PCPP Assignment 4

Group name: Thread Heresy
Real names : Nedas Surkus, Niclas Abelsen,
Github username: nesu, niab

November 2022

Contents

7.1

1-2

See *LambdaExample.java*.

4

See *LambdaExample.java*.

7.2

1

```
Sequential                  12035959.4 ns  243332.85         32
IntStream                         0.7 ns       0.00  536870912
Parallel                          2.6 ns       0.02  134217728
ParallelStream                    2.7 ns       0.02  134217728
```

2

```
Sequential                  11831076.6 ns  143556.18         32
IntStream                   12002457.5 ns   48583.18         32
Parallel                          0.7 ns       0.01  536870912
ParallelStream                    2.6 ns       0.01  134217728
```

3

See *PrimeCountingPerf.java*.

4

```
Sequential                  11731206.6 ns   28359.49         32
IntStream                   11754149.4 ns   19636.76         32
Parallel                     2141519.8 ns    9246.52        128
ParallelStream                    0.7 ns       0.01  536870912
```

5

```
Sequential                  11731174.7 ns   10218.10         32
IntStream                   11802634.7 ns   10463.31         32
Parallel                     2129965.5 ns    6515.89        128
ParallelStream               2141068.0 ns    5652.60        128
```

7.3

1-5

See *TestWordStream.java*.

6

Yes. We can measure it using Mark7 imported from Benchmark.java

```
Sequential                  31913260.0 ns  324058.51          8
Parallel                    13720863.1 ns   83835.28         32
```

7-8

See *TestWordStream.java*.

9.1

1

See stopwatchUI.java

2

Interleaving can occur but is not noticeable the first minutes. It takes very few nanoseconds to perform the time conversations which results in the timer being behind. This is not crucial for short time measurements but can have an affect when measuring for longer periods.

This can be fixed by synchronizing the method or using locks.

3

See stopwatch2.java

4

See stopwatchN.java

9.2

1

Yes

2

Yes

3

We take 3 items. A, AB, ABC and convert them into an observable source and then we use flat map.

We then map value v (which is one of the three items) and we specify a random delay.

Before we do next. We subscribe it on a scheduler which gives us a thread.

Once the delay is finished we will output a string with current thread.

After we map it, each of the values is a observable source and we can subscribe on them.

Once we subscribe , we will output another string with thread name and item length.

The reason why it does not execute one after another is due to the random delay we mapped values to.

Additionally, it takes some time to assign a value to the scheduler thus we will not get any of the inputs in a predictable manner.

But rather, we will get them as they become available.

9.3

1

See StopwatchRx.java

2

See stopwatchUI.java

9.4

1-5

See TestWordStream.java