1. **Concurrent Workers**

public class WorkerThread implements Runnable {

    private int workerId;

    public WorkerThread(int id) {

        this.workerId = id;

    }

    public void run() {

        System.out.println("Worker " + this.workerId + " started working");

        try {

            Thread.sleep(2000); // simulate work by sleeping for 2 seconds

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

        System.out.println("Worker " + this.workerId + " finished working");

    }

    public static void main(String[] args) {

        Thread[] threads = new Thread[5];

        for (int i = 0; i < threads.length; i++) {

            threads[i] = new Thread(new WorkerThread(i+1));

            threads[i].start();

        }

        // Wait for all threads to finish before exiting main thread

        for (Thread thread : threads) {

            try {

                thread.join();

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

}

1. **Message Thread**
2. **Prime Number Check**
3. **Producer Consumer**