UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES

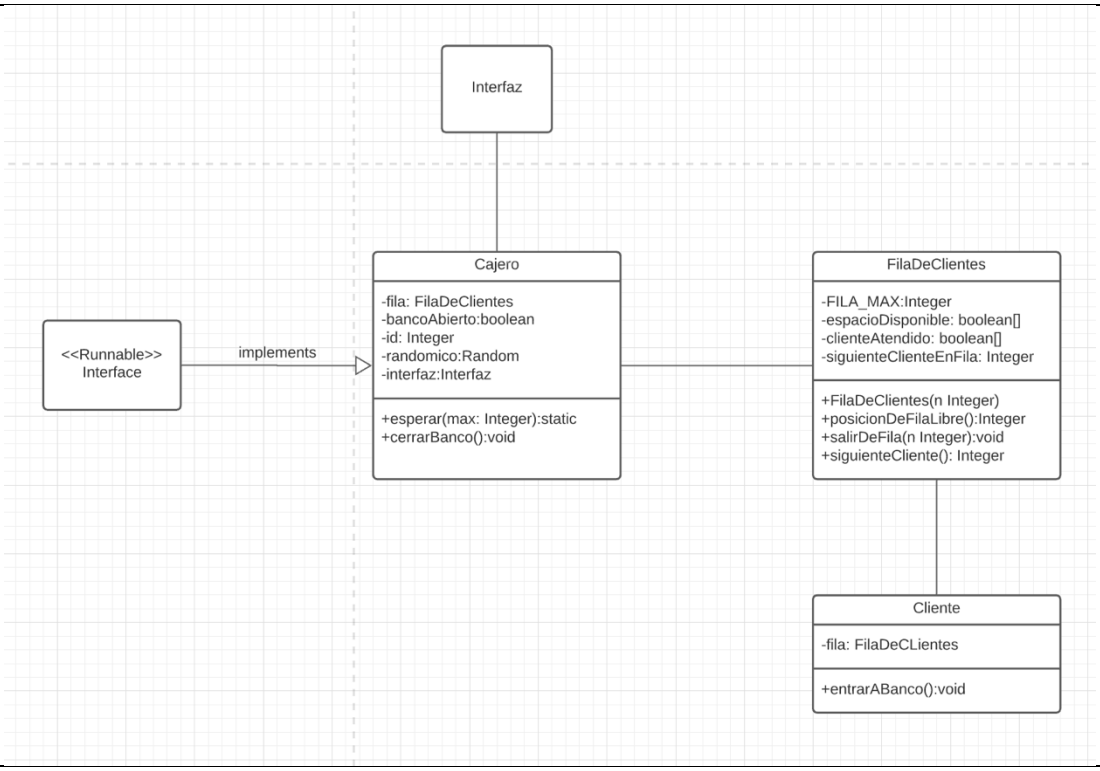
CARRERA: Computación		**ASIGNATURA**: Programación Aplicada
NRO. PRÁCTICA:		**TÍTULO PRÁCTICA**: Prueba practica 2

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

ACTIVIDADES DESARROLLADAS

https://github.com/ddutan2000/PruebaPractica2.git

1.



2. Se pidió crear un programa que simule un banco que ofrece tres cajero y 100 clientes que operen esos cajeros. Aparte los clientes deberán depositar o retirar dinero al azar. Para este programa se crearon tres paquetes:

- **Ec.edu.ups.modelo**
- **Ec.edu.ups.imagen**
- **Ec.edu.ups.test**

3. **Ec.edu.ups.modelo**
Este paquete contiene todas las clases que se utilizaran para simular el banco.

- Cajero.java
- Cliente.java
- FilaDeCliente.java

4. **filaDeCliente.java**
Esta clase se creo para manejar el flujo de entrada de los clientes. De esta manera cada cliente será atendido por cada cajero disponible.

