Producer Consumer Assignment

Design

1. You will implement Java code for the bounded buffer problem discussed in the lecture on threads and observe a race condition. The file Factory.java on the course website's Notes webpage can help with this assignment.

2. The buffer is a large array of n integers, initialized to all zeros.

3. The producer and the consumer are separate concurrent threads in your program.

4. The producer executes short bursts of random duration. During each burst of length k1, the producer adds a 1 to the next k1 slots of the buffer, modulo n.

5. The consumer also executes short bursts of random duration. During each burst of length k2, the consumer reads the next k2 slots, modulo n, and resets each to 0.

6. If any slot contains a number greater than 1, then a race condition has been detected: the consumer was unable to keep up and thus the producer has added a 1 to a slot that has not yet been reset.

7. Both producer and consumer sleep periodically for random time intervals to emulate unpredictable execution speeds.

8. The producer thread pseudo code:

```
while (true)
{
  generate random integer k1 using k.
  for i from 0 to (k1 – 1)
  {
    buffer[(next_in + i) mod n] += 1
  }
  next_in = (next_in + k1) mod n
  generate random integer t1 using t
  sleep for t1 seconds
}
```

9. The consumer thread pseudo code:

```
while (true)
{
  generate random integer t2 using t
  sleep for t2 seconds
  generate random integer k2 using k
  for i from 0 to (k2 – 1)
  {
    data = buffer[(next_out + i) mod n]
    if (data > 1) exit and report race condition
    buffer[(next_out + i) mod n] = 0
  }
  next_out = (next_out + k2) mod n
}
```

10. A thread which executes a call to the Thread.sleep(m) method sleeps, does nothing, for m milliseconds. Since the values t1 and t2 are the number of seconds to put a thread to sleep then that value must be multiplied by 1,000 before passing it as the parameter to the Thread.sleep method.

11. Create a driver class and make the name of the driver class **Assignment2** containing only one method:
    `public static void main(String args[]).`
    The main method receives, via the command line arguments, the values for n, k, and t, in that order. The main method creates and initializes the buffer array, then creates the producer and consumer objects, then starts the producer and consumer objects, and then executes a join on the producer and the consumer objects. Executing a join on a thread object is done by calling its join() method.

    I compile and run your program via the command line using the Java JDK. Therefore, the command I type to execute your program is **java Assignment1**. I will test your program with my own values for n, k, and t.

    For example, if n = 100, k = 22, and t = 60 then the command to run the program is:

    ```
    java Assignment1 100 22 60
    ```

12. Experiment with different values of n, k, and t until you observe a race condition. Reporting the race condition means the consumer prints the message that a race condition occurred. Your program will not terminate on its own. You will terminate it yourself when you experiment with the different values of n, k, and t. If, after a reasonable amount of time, your program does not print the race condition message, then just stop your program. If your program does print the race condition message, then just stop your program. When you find a combination of values for n, k, and t that cause a race condition, add a comment to the top of your Assignment2.java file listing those values. It will be at this point that you can submit your assignment.

13. You must declare public each class you create which means you define each class in its own file.

14. You must declare private all the data members in every class you create.

15. You can use "**implements Runnable**" or "**extends Thread**" to implement the classes defining the threads in this assignment. You cannot use extends in this assignment in defining any class except if you are using "**extends Thread**" to define a thread class.

16. **Tip:** Make your program as modular as possible, not placing all your code in one .java file. You can create as many classes as you need in addition to the classes described above. Methods being reasonably small follow the guidance that "A function does one thing and does it well." You will lose a lot of points for code readability if you don't make your program as modular as possible. But, do not go overboard on creating classes and methods. Your common sense guides your creation of classes and methods.

17. Do **NOT** use your own packages in your program. If you see the keyword **package** on the top line of any of your .java files then you created a package. Create every .java file in the **src** folder of your Eclipse project, if you're using Eclipse.

18. Do **NOT** use any graphical user interface code in your program!

19. Do **NOT** type any comments in your program except as stated in number 12 above. If you do a good job of programming by following the advice in number 16 above then it will be easy for me to determine the task of your code.

Grading Criteria

The total assignment is worth 20 points, broken down as follows:

1. If your code does not implement the task described in this assignment then the grade for the assignment is zero.
2. If your program does not compile successfully then the grade for the assignment is zero.
3. If your program produces runtime errors which prevents the grader from determining if your code works properly then the grade for the assignment is zero.

If the program compiles successfully and executes without significant runtime errors then the grade computes as follows:
Followed proper submission instructions, 4 points:

1. Was the file submitted a zip file.
2. The zip file has the correct filename.
3. The contents of the zip file are in the correct format.
4. The keyword **package** does not appear at the top of any of the .java files.

Program execution, 4 points:

- Program input, the program properly reads, processes, and uses the input.
- Program output, the program produces the correct results for the input.

Code implementation, 8 points:

- The driver file has the correct filename, **Assignment2.java** and contains only the method **main** performing the exact tasks as described in the assignment description.
- The code performs all the tasks as described in the assignment description.
- The code is free from logical errors.

Code readability, 4 points:

- Good variable, method, and class names.
- Variables, classes, and methods that have a single small purpose.
- Consistent indentation and formatting style.
- Reduction of the nesting level in code.

Late submission penalty: assignments submitted after the due date are subjected to a 2 point deduction for each day late.

Late submission policy: you **CAN** submit your assignment early, before the due date. You are given plenty of time to complete the assignment well before the due date. Therefore, I do **NOT** accept any reason for not counting late points if you decide to wait until the due date (and the last possible moment) to submit your assignment and something happens to cause you to submit your assignment late.

Submission Instructions

Go to the folder containing the .java files of your assignment and select all (and **ONLY**) the .java files which you created for the assignment in order to place them in a Zip file. The file can **NOT** be a **7z** or **rar** file! Then, follow the directions below for creating a zip file depending on the operating system running on the computer containing your assignment's .java files.

Creating a Zip file in Microsoft Windows (any version):

1. Right-click any of the selected .java files to display a pop-up menu.
2. Click on **Send to**.
3. Click on **Compressed (zipped) Folder**.
4. Rename your Zip file as described below.
5. Follow the directions below to submit your assignment.

Creating a Zip file in Mac OS X:

1. Click **File** on the menu bar.
2. Click on **Compress ? Items** where ? is the number of .java files you selected.
3. Mac OS X creates the file **Archive.zip**.
4. Rename **Archive** as described below.
5. Follow the directions below to submit your assignment.

Save the Zip file with the filename having the following format:

your last name,
followed by an underscore _,
followed by your first name,
followed by an underscore _,
followed by the word **Assignment2**.

For example, if your name is John Doe then the filename would be: **Doe_John_Assignment2**

Once you submit your assignment you will not be able to resubmit it!
Make absolutely sure the assignment you want to submit is the assignment you want graded.
There will be **NO** exceptions to this rule!

You will submit your Zip file via your CUNY Blackboard account.
The only accepted submission method!

Follow these instructions:

Log onto your CUNY BlackBoard account.
Click on the CSCI 340 course link in the list of courses you're taking this semester.
Click on **Content** in the green area on the left side of the webpage.
You will see the **Assignment 2 – Producer Consumer Assignment**.
Click on the assignment.
Upload your Zip file and then click the submit button to submit your assignment.

Due Date: Submit this assignment by Thursday, May 14, 2020.