Threads

1. Write a class(ExtendedThread) which extends Thread. Inside the run methods print the current thread name, priority & its state. In the runner class Spawn a new thread using the ExtendedThread class. Print the current thread name, priority & its state in the runner class before & after calling the thread.start() method.
2. Write a class (RunnableThread) which implements Runnable. Inside the run methods print the current thread name, priority & its state. In the runner class Spawn a new thread using the RunnableThread class.Print the current thread name, priority & its state in the runner class before & after calling the thread.start() method.
3. Spawn a new Thread using the ExtendedThread and provide the name for that thread as ExtendedThread. Spawn a new Thread using the RunnableThread & set the name as RunnableThread. Print the current thread name, priority & its state in the runner class before & after calling the thread.start() method of each thread.
4. In the run method of ExtendedThread, print a message “Going to Sleep: “+threadName. After that sleep for 60 secs. Then print After sleeping: “+threadName.In the run method of RunnableThread,print a message “Going to Sleep: “+threadName. After that sleep for45 secs. Then print After sleeping: “+threadName.From the runner class spawn 5 ExtendedThreads with your custom Thread name for each thread & 5 more RunnableThreads with your custom name for each Thread. Observe what happens by analysing the output prints.
5. In the above task instead of hard coding the sleep time, try to accept the sleep time as an argument while creating the Thread itself for each thread. Try to provide a different sleep time for each Thread.
6. In continuation to the above task, have some condition based while loop in the run methods of RunnableThread & ExtendedThread class. That is once the thread is started it should not exit until the while loop condition is made false. That while loop condition(variable) should be instance specific & should not be static. You can have a setter method for the variable to toggle the value from outside classes. Now once the runner starts all the threads, after 2 minutes, take 3 Thread Dumps at an interval of 30 seconds in between each & analyse the state of each Thread.
7. In continuation to the above task, once all the threads are started. From the runner class try to stop the execution of Thread one by one by toggling the condition variable of the thread. The difference between two stops can be 60 seconds from the runner class. At the end check if all the Threads are alive or not. If all the 10 threads exited, you can print tasks completed. After 2 minutes of starting the runner class, start taking thread dumps(10 count) at 45 secs interval. Please analyze the difference between each thread dump. Take a final thread dump after the tasks completed message.