Hard Working Threads

Purpose

To review concepts related to threads

Directions

You will need to download the file Lab 8b Starting Code.zip from Pilot for this lab. This code involves one taskmaster thread and several worker threads. The taskmaster thread creates new tasks and adds them to the order queue. Each task takes a certain number of milliseconds to complete. The worker threads pick up tasks from the order queue and simulate working on them by sleeping for the specified number of milliseconds. The program continues until a total of 100 tasks have been created.

Your tasks for this lab are:

- Right now the queue can behave inconsistently. For example, occasionally two workers can both accept the same task (see below). Modify the program so that this cannot occur.

```
Worker 3 accepted Task 1: 1007 ms
Worker 1 accepted Task 2: 896 ms
Worker 2 accepted Task 1: 1007 ms
```

- Right now the program can say it is finished before all of the tasks have been made and completed (see below). Modify the program so that this cannot occur.

```
Worker 1 accepted Task 1: 1175 ms
Finished?
Worker 1 finished Task 1: 1175 ms
```

- Right now if a worker requests a task to work on but the queue is empty, the acceptTask method will loop until the queue is no longer empty, as shown below. This is called a "busy loop" and it is not an efficient use of CPU time. Modify this code so that the method instead waits until it has been notified that a new task has been created.

```
public Task acceptTask() {
    while (orders.isEmpty()) {
        // looping until there is a task in the queue
    }
    return orders.poll();
}
```

- Change the program so that if there are five tasks in the queue, the program will wait until the number of tasks falls below five before adding any more tasks to the queue.

Hint: You can solve all of the problems above by only making changes to two files: Lab8PracticeDriver.java and OrderQueue.java