

Informe de las prácticas de experimentación y aplicación de los aprendizajes

1. Datos Informativos:

Facultad:	<i>CIENCIAS ADMINISTRATIVAS GESTIÓN EMPRESARIAL E INFORMÁTICA</i>
Carrera:	<i>SOFTWARE</i>
Asignatura:	<i>PROGRAMACION ORIENTADA A OBJETOS</i>
Ciclo:	<i>2</i>
Docente:	<i>ING. MÓNICA BONILLA</i>
Título de la práctica:	<i>IMPLEMENTACIÓN DE HILOS EN PROGRAMACIÓN ORIENTADA A OBJETOS</i>
No. de práctica:	<i>4</i>
Escenario o ambiente de aprendizaje de la practica	<i>En casa</i>
No. de horas:	<i>4</i>
Fecha:	<i>14/12/2024</i>
Estudiantes:	<i>Ariel Alejandro Calderón</i>
Calificación	

2. Introducción:

El uso de hilos de ejecución (o *threads*) en POO es crucial para mejorar el rendimiento y la capacidad de respuesta de las aplicaciones, especialmente en programas que requieren la ejecución concurrente de tareas. La práctica realizada tiene como propósito entender la implementación y gestión de hilos en el lenguaje Java, aplicando conceptos fundamentales de la POO. Esto incluye la creación, control y sincronización de hilos, para lograr una ejecución eficiente y sin errores en sistemas multihilo.

3. Objetivo de la práctica:

El objetivo principal de esta práctica es implementar hilos en un entorno de programación orientada a objetos, utilizando el lenguaje Java. Se busca que el estudiante comprenda cómo crear y gestionar múltiples hilos dentro de un programa, así como los mecanismos de sincronización necesarios para evitar problemas como la condición de carrera y el acceso simultáneo a recursos compartidos. Al finalizar la práctica, los estudiantes deberán ser capaces de crear aplicaciones multihilo, mejorando la eficiencia de sus programas.

4. Descripción del desarrollo de la práctica:

En esta práctica, se centró en la implementación de hilos en Java para ejecutar tareas concurrentes. La práctica comenzó con una explicación teórica sobre los hilos, su creación mediante la clase Thread y la interfaz Runnable, y los problemas que pueden surgir al trabajar con múltiples hilos, como las condiciones de carrera o el acceso simultáneo a recursos compartidos.

A continuación, se realizó un ejercicio práctico en el que se implementaron hilos para realizar varias operaciones matemáticas de manera concurrente. El escenario propuesto fue una calculadora simple que debía ejecutar diferentes operaciones (como sumar, restar, multiplicar y dividir) al mismo tiempo, lo que ilustró cómo los hilos pueden mejorar la eficiencia de un programa cuando se ejecutan tareas independientes.

Código del ejemplo:

```
public class CalculadoraMultihilo {

    // Clase interna para la operación de suma
    static class Suma implements Runnable {
        private int a, b;

        public Suma(int a, int b) {
            this.a = a;
            this.b = b;
        }

        @Override
        public void run() {
            int resultado = a + b;
            System.out.println("Resultado de la suma: " + resultado);
        }
    }

    // Clase interna para la operación de resta
    static class Resta implements Runnable {
        private int a, b;

        public Resta(int a, int b) {
            this.a = a;
```

```
this.b = b;
}

@Override
public void run() {
    int resultado = a - b;
    System.out.println("Resultado de la resta: " + resultado);
}

}

// Clase principal para la ejecución de los hilos
public static void main(String[] args) {
    // Crear hilos para las operaciones
    Thread hiloSuma = new Thread(new Suma(10, 5));
    Thread hiloResta = new Thread(new Resta(10, 5));

    // Iniciar los hilos
    hiloSuma.start();
    hiloResta.start();

    try {
        // Esperar que los hilos terminen su ejecución
        hiloSuma.join();
        hiloResta.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    System.out.println("Cálculos completados.");
}
}
```

