Spring ’16 CIS 212 Assignment 7 – 110/100 points possible – Due Wednesday, 5-25, 11:59 PM

The goal of this assignment is to gain experience working with concurrent programming via multiple threads and thread synchronization. You’ll write an implementation that simulates producer and consumer processes. The simulated consummation process will take far more time to complete than the production process, causing the machine to run out of memory if the threads aren’t synchronized in such a way that no more than a specified number of units are produced prior to being consumed.

1. [10] Create a new class with a public static void main method that first creates a java.util.concurrent.LinkedBlockingQueue with enough capacity for 100,000 String entries.

2. [20] Create a new class which implements Runnable to simulate the producer. When executed, this process should add 2,000,000 random Strings (hint: see the UUID class) to the above queue, waiting until there is space in the queue if necessary (i.e., using the LinkedBlockingQueue put() method). Print your progress once every 1000 Strings produced.

3. [40] Create another new class which implements Runnable to simulate the consumer. When executed, this process should consume Strings from the queue, keeping track of the overall max String found (i.e., using the String compareTo() method). The process should continue as long as there are Strings in the queue or the producer hasn’t finished (i.e. so the consumer doesn’t quit if the queue happens to be empty before the producer finishes). Use the Thread sleep() method to wait up to 10 milliseconds (i.e., a random number of milliseconds on the range [0, 10)) between each String comparison to ensure that the consumer takes longer to execute than the producer. Print your progress once every 1000 Strings consumed. Print the total number of Strings consumed and the max String found when the process completes.

There are some important subtleties to note here. You’ll want to implement your consumer in such a way that your application supports an arbitrary number of consumers. The consumers are expected to consume at different rates, so consumers should not simply consume a predetermined number of Strings and then quit; one consumer may complete significantly earlier than the others and would be idle waiting for the other processes in this case. As such, you’ll want a way for your producer to signal its completion to the consumers. You’ll also need to ensure that you don’t have a consumer potentially waiting indefinitely for a String to be produced after the producer has completed (hint: see the LinkedBlockingQueue poll() method that allows a timeout). We’ll discuss this in more detail in class.

4. [10] Use a java.util.concurrent.ExecutorService to execute your producer and at least two consumers concurrently. Shut down the service after starting the two processes.

5. [20] Write code that is clear and efficient. Specifically, your code should be indented with respect to code blocks, avoid unnecessarily repetitive code, avoid code that is commented out or otherwise unused, use descriptive and consistent class/method/variable names, etc.

6. [+10] (Extra credit) Use a synchronized LinkedList in part 1 rather than a LinkedBlockingQueue. You'll then need to use the Object wait() method in part 2 to halt the producer process and an Object notify() or notifyAll() method in part 3 to resume the producer.

The following is output for producing 10000 Strings with a queue of size 1000 and 2 consumers:

```
produced: 1000

produced: 2000

consumer 2 consumed: 1000

produced: 3000

consumer 1 consumed: 1000

produced: 4000

consumer 2 consumed: 2000

produced: 5000

consumer 1 consumed: 2000

produced: 6000

consumer 2 consumed: 3000

produced: 7000

consumer 1 consumed: 3000

produced: 8000

consumer 2 consumed: 4000

produced: 9000

consumer 1 consumed: 4000

produced: 10000

done producing! 10000 produced

consumer 2 consumed: 5000

consumer 1 done consuming! 4968 consumed

consumer 1 max String: fff60361-d394-46c7-92e4-ce7935da1518
```

```
consumer 2 done consuming! 5032 consumed

consumer 2 max String: fffd0f28-e265-4e89-ad1a-8ac4a8281501
```

Zip the Assignment7 folder in your Eclipse workspace directory, name the .zip file <Your Full Name>Assignment7.zip (e.g., EricWillsAssignment7.zip), and upload the .zip file to Canvas (see Assignments section for submission link).