Actividad 5: ejercicio de hilos

2º Desarrollo de Aplicaciones Multiplataforma
Programación de servicios y procesos

19/11/2021

Martínez Díez, Ángel Mori

Contenido

1. Abre el archivo de apoyo "Multitarea e hilos en Java..." introduce y ejecuta el ejercicio de las cajeras.

a. Original

Código

```java
public class Cajera {
    private String nombre;

    public Cajera(String nombre) {
        this.nombre = nombre;
    }

    public String getNombre() {
        return this.nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public void procesarCompra(Cliente cliente, long timeStamp) {
        System.out.println("La cajera " + this.nombre + " COMIENZA A PROCESAR LA COMPRA DEL CLIENTE "
                + cliente.getNombre() + " EN EL TIEMPO: " + (System.currentTimeMillis() - timeStamp) / 1000 + "seg");

        for (int i = 0; i < cliente.getCarroCompra().length; i++) {
            this.esperarXsegundos(cliente.getCarroCompra()[i]);
            System.out.println("Procesado el producto " + (i + 1) + " ->Tiempo: "
                    + (System.currentTimeMillis() - timeStamp) / 1000 + "seg");
        }
        System.out.println("La cajera " + this.nombre + " HA TERMINADO DE PROCESAR " + cliente.getNombre()
                + " EN EL TIEMPO:" + (System.currentTimeMillis() - timeStamp) / 1000 + "seg");
    }

    private void esperarXsegundos(int segundos) {
        try {
            Thread.sleep(segundos * 1000);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
    }
}
```

```java
public class Cliente {
    private String nombre;

    private int[] carroCompra;

    // Constructor, getter y setter

    public Cliente(String nombre, int[] carroCompra) {
        this.nombre = nombre;
        this.carroCompra = carroCompra;
    }

    public String getNombre() {
        return this.nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int[] getCarroCompra() {
        return this.carroCompra;
    }

    public void setCarroCompra(int[] carroCompra) {
        this.carroCompra = carroCompra;
    }

}
```

```java
public class Main {
    Run | Debug
    public static void main(String[] args) {
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });
        Cajera cajera1 = new Cajera("Cajera 1");
        Cajera cajera2 = new Cajera("Cajera 2");
        // Tiempo inicial de referencia
        long initialTime = System.currentTimeMillis();
        cajera1.procesarCompra(cliente1, initialTime);
        cajera2.procesarCompra(cliente2, initialTime);
    }
}
```

Salida

```
[Running] cd "/Users/angel/Documents/GitHub/DAM/Servicios y procesos/UT2-Hilos/Tareas/Ejercicio de hilos/src/" && javac Main.java && java Main
La cajera Cajera 1 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO:15seg
La cajera Cajera 2 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIEMPO: 15seg
Procesado el producto 1 ->Tiempo: 16seg
Procesado el producto 2 ->Tiempo: 19seg
Procesado el producto 3 ->Tiempo: 24seg
Procesado el producto 4 ->Tiempo: 25seg
Procesado el producto 5 ->Tiempo: 26seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO:26seg

[Done] exited with code=0 in 27.578 seconds
```

b. Thread

Código

Cliente queda como en el original.

```java
public class CajeraThread extends Thread {
    private String nombre;

    private Cliente cliente;

    private long initialTime;

    public CajeraThread(String nombre, Cliente cliente, long initialTime) {
        this.nombre = nombre;
        this.cliente = cliente;
        this.initialTime = initialTime;
    }

    public Cliente getCliente() {
        return this.cliente;
    }

    public void setCliente(Cliente cliente) {
        this.cliente = cliente;
    }

    public long getInitialTime() {
        return this.initialTime;
    }

    public void setInitialTime(long initialTime) {
        this.initialTime = initialTime;
    }

    public String getNombre() {
        return this.nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    @Override
    public void run() {
        System.out.println(
                "La cajera " + this.nombre + " COMIENZA A PROCESAR LA COMPRA DEL CLIENTE " + cliente.getNombre()
                        + " EN EL TIEMPO: " + (System.currentTimeMillis() - this.initialTime) / 1000 + "seg");
        for (int i
            this.es
            System.
        }
        System.out.println("La cajera " + this.nombre + " HA TERMINADO DE PROCESAR " + cliente.getNombre()
                + " EN EL TIEMPO:" + (System.currentTimeMillis() - this.initialTime) / 1000 + "seg");
    }
}
```

```
void java.io.PrintStream.println(String x)

Prints a String and then terminate the line. This method behaves as though it invokes
print(String) and then println().

• Parameters:
  ◦ x The String to be printed.
```

```java
    private void esperarXsegundos(int segundos) {
        try {
            Thread.sleep(segundos * 1000);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
    }
}
public class MainThread {
    Run | Debug
    public static void main(String[] args) {
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });

        // Tiempo inicial de referencia
        long initialTime = System.currentTimeMillis();

        CajeraThread cajera1 = new CajeraThread("Cajera 1", cliente1, initialTime);
        CajeraThread cajera2 = new CajeraThread("Cajera 2", cliente2, initialTime);

        cajera1.start();
        cajera2.start();
    }
}
```

