

## The Tutorial Questions and Lab Projects of Week 4

### Tutorial Questions

1. Discuss each of the tasks of encapsulation, concurrent processing, protection, name resolution, communication of parameters and results, and scheduling in the case of the UNIX file service (or that of another kernel that is familiar to you).
2. Why are some system interfaces implemented by dedicated system calls (to the kernel), and others on top of message-based system calls?
3. Smith decides that every thread in his processes ought to have its own protected stack – all other regions in a process would be fully shared. Does this make sense?
4. Should signal (software interrupt) handlers belong to a process or to a thread?
5. A file server uses caching, and achieves a hit rate of 80%. File operations in the server cost 5 ms of CPU time when the server finds the requested block in the cache, and take an additional 15 ms of disk I/O time otherwise. Explaining any assumptions you make, estimate the server's throughput capacity (average requests/sec) if it is:

- i) single-threaded;
  - ii) two-threaded, running on a single processor;
  - iii) two-threaded, running on a two-processor computer.
6. A client makes RMI's to a server. The client takes 5 ms to combine the arguments for each request, and the server takes 10 ms to process each request. The local OS processing time for each send or receive operation is 0.5 ms, and the network time to transmit each request or reply message is 3 ms. Marshalling or unmarshalling takes 0.5 ms per message. Estimate the time taken by the client to generate and return from 2 requests:
    - (i) if it is single-threaded, and
    - (ii) if it has two threads which can make requests concurrently on a single processor. Is there a need for asynchronous RMI if processes are multi-threaded?

### Lab Projects

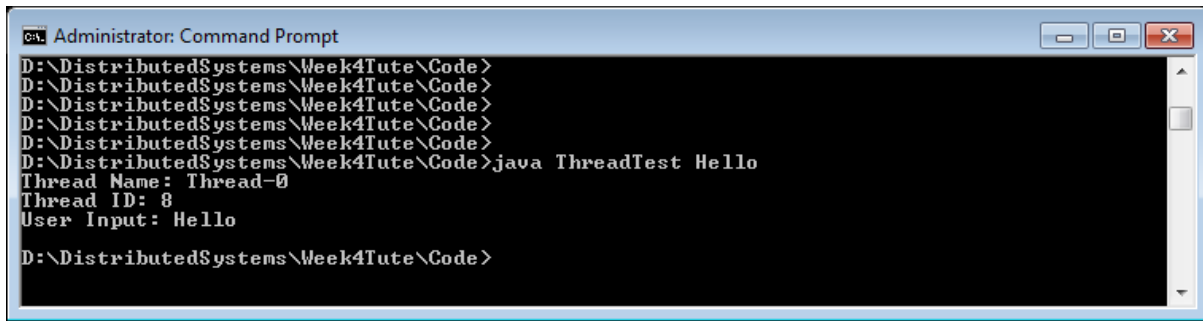
In Java, there are two ways to implement concurrency: implementing the `Runnable` interface or extending the `Thread` class. Read the Java concurrency document at:

<http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html>

After reading the document, you need to complete the following tasks.

#### Task 1:

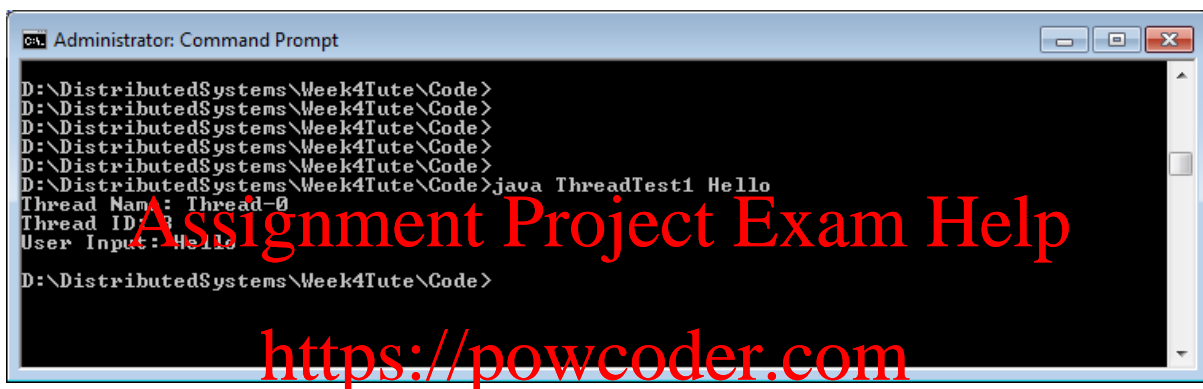
Define a Java class to implement the `Runnable` interface. The program should have the function to accept the user's input and output: the user's input, the thread's identifier and the thread's name. The outputs of the program should like the following screenshot of `ThreadTest` program.



```
C:\> Administrator: Command Prompt
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>java ThreadTest Hello
Thread Name: Thread-0
Thread ID: 8
User Input: Hello
D:\DistributedSystems\Week4Tute\Code>
```

