



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

# Ingeniería en computación

## Laboratorios de computación salas A y B

*Profesor:* \_\_\_\_\_ Ing. Rene Adrian Davila Perez

*Asignatura:* \_\_\_\_\_ Programación Orientada a Objetos

*Grupo:* \_\_\_\_\_ 1

*No. de práctica:* \_\_\_\_\_ Practica 12

*Integrante:* \_\_\_\_\_ 424121462

*Semestre:* \_\_\_\_\_ 2025-1

*Fecha de entrega:* \_\_\_\_\_ 20 de noviembre de 2024

*Observaciones:* \_\_\_\_\_

CALIFICACIÓN: \_\_\_\_\_

# Índice

1. Introducción	..... (2)
<hr/>	
2. Marco Teórico	..... (3)
<hr/>	
3. Desarrollo	..... (4-5)
<hr/>	
4. Resultados	..... (6-7)
<hr/>	
5. Conclusiones	..... (7-8)
<hr/>	
6. Referencias	..... (8)
<hr/>	

## Introducción:

A continuación, se describe brevemente el planteamiento de los ejercicios:

- **Ejercicio 1.** Escribir un programa en Java que cree dos hilos para encontrar e imprimir números pares e impares en un rango del 1 al 20.
- **Ejercicio 2.** Crear un programa en Java que ordene un arreglo de enteros mediante múltiples hilos. El programa dividirá la tarea de ordenamiento, utilizando hilos para realizar un procesamiento en paralelo y así optimizar el rendimiento.
- **Ejercicio Tarea.** Desarrollar un programa en Java para simular una cuenta bancaria con operaciones de depósito y retiro utilizando hilos. El programa debe considerar posibles conflictos, como el caso de fondos insuficientes.

Esta práctica tiene como finalidad comprender el uso de hilos en Java, permitiendo el desarrollo de aplicaciones que puedan realizar múltiples tareas de manera simultánea. La programación concurrente es fundamental en escenarios donde se optimiza el rendimiento mediante la paralelización de tareas, reduciendo así los tiempos de ejecución en procesos de alta demanda.

Estos ejercicios nos permiten comprender de mejor manera la implementación de procesos paralelos y el manejo seguro de recursos, fundamentales para aplicaciones robustas y eficientes.

Adicionalmente se tienen los siguientes objetivos:

- Aplicar los conceptos de hilos en Java para lograr concurrencia en tareas específicas.
- Implementar un sistema de manejo seguro de recursos compartidos, donde los hilos puedan ejecutar operaciones en paralelo de forma eficiente.
- Introducir el uso de Lock y ReentrantLock para gestionar accesos concurrentes y prevenir conflictos.
- Estructurar los ejercicios según los principios de encapsulación y organización mediante paquetes.
- Generar archivos jar y javadoc para una correcta distribución y documentación del proyecto.
- Crear diagramas UML que representen el diseño estático y la ejecución dinámica del sistema.

Esta práctica nos ayudará a entender de mejor manera la programación concurrente, preparándonos para manejar la complejidad y la eficiencia en aplicaciones Java mediante el uso correcto de hilos y recursos compartidos.

## Marco Teórico:

Los hilos en Java son mecanismos que permiten la ejecución de múltiples tareas dentro de un mismo programa, optimizando el uso de recursos del sistema. Su función principal es habilitar la paralelización de procesos, permitiendo que el programa realice varias operaciones al mismo tiempo. Esto resulta fundamental para desarrollar aplicaciones más rápidas y eficientes.

