Instituto Politécnico Nacional

Escuela Superior de Cómputo

Sistemas Distribuidos

PRACTICA 1:

Hilos En Java

Alumno:

Meza Díaz Arturo Rodrigo 2022630251

Grupo: 7CM1

Maestro: Carreto Arellano Chadwick

Fecha de realización: 20 de febrero de 2025

Fecha de entrega: 20 de febrero de 2025

Índice.

1. Antecedentes.

Un proceso es un programa en ejecución que puede estar compuesto por varios hilos de ejecución. Un hilo es la unidad mínima de procesamiento dentro de un proceso, que permite la ejecución concurrente de tareas. La programación concurrente es fundamental en el desarrollo de aplicaciones modernas, ya que permite aprovechar mejor los recursos del sistema y mejorar la eficiencia de las tareas.

En Java, los hilos se pueden manejar a través de la clase Thread o implementando la interfaz Runnable. La ejecución de hilos permite optimizar el rendimiento de los programas, mejorando la eficiencia y permitiendo la realización de tareas simultáneas. Además, Java proporciona herramientas como ExecutorService y Callable para una gestión más avanzada de los hilos, lo que permite una mejor administración de los recursos.

2. Planteamiento del problema.

Un proceso es un programa en ejecución que puede estar compuesto por varios hilos de ejecución. Un hilo es la unidad mínima de procesamiento dentro de un proceso, que permite la ejecución concurrente de tareas. La programación concurrente es fundamental en el desarrollo de aplicaciones modernas, ya que permite aprovechar mejor los recursos del sistema y mejorar la eficiencia de las tareas.

En Java, los hilos se pueden manejar a través de la clase Thread o implementando la interfaz Runnable. La ejecución de hilos permite optimizar el rendimiento de los programas, mejorando la eficiencia y permitiendo la realización de tareas simultáneas. Además, Java proporciona herramientas como ExecutorService y Callable para una gestión más avanzada de los hilos, lo que permite una mejor administración de los recursos.

3. Propuesta de solución.

Para solucionar el problema, se propone implementar una clase Impresora que implemente Runnable, permitiendo ejecutar cada impresora en un hilo separado. Se utilizará una cola para gestionar los trabajos de impresión y simular el tiempo de procesamiento mediante Thread.sleep(). Esto permitirá que los trabajos sean procesados de manera paralela sin interrupciones.

El diseño de la solución incluirá dos hilos separados, cada uno representando una impresora, con su propia cola de trabajos. Cada impresora tomará los trabajos de la cola y los procesará uno por uno, simulando el tiempo necesario para imprimir cada documento. Una vez finalizados los trabajos, el hilo de la impresora indicará que no hay más documentos pendientes.

4. Materiales y métodos.

Lenguaje:

- Java.

Librerías utilizadas:

- java.util.Queue: Para manejar la cola de trabajos de impresión.
- java.util.LinkedList: Implementación de la cola de trabajos.
- java.lang.Thread: Para la gestión de hilos.

Métodos implementados:

- agregarTrabajo(String trabajo): Agrega un trabajo a la cola de la impresora.
- run(): Procesa los trabajos en la cola hasta que esté vacía.
- Thread.sleep(): Simula el tiempo de procesamiento de cada trabajo.

5. Desarrollo de solución.

El código desarrollado implementa dos impresoras que trabajan en paralelo. Cada impresora recibe una lista de trabajos y los procesa en orden. Se utilizan hilos para ejecutar ambas impresoras simultáneamente, asegurando que los trabajos sean manejados de manera concurrente sin interferencias.

```java
import java.util.LinkedList;
import java.util.Queue;

public class Practica1 {

    // Clase que implementa Runnable para simular una impresora
    static class Impresora implements Runnable {
        private String nombre;
        private Queue<String> colaDeTrabajos;
        //crea una cola de trabajos para cada impresora
        public Impresora(String nombre) {
            this.nombre = nombre;
            this.colaDeTrabajos = new LinkedList<>();
        }

        // Método para agregar trabajos a la cola de la impresora
        public void agregarTrabajo(String trabajo) {
            colaDeTrabajos.add(trabajo);
        }

        @Override
        public void run() {
            //bucle que se ejecuta mientras haya trabajos en la cola
            while (!colaDeTrabajos.isEmpty()) {
                // Obtener el siguiente trabajo de la cola mediante poll(), que devuelve null si la cola
está vacía
                String trabajo = colaDeTrabajos.poll(); // Obtener el siguiente trabajo
                System.out.println(nombre + " está procesando: " + trabajo);

                // Simular el tiempo de procesamiento del trabajo
                try {
                    Thread.sleep(2000); // 2 segundos por trabajo
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }

                System.out.println(nombre + " ha terminado: " + trabajo);
            }
            System.out.println(nombre + " no tiene más trabajos.");
        }
    }

    public static void main(String[] args) {
        // Crear dos impresoras (hilos)
        Impresora impresora1 = new Impresora("Impresora 1");
        Impresora impresora2 = new Impresora("Impresora 2");

        // Agregar trabajos a las impresoras
        impresora1.agregarTrabajo("Trabajo 1");
        impresora1.agregarTrabajo("Trabajo 2");
        impresora1.agregarTrabajo("Trabajo 3");

        impresora2.agregarTrabajo("Trabajo A");
        impresora2.agregarTrabajo("Trabajo B");

       // Iniciar las impresoras (hilos)
        Thread hiloImpresora1 = new Thread(impresora1);
        Thread hiloImpresora2 = new Thread(impresora2);
        //inicia los hilos
        hiloImpresora1.start();
        hiloImpresora2.start();
    }
    //metodo toString
    @Override
    public String toString() {
        return "Practica1 []";
    }
}
```

6. Resultados.

Al ejecutar el programa, se observa que las impresoras procesan trabajos de manera concurrente. Cada impresora procesa su propia cola de trabajos sin interferir con la otra. La salida en consola es:

```
PS C:\Users\yoyia\OneDrive\Escritorio\ESCOM\OCTAVO\SistDistribuidos>  c:; cd '
in\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp
.java\jdt_ws\SistDistribuidos_1588b7f9\bin' 'Practica1'
Impresora 2 está procesando: Trabajo A
Impresora 1 está procesando: Trabajo 1
Impresora 2 ha terminado: Trabajo A
Impresora 2 está procesando: Trabajo B
Impresora 1 ha terminado: Trabajo 1
Impresora 1 está procesando: Trabajo 2
Impresora 2 ha terminado: Trabajo B
Impresora 2 no tiene más trabajos.
Impresora 1 ha terminado: Trabajo 2
Impresora 1 está procesando: Trabajo 3
Impresora 1 ha terminado: Trabajo 3
Impresora 1 no tiene más trabajos.
PS C:\Users\yoyia\OneDrive\Escritorio\ESCOM\OCTAVO\SistDistribuidos>
```

Esto demuestra que los hilos funcionan correctamente y procesan los trabajos de manera paralela.

7. Conclusiones.

En esta práctica implementé el uso de hilos en Java para la simulación de impresoras trabajando de manera concurrente. Logré la ejecución simultánea de tareas y la correcta gestión de colas de trabajos. El uso de Runnable y Thread me permitió la creación de hilos eficientes.

Por tanto, concluyo que el manejo de hilos es una herramienta útil para mejorar el rendimiento en aplicaciones con tareas concurrentes. Este conocimiento es esencial en el desarrollo de software moderno, ya que permite optimizar el uso de recursos y mejorar la eficiencia de los sistemas multitarea.