# JAVA THREADS AND THREAD POOLS

Sandra Kumi

REVIEWER  Mr. Thomas Darko

# Java Threads and Thread Pools

## 1. Thread Concepts

In Java, threads allow concurrent execution of tasks. Each thread represents a separate path of execution in a program.

Understanding threads and their lifecycle is key to developing responsive and efficient applications.

1.1. Creating Threads:

Threads in Java can be created in two ways:

- By implementing the Runnable interface.
- By extending the Thread class.

1.2. Thread Lifecycle:

A thread in Java goes through the following states:

- NEW: When the thread is created.
- RUNNABLE: When the thread is ready to run or is currently running.
- BLOCKED: When the thread is waiting for a monitor lock.
- WAITING: When the thread is waiting indefinitely for another thread.
- TIMED_WAITING: When the thread is waiting for a specified time.
- TERMINATED: When the thread has finished executing.

1.3. Thread Synchronization:

Synchronization ensures that only one thread accesses critical sections of the code at a time.

In Java, the `synchronized` keyword is used to achieve this.

- Synchronized Methods: Lock the entire method for exclusive access.

- Synchronized Blocks: Lock a specific block of code within a method.

## 2. Thread Pools

Thread pools manage a fixed set of threads to execute tasks efficiently. Instead of creating new threads for each task, a thread pool reuses a set of threads, leading to better resource utilization and task management.

2.1. Why Use Thread Pools:

- Improved performance by reusing threads and reducing thread creation overhead.
- Efficient resource management with a limited number of threads.
- Simplified task management by assigning tasks to a pool of threads.

2.2. Types of Thread Pools in Java:

- Fixed Thread Pool: A pool with a fixed number of threads.
- Cached Thread Pool: A pool that creates new threads as needed and reuses idle threads.
- Scheduled Thread Pool: A pool that schedules tasks for execution after a delay or at fixedintervals.

2.3. Example - Fixed Thread Pool:

```
ExecutorService executor = Executors.newFixedThreadPool(3);
for (int i = 1; i <= 5; i++) {     executor.submit(new
Task("Task " + i));
}
executor.shutdown();
```

```

In this example, we create a thread pool with 3 threads. The tasks are submitted to the pool, and the threads are reused to process them.
```