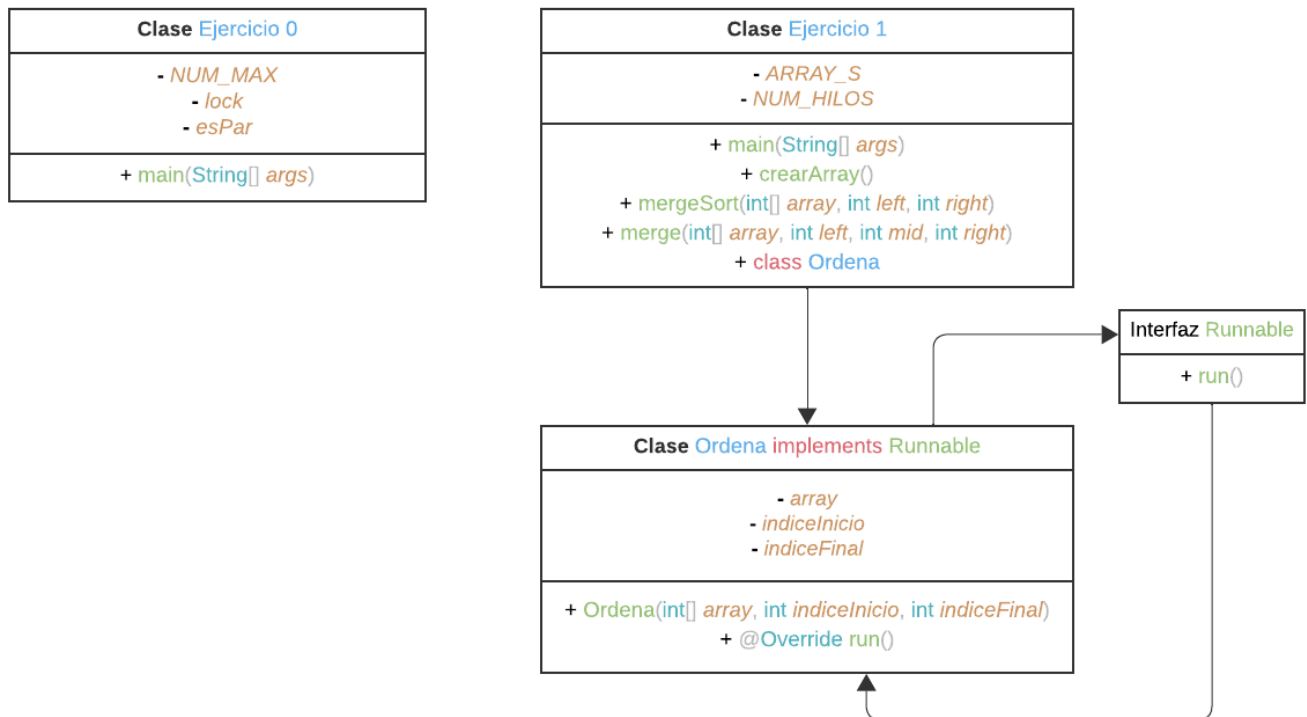
Un hilo es una unidad de procesamiento dentro de un proceso más amplio, y todos los hilos en Java son instancias de la clase `Thread` o de clases que implementan la interfaz `Runnable`. Cada hilo tiene su propio ciclo de vida, que incluye etapas como `New`, `Runnable`, `Waiting`, `Timed Waiting` y `Terminated`. Inicialmente, un hilo comienza en el estado `New`, y al ser iniciado, transita al estado `Runnable`, donde puede comenzar a ejecutar su tarea. Un hilo puede pasar temporalmente al estado `Waiting` si necesita que otro hilo complete una tarea antes de continuar, o al estado `Timed Waiting` cuando se espera por un período específico. Finalmente, el hilo llega al estado `Terminated` una vez que ha completado su tarea o ha sido finalizado.

El manejo de hilos en Java introduce la complejidad de la concurrencia, ya que varios hilos pueden intentar acceder y modificar el mismo recurso, creando potencialmente problemas como condiciones de carrera y bloqueos. Java proporciona el paquete **`java.util.concurrent`**, que incluye estructuras como `Lock` y `ReentrantLock`, ofreciendo un mayor control sobre el acceso a los recursos y permitiendo la sincronización de tareas para evitar conflictos de concurrencia.

Para crear hilos en Java, existen dos métodos principales: extender la clase `Thread` o implementar la interfaz `Runnable`. Ambos enfoques permiten definir el comportamiento del hilo a través del método `run`. La implementación de la interfaz `Runnable` es preferida en aplicaciones complejas, ya que permite a las clases heredar de otras y adherirse mejor a los principios de encapsulación y reutilización de código. Además, cada hilo tiene asignada una prioridad que influye en el orden en el que los hilos reciben tiempo de procesador. Estas prioridades se establecen en un rango de `MIN_PRIORITY` (1) a `MAX_PRIORITY` (10), siendo la predeterminada `NORM_PRIORITY` (5). Sin embargo, la prioridad de los hilos no garantiza un orden específico de ejecución, pues depende del sistema operativo. [1, 2]

En conclusión, el manejo de hilos en Java es esencial para construir aplicaciones concurrentes, robustas y de alto rendimiento. Estas prácticas son clave para optimizar el rendimiento en aplicaciones que requieren una alta velocidad de procesamiento y una respuesta rápida a múltiples tareas simultáneas.

## Desarrollo:

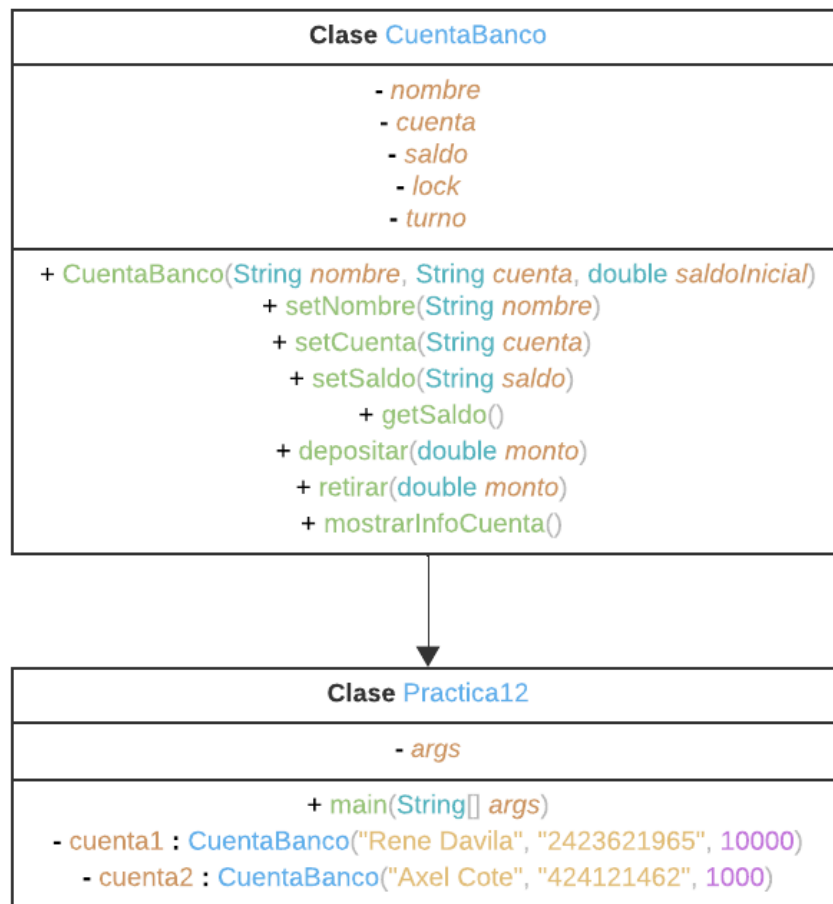


Para el Ejercicio 0 se realizaron las asignaciones de dos hilos, uno con el cual se detectara los números impares mediante un ciclo for dado por el `NUM_MAX` y otro hilo que detectara los pares con un ciclo similar, una vez inicializados los hilos, se mandaran a iniciar y a que terminen su ejecución de buena manera.

Para realizar las pruebas solo se le asignó el valor de 20 a `NUM_MAX` para así detectar a los pares e impares dentro del rango de 1 a 20, después de la asignación solo se ejecutó el programa.

Para el Ejercicio 1, se generará un arreglo de tamaño 400 con valores aleatorios mediante el método `crearArray`, además se define un número de hilos = 4 para dividir el trabajo en partes iguales. Los hilos se inician con el método `start`. Una vez que los hilos terminan de usarse para arreglarlo, las secciones del arreglo ordenadas se combinan en un solo arreglo ordenado gracias al algoritmo `mergeSort`.

Para probar el programa, se ejecuta con un arreglo de tamaño 400 lleno de valores aleatorios entre 0 y 399. Esperando que su ejecución se realice de manera correcta y mas rapida que lo común.