## Task 2:

For the same function as those of the program in Task 1, implement a program by extending Java Thread class. For example, the Threadtest1 program inherits Thread class with the same function implemented.



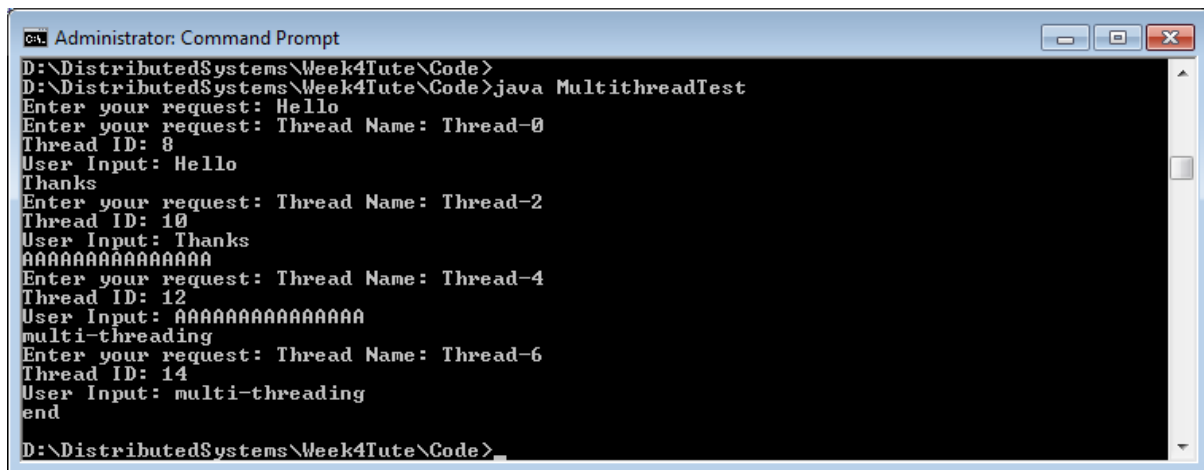
```
C:\> Administrator: Command Prompt
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>java ThreadTest1 Hello
Thread Name: Thread-0
Thread ID: 8
User Input: Hello
D:\DistributedSystems\Week4Tute\Code>
```

Assignment Project Exam Help  
<https://powcoder.com>

## Task 3:

1. Revise the program of Task 1 or Task 2 so that the program continuously accepts the user's inputs as a string till the input is the string 'end', which enables the program to exit normally.
2. For each input, the program creates a thread to process. The thread just outputs: the thread's name, thread's identifier and the user's input.

For example, the MultithreadTest program is an implementation for the above task. The screenshot of the program is as follows.



```
C:\> Administrator: Command Prompt
D:\DistributedSystems\Week4Tute\Code>
D:\DistributedSystems\Week4Tute\Code>java MultithreadTest
Enter your request: Hello
Enter your request: Thread Name: Thread-0
Thread ID: 8
User Input: Hello
Thanks
Enter your request: Thread Name: Thread-2
Thread ID: 10
User Input: Thanks
AAAAAAAAAAAAAAAAAAAA
Enter your request: Thread Name: Thread-4
Thread ID: 12
User Input: AAAAAAAAAAAAAAAAAA
multi-threading
Enter your request: Thread Name: Thread-6
Thread ID: 14
User Input: multi-threading
end
D:\DistributedSystems\Week4Tute\Code>_
```

#### Task 4:

Have your program produced outputs which are similar to those in the above screenshot? If so, give your analysis about why the outputs seem strange. For example, why is the output of Thread Name on the same line as the input prompt?

## Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder