CIFP JUAN DE COLONIA

Programación de Procesos y Servicios

UT2: Práctica Cuenta Bancaria

Alexis López Briongos Dam2t

24/11/2023

Índice

Clase CuentaBancaria

- En esta clase vamos a controlar las operaciones de ingreso o retiro de una cuenta bancaria.

```java
public class CuentaBancaria {

    private int saldoInicial;
    private int saldoMaximo;

    public CuentaBancaria(int saldoInicial, int saldoMaximo) {
        this.saldoInicial = saldoInicial;
        this.saldoMaximo = saldoMaximo;
    }

    public synchronized boolean ingresoDinero(int cantidad) {
        boolean ingresoSatisfactorio; // Esta variable sera la que decida si se ha realizado con exito la operacion o
                                      // no
        if (cantidad > 0 && cantidad < saldoMaximo) {// Comprobamso que cantidad sea un numero válido para el ingreso
            double saldoTemp = saldoInicial; // Simplemente una variable para mostrar el saldo anterior a la operación
            saldoInicial += cantidad;
            System.out.println(
                    "El usuario " + Thread.currentThread().getName() + " ha ingresado " + cantidad + " a " + saldoTemp);
            System.out.println("Saldo actual: " + saldoInicial);
            ingresoSatisfactorio = true; //Establecemos a true la variable de infresoSatisfactorio
        } else {
            System.err.println(
                    "El usuario " + Thread.currentThread().getName() + " no  puede realizar el ingreso por que "
                            + cantidad + " supera al saldo máximo permitido: " + saldoMaximo);
            ingresoSatisfactorio = false; //Establecemos a false la variable de infresoSatisfactorio
        }

        return ingresoSatisfactorio; //retornamos la variable booleana

    }
```

- Comprobamos si la cantidad a ingresar es válida y en consecuencia agregamos el saldo o no y cambiamos el valor de la variable booleana con lo que trabajaremos en la clase Persona para poder identificar los ingresos o retiros satisfactorios o los errores cometidos en las operaciones.

- Misma metodología para el método de retirar dinero.

Clase Persona

- En esta clase controlaremos el acceso de las operaciones de la clase CuentaBancaria.

```java
public class Persona extends Thread {
    private String nombre;
    private CuentaBancaria cb;
    private int contIngresos;
    private int contRetiros;
    private int contFallos;

    public Persona(String nombre, CuentaBancaria cb) {
        this.nombre = nombre;
        this.cb = cb;
        this.setName(nombre);
        contIngresos = 0;
        contRetiros = 0;
        contFallos = 0;

    }

    @Override
    public void run() {

        while (!isInterrupted()) { //
            int aleatorio = (int) (Math.random() * 500 + 1); // Generamos ingreso o retiro aleatorio
            if (cb.ingresoDinero(aleatorio)) { // Si se cumple el ingreso sumamos 1 al contador de ingresos
                contIngresos++;
            } else {
                contFallos++; // Si no sumamos 1 al contador de fallos
                synchronized (this) { // Sincronizamos el bloque para poder utilizar el método wait()
                    try {
                        System.err.println("El usuario " + nombre + " ha fallado en la operación"
                                + "de ingreso. Intentos restantes: " + contFallos + "/3");
                        wait(5000); // si no se puede realizar la operación hacemos esperar al hilo 5 segundos
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
```

- En el constructor agregamos el nombre de la persona y la cuenta bancaria con la que trabajaremos.

- En el método run() realizamos un bucle while hasta que el hilo sea interrumpido (más adelante veremos cómo lo hacemos).

- Inicializamos una variable de tipo int para que la cantidad de retiro o ingreso sea aleatoria. Comprobamos con el método ingresoDinero() pasándole la cantidad aleatoria si es válida y se ha podido realizar el ingreso o no.

- En consecuencia al resultado del método ingresoDinero() vamos a aumentar el contadorIngresos el cual controla el numero de ingresos que podemos realizar por usuario (2 por operación). Si no se cumple la condición aumentamos el contador de Fallos (3 max) y ponemos en espera al hilo durante 5000 milisegundos (5 segundos) siendo necesario establecer como sincronizado el bloque en el que vamos a ejecutar este método wait().