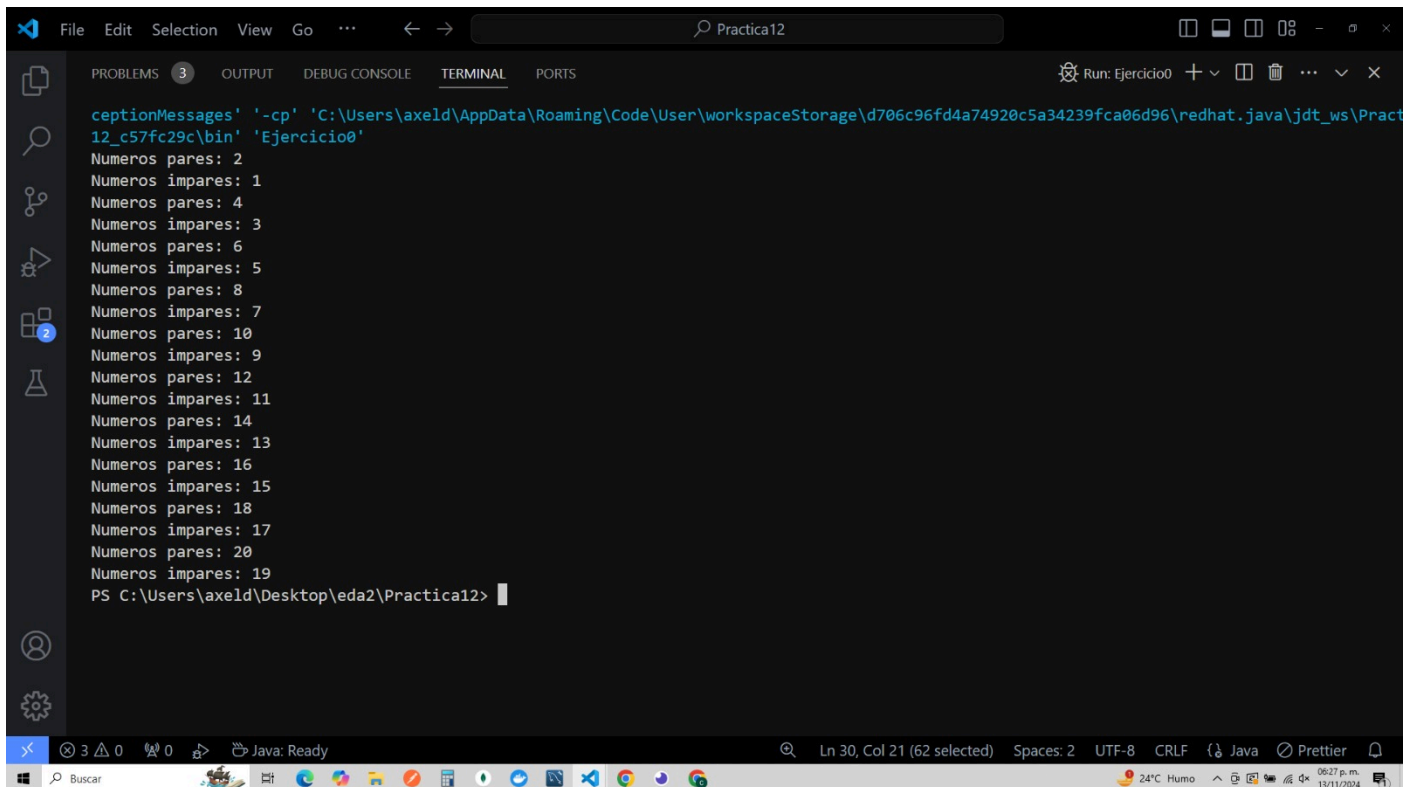


Para la clase CuentaBanco se creará un programa que gestione las operaciones de depósito y retiro de una cuenta bancaria mediante el uso de hilos. Para asegurarnos de que haya un acceso seguro de los recursos compartidos, se utiliza un objeto de bloqueo. Además, se controla el turno entre operaciones de depósito y retiro mediante el turno, asegurando que las operaciones sean alternadas y así se eviten conflictos de concurrencia.

El método depositar incrementa el saldo de la cuenta y notifica al siguiente hilo para continuar, mientras que el método retirar verifica que haya fondos suficientes antes de realizar la transacción.

Para la clase Practica12, que interactúa con la clase CuentaBanco para realizar operaciones de depósito y retiro utilizando hilos. Se crearán dos cuentas bancarias con saldos diferentes, y mediante dos hilos se realizan depósitos de 1000 y retiros de 500 en ambas cuentas. Al iniciar el programa, se muestra la información inicial de las cuentas, tras realizar las operaciones, el programa actualiza los saldos de las cuentas según los depósitos y retiros. Al finalizar las operaciones se vuelve a mostrar en pantalla la información de la cuenta.

