Assignment 2:
Due: April 10, 2022, 11 PM

Overview: In this project you will implement a CPU scheduler using a priority queue.

Background: Operating systems uses queues and a concept known as time slicing to enable multitasking. As new processes are created, they are added to the queue. Each process in turn is

1. dequeued
2. run on the CPU for a certain length of time (one time slice)
3. enqueued

These steps are repeated for each process until it has finished.

Some processes are more important than others, however, and so the queue in question will be a priority queue.

Instructions:
Create a class, **Job**, that has these variables (it may have others):

priority (-20 to +19, where -20 is highest)
length (in time slices, between 1-100)
timeWaiting (in time slices) and
name (a String, which may include spaces)

Each iteration of a loop will represent one time slice, and in that iteration perform the following tasks:

1. dequeue a job for execution
2. for all *waiting* processes, increment timeWaiting, and should it exceed maxWaitingTime, increase priority by 1 (and reset timeWaiting to 0)
3. process one instruction from **jobs.txt**. Each instruction should have the form
    1. add job *name* with length *n* and priority *p* or
    2. no new job this slice
    3. Note: If an instruction is malformed, ignore it.

When jobs.txt has been completely read and there are no more processes in need of CPU time, print out Done!

Sample Output:
This shows 3 distinct runs of the program.

```
A (priority: -15, time remaining: 4)
A (priority: -15, time remaining: 3)
A (priority: -15, time remaining: 2)
A (priority: -15, time remaining: 1)
A (priority: -15, time remaining: 0)
B (priority: -14, time remaining: 2)
B (priority: -14, time remaining: 1)
B (priority: -14, time remaining: 0)
C (priority: 4, time remaining: 6)
C (priority: 4, time remaining: 5)
C (priority: 4, time remaining: 4)
C (priority: 4, time remaining: 3)
C (priority: 4, time remaining: 2)
C (priority: 4, time remaining: 1)
C (priority: 4, time remaining: 0)
Done!
```

```
# Output 1
#this output was with maxWaitingTime of 2
add job A with length 5 and priority -15
add job B with length 3 and priority -12
add job C with length 7 and priority 8
no new job this slice
```

```
A (priority: -15, time remaining: 4)
A (priority: -15, time remaining: 4)
A (priority: -15, time remaining: 3)
A (priority: -15, time remaining: 2)
A (priority: -15, time remaining: 1)
with length (priority: -19, time remaining: 3)
with length (priority: -19, time remaining: 2)
with length (priority: -19, time remaining: 1)
with length (priority: -19, time remaining: 0)
A (priority: -17, time remaining: 0)
B (priority: -18, time remaining: 2)
B (priority: -18, time remaining: 1)
B (priority: -18, time remaining: 0)
C (priority: 0, time remaining: 6)
C (priority: 0, time remaining: 5)
C (priority: 0, time remaining: 4)
C (priority: 0, time remaining: 3)
C (priority: 0, time remaining: 2)
C (priority: 0, time remaining: 1)
C (priority: 0, time remaining: 0)
Done!
```

Output 2
#this output was also with maxWaitingTime of 2
add job A with length 5 and priority -15
add job B with length 3 and priority -12
add job C with length 7 and priority 8
no new job this slice
add job with length with length 4 and priority -19
add job seriously?? with length 4 and priority -20
add job D with length x and priority 2

```
A (priority: -15, time remaining: 4)
A (priority: -15, time remaining: 3)
A (priority: -15, time remaining: 2)
A (priority: -15, time remaining: 1)
with length (priority: -19, time remaining: 3)
with length (priority: -19, time remaining: 2)
with length (priority: -19, time remaining: 1)
with length (priority: -19, time remaining: 0)
A (priority: -19, time remaining: 0)
B (priority: -19, time remaining: 2)
B (priority: -19, time remaining: 1)
B (priority: -19, time remaining: 0)
C (priority: -2, time remaining: 6)
C (priority: -2, time remaining: 5)
C (priority: -2, time remaining: 4)
C (priority: -2, time remaining: 3)
C (priority: -2, time remaining: 2)
C (priority: -2, time remaining: 1)
C (priority: -2, time remaining: 0)
Done!
```

Output 3
#this output was with maxWaitingTime of 0
add job A with length 5 and priority -15
add job B with length 3 and priority -12
add job C with length 7 and priority 8
no new job this slice
add job with length with length 4 and priority -19
add job seriously?? with length 4 and priority -20
add job D with length x and priority 2

Notes:

In the event of equal priority, you may choose whichever job you wish to process first.

Requirements:

- Identify the major units of functionality in this assignment, and use classes and methods to encapsulate them. All methods must be commented using Javadoc and tested.
- Your code must be stored in a GitHub repo, and you must make multiple (at least 4) commits and push them to GitHub
- Use net.datastructures implementations of any data structures required for this assignment.
- Review all coding standards and make sure you have adhered to them.

    - Your method comments, in particular, should *briefly* but clearly explain the intent of (almost) every line: walk me through your code. Write comments to the side of statements, not above them.
    - If an identifier name is unclear, refactor it (use F2 in VSC).
    - Abstract away significant complexity into separate methods. There is one area in particular, in this assignment, that will make your head hurt and your grade drop if you do not heed this
    - Do not use any regular expressions for parsing instructions: use only the basic String methods
    - No methods should be more than approximately 30 lines long
    - Write comments
- Your output should be formatted identically to mine

Deliverables:

1. Your entire Visual Studio Code project, zipped
2. Unzipped screenshots of your output, uploaded separately from the zipped project.
    - there should be 3 screenshots, corresponding to the 3 files
3. An unzipped screenshot of your **commit history** on GitHub
    - To see your commit history, In the <> Code tab on GitHub, click the commits button
    - The window that then appears is your commit history
4. In the submission comments, a link to your GitHub repo