- Si estas condiciones anteriormente mencionadas se cumplen se interrumpe el hilo.

```
        try {
            Thread.sleep(3000); // Tiempo en milisegundos para alternar entre las operaciones de ingreso o
                                // retiro
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        }
        // Misma metodología para los retiros
        if (cb.retiroDinero(aleatorio)) {
            contRetiros++;
        } else {
            contFallos++;
            synchronized (this) {
                try {
                    System.err.println("El usuario " + nombre + " ha fallado en la operación"
                            + "de ingreso. Intentos restantes: " + contFallos + "/3");
                    wait(5000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
        if (contIngresos == 2 && contRetiros == 2) { // Si llega el contador de ingresos a 2 y retiros a 2 se
                                                     // interrumpe el hilo
            interrupt();
        } else if (contFallos > 2) { // si llega el contador de fallos a 2 , se interrumpe el hilo
            System.err.println("El usuario " + nombre + " ha superado el límite de intentos de operaciónes. "
                    + "Intentélo de nuevo mas tarde.");
            interrupt();
        }
    }
```

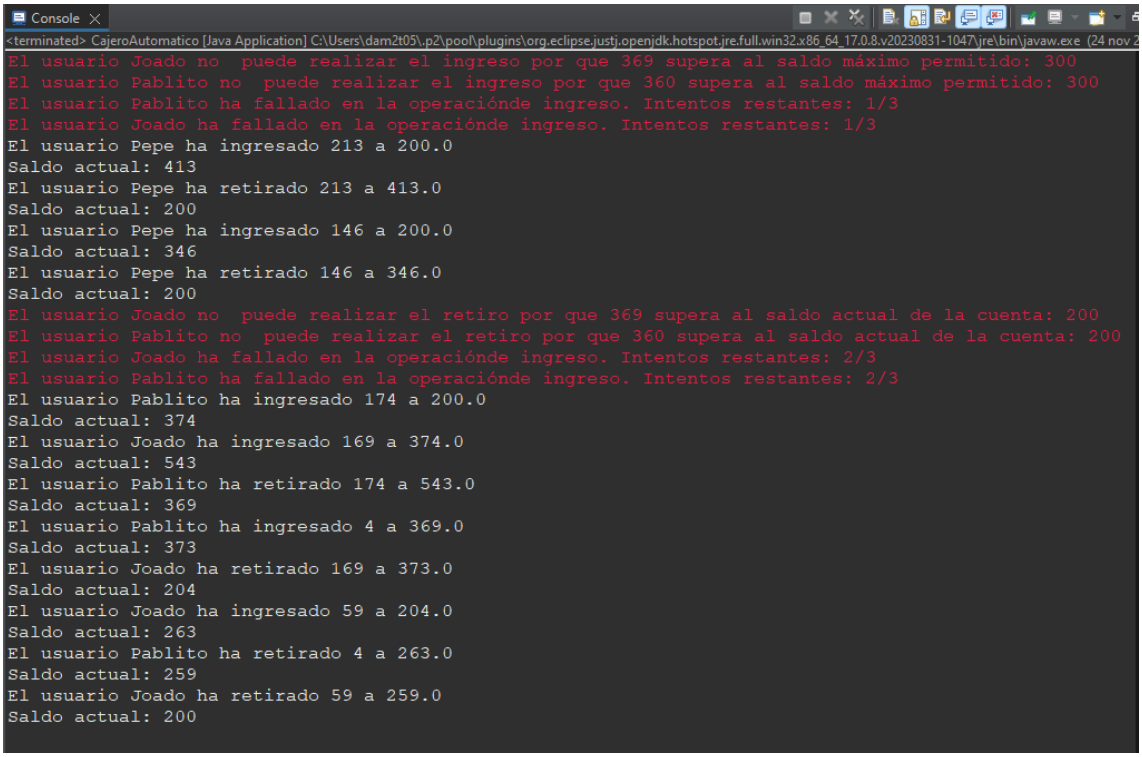
- Establecemos un periodo de espera entre las operaciones Ingreso y retiro con un Thread.sleep() de 3000 milisegundos(3 segundos).
- Evaluamos las variables contIngresos , contRetiros y contFallos por cada iteración. Si se cumple cualquiera de estas condiciones definidas el hilo se interrumpe.

Clase Cajero Automático

```
public class CajeroAutomatico {

    public static void main(String[] args) {
        CuentaBancaria cb = new CuentaBancaria(200,300);
        Persona Joado = new Persona("Joado",cb);
        Persona Pablito = new Persona("Pablito",cb);
        Persona Pepe = new Persona("Pepe",cb);

        Joado.start();
        Pablito.start();
        Pepe.start();


        try {
            Joado.join();
            Pablito.join();
            Pepe.join();
        } catch (Exception e) {

        }

    }
```

- En esta clase simplemente instanciamos las clases, asignamos misma cuenta bancaria para que todas operen en la misma e inicializamos los hilos recordando que la clase Persona extiende de Thread.

Resultado consola



- Este sería el resultado de la ejecución de los hilos Persona.
- Podemos visualizar que Joado realiza una operación errónea y no puede volver a intentar una operación hasta pasado 5 segundos. Mismo caso con Pablito.
- Sin embargo Pepe realiza los ingresos y los retiros de forma intercalada sin ningún error ocasionado pudiendo finalizar su hilo con total normalidad.
- En el caso de Joado y Pablito al no haber cometido 3 errores pueden volver a intentar realizar operaciones y así las realizan de forma intercalada hasta completar su número de operaciones.