Fall ’14 CIS 212 Assignment 7 – 110/100 points possible – Due Wednesday, 11-26, 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 production procedure will take far less time to complete than the consummation procedure, therefore causing the machine to run out of memory if the threads aren’t synchronized such that no more than a specified number of units are produced prior to being consumed.

1. [20] Create a new class with a public static void main method that first creates a java.util.concurrent.ArrayBlockingQueue with enough capacity for 100,000 String entries.

2. [30] 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 (i.e., using the ArrayBlockingQueue put() method), calling Thread.yield() between each addition (we’ll discuss this in class). Print your progress once every 1000 additions (e.g., “1000 Strings produced”). Note that creating and adding 2,000,000 String to a simple ArrayList (or LinkedList, etc) should cause Java to run out of memory (at least it does on my machine)!

3. [30] Create another new class which implements Runnable to simulate the consumer. When executed, this process should find and print the overall max String (i.e., using the ArrayBlockingQueue take() method to get each String and the String compareTo() method for the comparisons) of the 2,000,000 random Strings above, calling Thread.sleep(1) between each comparison (i.e., to simulate a more complicated process). Print your progress once every 1000 additions (e.g., “1000 Strings consumed”). This process is going to take some time to complete, so I recommend starting with a smaller queue size and number of Strings for initial testing.

4. [20] Use a java.util.concurrent.ExecutorService to execute your producer and consumer concurrently. Shut down the service after starting the two processes.

5. [+10] (Extra credit) Use a synchronized ArrayList in part 1 rather than a ArrayBlockingQueue (hint: see the Collections.synchronizedList method). You’ll then need to use the Object wait method in part 2 to halt the producer process and the Object notify or notifyAll method in part 3 to resume the producer.

Zip the Assignment7 folder in your Eclipse workspace directory and upload the .zip file to Blackboard (see Assignment 7 assignment in the Assignments area).