Threads & Multithreading in Java

What is a Thread?

A Thread is the smallest unit of execution in a Java program.
It represents a separate path of execution. When a Java program runs, the JVM starts with one main thread by default (the one that executes the main() method).

What is Multithreading?

Multithreading is the ability of Java to run multiple threads concurrently.
It allows programs to do multiple tasks simultaneously, improving performance and responsiveness.

✅ Benefits:

- Faster execution
- Better CPU utilization
- Improved user experience
- Useful in real-time systems, GUI apps, servers, etc.

Ways to Create a Thread in Java

1. Extending the Thread class

```
class MyThread extends Thread {
   public void run() {
      System.out.println("Thread running...");
   }
}
```

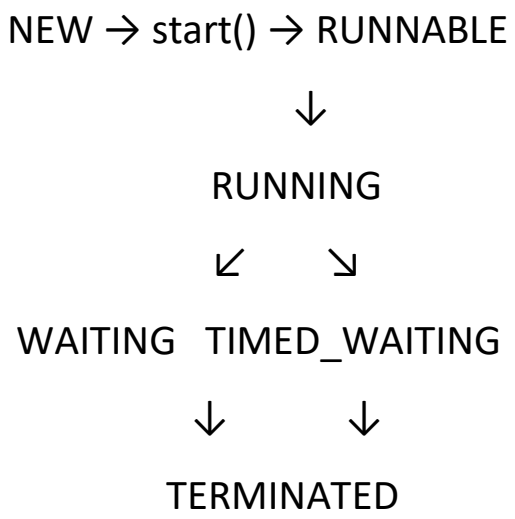
2. Implementing the Runnable interface

```
class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Runnable running...");
    }
}
```

Thread Lifecycle (States)

State	Description
NEW	Thread created but not started
RUNNABLE	Thread is ready to run
RUNNING	Thread is executing run()
BLOCKED	Waiting for lock
WAITING	Waiting indefinitely for another thread
TIMED_WAITING	Waiting for a specific time (e.g., sleep())
TERMINATED	Thread has finished execution

Thread lifecycle diagram



Real-world Use Case Example

In my project, when a feedback form is submitted:

- One thread saves data
- One sends confirmation email
- One updates analytics
  This improves responsiveness and user experience.

Thread Safety

When multiple threads access shared data, proper synchronization is needed to avoid inconsistent states. Use

- synchronized keyword
- volatile variables
- Concurrent APIs (like AtomicInteger, ConcurrentHashMap, etc.)