

# **Práctica #10 – Programación concurrente.**

Laboratorio de Programación Orientada a Objetos.

**Alumno:** Jeremy Uriel Rossell Segura

**Matrícula:** 2173396

**Materia:** Laboratorio de Programación Orientada a Objetos.

**Docente:** Jorge Alberto Islas Pineda

# Compilación

El programa compiló adecuadamente.

The screenshot shows the IntelliJ IDEA interface with the project '2173396\_practica10' open. The code editor displays Main.java with the following content:

```
package empresa;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
public class Main {
    public static void main(String[] args) {
        crearArchivosDeEjemplo();
        // crear hilos para cada sucursal
        LectorArchivo sucursal1 = new LectorArchivo("sucursal1.txt");
    }
}
```

The 'Run' tab is selected, showing the output of the run:

```
/usr/java/jdk-25-oracle-x64/bin/java -javaagent:/home/jeremy/.local/share/JetBrains/Toolbox/apps/intellij-idea-ultimate/lib/Performance
[✓] Archivo creado: sucursal1.txt
[✓] Archivo creado: sucursal2.txt
[✓] Archivo creado: sucursal3.txt
Hilo-1 leyendo: Ana
Hilo-2 leyendo: Carlos
Hilo-3 leyendo: Laura
Hilo-1 leyendo: Luis
Hilo-2 leyendo: Marta
Hilo-3 leyendo: Jorge
Hilo-1 leyendo: Sofia
Hilo-2 leyendo: Pedro
Hilo-3 leyendo: Diego
Process finished with exit code 0
```

The CPU and Heap Memory sections are visible on the right.

# Depuración

El programa se depuró adecuadamente.

The screenshot shows the IntelliJ IDEA interface with the project '2173396\_practica10' open. The code editor displays Main.java with the same content as before. The 'Debug' tab is selected, showing the debug session output:

```
/usr/java/jdk-25-oracle-x64/bin/java -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:58861,suspend=y,server=n -javaagent:/home/jeremy/.local/share/JetBrains/Toolbox/apps/intellij-idea-ultimate/lib/Performance
Connected to the target VM, address: '127.0.0.1:58861', transport: 'socket'
Hilo-3 leyendo: Laura
Hilo-1 leyendo: Ana
Hilo-2 leyendo: Carlos
Hilo-3 leyendo: Jorge
Hilo-1 leyendo: Luis
Hilo-2 leyendo: Marta
Hilo-3 leyendo: Diego
Hilo-1 leyendo: Sofia
Hilo-2 leyendo: Pedro
Disconnected from the target VM, address: '127.0.0.1:58861', transport: 'socket'
Process finished with exit code 0
```

A warning message at the bottom right states: 'Cannot record performance: Cannot start the profiler: kernel variables are not configured'. There are also buttons for 'Disable Recording Time with Run to Cursor' and 'Configure'.

# Código

## Main.java y LectorArchivo.java

The screenshot shows the IntelliJ IDEA interface with two files open: Main.java and LectorArchivo.java.

**Main.java:**

```
public class Main {
    public static void main(String[] args) {
        LectorArchivo sucursal1 = new LectorArchivo("sucursal1");
        LectorArchivo sucursal2 = new LectorArchivo("sucursal2");
        LectorArchivo sucursal3 = new LectorArchivo("sucursal3");

        // asignar nombres
        sucursal1.setName("Hilo-1");
        sucursal2.setName("Hilo-2");
        sucursal3.setName("Hilo-3");

        // ejecutar hilos simultáneamente
        sucursal1.start();
        sucursal2.start();
        sucursal3.start();
    }

    // Método auxiliar: crea archivos con nombres de empleados
    private static void crearArchivosEjemplo() {
        String[][] datos = {
            {"Ana", "Luis", "Sofía"}, {"Carlos", "Marta", "Pedro"}, {"Laura", "Jorge", "Diego"}};

        for (int i = 0; i < 3; i++) {
            File archivo = new File(pathname: "sucursal" + (i + 1) + ".txt");
            if (!archivo.exists()) {
                try (BufferedWriter bw = new BufferedWriter(new FileWriter(archivo))) {
                    for (String nombre : datos[i]) {
                        bw.write(nombre);
                        bw.newLine();
                    }
                }
                System.out.println("[✓] Archivo creado: " + archivo.getName());
            } catch (IOException e) {
                System.out.println("[-] Error al crear archivo: " + e.getMessage());
            }
        }
    }
}
```