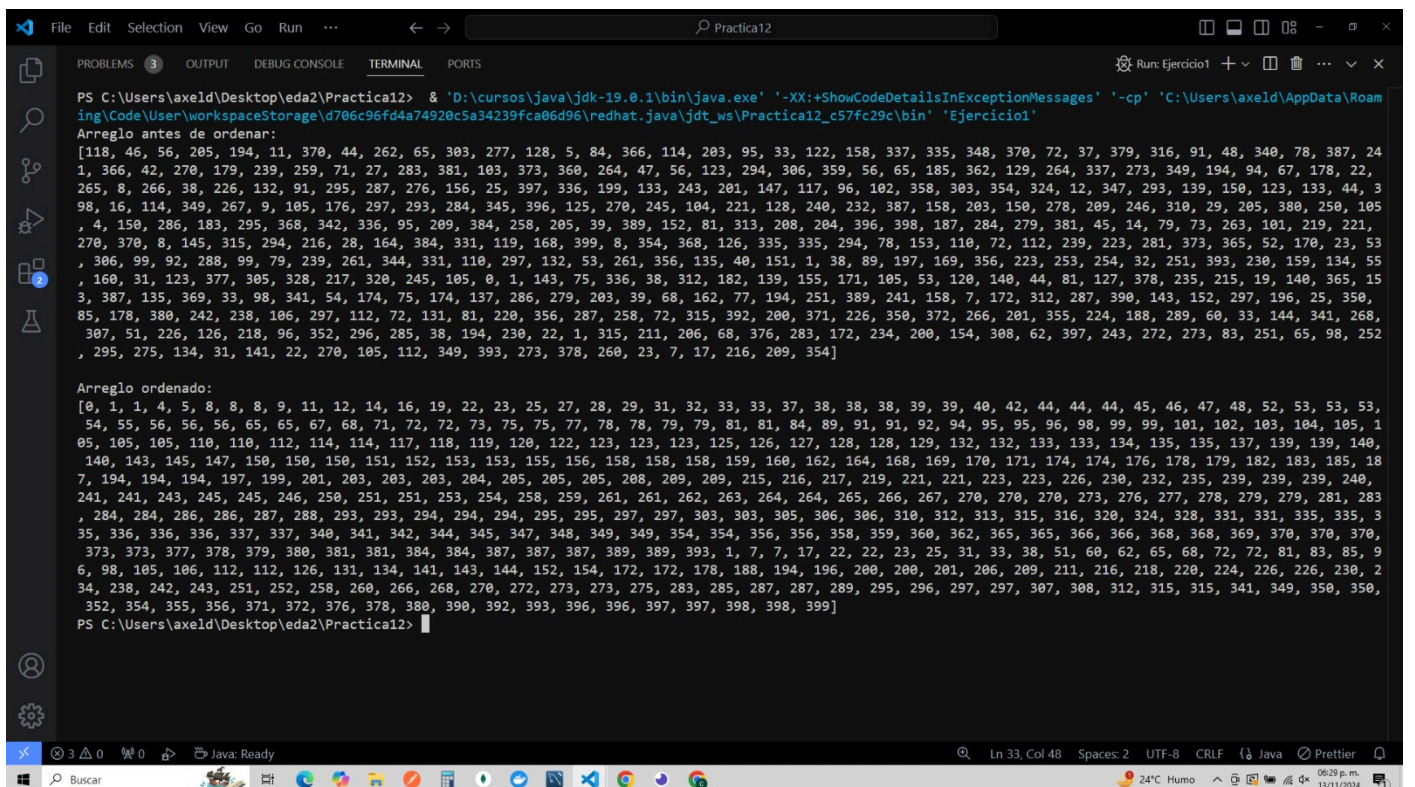
## Resultados:



```
ceptionMessages' '-cp' 'C:\Users\axeld\AppData\Roaming\Code\User\workspaceStorage\d706c96fd4a74920c5a34239fca06d96\redhat.java\jdt_ws\Practica12_c57fc29c\bin' 'Ejercicio0'

Numeros pares: 2
Numeros impares: 1
Numeros pares: 4
Numeros impares: 3
Numeros pares: 6
Numeros impares: 5
Numeros pares: 8
Numeros impares: 7
Numeros pares: 10
Numeros impares: 9
Numeros pares: 12
Numeros impares: 11
Numeros pares: 14
Numeros impares: 13
Numeros pares: 16
Numeros impares: 15
Numeros pares: 18
Numeros impares: 17
Numeros pares: 20
Numeros impares: 19
PS C:\Users\axeld\Desktop\eda2\Practica12>
```

Como se muestra en pantalla se observa la impresión de los números pares e impares gracias a los ciclos y los hilos usados.

