Ahmad Hakam Alhafi

Part 1:

A:

A screenshot of a computer

Description automatically generated

Description:

**I made 3 Threads (t1, t2, t3) by implementing the predefined interface Runnable for the three classes (max, min, avg) which I put the appropriate operations in their run() method.**

**Then I start the threads by calling start() method, and I invoked join() method after starting the threads, which means that the calling thread goes into a waiting state. It remains in a waiting state until the referenced thread terminates, and it may also return if the referenced thread was interrupted. In this case, the method throws an InterruptedException.**

**And the output for this array as an example {90, 81, 78, 95, 79, 72} was as you see in the screenshot.**

B:

A screenshot of a computer

Description automatically generated with medium confidence

Description:

**I made one thread by extending the predefined class Thread for my class Prime which I put the appropriate operations in its run() method.**

**Then I start the thread by calling start() method, and I invoked join() method after starting the thread, which means that the calling thread goes into a waiting state. It remains in a waiting state until the referenced thread terminates, and it may also return if the referenced thread was interrupted. In this case, the method throws an InterruptedException.**

**And the output for inserting number 15 for example was as you see in the screenshot; all the prime numbers less than or equal to 15.**

C:

A screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generated

Description:

**You can see the execution time for each case in the screenshots :).**

**In task 1, it takes execution time with threads more than the case without threads.**

**creating a thread is a relatively expensive OS operation, and context switching is again a relatively expensive operation.**

**We will be waiting for every thread to finish its work, so the time will be like (the sequential solution time+ creating threads time).**

Text

Description automatically generated Text

Description automatically generated

Description:

**You can see the execution time for each case in the screenshots :).**

**In task 2, it takes execution time without threads more than the case with threads.**

**I just made a one thread, so there is not much context switching time, and there is I/O operations, which is a case that multithreading can be more useful with.**

Part 2:

A:

A screenshot of a computer

Description automatically generated with medium confidence

Description:

**I made 5 Threads p1, p2, p3, p4, p5 (4 threads for sorting and 1 for merging) by implementing the predefined interface Runnable for the 2 classes (Sort, Merge) which I put the appropriate operations in their run() method.**

**Then I start the threads by calling start() method, and I invoked join() method after starting the threads, which means that the calling thread goes into a waiting state. It remains in a waiting state until the referenced thread terminates, and it may also return if the referenced thread was interrupted. In this case, the method throws an InterruptedException.**

**And the output for this array as an example {2, 0, 1, 4, 3, 5, 8, 6, 7, 10, 9, 11} was as you see in the screenshot; sorted list.**

B:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

Description:

**You can see the execution time for each case in the screenshots :).**

**In task 1, it takes execution time with threads more than the case without threads.**

**creating a thread is a relatively expensive OS operation, and context switching is a relatively expensive operation too.**

**We will be waiting the first 4 threads to finish their works, then the fifth thread will start working, and we will be waiting again until it finishes its work.**

**So, the execution time will be like (the sequential solution time+ creating threads time).**

**Thanks for ur time ♥**