

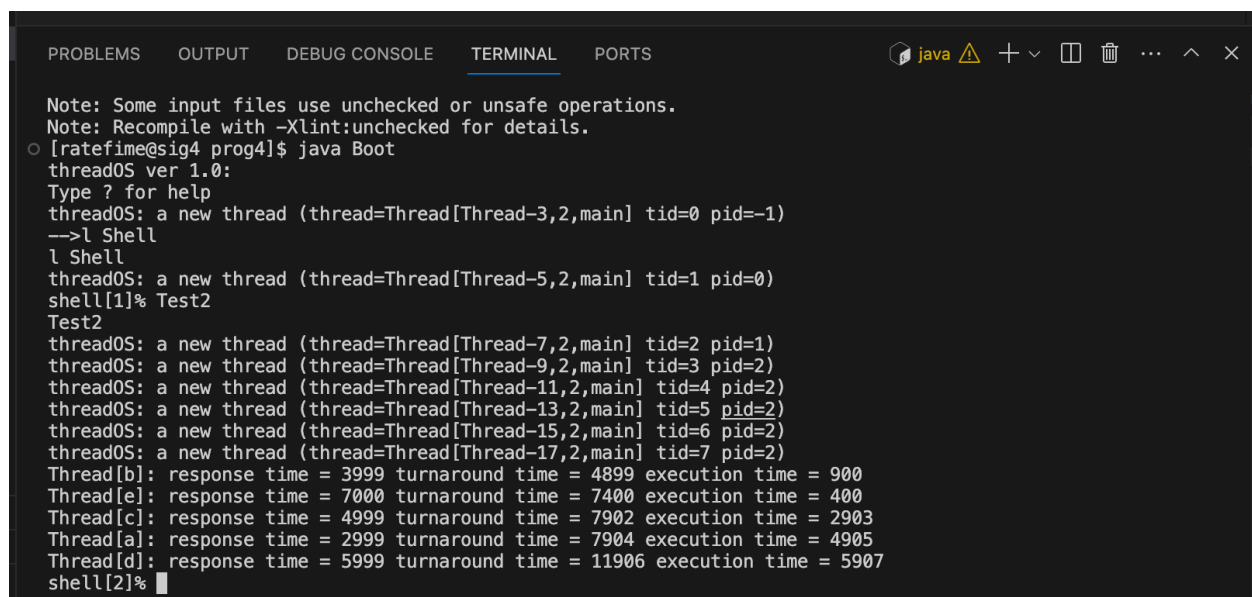
Part1)

First, I created SyncQueue.java with the specification provided on Programming4.

SyncQueue.java: according to the structure, first, I created an array of QueueNode objects, each dedicated to a different condition, allowing threads to wait or be notified about certain events efficiently. The SyncQueue provides two constructors: one with a default condition size of 10 and another allowing customization for the number of conditions. enqueueAndSleep(int condition) method, which places a calling thread into a waiting state until its condition is satisfied, and dequeueAndWakeup(int condition, int tid), tasked with resuming a waiting thread when its condition is met. An overloaded version of dequeueAndWakeup accepts a thread ID as an additional argument, contributing control by specifying which thread to wake.

Second, QueueNode.java: the QueueNode improves thread synchronization by using a LinkedList<Integer> to manage thread IDs. QueueNode allows for targeted waking of threads based on specific conditions. The sleep() method puts a thread into a wait state until a condition is met, marked by an entry in the queue. The wake(int tid) method selectively awakens threads, offering a more precise control over synchronization.

Third, Kernel.old.java: I followed the instruction, first, I needed to get the current thread id by using scheduler.getMyTcb(). Second, make the current thread sleep in waitQueue. Third, I needed to get the current parent's thread id and wakeup the threads. Finally, I needed to delete the current thread.



```

Note: Some input files use unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
○ [ratefime@sig4 prog4]$ java Boot
thread0S ver 1.0:
Type ? for help
thread0S: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Shell
l Shell
thread0S: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
shell[1]% Test2
Test2
thread0S: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
thread0S: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=2)
thread0S: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=2)
thread0S: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=2)
thread0S: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=2)
thread0S: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=2)
Thread[b]: response time = 3999 turnaround time = 4899 execution time = 900
Thread[e]: response time = 7000 turnaround time = 7400 execution time = 400
Thread[c]: response time = 4999 turnaround time = 7902 execution time = 2903
Thread[a]: response time = 2999 turnaround time = 7904 execution time = 4905
Thread[d]: response time = 5999 turnaround time = 11906 execution time = 5907
shell[2]%

```

Part2)

For part2, I used the Kernel.java which I implemented in part 1, and according to the structure, I added enqueueAndSleep for Rawread, Rawwrite, and Sync. Then I enabled the INTERRUPT_DISK case. Since we want to wake up threads on either disk_req or disk_fin, we do not need to use if statement, so just wake up threads on either case.

Test3.java and TestThread3.java : First, Test3.java does two main types of operations: numerical computations and read and write many blocks across the disk. It works by using another class named TestThread3.java. Within TestThread3.java, there are specific instructions for what to do: if it's about computing, it performs complex algorithms to evaluate the CPU's performance. If it's about disk operations, it handles reading from and writing to the disk based on a structure provided in Programming4. After running these operations a certain number of times, Test3.java calculates the total time taken for these operations.

Kernel.old.java: 1 Test3 6

```
Type ? for help
thread0S: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Test3 6
l Test3 6
thread0S: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
thread0S: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-23,2,main] tid=10 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-25,2,main] tid=11 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-27,2,main] tid=12 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-29,2,main] tid=13 pid=1)
it is reading and writing on the disk!!
elapsed time is = 302359 msec
-->█
```

Kernel.java: l Test3 6

```
Type ? for help
thread0S: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Test3 6
l Test3 6
thread0S: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
thread0S: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-23,2,main] tid=10 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-25,2,main] tid=11 pid=1)
it is reading and writing on the disk!!
thread0S: a new thread (thread=Thread[Thread-27,2,main] tid=12 pid=1)
it is computing!!
thread0S: a new thread (thread=Thread[Thread-29,2,main] tid=13 pid=1)
it is reading and writing on the disk!!
elapsed time is = 302881 msec
-->
```

I tested Test3 from 2 up to 6 on both Kernel.old and Kernel:

| | A | B | C |
|---|-----------|-----------------|-------------|
| 1 | Test3 | Kernel.old.java | kernel.java |
| 2 | l Test3 2 | 93199 msec | 110713 msec |
| 3 | l Test3 3 | 157416 msec | 145165 msec |
| 4 | l Test3 4 | 190500 msec | 190538 msec |
| 5 | l Test3 5 | 251886 msec | 243287 msec |
| 6 | l Test3 6 | 302359 msec | 302881 msec |
| 7 | | | |

Report)

When we look at the results it seems that the kernel.java, does better sometimes, like in Test3 3, but not all the time. For example, in the Test3 2 test, the old kernel actually finished faster. When lots of threads are running, as in Test3 5 and Test3 6, the times are pretty close, but the new kernel is a bit slower, which might mean it struggles a bit more when things get really busy. I am

not sure if my complex computing did the best job to show the results, that might be why the results sometimes are really close.