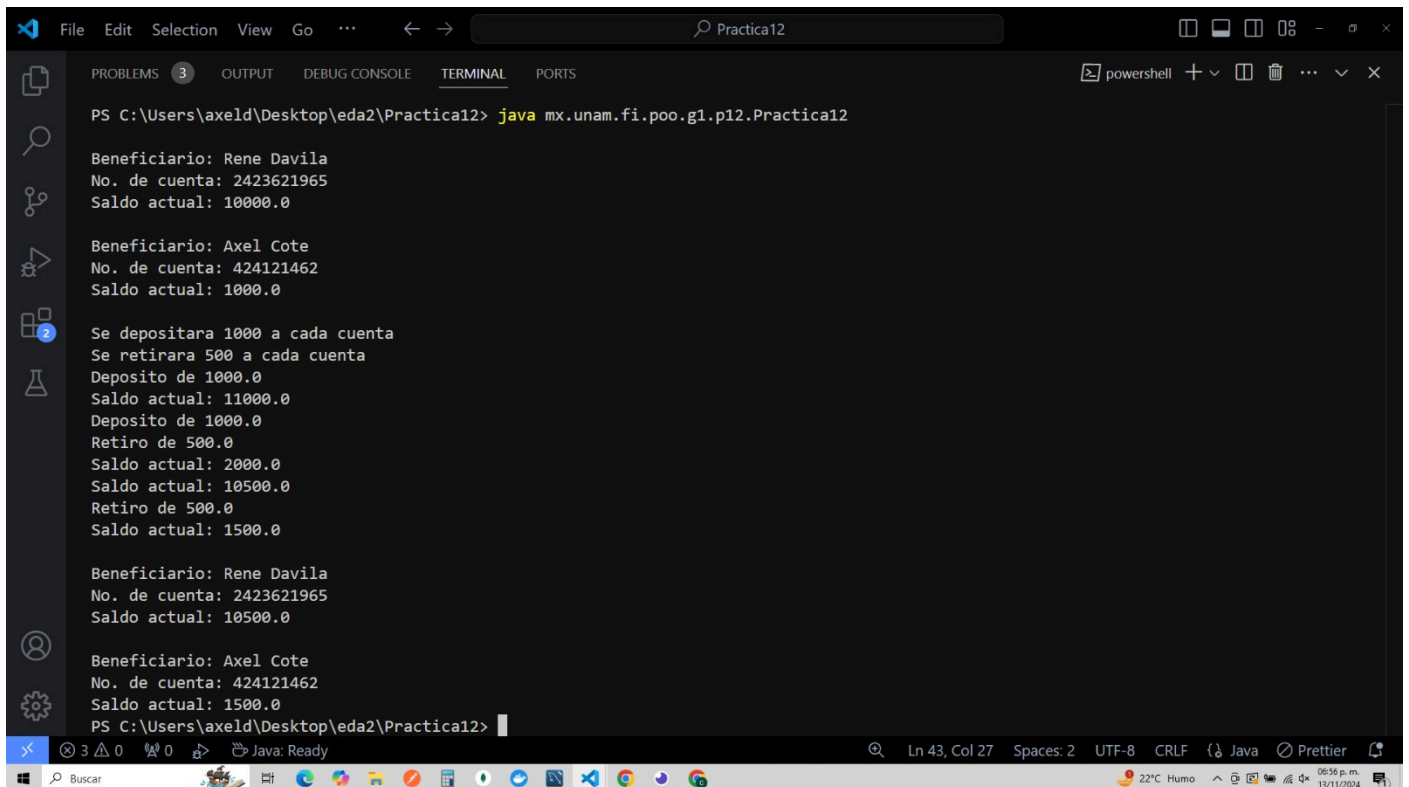


```
PS C:\Users\axeld\Desktop\eda2\Practica12> & 'D:\cursos\java\jdk-19.0.1\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\axeld\AppData\Roaming\Code\User\workspaceStorage\d706c96fd4a74920c5a34239fca06d96\redhat.java\jdt_ws\Practica12_c57fc29c\bin' 'Ejercicio1'

Arreglo antes de ordenar:
[118, 46, 56, 205, 194, 11, 370, 44, 262, 65, 303, 277, 128, 5, 84, 366, 114, 203, 95, 33, 122, 158, 337, 335, 348, 370, 72, 37, 379, 316, 91, 48, 340, 78, 387, 24
1, 366, 42, 270, 179, 239, 259, 71, 27, 283, 381, 103, 373, 360, 264, 47, 56, 123, 294, 306, 359, 56, 65, 185, 362, 129, 264, 337, 273, 349, 194, 94, 67, 178, 22,
265, 8, 266, 38, 226, 132, 91, 295, 287, 276, 156, 25, 397, 336, 199, 133, 243, 201, 147, 117, 96, 102, 358, 303, 354, 324, 12, 347, 293, 139, 150, 123, 133, 44, 3
98, 16, 114, 349, 267, 9, 105, 176, 297, 293, 284, 345, 396, 125, 270, 245, 104, 221, 128, 240, 232, 387, 158, 203, 150, 278, 209, 246, 310, 29, 205, 380, 250, 105
, 4, 150, 286, 183, 295, 368, 342, 336, 95, 209, 384, 258, 205, 39, 389, 152, 81, 313, 208, 204, 396, 398, 187, 284, 279, 381, 45, 14, 79, 73, 263, 101, 219, 221,
270, 370, 8, 145, 315, 294, 216, 28, 164, 384, 331, 119, 168, 399, 8, 354, 368, 126, 335, 335, 294, 78, 153, 110, 72, 112, 239, 223, 281, 373, 365, 52, 170, 23, 53
, 306, 99, 92, 288, 99, 79, 239, 261, 344, 331, 110, 297, 132, 53, 261, 356, 135, 40, 151, 1, 38, 89, 197, 169, 356, 223, 253, 254, 32, 251, 393, 230, 159, 134, 55
, 160, 31, 123, 377, 305, 328, 217, 320, 245, 105, 0, 1, 143, 75, 336, 38, 312, 182, 139, 155, 171, 105, 53, 120, 140, 44, 81, 127, 378, 235, 215, 19, 140, 365, 15
3, 387, 135, 369, 33, 98, 341, 54, 174, 75, 174, 137, 286, 279, 203, 39, 68, 162, 77, 194, 251, 389, 241, 158, 7, 172, 312, 287, 390, 143, 152, 297, 196, 25, 350,
85, 178, 380, 242, 238, 106, 297, 112, 72, 131, 81, 220, 356, 287, 258, 72, 315, 392, 200, 371, 226, 350, 372, 266, 201, 355, 224, 188, 289, 60, 33, 144, 341, 268,
307, 51, 226, 126, 218, 96, 352, 296, 285, 38, 194, 230, 22, 1, 315, 211, 206, 68, 376, 283, 172, 234, 200, 154, 308, 62, 397, 243, 272, 273, 83, 251, 65, 98, 252
