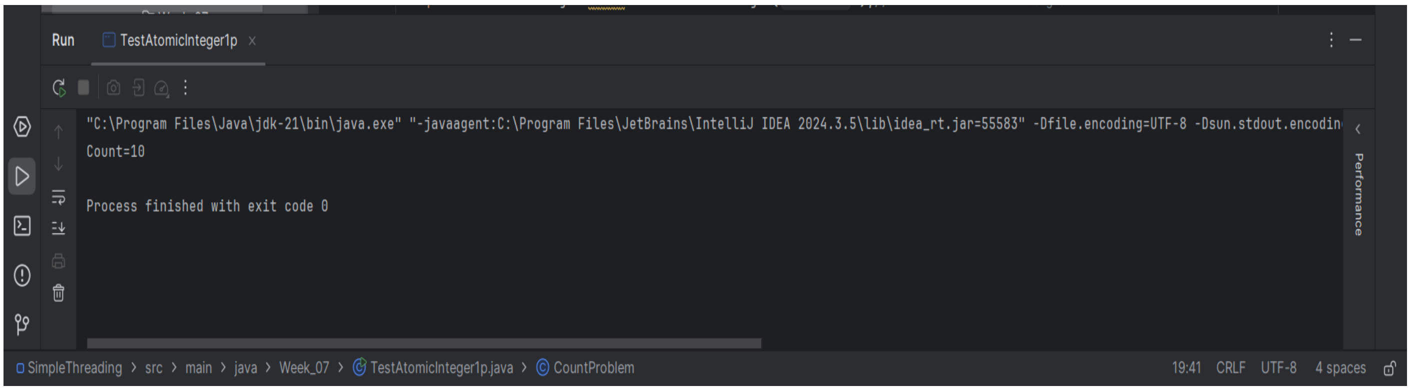
STIWK3014 REAL TIME PROGRAMMING

Tutorial / Exercise 8: TestAtomicInterger1p.java & Synchronization.java

TASK 1

```java
package Week_07;
import java.util.concurrent.atomic.AtomicInteger;//New modification

public class TestAtomicInteger1p {

    public static void main(String[] args) throws InterruptedException{

        CountProblem pt = new CountProblem ();
        Thread t1 = new Thread (pt, "t1");
        Thread t2 = new Thread (pt, "t2");
        t1.start ();
        t2.start ();
        t1.join();
        t2.join();
        System.out.println("Count=" + pt.getCount());
    }
}

class CountProblem implements Runnable {
    private AtomicInteger count = new AtomicInteger(0);//New modification

    @Override
    public void run() {

        for (int i = 0; i <= 4; i++) {//New modification
            processSomething(i);
            count.incrementAndGet();//New modification

        }
    }
    public int getCount() {
        return this.count.get();
    }
    private void processSomething(int i) {
        try{
            Thread.sleep(i*200);
        }catch(InterruptedException e){
            e.printStackTrace();
        }
    }

}
```

Output:



Run
TestAtomicInteger1p
Performance
SimpleThreading › src › main › java › Week_07 › TestAtomicInteger1p.java › CountProblem
19:41 CRLF UTF-8 4 spaces

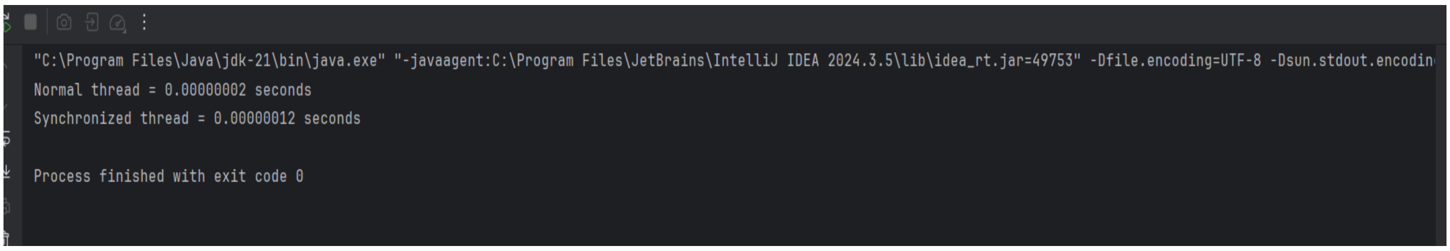
```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.5\lib\idea_rt.jar=55583" -Dfile.encoding=UTF-8 -Dsun.stdout.encodin
Count=10

Process finished with exit code 0
```

TASK 2

Comparison of Normal Thread and Synchronized Thread and the Outputs

```java
package Week_07;

public class ThreadComparison {
    static int counter = 0;
    static final int threadCount = 10;

    static class NormalThread extends Thread {
        public void run() {
            for (int i = 0; i < 100000; i++) {
                counter++;
            }
        }
    }

    static class SynchronizedThread extends Thread {
        private static final Object lock = new Object();s
        public void run() {
            for (int i = 0; i < 100000; i++) {
                synchronized (ThreadComparison.class) {
                    counter++;
                }
            }
        }
    }

    public static void main(String[] args) throws InterruptedException {
        counter = 0;
        Thread[] normalThreads = new Thread[threadCount];
        long normalStart = System.currentTimeMillis();

        for (int i = 0; i < threadCount; i++) {
            normalThreads[i] = new NormalThread();
            normalThreads[i].start();
        }
        for (int i = 0; i < threadCount; i++) {
            normalThreads[i].join();
        }

        long normalEnd = System.currentTimeMillis();
        double normalTime = (normalEnd - normalStart) / 1_000_000_000.0;

        // Synchronized Thread Execution
        counter = 0;
        Thread[] syncThreads = new Thread[threadCount];
        long syncStart = System.currentTimeMillis();

        for (int i = 0; i < threadCount; i++) {
            syncThreads[i] = new SynchronizedThread();
            syncThreads[i].start();
        }
        for (int i = 0; i < threadCount; i++) {
            syncThreads[i].join();
        }

        long syncEnd = System.currentTimeMillis();
        double syncTime = (syncEnd - syncStart) / 1_000_000_000.0;

        // Format to 8 decimal places
        System.out.printf("Normal thread = %.8f seconds\n", normalTime);
        System.out.printf("Synchronized thread = %.8f seconds\n",
syncTime);
    }
}
```

Output :



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.3.5\lib\idea_rt.jar=49753" -Dfile.encoding=UTF-8 -Dsun.stdout.encodin
Normal thread = 0.00000002 seconds
Synchronized thread = 0.00000012 seconds

Process finished with exit code 0
```