Threads & Parallel Computing in Java

Team Members

- Ahmed Moatz (Sec 1)
- Samaa Abdelmohsen (Sec 2)
- Saad Mohammed (Sec 2)
- Youssef sameh (Sec 6)
- Ali Elsayed (Sec 3)
- .Shaaban Mosaad (sec3)

Course : Parallel Programming | **Instructor:** En. Ahmed Maher

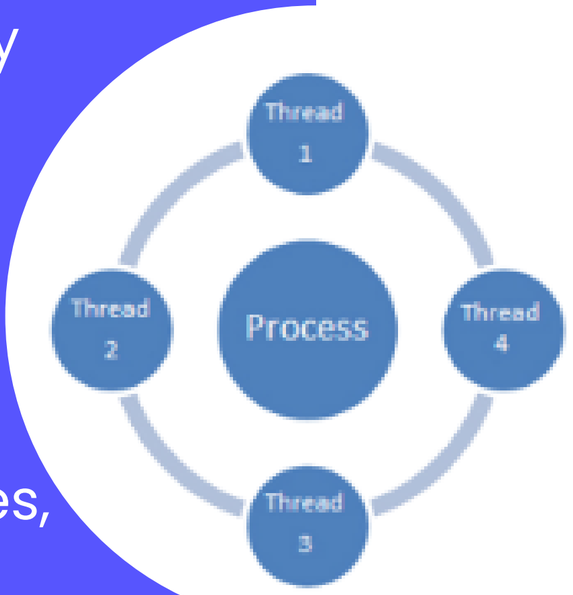
Introduction to Threads

2

A thread is the smallest unit of execution within a program.
It allows us to run tasks concurrently, sharing the same memory space.
In Java, we can create threads using:

- The Thread class
- The Runnable interface
- Executors (for more advanced handling)

Threads are essential for performance in applications like games, image processing, and server-side apps.



Why Parallel Computing?

5

Faster Image Processing

Using multithreading allows us to apply image filters in parallel, which reduces processing time dramatically.

Optimized Resource Usage

By distributing the workload across CPU cores, we ensure better use of system resources during image operations.

Scalable Design

The system supports both single-threaded and multi-threaded modes, making it adaptable to different performance needs.

Real-Time Feedback

Multithreading enables responsive UI updates and real-time progress output, enhancing the user experience.

Thread Lifecycle in Java

6

lifecycle of a thread

🟡 New

Thread is created but hasn't started yet.

→ **Thread t = new Thread();**

🟡 Ready (Runnable)

Thread is ready to run but waiting for CPU.

→ **t.start();**

🟡 Running

Thread is executing.

→ **It is scheduled by the JVM thread scheduler.**

🟡 Blocked / Waiting / Sleeping

Thread is paused for:

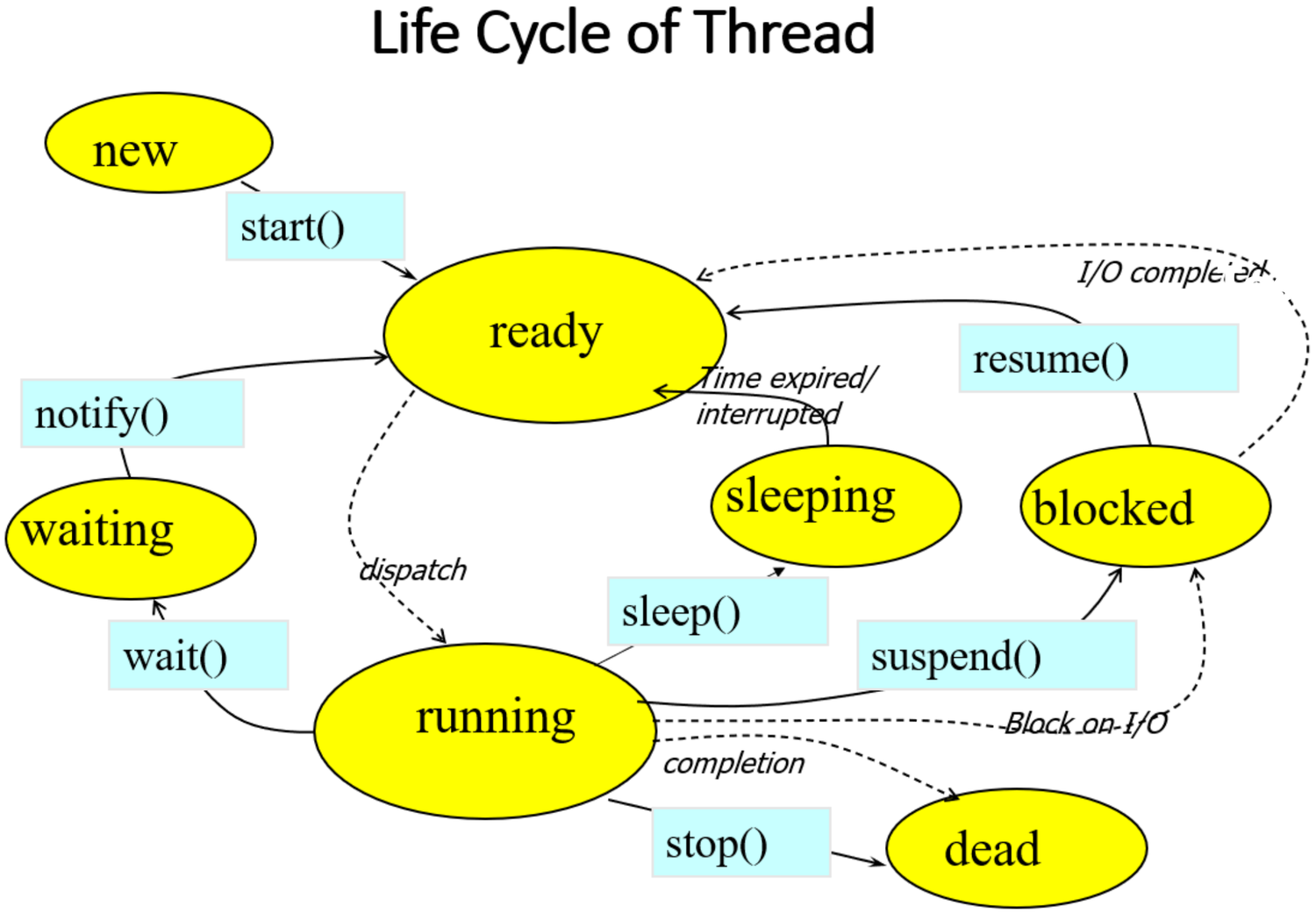
- **I/O**
- **Sleeping via Thread.sleep()**
- **Waiting for another thread via wait()**

