Day 13 – Multithreading and Synchronization in Java

Objective:

To understand the concept of multithreading in Java, learn how to create and manage threads, and explore synchronization techniques to handle concurrent execution safely.

Content:

Today, I studied **multithreading**, a feature in Java that allows the simultaneous execution of two or more parts of a program.
Each part of such a program is called a **thread**, and it runs independently, helping improve performance and resource utilization.

---

1. What is a Thread?

A thread is a lightweight subprocess — the smallest unit of CPU execution.
Java supports multithreading through the `Thread` class and the `Runnable` interface.

---

2. Creating Threads in Java

(a) By Extending the Thread Class

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running...");
    }
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        t1.start();  // starts the thread
    }
}
```

(b) By Implementing Runnable Interface

```
class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Thread is running using Runnable...");
    }
    public static void main(String[] args) {
        Thread t1 = new Thread(new MyRunnable());
        t1.start();
    }
}
```

---

3. Thread Methods

Method	Description
`start()`	Begins execution of a thread
`run()`	Contains the code to execute in a thread
`sleep(ms)`	Makes the thread pause temporarily
`join()`	Waits for a thread to finish execution
`setPriority()`	Changes the execution priority

---

4. Synchronization

When multiple threads access shared resources, data inconsistency can occur.
Synchronization ensures that only one thread accesses the resource at a time.

Example:

```
class Counter {
    synchronized void displayCount(){
        for (int i = 1; i <= 3; i++){
            System.out.println(i);
        }}}
```

Learning Outcome:

Understood how to create and execute multiple threads in Java.
 Learned to use thread lifecycle methods and synchronization to handle concurrency safely.
 Gained knowledge of managing parallel tasks effectively to enhance program performance.