This program runs for 10 seconds, then the producer will put a stop message, so that the consumer knows to stop.

1. **Complete the missing pieces. The program should behave correctly, in that no messages should be lost. Make sure that the sum of all sent messages equals the sum of all received messages.**

Blocking queues provide blocking put and take methods as well as the timed

equivalents offer and poll.

“blocks” means stoppers.

Blocking queues support the producer-consumer design pattern.

A producerconsumer design separates the identification of work to be done from the execution of that work by placing work items on a “to do” list for later processing

5.3 Blocking queues

- We need to use blocking queues for this problem (from book)

- The object type used here is BlockingQueue. For some reason this gave an error. So we use ArrayBlockingQueu instead.

ArrayBlockingQueue queue = new ArrayBlockingQueue(10);

2. **Change the main method so that two consumer threads are created, both of them consuming messages from the single queue. Your program should stop gracefully, as in 1.**

After adding thread 2, the messages sent were 199 and received only 98. Something is wrong.

Problem: One consumer would get the stop message from the original given code

public void stop() {

running = false;

} in Producer.java

, but the second consumer doesn’t get the stop message, so it didn’t stop.

Solution: //Add this line, after the consumer stops, add another stop message to the queue

queue.offer(new Message("stop"));

So the first consumer thread will put the stop message again for the next consumer thread to know to stop. If there are more consumer threads, this will continue so that the next ones know when to stop.

**3. Change the main method so that, additionally to the two consumer threads of the previous point, two producer threads are created, both of them sending messages to the single queue.**

**The inputs are how many threads of consumers and how many threads of producers you want.**

Change the inputs from Configuration, application, program arguments.