Spring 2020

CS5373 – Homework3

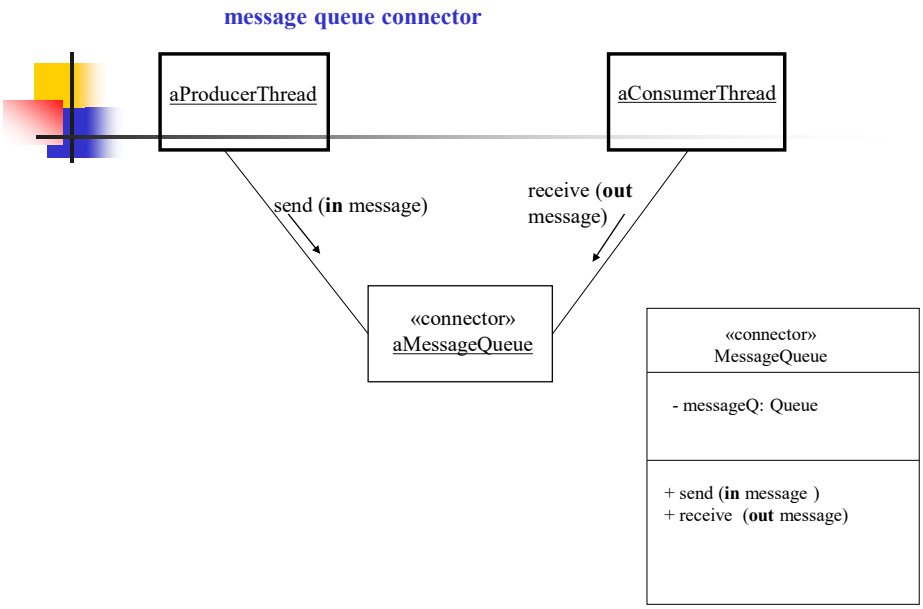
You are asked to implement a “message queue connector” class using Java. The following figure depicts a message queue connector class, followed by the detailed specifications of operations. The message queue connector class has send() and receive(), and the size of queue is 3.

To test your implementation, your program should create two separate threads, a producer thread and a consumer thread. The producer thread sends a message having a structure, (string, integer) – e.g., (“add”, 3) or (“multiply”, 7) - to a consumer thread via a message queue connector. There are AddCalculation class, which has an add() operation and MultiplyCalculation class, which has a multiply() operation. **You should implement the AddCalculation and MultiplyCalculation classes**. When the consumer thread receives a message from the connector, it extracts the message and then calls one operation on the AddCalculation or MultiplyCalculation class depending on the message. For example, the consumer thread calls the add() operation of AddCalucation class if it receives a message like (“add”, 3), while it calls the multiply() operation of MultiplyCalculation class if it receives a message like (“multiply”, 3). The add() operation adds 10 to the integer in a message from the producer thread and displays the result on the screen. The multiply() operation multiplies the integer by 10 and displays the result on the screen.

The producer thread should displays the messages on the screen before it sends the messages to the consumer thread. The messages are (“add”, 4), (“multiply”, 1), (“multiply”, 8), (“add”, 2), (“add”, 3), (“add”, 99), (“multiply”, 53) that are sent to the consumer.

The producer thread sends the consumer thread the ending message (“end”, 0) that is the last message. When the consumer thread receives the ending message, the program terminates.

Please make a zip file containing all source code file and executable files for this HW3 and submit it to blackboard.



Message queue connector

```
monitor MessageQueue
 -- Encapsulate message queue that holds max of maxCount messages
 -- Monitor operations are executed mutually exclusively;
private maxCount : Integer;
private messageCount : Integer = 0;

public send (in message)
  while messageCount = maxCount do wait;
  place message in buffer;
  Increment messageCount;
  if messageCount = 1 then signal;
end send;

public receive (out message)
  while messageCount = 0 do wait;
  remove message from buffer;
  Decrement messageCount;
  if messageCount = maxCount-1 then signal;
end receive;
end MessageQueue;
```