🔴 Dead

Thread has completed or stopped.

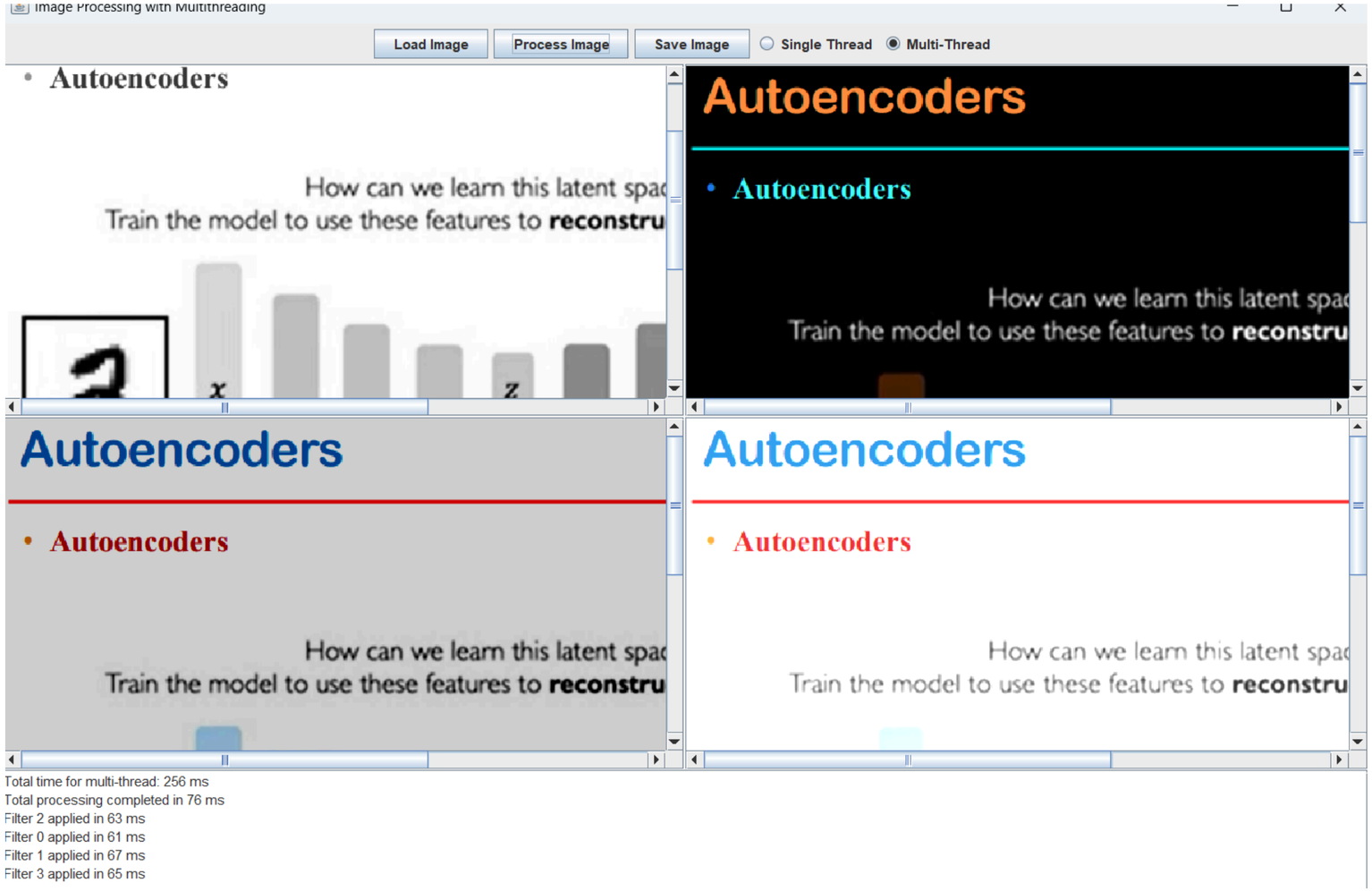
→ **stop( ) or it finishes its run() method.**