**LectorArchivo.java:**

```
package empresa;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class LectorArchivo extends Thread {
    private String nombreArchivo;

    public LectorArchivo(String nombreArchivo) {
        this.nombreArchivo = nombreArchivo;
    }

    @Override
    public void run() {
        try (BufferedReader br = new BufferedReader(new FileReader(nombreArchivo))) {
            String linea;
            while ((linea = br.readLine()) != null) {
                System.out.println(getName() + " leyendo: " + linea);
                Thread.sleep(500);
            }
        } catch (IOException e) {
            System.out.println(getName() + " [-] Error al leer: " + e.getMessage());
        } catch (InterruptedException e) {
            System.out.println(getName() + " interrumpido.");
        }
    }
}
```

# Archivos

## Texto

The screenshot shows the IntelliJ IDEA file browser displaying the project structure of 2173396\_practica10.

- Project
- 2173396\_practica10
- .idea
- .gitignore
- misc.xml
- modules.xml
- workspace.xml
- out
- production
- 2173396\_practica10
- empresa
- LectorArchivo
- Main
- src
- empresa
- LectorArchivo
- Main
- .gitignore
- 2173396\_practica10.iml
- sucursal1.txt
- sucursal2.txt
- sucursal3.txt

## Preguntas

### 1. ¿Qué sucede si se ejecutan los tres hilos sin el Thread.sleep(500)?

Todos los hilos se podrían decir que “competirán” por tiempo de CPU y leerán los archivos demasiado rápido, por lo que la salida aparecerá casi al mismo tiempo, por lo que puede verse desordenada o agrupar varias líneas sin pausas

### 2. ¿Por qué el orden de lectura varía en cada ejecución?

Porque los hilos son administrados por el planificador del sistema operativo, que decide en qué orden se ejecutan los fragmentos de código, a esto en ciencias computacionales se le conoce como time slicing, que hace que el sistema pueda dar prioridad a distintos hilos en cada ejecución

### 3. ¿Qué pasaría si todos los hilos intentaran escribir en el mismo archivo?

Se producirían race conditions, el contenido podría mezclarse o peor, sobrescribirse, ya que varios hilos escribirían simultáneamente en la misma posición del archivo

### 4. ¿Cómo podrías sincronizar el acceso al archivo compartido?

Usando la keyword “synchronized”, o utilizando clases seguras para hilos como ReentrantLock, o una BlockingQueue

### 5. ¿Qué clase se usa para crear o verificar un archivo en el sistema?

La clase java.io.File, la cual nos deja verificar existencia con exists(), crear archivo con createNewFile(), y eliminar con delete()

### 6. ¿Qué diferencia hay entre start() y run() en un hilo?

start() crea un nuevo hilo de ejecución independiente y luego llama internamente a run(), pero si llamas directamente a run(), el código se ejecuta en el mismo hilo actual, sin concurrencia real, esa es su diferencia

### 7. ¿Qué pasa si no cierras correctamente un BufferedReader o BufferedWriter?

El archivo puede quedar totalmente bloqueado, los datos no se guardan correctamente, o se pierden recursos del sistema mediante fugas de memoria, por eso usamos try-with-resources

### 8. ¿Cómo podrías modificar el programa para que cada hilo escriba en un nuevo archivo en lugar de leer?

Cambiando BufferedReader por un BufferedWriter dentro del run() y escribir:

```
```java
try (BufferedWriter bw = new BufferedWriter(new FileWriter(nombreDelArchivo))) {
    bw.write("texto generado por " + getName());
}
```

```

### 9. ¿Qué ventajas tiene usar varios hilos en lugar de leer los tres archivos uno por uno?

Mayor eficiencia ya que pueden leerse simultáneamente si hay varios núcleos, y aprovechas el poder del CPU junto a su buffer y clock speed, también no bloquea la ejecución, mientras un hilo espera la entrada/salida otro puede avanzar; y sobre todo escalabilidad, ya que es más fácil procesar grandes volúmenes de archivos y datos

### 10. ¿Podrías agregar un hilo adicional que, al final, cuente cuántas líneas totales se leyeron entre los tres archivos?

Sí, podemos crear un hilo extra al que podemos llamar ContadorLineas que lea los tres archivos y sume el total, no es difícil

```
```java
class ContadorLineas extends Thread {
    @Override
    public void run() {
        int total = 0;
```

```

```
for (int i = 1; i <= 3; i++) {  
    try (BufferedReader br = new BufferedReader(new FileReader("sucursal" + i + ".txt"))) {  
        while (br.readLine() != null) total++;  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}  
System.out.println("Total de líneas leídas: " + total);  
}  
}  
```
```

y luego en el main():

```
```java  
ContadorLineas contador = new ContadorLineas();  
contador.start();  
```
```