Atributos:

- private int FILA_MAX;
- private boolean[] espacioDisponible;
- private boolean[] clienteAtendido;
- private int siguienteClienteEnFila = 0;

El constructor de esta clase recibe un objeto de tipo entero en el cual definirá el tamaño de la fila. Aparte el constructor inicializa todos sus atributos dentro del constructor.

Métodos:

- posicionDeFilaLibre: este método esta sincronizado y retorna un valor entero. Al utilizar este método se inicializa un atributo el cual definirá la posición. Este método cambia los estados de la lista para que aparezca como que cierta posición en la lista ya esta ocupado.
- salirDeFila: Este método recibe un numero entero en su parámetro. Este método realiza la tarea de vaciar el puesto que ocupaba el cliente y permite que otro cliente pueda ingresar en el.
- siguienteCliente: Este método retorna un objeto de tipo entero y aparte es un método sincronizado. Este método cumple con varias tareas a la vez. Primero revisa si el cliente esta en la fila y luego revisa si es que ya fue atendido, en caso de cumplir con ambas condiciones se le cambia el estado del cliente a true. Y retornamos el numero del cliente que ya fue atendido. En caso de no cumplir simplemente se va sumando un valor mas para atender al siguiente cliente. Otra condición que tiene este método es que si el valor de búsqueda es igual al máximo valor de la fila se reinicia la búsqueda del cliente. La ultima condición que tiene este método es que si la búsqueda del cliente es el mismo valor del cliente que sigue se sale de while.

5. Cliente.java

Esta clase se utiliza para simular si es que los clientes se quedaron son espacio en la fila.

Atributos:

- private FilaDeClientes fila;

el constructor de este método solamente recibe un objeto de la clase FilaDeCliente.

Métodos:

- entrarABanco: Este metodo realiza la tarea solamente de notificar al banco si es que hay suficiente espacio en la fila. En caso de que no exista espacio el cliente se va, caso contrario se una a la fila.

6. Cajero.java

Esta clase simula las actividades de un cajero, y a la misma vez se debe tener en cuenta que esta clase implementa la interfaz Runnable.

Atributos:

- private FilaDeClientes fila;
- private boolean bancoAbierto;
- private int id;
- private Random randomico = new Random();
- private Interfaz interfaz;

El constructor de esta clase recibe en su parámetro los siguientes objetos: FilaDeCliente de la clase FilaDeCliente, un objeto de tipo entero, y la interfaz donde se simulará el proceso.

