NUR AZYYATI BITNI ABU BAKAR | 291560

TUTORIAL 8:
THE CODE: TASK 1

TestAtomicInteger1P.java

```java
class TestAtomicInteger1P {
    public static void main(String[] args) throws InterruptedException {
        CountProblem pt = new CountProblem();
        Thread t1 = new Thread(pt, name: "t1");
        Thread t2 = new Thread(pt, name: "t2");
        t1.start();
        t2.start();
        t1.join();
        t2.join();

        System.out.println("Count= " + pt.getCount());
    }
}

class CountProblem implements Runnable {  2 usages

    private int count;  2 usages

    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            processSomething(i);
            count++;
        }
    }
    public int getCount() {  1 usage
        return this.count;
    }

    private void processSomething(int i) {  1 usage
        try {
            Thread.sleep( millis: i * 200);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

THE OUTPUT:

```
"C:\Program Files\Java\jdk-16.0.1\bi
Count= 10

Process finished with exit code 0
```

TestAtomicInteger1P.java (MODIFY)

```java
import java.util.concurrent.atomic.AtomicInteger; //New modification
class TestAtomicInteger1P {
    public static void main(String[] args) throws InterruptedException {
        CountProblem pt = new CountProblem();
        Thread t1 = new Thread(pt, name: "t1");
        Thread t2 = new Thread(pt, name: "t2");
        t1.start();
        t2.start();
        t1.join();
        t2.join();

        System.out.println("Count= " + pt.getCount());
    }
}

class CountProblem implements Runnable {  2 usages

    private AtomicInteger count = new AtomicInteger( initialValue: 0); //New modification

    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            processSomething(i);
            count.incrementAndGet(); //New modification
        }
    }

    public int getCount() {  1 usage
        return count.get(); //New modification
    }

    private void processSomething(int i) {  1 usage
        try {
            Thread.sleep( millis: i * 200);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

THE OUTPUT: (MODIFY)

```
"C:\Program Files\Java\jdk-16.0.1\
Count= 10

Process finished with exit code 0
```

THE CODE: TASK 2 Comparison of Normal Thread and Synchronized Thread

NormalThread.java

```
public class NormalThread extends Thread {  1 usage
    @Override
    public void run() {
        for (int i = 0; i < 1000; i++) {
            int x = i * i;
        }
    }
}
```

Synchronized.java

```
public class SynchronizedThread extends Thread {  1 usage
    private static final Object lock = new Object();  1 usage

    @Override
    public void run() {
        synchronized (lock) {
            for (int i = 0; i < 1000; i++) {
                int x = i * i;
            }
        }
    }
}
```

ComparisonNormalAndSynchronizedThread.java

```java
public class ComparisonNormalAndSynchronizedThread {
    public static void main(String[] args) throws InterruptedException {
        long startTime, endTime;

        startTime = System.nanoTime();
        Thread[] normalThreads = new Thread[10];
        for (int i = 0; i < 10; i++) {
            normalThreads[i] = new NormalThread();
            normalThreads[i].start();
        }
        for (Thread t : normalThreads) {
            t.join();
        }
        endTime = System.nanoTime();
        double normalDuration = (endTime - startTime) / 1_000_000_000.0;
        startTime = System.nanoTime();
        Thread[] syncThreads = new Thread[10];
        for (int i = 0; i < 10; i++) {
            syncThreads[i] = new SynchronizedThread();
            syncThreads[i].start();
        }
        for (Thread t : syncThreads) {
            t.join();
        }
        endTime = System.nanoTime();
        double syncDuration = (endTime - startTime) / 1_000_000_000.0;

        System.out.printf("Normal thread = %.9f seconds\n", normalDuration);
        System.out.printf("Synchronized thread = %.9f seconds\n", syncDuration);
    }
}
```

THE OUTPUT:

```
"C:\Program Files\Java\jdk-16.0.1\bin\java.exe
Normal thread = 0.009900500 seconds
Synchronized thread = 0.015148100 seconds

Process finished with exit code 0
```