Explicación del código:

- **Clases Suma y Resta:** Estas clases implementan la interfaz Runnable y realizan las operaciones de suma y resta, respectivamente, en sus métodos run().
- **Creación de hilos:** En el método main, se crean dos objetos Thread, uno para la operación de suma y otro para la resta. Cada hilo se ejecuta independientemente, lo que significa que ambas operaciones se realizarán al mismo tiempo si el sistema lo permite.
- **Sincronización:** Después de iniciar los hilos con start(), se utiliza el método join() - para asegurar que el programa principal espere a que ambos hilos finalicen antes

de imprimir el mensaje final. Esto asegura que los resultados de las operaciones se muestren correctamente en la consola, una vez que ambas tareas se hayan completado.

Ventajas de implementar hilos en este ejemplo:

- **Mejora en la eficiencia:** Al ejecutar operaciones de manera concurrente, el tiempo total de ejecución de todas las tareas se reduce.
- **Paralelismo:** Al dividir las operaciones en hilos separados, es posible aprovechar el paralelismo del hardware moderno. Si el sistema tiene múltiples núcleos, cada hilo puede ser ejecutado por un núcleo diferente, lo que mejora el rendimiento general.
- **Responsividad:** Si se implementan hilos en aplicaciones interactivas (por ejemplo, aplicaciones gráficas o servidores), los hilos pueden ejecutarse en segundo plano sin bloquear la interfaz de usuario. Esto mejora la experiencia del usuario.

Desventajas de implementar hilos:

- **Complejidad adicional:** La implementación de hilos introduce una capa adicional de complejidad. Aunque el ejemplo anterior es sencillo, en aplicaciones más grandes, gestionar múltiples hilos puede ser difícil de manejar.
- **Posibles errores difíciles de depurar:** Los errores relacionados con la sincronización y la ejecución concurrente pueden ser difíciles de detectar.

5. Metodología:

La metodología seguida fue de aprendizaje activo, donde se documentó una breve introducción teórica sobre la creación y gestión de hilos en programación orientada a objetos. Posteriormente, se les proporcionaron ejercicios prácticos que requerían la implementación de hilos en diferentes escenarios.

6. Resultados obtenidos:

Conocer el uso de hilos en Java, sus ventajas y desventajas

7. Conclusiones:

La implementación de hilos en programación orientada a objetos es una habilidad

fundamental para desarrollar aplicaciones eficientes que aprovechen la concurrencia en sistemas multi-núcleo. A través de esta práctica se aprendió a cómo crear hilo y cómo sincronizarlos para manejar situaciones complejas que involucran múltiples procesos simultáneos.

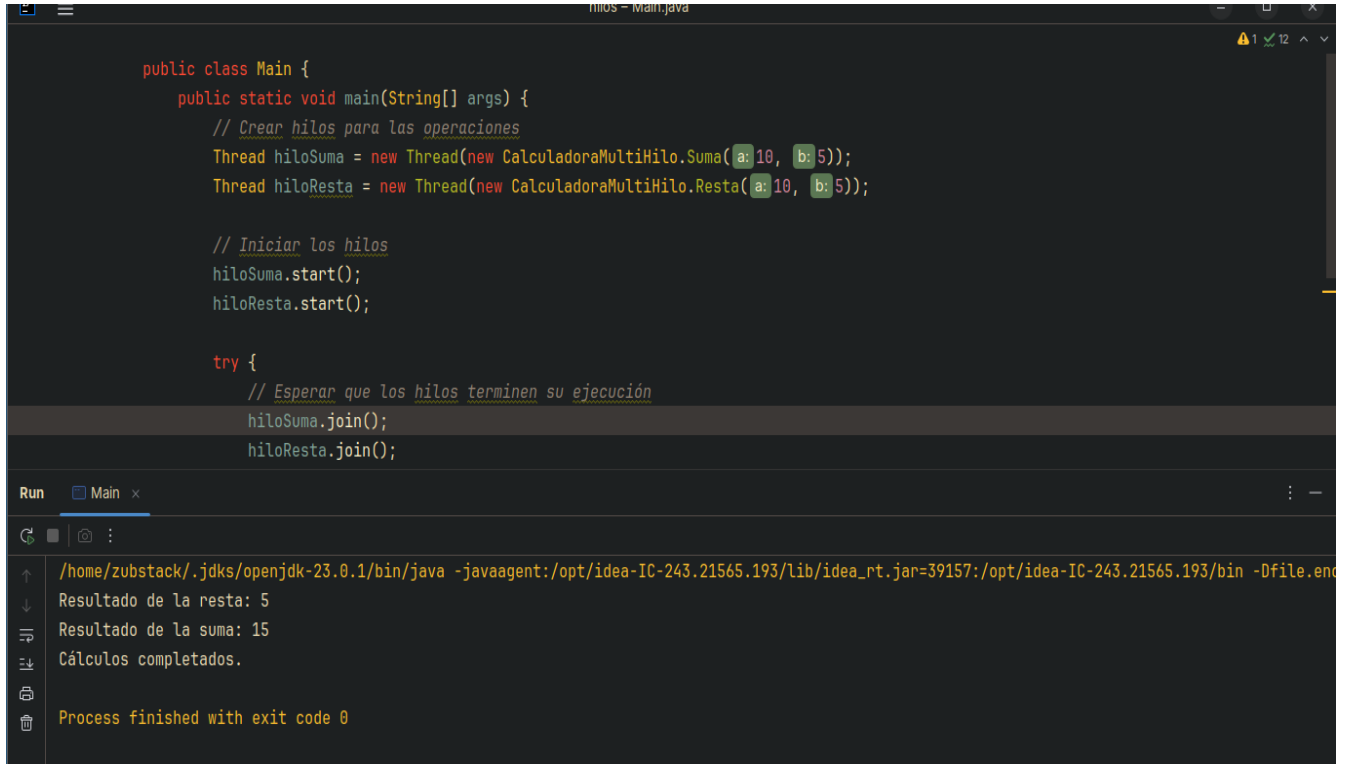
8. Recomendaciones:

Se recomienda practicar la implementación de hilos en proyectos más complejos, como el desarrollo de aplicaciones con interfaces gráficas de usuario (GUI) que requieren procesamiento en segundo plano. También se aconseja estudiar el impacto del uso de hilos en el rendimiento y cómo evitar el exceso de hilos, que puede llevar a una sobrecarga en el sistema.

9. Bibliografía:

- Eckel, B. (2006). *Thinking in Java* (4th ed.). Pearson Education.
- Bloch, J. (2008). *Effective Java* (2nd ed.). Addison-Wesley.
- Oracle Corporation. (2024). *Concurrency in Java*
<https://docs.oracle.com/javase/tutorial/essential/concurrency/>.

10. Anexos:



```
public class Main {
    public static void main(String[] args) {
        // Crear hilos para las operaciones
        Thread hiloSuma = new Thread(new CalculadoraMultiHilo.Suma(a: 10, b: 5));
        Thread hiloResta = new Thread(new CalculadoraMultiHilo.Resta(a: 10, b: 5));

        // Iniciar los hilos
        hiloSuma.start();
        hiloResta.start();

        try {
            // Esperar que los hilos terminen su ejecución
            hiloSuma.join();
            hiloResta.join();
        }
    }
}
```

Run Main x

/home/zubstack/.jdk/openjdk-23.0.1/bin/java -javaagent:/opt/idea-IC-243.21565.193/lib/idea_rt.jar=39157:/opt/idea-IC-243.21565.193/bin -Dfile.encoding=UTF-8

Resultado de la resta: 5
Resultado de la suma: 15
Cálculos completados.

Process finished with exit code 0

Anexo 1: Ejecucion de codigo ejemplo