Concurrent and Distributed Systems

Practical 2 – Producer-Consumer Problem

This lab contains a checkpoint at the end of Task 3. Please contact any lecturing staff when you reach this point to receive your credit. You will be expected to demo the program and to show the code.

Useful and reference material:

- Lectures
- You can take inspiration from the Java code for Player and MessageQueue seen in lectures and available on canvas, BUT here
    - a thread-based implementation is required;
    - you can forget the GUI implementation (which, as discussed was implicitly providing threads) and interaction with the user;
    - each player is either a producer or a consumer.

This lab is concerned with experimenting with the Producer Consumer Problem. The problem involves two communicating entities – a Producer and a Consumer. As was outlined in lectures, Producer and Consumer communicate via a Message Passing System. The Producer creates elements and puts them into a message queue. The Consumer reads and removes elements from the message queue and displays the information on the screen. Producer and Consumer operate concurrently.

Task 1: Create a new Eclipse project in your own home directory and create a system that includes a Producer and a Consumer. Both entities should be implemented using *threads*: you will be able to launch multiple producers and multiple consumers. Also provide a message queue that is used by the producer and consumer threads for communication. The message queue should be unbounded, i.e. the queue can never be full.
Hint: the Java classes Vector and LinkedList provide such a mechanism.
Producer and Consumer should be non-blocking, i.e. regardless of the state of the message queue, calls on the queue should always return immediately (regardless of the success of the operation). As a consequence, what checks do you need to include in the consumer?

Include some “debug messages” in your program, e.g. when an element is written to or removed from the buffer, when the buffer is empty. How big does the buffer get?

Task 2: Change the message queue in a way that its capacity is restricted to 10 elements. What impact does this have on the behaviour of the producer and consumer? For the non-blocking case, how do you need to change the Producer?

Task3: Change the message queue (restricted to 10 elements) to a blocking queue, i.e. calls to the send and receive messages only return on success of the operation. Can you remove some code from the Producer and Consumer classes?

Checkpoint.

Task 4: Change the program so that you have two (or more) producers and afterwards two (or more) consumers. How many producers and consumers can run at any one time, if the queue is limited to 4, 2, and 1 elements? With multiple producers or multiple consumers, is the program guaranteed to run successfully? – Do **not** modify your program!

Task 5: Is the class Vector thread-safe? If you are not sure, check in the JDK on-line help or the Java API reference. What does this mean? How would you need to change the methods in your message queue class, if you were to implement it using a LinkedList or ArrayList? If time permits implement the message queue using one of the two classes.