Salida

```
[Running] cd "/Users/angel/Documents/GitHub/DAM/Servicios y procesos/UT2-Hilos/Tareas/Ejercicio de hilos/src/" && javac MainThread.java && java MainThre
La cajera Cajera 2 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIEMPO: 0seg
La cajera Cajera 1 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg
Procesado el producto 1 ->Tiempo: 1seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 3 ->Tiempo: 9seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 5 ->Tiempo: 11seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO:11seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO:15seg

[Done] exited with code=0 in 16.854 seconds
```

c. *Runnable*

Código

Cajera y cliente quedan como en el original

```java
public class MainRunnable implements Runnable {

    private Cliente cliente;
    private Cajera cajera;
    private long initialTime;

    public MainRunnable(Cliente cliente, Cajera cajera, long initialTime) {
        this.cajera = cajera;
        this.cliente = cliente;
        this.initialTime = initialTime;
    }

    Run | Debug
    public static void main(String[] args) {
        Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2, 1, 5, 2, 3 });
        Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3, 5, 1, 1 });
        Cajera cajera1 = new Cajera("Cajera 1");
        Cajera cajera2 = new Cajera("Cajera 2");
        // Tiempo inicial de referencia
        long initialTime = System.currentTimeMillis();
        Runnable proceso1 = new MainRunnable(cliente1, cajera1, initialTime);
        Runnable proceso2 = new MainRunnable(cliente2, cajera2, initialTime);
        new Thread(proceso1).start();
        new Thread(proceso2).start();
    }

    @Override
    public void run() {
        this.cajera.procesarCompra(this.cliente, this.initialTime);
    }
}
```

Salida

```
[Running] cd "/Users/angel/Documents/GitHub/DAM/Servicios y procesos/UT2-Hilos/Tareas/Ejercicio de hilos/src/" && javac MainRunnable.java && java MainRu
La cajera Cajera 2 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIEMPO: 0seg
La cajera Cajera 1 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg
Procesado el producto 1 ->Tiempo: 1seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 3 ->Tiempo: 9seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 5 ->Tiempo: 11seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO:11seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO:15seg

[Done] exited with code=0 in 17.09 seconds
```