lifecycle of a thread



Project Overview – Image Processor GUI

7



🖼️ I created a Java Swing application that loads, processes, and saves images using:

Four filters:

- Grayscale, Invert, Brightness Down, Brightness Up
- GUI built with Swing
- Filters processed using Single-thread or Multi-thread modes

Applying Threads in the Project 10

```
for (int i = 0; i < 4; i++) {
    long startTime = System.currentTimeMillis();


    final int filterIndex = i;


    Thread thread = new Thread(() -> applyFilter(filterIndex));
    thread.start();
    try {
        thread.join();
    } catch (InterruptedException e) {
        JOptionPane.showMessageDialog(this, message:"Thread execution interrupted!", title:"Error", JOptionPane.ERROR_MESSAGE);
    }

    long endTime = System.currentTimeMillis();
    long filterTime = endTime - startTime;
    totalTime += filterTime;

    outputArea.append("Filter " + filterIndex + " applied in " + filterTime + " ms\n");
}
```

Single Thread Mode:

- One filter runs at a time
- Filters are applied sequentially
- Each filter waits for the previous one to finish

Applying Threads in the Project 10

```
rivate void processMultiThread() {
    Thread[] threads = new Thread[4];
    long[] filterTimes = new long[4];

    for (int i = 0; i < 4; i++) {
        int filterIndex = i;
        threads[i] = new Thread(() -> {
            long startTime = System.currentTimeMillis();
            applyFilter(filterIndex);
            long endTime = System.currentTimeMillis();
            filterTimes[filterIndex] = endTime - startTime;

            SwingUtilities.invokeLater(() -> {
                outputArea.append("Filter " + filterIndex + " applied in " + filterTimes[filterIndex] + " ms\n");
            });
        });
        threads[i].start();
    }

    for (Thread thread : threads) {
        try {
            thread.join(); // Wait for all threads to finish
        } catch (InterruptedException e) {
            JOptionPane.showMessageDialog(this, message:"Thread execution interrupted!", title:"Error", JOptionPane.ERROR_MESSAGE);
        }
    }
```

Multi-Thread Mode:

- All filters run in parallel.
- Each filter is processed in its own thread.
- Improves performance and reduces processing time.

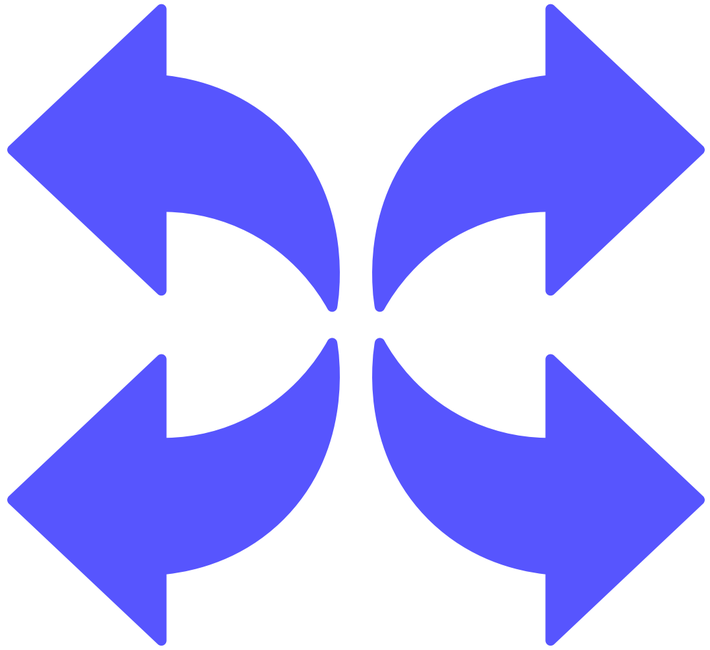
Conclusion

Enhanced Speed

- Threads significantly improve application speed and responsiveness. They allow simultaneous execution of filters, reducing total processing time.

Better User Experience

- Using multithreading ensures the GUI remains responsive, providing smoother interaction and faster feedback for the user.



Essential Lifecycle Knowledge

- Understanding the Thread Lifecycle is key to controlling thread creation, execution, and termination effectively.

Critical for Performance

- Parallel processing is crucial in modern applications, especially those handling image, video, or data processing in real time.

THANK YOU

Java™
Threading