, 295, 275, 134, 31, 141, 22, 270, 105, 112, 349, 393, 273, 378, 260, 23, 7, 17, 216, 209, 354]

Arreglo ordenado:
[0, 1, 1, 4, 5, 8, 8, 8, 9, 11, 12, 14, 16, 19, 22, 23, 25, 27, 28, 29, 31, 32, 33, 33, 37, 38, 38, 38, 39, 39, 40, 42, 44, 44, 44, 45, 46, 47, 48, 52, 53, 53, 53,
54, 55, 56, 56, 56, 65, 65, 67, 68, 71, 72, 72, 73, 75, 75, 77, 78, 78, 79, 79, 81, 81, 84, 89, 91, 91, 92, 94, 95, 95, 96, 98, 99, 99, 101, 102, 103, 104, 105, 1
05, 105, 105, 110, 110, 112, 114, 114, 117, 118, 119, 120, 122, 123, 123, 123, 125, 126, 127, 128, 128, 129, 132, 132, 133, 133, 134, 135, 135, 137, 139, 139, 140,
140, 143, 145, 147, 150, 150, 150, 151, 152, 153, 153, 155, 156, 158, 158, 159, 160, 162, 164, 168, 169, 170, 171, 174, 174, 176, 178, 179, 182, 183, 185, 18
7, 194, 194, 194, 197, 199, 201, 203, 203, 203, 204, 205, 205, 205, 208, 209, 209, 215, 216, 217, 219, 221, 221, 223, 223, 226, 230, 232, 235, 239, 239, 239, 240,
241, 241, 243, 245, 245, 246, 250, 251, 251, 253, 254, 258, 259, 261, 261, 262, 263, 264, 264, 265, 266, 267, 270, 270, 270, 273, 276, 277, 278, 279, 279, 281, 283
, 284, 284, 286, 286, 287, 288, 293, 293, 294, 294, 294, 295, 295, 297, 297, 303, 303, 305, 306, 306, 310, 312, 313, 315, 316, 320, 324, 328, 331, 331, 335, 335, 3
35, 336, 336, 336, 337, 337, 340, 341, 342, 344, 345, 347, 348, 349, 349, 354, 354, 356, 356, 358, 359, 360, 362, 365, 365, 366, 366, 368, 368, 369, 370, 370, 370,
373, 373, 377, 378, 379, 380, 381, 381, 384, 384, 387, 387, 389, 389, 393, 1, 7, 7, 17, 22, 23, 25, 31, 33, 38, 51, 60, 62, 65, 68, 72, 72, 81, 83, 85, 9
6, 98, 105, 106, 112, 112, 126, 131, 134, 141, 143, 144, 152, 154, 172, 172, 178, 188, 194, 196, 200, 200, 201, 206, 209, 211, 216, 218, 220, 224, 226, 226, 230, 2
34, 238, 242, 243, 251, 252, 258, 260, 266, 268, 270, 272, 273, 273, 275, 283, 285, 287, 287, 289, 289, 295, 296, 297, 297, 307, 308, 312, 315, 315, 341, 349, 350, 350,
352, 354, 355, 356, 371, 372, 376, 378, 380, 390, 392, 393, 396, 396, 397, 397, 398, 398, 399]
PS C:\Users\axeld\Desktop\eda2\Practica12>
```