2. Abre el archivo "último hilos" y realiza uno de los ejercicios, el que tú quieras.

```java
import java.util.Random;


class Resultados {
    public static int ganancias;
    public static long tiempo_espera;
    public static int clientes_atendidos;
}
```

```
class Caja {
    private static final int MAX_TIME = 1000;
    private boolean ocupada;

    public Caja() {
        this.ocupada = false;
    }

    public boolean ocupada() {
        return ocupada;
    }

    synchronized public void atender(int pago) throws InterruptedException {
        ocupada = true;
        int tiempo_atencion = new Random().nextInt(MAX_TIME);
        Thread.sleep(tiempo_atencion);
        Resultados.ganancias += pago;
        Resultados.clientes_atendidos++;
        ocupada = false;
    }
}

class Cola {
    class Nodo {
        int cliente;
        Nodo sig;
    }

    Nodo raiz, fondo;
    Caja cajas[];
    int N;

    public Cola(int N) {
        raiz = null;
        fondo = null;
        this.N = N;
        cajas = new Caja[N];
        for (int i = 0; i < N; i++) {
            cajas[i] = new Caja();
        }
```

```
}

private boolean vacia() {
    if (raiz == null)
        return true;
    else
        return false;
}

private int cajaLibre() {
    int i = 0;
    while (i < N) {
        if (!cajas[i].ocupada()) {
            break;
        }
        i++;
    }
    return i;
}

synchronized public int esperar(int id_cliente) throws InterruptedException {
    int caja_id;
    Nodo nuevo;
    nuevo = new Nodo();
    nuevo.cliente = id_cliente;
    nuevo.sig = null;
    if (vacia()) {
        raiz = nuevo;
        fondo = nuevo;
    } else {
        fondo.sig = nuevo;
        fondo = nuevo;
    }
    // Esperar hasta el turno
    while (((caja_id = cajaLibre()) == N) || (raiz.cliente != id_cliente)) {
        // Me bloqueo hasta que sea mi turno
        wait();
    }
    // Salgo de la cola
    raiz = raiz.sig;
```

```
        return caja_id;
    }

    public void atender(int id_caja, int pago) throws InterruptedException {
        cajas[id_caja].atender(pago);
    }

    synchronized public void finalizar_compra() throws InterruptedException {
        notify();
    }
}

class Cliente extends Thread {
    private static final int MAX_DELAY = 2000;
    private static final int MAX_COST = 100;
    private int id;
    private Cola cola;

    Cliente(int id, Cola cola) {

        this.id = id;
        this.cola = cola;
    }

    public void run() {
        try {
            int numero_caja;
            System.out.println("Cliente " + id + " realizando compra");
            Thread.sleep(new Random().nextInt(MAX_DELAY));
            long s = System.currentTimeMillis();
            numero_caja = cola.esperar(id);
            cola.atender(numero_caja, new Random().nextInt(MAX_COST));
            System.out.println("Cliente " + id + " atendido en caja " + numero_caja);
            cola.finalizar_compra();
            System.out.println("Cliente " + id + " finalizando");
            long espera = System.currentTimeMillis() - s;
            Resultados.tiempo_espera += espera;
            System.out.println("Cliente " + id + " saliendo despues de esperar " + espera);
        } catch (InterruptedException e) {
            e.printStackTrace();
```

```
        }
    }
}

public class ModernSuperMarket {
    public static void main(String[] args) throws InterruptedException {
        int N = Integer.parseInt(args[0]);
        Cola cola = new Cola(N);
        int M = Integer.parseInt(args[1]);
        Cliente clientes[] = new Cliente[M];
        for (int i = 0; i < M; i++) {
            clientes[i] = new Cliente(i, cola);
            clientes[i].start();
        }
        try {
            for (int i = 0; i < M; i++) {
                clientes[i].join();
            }
        } catch (InterruptedException ex) {
            System.out.println("Hilo principal interrumpido.");
        }
        System.out.println("Supermercado cerrado.");
        System.out.println("Ganancias: " + Resultados.ganancias);
        System.out.println("Tiempo medio de espera: " + (Resultados.tiempo_espera /
Resultados.clientes_atendidos));
    }
}
```

Se obtienen resultados diferentes con las mismas condiciones de ejecución, ya que hay factores aleatorios:

```
MacBook-Pro-de-Angel:Ejercicio de hilos angel$  cd "/Users/angel/Documents/GitHub/DAM/Servicios y procesos/UT2-Hilos/Tareas/Ejercicio d
e hilos" ; /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java -Dfile.encoding=UTF-8 -cp "/Users/
angel/Documents/GitHub/DAM/Servicios y procesos/UT2-Hilos/Tareas/Ejercicio de hilos/bin" ModerSuperMarket.ModernSuperMarket 2 2
Cliente 1 realizando compra
Cliente 0 realizando compra
Cliente 0 atendido en caja 0
Cliente 0 finalizando
Cliente 0 saliendo despues de esperar 64
Cliente 1 atendido en caja 0
Cliente 1 finalizando
Cliente 1 saliendo despues de esperar 569
Supermercado cerrado.
Ganancias: 92
Tiempo medio de espera: 316
MacBook-Pro-de-Angel:Ejercicio de hilos angel$  cd "/Users/angel/Documents/GitHub/DAM/Servicios y procesos/UT2-Hilos/Tareas/Ejercicio d
e hilos" ; /usr/bin/env /Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home/bin/java -Dfile.encoding=UTF-8 -cp "/Users/
angel/Documents/GitHub/DAM/Servicios y procesos/UT2-Hilos/Tareas/Ejercicio de hilos/bin" ModerSuperMarket.ModernSuperMarket 2 2
Cliente 0 realizando compra
Cliente 1 realizando compra
Cliente 0 atendido en caja 1
Cliente 0 finalizando
Cliente 0 saliendo despues de esperar 172
Cliente 1 atendido en caja 0
Cliente 1 finalizando
Cliente 1 saliendo despues de esperar 655
Supermercado cerrado.
Ganancias: 131
Tiempo medio de espera: 413
MacBook-Pro-de-Angel:Ejercicio de hilos angel$
```