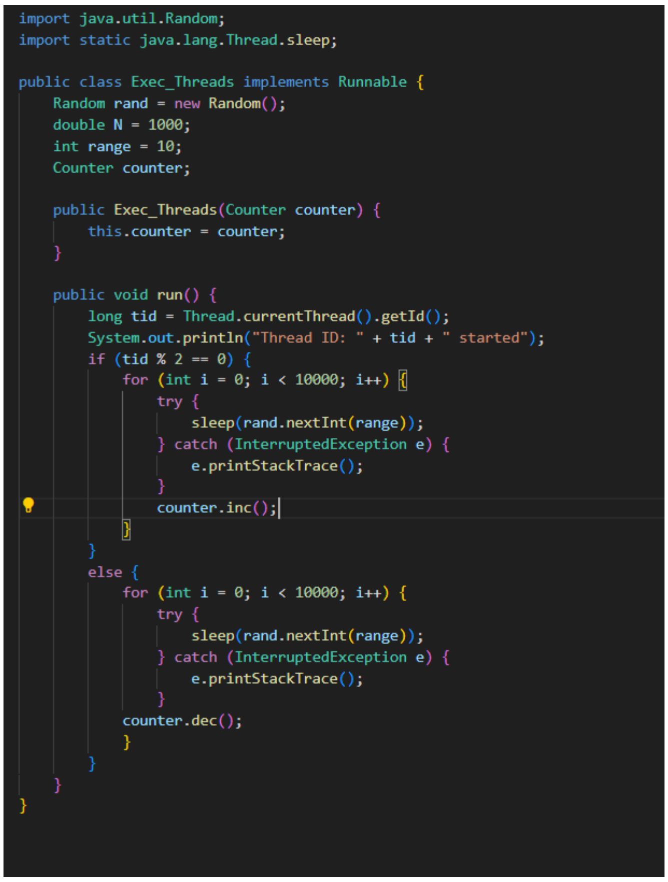
Parallel Computer Systems

Exercise 1 Dean Shalev

209707470



```java
import java.util.Random;
import static java.lang.Thread.sleep;

public class Exec_Threads implements Runnable {
    Random rand = new Random();
    double N = 1000;
    int range = 10;
    Counter counter;

    public Exec_Threads(Counter counter) {
        this.counter = counter;
    }

    public void run() {
        long tid = Thread.currentThread().getId();
        System.out.println("Thread ID: " + tid + " started");
        if (tid % 2 == 0) {
            for (int i = 0; i < 10000; i++) {
                try {
                    sleep(rand.nextInt(range));
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                counter.inc();
            }
        }
        else {
            for (int i = 0; i < 10000; i++) {
                try {
                    sleep(rand.nextInt(range));
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            counter.dec();
            }
        }
    }
}
```

1. We expect the value of sum to be 0 at the end of the program, because we incremented and decreased the same number of times.
2. The program does not always return 0.
3. Because we make no use of locks, we get race conditions, where one thread tries change the value of sum at the same time as the other thread does, which causes data corruption. This leads to an unexpected outcome.