Memoria Practica II Threads

Que

Se desea realizar una clase CuentaBancaria que simulará las Opes que se realizan en una cuenta. Para ello, crearemos diferentes hilos, cada uno de ellos con un tipo de operación. Además, como cada operación tiene diferentes frecuencias, también deberíamos indicar, cuánto tiempo deben dormir después de cada ejecución del hilo, quedando así:

- Nómina::ingreso::1200::3000 (concepto, Op, cantidad, tiempo_espera)
- Hipoteca::cobro::400::3000
- Luz::cobro::40::3000
- Agua::cobro::30::3000
- Compras::cobro::50::1000
- Retirada efectivo::cobro::20::300

Al inicializar el programa, se deberá indicar el saldo actual de nuestra cuenta. Cada operación de ingreso o cobro deberá tener un número de operación que se irá incrementando. Para evitar la condición de carrera, se deben sincronizar aquellas secciones que consideréis críticas mediante el uso de synchronized en métodos o sentencias.

Para que

Esta actividad tiene como objetivo proporcionar un aprendizaje básico sobre la utilización de hilos en Java. A través de esta práctica, exploramos de manera sencilla cómo trabajar con threads y comprender su funcionamiento. Esta práctica es útil, ya que nos brinda una introducción básica pero más abanzada que la anterior de cómo los hilos operan y las diferentes formas en que pueden ser utilizados y aplicados.

Pseudocodigo

CuentaBancaria

```
Crear clase CuentaBancaria
    Inicializar saldo con saldoInicial
    Crear método Ingresar(concepto, cantidad, tiempoEspera)
        Incrementar saldo en cantidad
        Imprime los datos de la operación
        Imprime el saldo actual
        Dormir en tiempoEspera
    Crear método Cobrar(concepto, cantidad, tiempoEspera)
        Decrementar saldo en cantidad
        Imprime los datos de la operación
        Imprime el saldo actual
        Dormir en tiempoEspera
    Crear método privado dormir(tiempo)
        Intentar dormir en tiempo
    Crear variable estática privada numeroOp inicializada en 1
    Crear método estático privado obtenerNumeroOp()
        Devolver numeroOp y luego incrementarlo
```

OpNomina

```
Crear clase OpNomina que extiende Thread
    Crear variable privada cuenta de tipo CuentaBancaria
    Crear constructor que recibe una cuenta
    Crear método run()
        Llamar a cuenta.Ingresar con parámetros "nómina", 1200, 3000
```

OpHipoteca

```
Crear clase OpHipoteca que extiende Thread
    Crear variable privada cuenta de tipo CuentaBancaria
    Crear constructor que recibe una cuenta
    Crear método run()
        Llamar a cuenta.Cobrar con parámetros "hipoteca", 400, 3000
```

OpLuz

```
Crear clase OpLuz que extiende Thread
    Crear variable privada cuenta de tipo CuentaBancaria
```

```
    Crear constructor que recibe una cuenta
    Crear método run()
        Llamar a cuenta.Cobrar con parámetros "luz", 40, 3000
```

OpAgua

```
Crear clase OpAgua que extiende Thread
    Crear variable privada cuenta de tipo CuentaBancaria
    Crear constructor que recibe una cuenta
    Crear método run()
        Llamar a cuenta.Cobrar con parámetros "agua", 30, 3000
```

OpCompras

```
Crear clase OpCompras que extiende Thread
    Crear variable privada cuenta de tipo CuentaBancaria
    Crear constructor que recibe una cuenta
    Crear método run()
        Llamar a cuenta.Cobrar con parámetros "compras", 50, 1000
```

OpRetiradaEfectivo

```
Crear clase OpRetiradaEfectivo que extiende Thread
    Crear variable privada cuenta de tipo CuentaBancaria
    Crear constructor que recibe una cuenta
    Crear método run()
        Llamar a cuenta.Cobrar con parámetros "retirada efectivo", 20, 300
```

Salir

```
Crear clase Salir que extiende Thread
    Crear método run()
        Crear scanner sc para entrada estándar
        Mientras sea verdad
            Leer entrada del usuario
            Si la entrada es "c"
                Salir del programa
```

SimuladorCuentaBancaria

```
Crear clase SimuladorCuentaBancaria
  En el método principal (main)
      Crear una cuenta bancaria con saldo inicial de 4000
      Crear un hilo Salir llamado salirThread
      Iniciar salirThread
      Loop infinito
            Crear hilos para cada operación: nominaThread, hipotecaThread,
luzThread, aguaThread, comprasThread, retiradaThread
          Iniciar cada hilo
```

Como

Conclusión

Esta práctica ha sido útil para aprender a programar con hilos en Java.