En la imagen anterior se puede mostrar la utilización del algoritmo MergeSort y la utilización de hilos para así poder hacer más eficiente el programa.



```
PS C:\Users\axeld\Desktop\eda2\Practica12> java mx.unam.fi.poo.g1.p12.Practica12

Beneficiario: Rene Davila
No. de cuenta: 2423621965
Saldo actual: 10000.0

Beneficiario: Axel Cote
No. de cuenta: 424121462
Saldo actual: 1000.0

Se depositara 1000 a cada cuenta
Se retirara 500 a cada cuenta
Deposito de 1000.0
Saldo actual: 11000.0
Deposito de 1000.0
Retiro de 500.0
Saldo actual: 2000.0
Saldo actual: 10500.0
Retiro de 500.0
Saldo actual: 1500.0

Beneficiario: Rene Davila
No. de cuenta: 2423621965
Saldo actual: 10500.0

Beneficiario: Axel Cote
No. de cuenta: 424121462
Saldo actual: 1500.0
PS C:\Users\axeld\Desktop\eda2\Practica12>
```

En la imagen mostrada se puede observar el funcionamiento de la aplicación de Banco que utiliza hilos para su procesamiento.

## Conclusiones:

En conclusión, el manejo de hilos en Java es una herramienta esencial para el desarrollo de aplicaciones eficientes y concurrentes. La prioridad de los hilos también desempeña un papel en la administración del tiempo de procesador, optimizando la secuencia en la que se ejecutan y permitiendo que se asignen a tareas según su relevancia, es importante señalar que esta prioridad depende del sistema operativo.

El paquete `java.util.concurrent` y sus estructuras de sincronización, como `Lock` y `ReentrantLock`, son fundamentales para gestionar la concurrencia y evitar problemas de acceso a recursos compartidos. Este control explícito sobre los recursos compartidos es esencial en aplicaciones donde se requiere que los hilos accedan de forma controlada a datos comunes.

Para concluir, la aplicación de estos conceptos y herramientas de manejo de hilos permite optimizar las capacidades de procesamiento en sistemas concurrentes. Su implementación fomenta la creación de aplicaciones robustas y confiables que maximizan el rendimiento y se adaptan eficientemente a situaciones que requieren la ejecución de múltiples tareas. La teoría y la



práctica en torno al manejo de hilos en Java resultan claves para abordar problemas complejos de concurrencia en entornos reales.

## Referencias:

[1] "Java Multithreading," *TutorialsPoint*. [En línea]. Disponible:

[https://www.tutorialspoint.com/java/java\\_multithreading.htm](https://www.tutorialspoint.com/java/java_multithreading.htm). [Accedido: Nov. 10, 2024].

[2] ChatGPT. [En línea]. Disponible: <https://chatgpt.com/>. [Accedido: Nov. 10, 2024]. Pregunta:

“que son los hilos en java”