Laboratory Exercise 3 – Consumer Producer Problem

Goal: Understand the consumer-producer problem using message passing model implementation using the Java Programming Language.

Type of Assignment : *INDIVIDUAL WORK*

Needed Files :

File Name	*File Description*
SleepUtilities	*Utilities for causing the thread to sleep. Contains 2 functions:* *1. void nap() – lets the thread nap from 0 to 5 seconds* *2. void nap(int duration) – lets the thread nap from 0 to duration seconds* *Note: No need to modify this to accomplish this lab exercise.*
Channel	*Provides an interface for the MessageQuees class (implements the message passing scheme). Contains interfaces for send(Object) and receive(Object) implementations of the MessageQueue class.*
MessageQueue	*Provides the implementation of the bounded buffer using message passing. Contains 1 constructor and 2 functions:* *1. MessageQueue() – constructor for the class which basically creates a vector to contain the elements/items produced by the Producer class* *2. send (Object item) – adds an element/item to the buffer* *3. receive(Object item) – it first returns an element/item to the caller then removes an element from the buffer*
Producer	*Provides the Producer implementation for the bounded-buffer problem. Contains 1 constructor and 1 run function:* *1. Producer(Channel m) – constructor for the class which creates a "mailbox" by instantiating a Channel class.* *2. run() – function to start the Producer thread. The elements/items created by the Producer are the current system date and time (until seconds). With the presence of the continuous while loop, nap() is being called to provide a short pause before proceeding with the succeeding instructions.*
Consumer	*Provides the Consumer implementation for the bounded-buffer problem. Contains 1 constructor and 1 run function:* *1. Consumer(Channel m) – constructor for the class which creates a "mailbox" by instantiating a Channel class.* *2. run() – function to start the Consumer thread. The elements/items consumer by the Consumer are the current system date and time (until seconds) created by the Producer. With the presence of the continuous while loop, nap() is being called to provide a short pause before proceeding with the succeeding instructions.*
Factory	*The main java file which creates the mailbox and the producer and consumer threads. Contains 1 constructor and the main(String args[]) function:* *1. Factory() – constructor for the main class which creates the mailbox and the producer and consumer threads. This is where the producer and the consumer threads are also started.* *2. main(String args[]) – main java function for the consumer-producer problem which just instantiates the Factory class.*

Required Activities:

1. Copy all required files from our LMS. Download it in the same directory in your individual terminals. Scan thru each of the files and quickly analyze how each function works. If you need help, refer to the above descriptions and to the documentation already placed in the codes.
2. Build them (using javac thru the command line or build button in your Java IDE). You can use javac *.java to build all at the same time in the command line. Make sure all java files are in the same directory.

3. Run Factory and observe how the consumer and producer work. A sample output is shown below: Theprogram will not end until it is interrupted by the user. If you are already acquainted with how the program works, then you may terminate by interrupting the program (Ctrl + C).

```
(base) ritchiemaegamot@Ritchies-MacBook-Air Consumer Producer % java Factory
Consumer wants to consume.
Consumer wants to consume.
Producer produced Wed Mar 06 15:39:07 PST 2024
Producer produced Wed Mar 06 15:39:08 PST 2024
Consumer wants to consume.
Consumer consumed Wed Mar 06 15:39:07 PST 2024
Consumer wants to consume.
Consumer consumed Wed Mar 06 15:39:08 PST 2024
Producer produced Wed Mar 06 15:39:10 PST 2024
Consumer wants to consume.
Consumer consumed Wed Mar 06 15:39:10 PST 2024
Consumer wants to consume.
Producer produced Wed Mar 06 15:39:12 PST 2024
Consumer wants to consume.
Consumer consumed Wed Mar 06 15:39:12 PST 2024
Producer produced Wed Mar 06 15:39:15 PST 2024
Producer produced Wed Mar 06 15:39:15 PST 2024
Producer produced Wed Mar 06 15:39:15 PST 2024
Consumer wants to consume.
Consumer consumed Wed Mar 06 15:39:15 PST 2024
```

4. Run Factory again.

   Task#1: Define how the consumer chooses which item to consume first.

5. Look at the Producer.java file, the presence of the while(true) control construct in the Producer java code allows it to infinitely produce many items/elements. Provide a modification to the appropriate java file(s) to get information regarding the number of elements/items currently in the buffer and print it on screen after every production. Hint: Modify MessageQueue.java and/or *Channel.java and/or Producer.java only.*

   Build the files and run Factory again.

   Task#2: What code(s) did you add or modify for each of the appropriate java files? Provide a screenshot of the code snippet to answer the question.

6. Look at the Producer.java and Consumer.java files, although the SystemUtilities class has two functions, nap(int) and nap(), only the latter is currently in use (or called) by both the Producer and Consumer classes which by default lets the producer and consumer pause from 0 to 5 seconds (look at each of their run() function). Use the nap(int) function of the utilities and allow the producer to pause longer than the consumer (i.e. consumer pauses from 0 to 5 seconds while the producer pauses from 0 to 15 seconds).

   Build the files and Run Factory again.

   Task#3: Provide a screenshot of the run result. It is up to you when to interrupt the process. See to it that you have substantial information for your screenshot. Describe the output in the screenshot.

7. Remove the restrictions you placed in the pausing time for both the producer and consumer. Use the nap() function instead of the nap(int) function (i.e. undo what you did in number 7). This step is in preparation for number 9 below.

   Build the files.

8. Look at the Producer.java file, by virtue of the while(true) construct, the producer produces infinitely many elements/items. Provide a bounded-buffer solution so that when the buffer is full, the producer will stop producing an item unless the consumer consumes it. Try to use a buffer which can accommodate up to 3 items only. (Hint: Modify the Producer.java files. You must have accomplished Number 5&6 by this time, it will be helpful to answer this number).

   Look at the Factory.java file. The Factory() constructor is the one responsible for starting both the producer and consumer threads, comment out the consumer.start() statement this time. This will assure that if your code modification was correct, it will produce only 3 items and halt producing from there.

Task#4: What code(s) did you add or modify for each of the appropriate java files? Provide a screenshot of the code snippet(s) to answer the question.

Build the files and Run Factory again.

Task#5: Provide a screenshot of the run result. It is up to you when to interrupt the process. See to it that you have substantial information for your screenshot.

9. Deliverables : PLACE ALLANSWERS FOR TASKS 1 TO 5 IN A pdf FILE. Name the file *<familyname>_conprod.pdf. SUBMIT TO THE assignment bin in CMSC 125 Canvas page.*