new Random();
        this.icono = icono;
        this.saldoInterfaz = saldoInterfaz;
        this.posicionEnFila = posicionEnFila;
        generarProcesos();
    }
}

```

```
public class Cajero {  
  
    private int id;  
    private Random random;  
    private Cliente cliente;  
  
    public Cajero(int id, Cliente cliente) {  
        this.id = id;  
        this.cliente = cliente;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public Cliente getCliente() {  
        return cliente;  
    }  
  
    public void setCliente(Cliente cliente) {  
        this.cliente = cliente;  
    }  
}  
  
public class Fila {  
  
    private List<Cliente> fila;  
    private List<Cliente> enEspera;  
    private Random random;  
    private boolean isLibre;  
    private final int INICIO_X = 18;  
    private final int INICIO_Y = 24;  
    private int tiempo;  
  
    public Fila(List<Cliente> clientes) {  
        enEspera = new ArrayList<>();  
        isLibre = true;  
        fila = clientes;  
        random = new Random();  
    }  
  
    public synchronized void ingresarEnFila(Cliente cliente) {
```

```
public class Proceso {

    private Random random = new Random();
    private int valor;

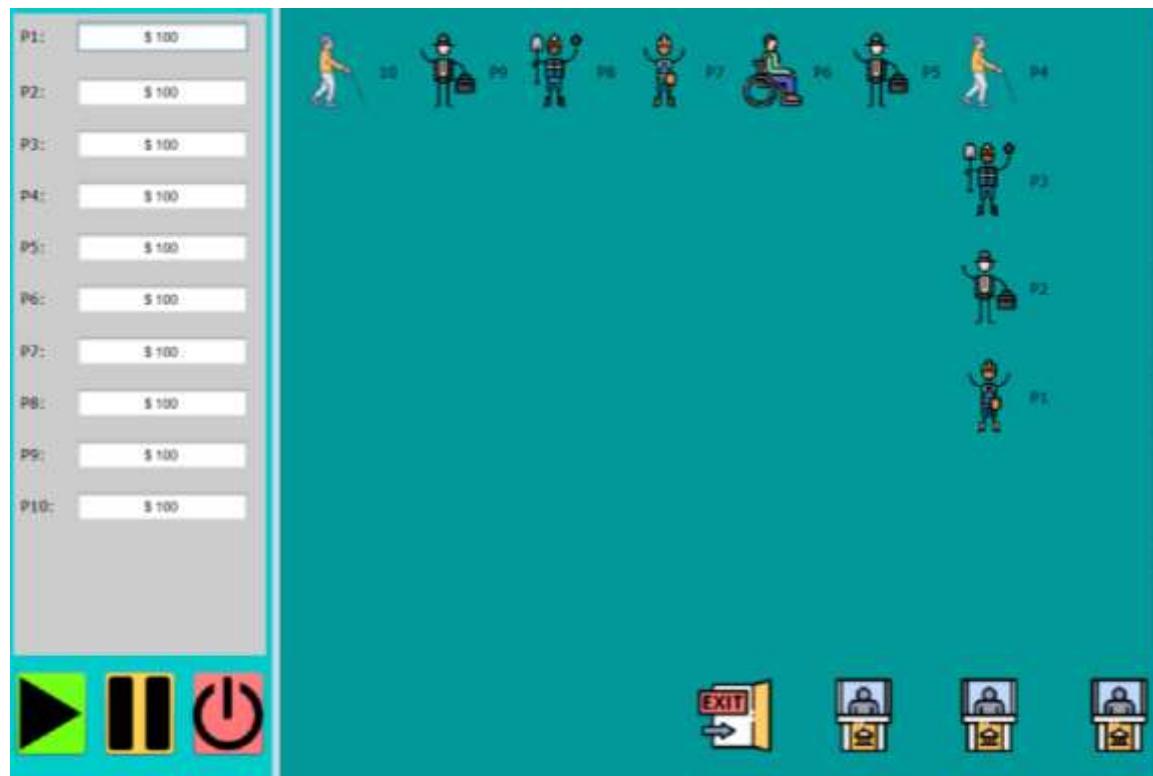
    public Proceso(int valor) {
        this.valor = valor;
        this.random = new Random();
    }

