Davante

sistemas de gestión empresarial

Curso 2025/2026

EJERCICIO 1

```
public class Ejercicio1{
static class hilo extends Thread {
        private final String hijo;
        public hilo(String hijo) {
 super(hijo);
this.hijo = hijo;
  }
        @Override
        public void run() {
 for (int i = 1; i <= 5; i++) Sout(Thread.currentThread().getName() + " número " + i);
                try {
     Thread.sleep(50);
          } catch (InterruptedException e) {                Sout(Thread.currentThread().getName() + " error");
                    return;
    }
    }
    }
    public static void main(String[] args) throws InterruptedException {
        hilo h1 = new hilo("Hilo1");
```

```
        hilo h2 = new hilo("Hilo2");
        h1.start();
        h2.start()
        h1.join();
        h2.join();

Sout("Todos los hilos se han ejecutado");
    }
```

EJERCICIO 2

```
public class Ejercicio2{
    static class Proceso implements Runnable {
      private final String proceso1;
       public  Proceso(String proceso1) {
           this.proceso1 = proceso1;
        }
        @Override
        public void run() {
for (int i = 1; i <= 5; i++) {            System.out.println(Thread.currentThread().getName() + " ejecuta " + proceso1 + i);
      try {
                Thread.sleep(50);
  } catch (InterruptedException e) {            System.out.println(Thread.currentThread().getName() + " interrumpido.");
```

```
                return;
            }
        }
        System.out.println(Thread.currentThread().getName() + " ha terminado la tarea " + nombreTarea +
".");
    }
  }
  public static void main(String[] args) throws InterruptedException {
 Proceso procesoA  = new Proceso("procesoA");
 Proceso procesoB = new Proceso("procesoB");
 Thread numero1 = new Thread(procesoA, "Worker-A");
 Thread numero2 = new Thread(procesoB, "Worker-B");
      numero1.start();
     numero2.start();
      numero1.join();
      numero2.join();
  }
}
```

EJERCICIO 3

```
jimport java.util.ArrayList;
import java.util.List;
```

```
public class SupermercadoBasicoComentado {

    // Clase Cliente implementa Runnable
    static class Cliente implements Runnable {
        String nombre;

        Cliente(String nombre) {
            this.nombre = nombre;
        }

        @Override
        public void run() {
//comienza la simulación del supermercado
System.out.println(Thread.currentThread().getName() + " atiende a " + nombre);
          }
}
//creamos la clase caja y le damos una lista
    static class Caja implements Runnable {
        int numero;          // número de la caja
        List<Cliente> cola;   // lista de clientes asignados a esta caja

        Caja(int numero, List<Cliente> cola) {//constructor
            this.numero = numero;
            this.cola = cola;
```

```
    }

    @Override
    public void run() {
        // esto se mostrará desde la caja q esté
        System.out.println("Caja " + numero + " abre.");

        // Recorremos la cola de clientes y los atendemos
        // bucle donde el cliente es atendido en la caja según su posición en la cola.
        for (Cliente c : cola) {
            c.run(); // ejecución del cliente en el hilo de la caja
        }

        //el sout saldrá cuando la caja esté totalmente vacía después de atender a todos los clientes
        System.out.println("Caja " + numero + " cierra.");
    }
}

public static void main(String[] args) throws Exception {

    int NUM_CAJAS = 4; // Número de cajas disponibles

    // Creamos 12 clientes
    List<Cliente> clientes = new ArrayList<>();
```

```
for (int i = 1; i <= 12; i++) {
    clientes.add(new Cliente("Cliente-" + i));
}

// Creamos una lista de colas
List<List<Cliente>> colas = new ArrayList<>();
for (int i = 0; i < NUM_CAJAS; i++) colas.add(new ArrayList<>());

// Repartimos los clientes en las colas
for (int i = 0; i < clientes.size(); i++) {
    colas.get(i % NUM_CAJAS).add(clientes.get(i));
}

// Creamos los hilos para las cajas
Thread caja1 = new Thread(new Caja(1, colas.get(0)), "Caja-1");
Thread caja2 = new Thread(new Caja(2, colas.get(1)), "Caja-2");
Thread caja3 = new Thread(new Caja(3, colas.get(2)), "Caja-3");
Thread caja4 = new Thread(new Caja(4, colas.get(3)), "Caja-4");

// Arrancamos las 4 caja ejecutándose en paralelo
caja1.start();
caja2.start();
caja3.start();
caja4.start();
```

```
        // join() para que el hilo principal espere a que todas las cajas acaben
        caja1.join();
        caja2.join();
        caja3.join();
        caja4.join();

        System.out.println("\nTodas las cajas han terminado de atender a sus clientes.");
            System.out.println("Observa que los mensajes aparecen mezclados: eso indica ejecución concurrente.");
    }
}


}
```