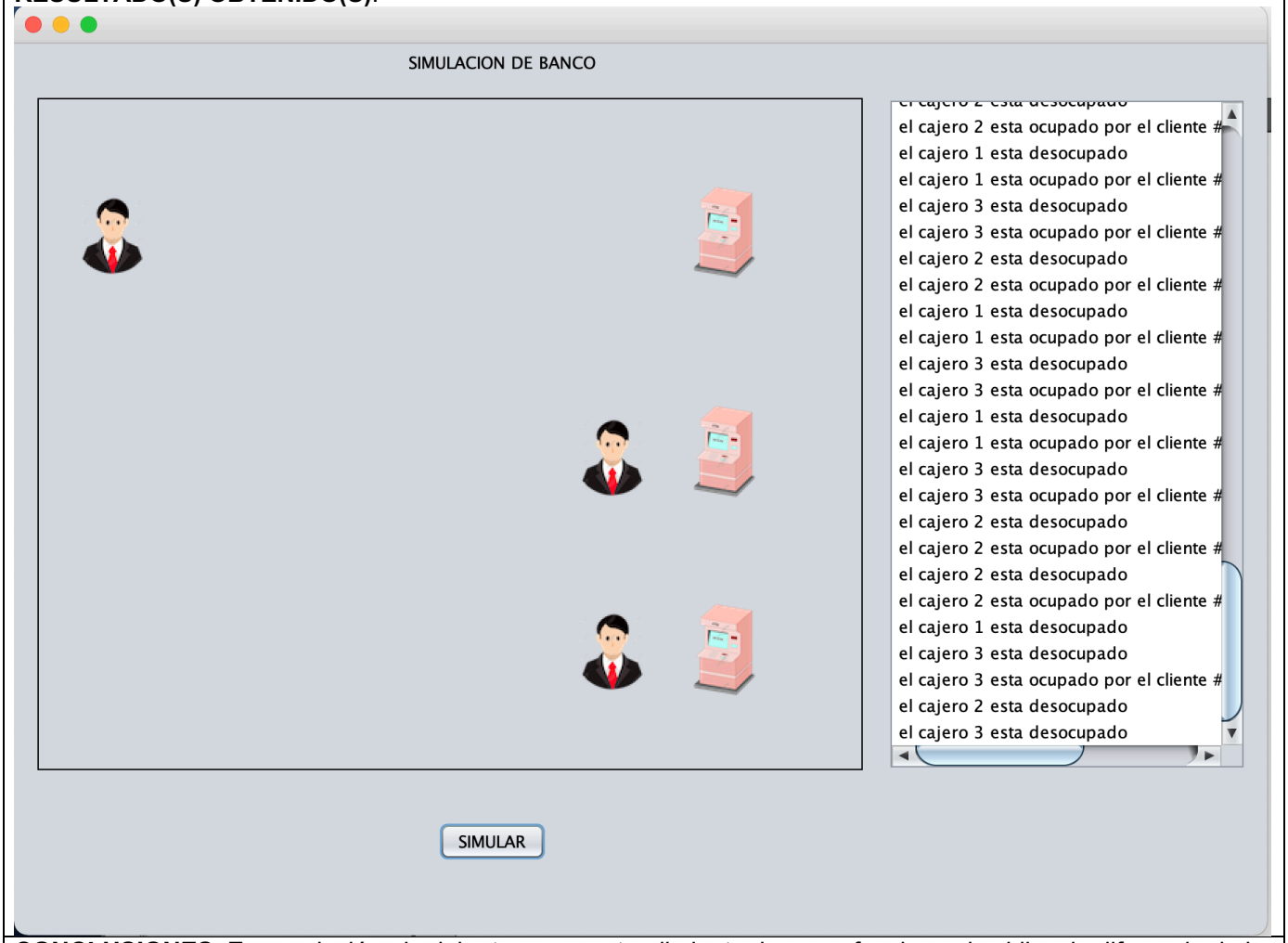
Métodos:

- Esperar: Este método recibe un objeto de tipo entero en su parámetro, también se debe tener en cuenta que este método es un método estático. Este método realiza la tarea de dormir al hilo por un tiempo al azar al convertir el numero entero en su parámetro en milisegundos.
- cerrarBanco: Este método solamente realiza la tarea de finalizar los hilos.

- Run: Este método esta sobrescrito de la interfaz Runnable. Realiza la tarea de que los cajeros empiezan a ser atendidos por los clientes y cuando los clientes terminan de utilizar los cajeros rápidamente son ocupados por los que siguen en la fila.

7. **Ec.edu.ups.imagen**
   Este paquete solamente contiene las imágenes con las cuales se simularan los procesos.
8. **Ec.edu.ups.test**
   Este paquete contiene la interfaz para visualizar los procesos.
   - interfaz
9. **interfaz.java**
   Este paquete es la interfaz donde se simularán los procesos del banco.
10.

RESULTADO(S) OBTENIDO(S):



CONCLUSIONES: En conclusión, de debe tener un entendimiento de como funcionan los hilos, la diferencia de la clase Thread con la interfaz Runnable. También se debe tener un entendimiento de como utilizar el método sychornized, la cual permite que algunos métodos se realicen a la misma vez sin tener que esperar que se acabe de ejecutar algunos métodos. Por ultimo se debe tener un buen manejo de la clase Thread y aparte se debe aprender a como sincronizar procesos eficazmente ya que si no se sincroniza adecuadamente el proceso que deseamos realizar no se ejecutara con éxito.

RECOMENDACIONES:

***Nombre de estudiante*: _____Denys Dutan_____**

***Firma de estudiante*: _____**