    public int getValor() {
        return valor;
    }

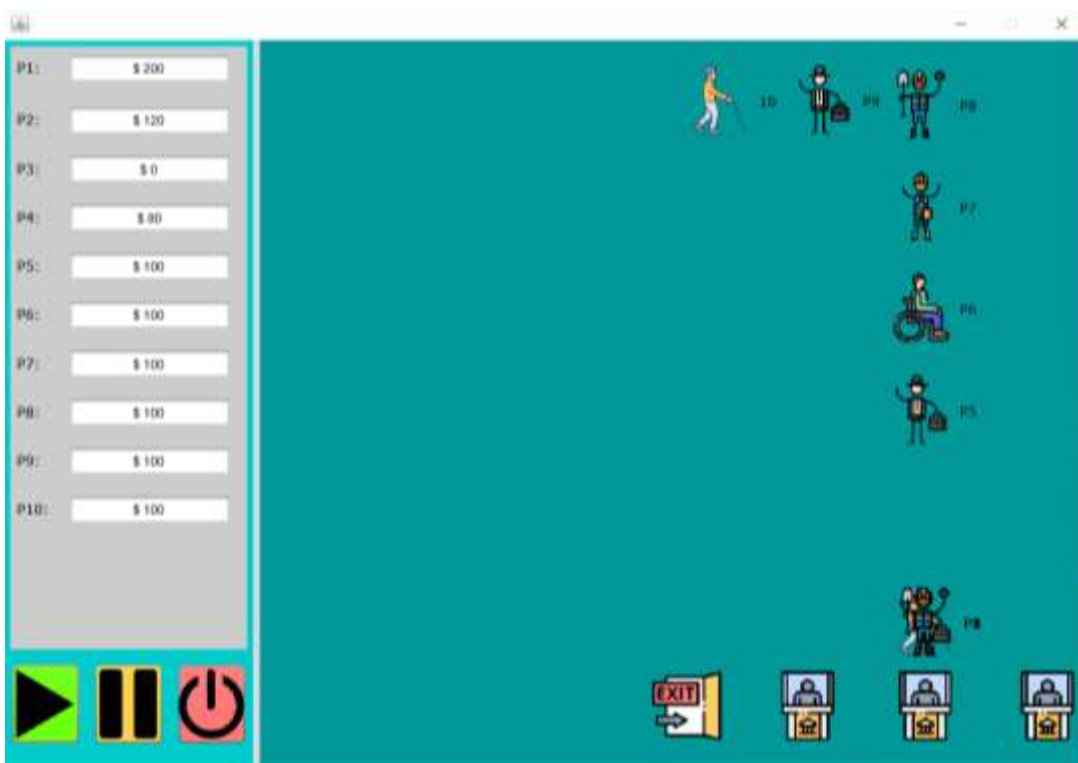
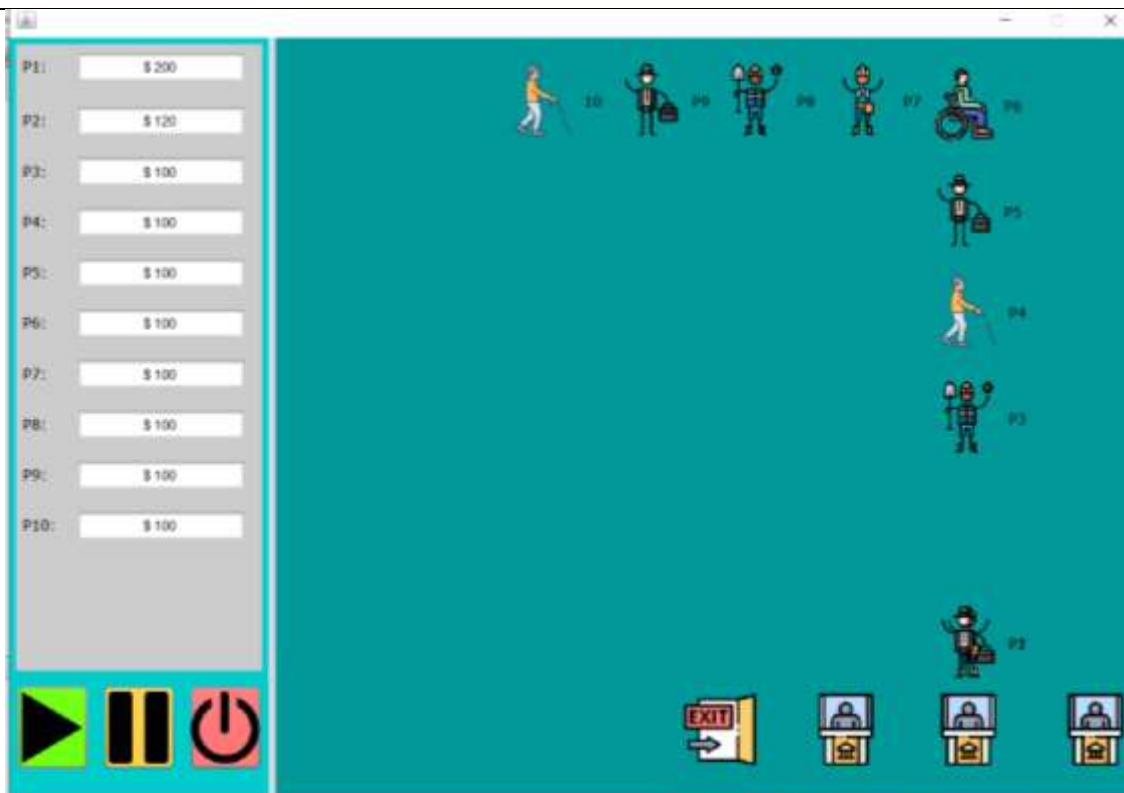
    public void setValor(int valor) {
        this.valor = valor;
    }


}
```

- Comprobación de las cuentas bancarias e interfaz gráfica.







	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

4. Deberá generar un informe de la practica en formato PDF y en conjunto con el código se debe subir al GitHub personal y AVAC.

<https://github.com/VazquezAriel/PruebaPractica2.git>

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 graficas 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: Ariel Vazquez

Firma de Estudiante:

