INSTITUTO POLITECNICO NACIONAL

Escuela Superior de Computo

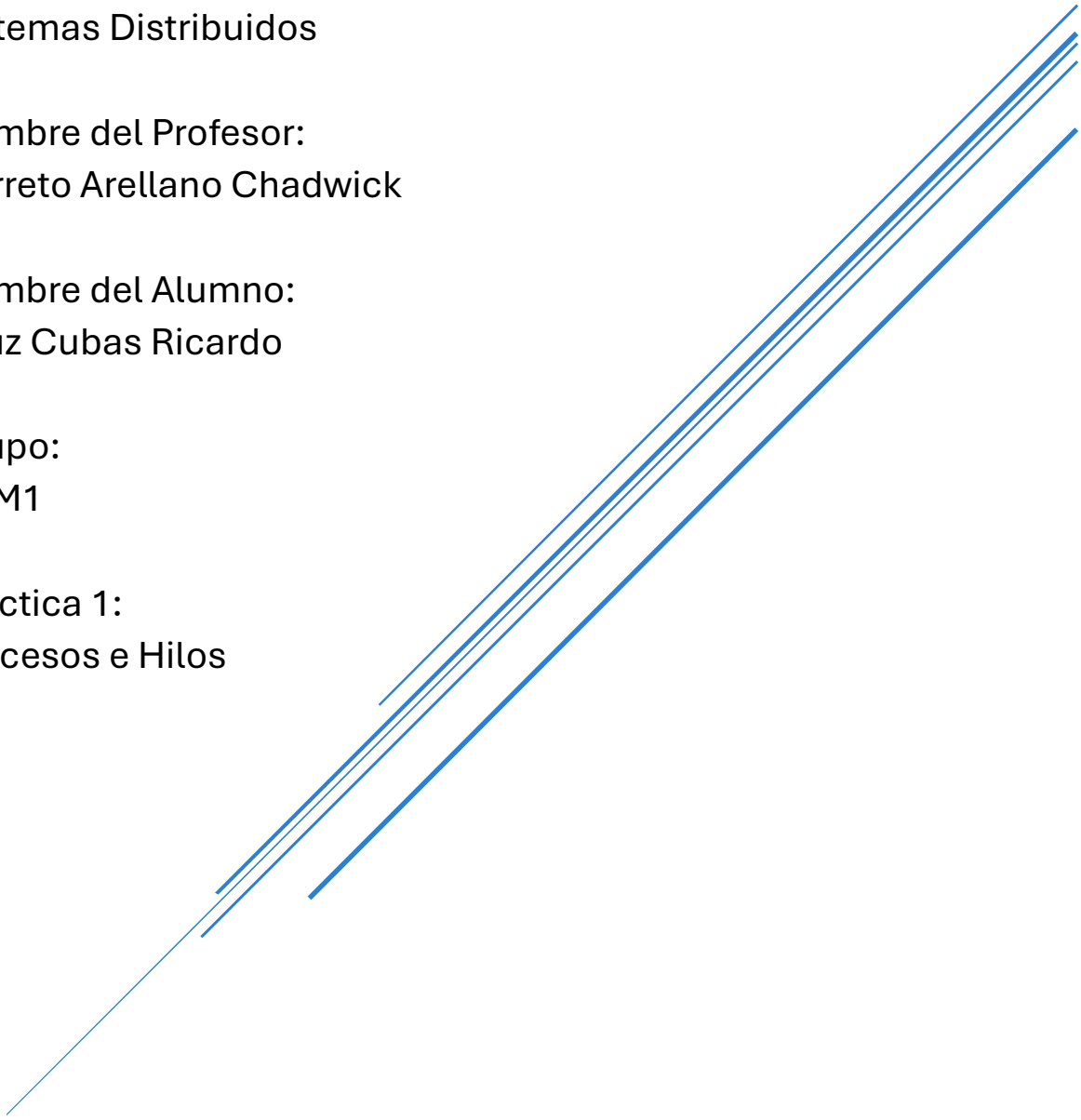
Unidad de Aprendizaje:
Sistemas Distribuidos

Nombre del Profesor:
Carreto Arellano Chadwick

Nombre del Alumno:
Cruz Cubas Ricardo

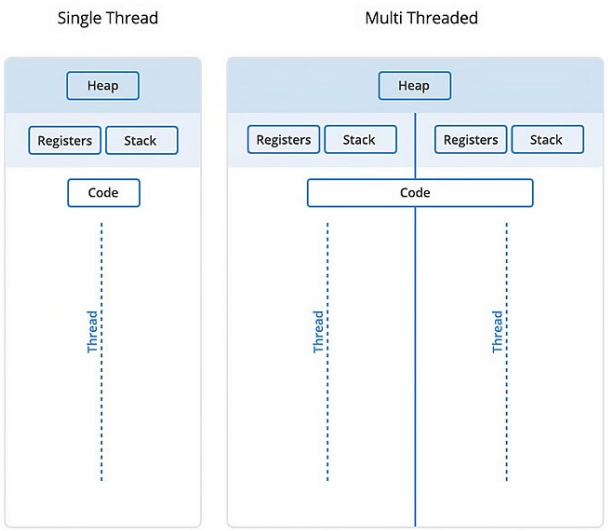
Grupo:
7CM1

Practica 1:
Procesos e Hilos



Antecedentes

Los procesos son la unidad elemental que tiene el sistema operativo para la programación y asignación de recursos. Un proceso en ejecución, indica la existencia de un programa que se encuentra interactuando con el ordenador. Al levantar un proceso, el sistema operativo asigna memoria y recursos para su ejecución. Al terminar un proceso, el sistema operativo libera esos recursos.



Los procesos son la unidad elemental que tiene el sistema operativo para la programación y asignación de recursos. Un proceso en ejecución, indica la existencia de un programa que se encuentra interactuando con el ordenador. Al levantar un proceso, el sistema operativo asigna memoria y recursos para su ejecución. Al terminar un proceso, el sistema operativo libera esos recursos.

El concepto de procesos e hilos surge con el desarrollo de los sistemas operativos modernos, a medida que las computadoras evolucionaban desde sistemas secuenciales hasta sistemas multitarea y multiprocesador.

Inicialmente, las computadoras ejecutaban un solo programa a la vez en un modo secuencial. Sin embargo, con el crecimiento de la capacidad de cómputo y la necesidad de optimizar el uso de los recursos, se introdujeron mecanismos para manejar múltiples programas de manera más eficiente.

Década de 1960: Multiprogramación y Procesos

- Se implementó el concepto de multiprogramación, donde varios programas podían estar en memoria al mismo tiempo, permitiendo cambiar de uno a otro para optimizar el uso del procesador.
- Los sistemas operativos comenzaron a administrar estos programas como procesos, que incluían un espacio de memoria separado, recursos asignados y un contexto de ejecución.

Década de 1970: Sistemas Multitarea

- Aparecieron los sistemas operativos multitarea, donde varios procesos podían ejecutarse aparentemente al mismo tiempo gracias a la conmutación de procesos (context switching).

- Se introdujeron estados del proceso (Ejecutando, Listo, Bloqueado) y mecanismos de planificación para mejorar la eficiencia.

Aunque los procesos permitieron la multitarea, cada uno tenía su propio espacio de memoria, lo que implicaba un alto costo de cambio de contexto. Para mejorar la eficiencia, se introdujo el concepto de hilos.

Aunque los procesos permitieron la multitarea, cada uno tenía su propio espacio de memoria, lo que implicaba un alto costo de cambio de contexto. Para mejorar la eficiencia, se introdujo el concepto de hilos.

Década de 1980: Aparición de los Hilos

- Los sistemas operativos empezaron a incluir hilos de ejecución, que permiten dividir un proceso en múltiples tareas concurrentes que comparten el mismo espacio de memoria.
- Ejemplo: En un procesador de texto, un hilo puede manejar la interfaz gráfica mientras otro guarda el documento en segundo plano.

Década de 1990: Multiprocesamiento y Hilos a Nivel de Núcleo

- Con el aumento de los procesadores multinúcleo, los sistemas operativos introdujeron hilos a nivel de núcleo, que permiten a los sistemas aprovechar múltiples núcleos para ejecutar tareas en paralelo.
- Se mejoraron los modelos de programación multihilo, como POSIX Threads (Pthreads) en UNIX y Java Threads en la Máquina Virtual de Java (JVM).

Hoy en día, los sistemas operativos modernos utilizan procesos e hilos de manera eficiente para maximizar el rendimiento. Algunos avances recientes incluyen:

- Computación paralela: Uso de múltiples hilos en procesadores multinúcleo para tareas intensivas.
- Hilos en la nube: Infraestructuras que manejan procesos concurrentes en servidores distribuidos.
- Lenguajes concurrentes: Como Go y Rust, que integran soporte nativo para la concurrencia.

Planteamiento del Problema

En esta práctica, se busca desarrollar una aplicación que implemente el uso de hilos de ejecución con el objetivo de comprender su funcionamiento, administración y beneficios dentro de un sistema.

Los hilos permiten la ejecución concurrente de múltiples tareas dentro de un mismo proceso, lo que mejora el rendimiento y la eficiencia en sistemas modernos. Sin embargo, su correcta implementación requiere el manejo adecuado de sincronización, compartición de recursos y planificación, aspectos fundamentales en la programación concurrente.

A través de esta práctica, se explorará cómo crear, gestionar y sincronizar hilos, además de analizar los desafíos comunes asociados, como las condiciones de carrera y los bloqueos. Esto permitirá comprender mejor la importancia de los hilos en el desarrollo de aplicaciones eficientes y responsivas.

Propuesta de Solución

Se propone desarrollar una aplicación que implemente hilos de ejecución utilizando un lenguaje de programación adecuado, como Java el cual cuentan con bibliotecas y mecanismos específicos para la gestión de hilos.

La solución se estructurará en las siguientes fases:

Investigación y Diseño:

- Analizar el concepto de hilos y su administración en diferentes lenguajes de programación.
- Definir la arquitectura de la aplicación, estableciendo el número de hilos y su interacción.

Implementación de Hilos:

- Crear un programa que utilice múltiples hilos dentro de un mismo proceso.
- Implementar tareas concurrentes para demostrar la ejecución en paralelo.
- Utilizar estructuras de control adecuadas para la gestión de hilos (clases Thread en Java).

Sincronización y Comunicación:

- Implementar mecanismos para coordinar la ejecución de los hilos y evitar problemas de concurrencia.
- Probar diferentes técnicas de sincronización, como bloqueos, semáforos y exclusión mutua.

Pruebas y Evaluación de Desempeño:

- Ejecutar la aplicación en distintos escenarios para analizar su comportamiento.
- Medir el impacto del uso de hilos en el rendimiento del programa.
- Identificar posibles mejoras en la gestión de los hilos y la sincronización.

Materiales y Métodos Empleados

Software:

- Java Development Kit (JDK) 8 o superior – Para compilar y ejecutar programas en Java.
- Entorno de desarrollo integrado (IDE) de Eclipse
- Bibliotecas estándar de Java – java.lang.Thread y java.util.concurrent para la gestión de hilos y sincronización.

Desarrollo de la Solución

En esta práctica, se desarrolló una aplicación en Java para implementar el uso de hilos de ejecución, utilizando el concepto de concurrencia. Se creó una simulación de una carrera entre dos corredores, cada uno representado por un hilo independiente. El objetivo de la práctica es comprender cómo se crean, inician y ejecutan hilos en un entorno multihilo, además de explorar la aleatoriedad en la ejecución de procesos concurrentes.

El código se divide en dos partes principales:

- La clase Corredor, que extiende Thread y define el comportamiento de cada corredor.
- La clase practica1, que crea e inicia los hilos para ejecutar la carrera.

```
import java.util.Random;

class Corredor extends Thread {
    private String nombre;

    public Corredor(String nombre) {
        this.nombre = nombre;
    }
```

Ahora bien, ¿cómo hacemos que el corredor realmente avance? Aquí entra en juego el método run(). Este método es lo que ejecutará cada hilo cuando lo iniciemos. Lo que hacemos dentro de run() es establecer una distancia de diez posiciones y luego usamos un ciclo for para que el corredor avance de una en una. En cada paso, imprimimos su posición actual y luego hacemos que se detenga un momento con Thread.sleep(). Pero no queremos que todos los corredores avancen exactamente al mismo ritmo, así que usamos un número aleatorio entre 0 y 500 milisegundos para que cada uno haga pausas distintas. Esto hace que la carrera sea más

realista, porque a veces un corredor avanzará más rápido que el otro, pero en la siguiente iteración quizá el otro se adelante.

Después de recorrer las diez posiciones, el corredor imprime un mensaje indicando que ha terminado. Así podemos ver cómo cada hilo avanza a su propio ritmo hasta que ambos terminan la carrera.

```
public void run() {
    Random random = new Random();
    int distancia = 10;

    for (int i = 1; i <= distancia; i++) {
        System.out.println(nombre + " avanzó a la posición " + i);
        try {
            Thread.sleep(random.nextInt(500));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    System.out.println(nombre + " ha terminado la carrera.");
}
```

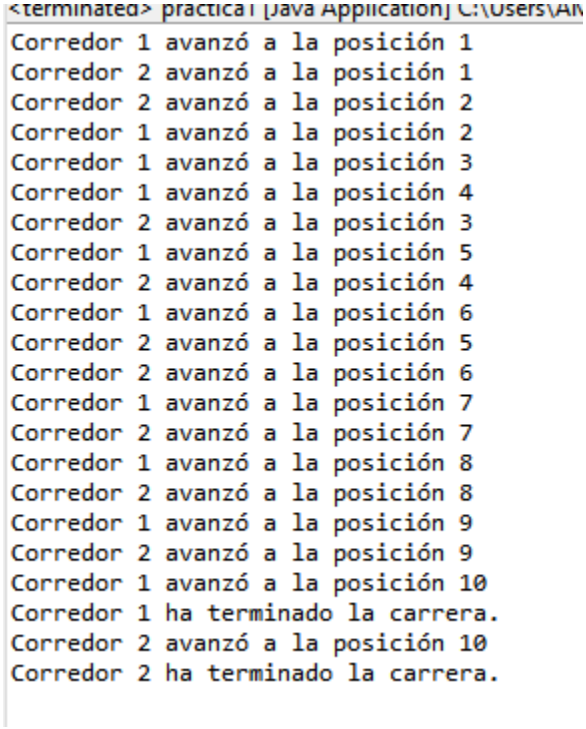
Ahora pasemos a la parte principal del programa, la clase practica1. Aquí es donde realmente ponemos en marcha la carrera. Dentro del main(), creamos dos objetos de la clase Corredor, uno llamado "Corredor 1" y el otro "Corredor 2". Una vez creados, simplemente llamamos a start() en cada uno, lo que hace que comiencen a ejecutarse de manera simultánea. En este punto, cada hilo sigue su propio camino y se ejecuta en paralelo con el otro, sin necesidad de esperar a que el otro termine.

```
public class practica1 {
    public static void main(String[] args) {
        Corredor corredor1 = new Corredor("Corredor 1");
        Corredor corredor2 = new Corredor("Corredor 2");

        corredor1.start();
        corredor2.start();
    }
}
```

Resultados

Si ejecutamos el programa, veremos en la consola cómo los corredores avanzan de manera intercalada. Tal vez el Corredor 1 empiece primero, pero luego el Corredor 2 lo alcance y lo rebase. O en otra ejecución podría ser al revés. Como los tiempos de espera son aleatorios, nunca sabemos quién terminará primero hasta que la carrera realmente se ejecute



Conclusión

Este código es que nos muestra cómo funciona la ejecución de hilos en Java. A diferencia de un programa normal donde las instrucciones se ejecutan en orden, aquí tenemos varias tareas corriendo al mismo tiempo, lo que hace que la salida varíe en cada ejecución. También vemos cómo el sistema operativo maneja los hilos y les da tiempo de CPU según sus propias reglas, lo que hace que el orden de los mensajes en la consola no sea predecible.