Assignment – 5

Shadril Hassan Shifat (00-30135)

J2EE – BJIT Academy

---

1. **Is there any possibility of a deadlock occurring in this code? If so, under what conditions could it happen, and what could be potential solutions to avoid it?**

Answer: I think there is no possibility of a deadlock occurring in this code. Deadlock describes a condition in which two or more threads are blocked forever because they are waiting for each other.

In this code, the Producer thread enters synchronized(buffer) block first and fills the buffer to its maxSize. After reaching the buffer to its maxSize, the Producer thread notifies all the Consumer threads to consume the buffer and waits until the buffer is empty.

Also, when the Producer thread waits, the Consumer thread enters synchronized(buffer) block first and consumes the buffer until the buffer is empty. When the buffer is empty then the Consumer thread notifies all the Producer threads to fill the buffer and waits until the buffer reaches its maxSize.

In this above situation, the Producer thread waits on a condition until the buffer is empty and the consumer thread wait on a condition until the buffer is full. So, the Producer thread and Consumer thread both aren't waiting for each other at the same time. That's why there is no possibility of a deadlock occurring.

2. **The producer code limits the buffer size to maxSize, but what happens if the producer thread attempts to add an item when the buffer is already full? How is this situation handled?**

Answer: If the Producer thread attempts to add an item to the buffer when the buffer is already full, the producer thread will become blocked. The producer thread will remain in a waiting state until the Consumer thread removes all items from the buffer, freeing up space. After the buffer becomes empty, then the Producer thread can again add items to the buffer.

3. **The consumer code removes items from the buffer, but what happens if the consumer thread attempts to remove an item when the buffer is empty? How is this situation handled?**

Answer: If the Consumer thread attempts to remove an item from the buffer when the buffer is empty, the Consumer thread will become blocked. The consumer thread will remain in a waiting state until the producer thread adds item to the buffer to its maxSize. After the buffer becomes full again (to its maxSize), then the Consumer thread can remove items from the buffer.

4. **Both the producer and consumer use synchronization on the buffer object to ensure thread safety. What purpose does this synchronization serve, and why is it necessary?**

Answer: I think the synchronization on the buffer object ensures that only one thread can access the buffer at a time. This prevents the Producer thread and Consumer thread from interfering with each other. For synchronization, the Producer thread can't add an item to the buffer while the Consumer thread is removing an item from the buffer. Also, the Consumer thread can't remove items from the buffer while the Producer thread is adding items to the buffer. And that's why no deadlock occurs.

5. **The code uses buffer.notifyAll() to wake up waiting threads. Why is notifyAll() used instead of notify()? What could be the potential issues if notify() were used instead?**

Answer: The code uses buffer.notifyAll() to wake up all waiting threads. This is because there could be multiple threads waiting on the buffer. The notifyAll() method will wake up all of the waiting threads, so that they can all compete for the lock on the buffer.

If notify() were used instead, then only one of the waiting threads would be woken up. For this, only one thread would always be able to access the buffer and the other threads would never be able to access it.