Week3

생산자 소비자 문제(한정된 버퍼 문제)

- 생산자와 소비자가 공통된 자원에 접근하려는 상황에서 버퍼가 가득차거나 비어있는경우 데이터를 계속 기다리는 문제

해결방법)

1. 데이터를 버리는 방법

버퍼가 가득 찬 경우: 생산자의 데이터를 버린다.

버퍼에 데이터가 없는 경우: 소비자는 데이터를 획득할 수 없다.

2. 반복문으로 스레드 대기

스레드가 BLOCKED 상태로 대기하게 됨

3. Object클래스에서 제공하는 메소드인 wait(), notify(), notifyAll() 이용 (synchronized내에서 작동)

wait() : 락을 반납하고 WATING 하며 notify()가 올때까지 대기한다.
notify() : 대기중인 스레드 하나를 깨운다
notofyAll() : 대기중인 모든 스레드를 깨운다.

wait, notify 한계)
버퍼가 비어있는 상태에서 소비자가 먼저 실행시 다른 소비자 스레드를 깨우게 되는 비효율이 발생한다.

4. Lock Condition 사용

synchronized를 사용하지 않고 대신에 ReentrantLock 사용한다.
(임계영역이 필요한 메소드에서 lock(), unlock()을 사용한다.)

생산자, 소비자 스레드가 각각 대기하는 공간 생성
private final Condition producerCond = lock.newCondition();
private final Condition consumerCond = lock.newCondition();

생산자 -> 소비자 , 소비자 -> 생산자 를 깨우는 방식으로 비효율 문제 해결
consumerCond.await -> producerCond.signal()
producerCond.await() -> consumerCond.signal()

5. BlockingQueue

내부적으로 Lock Condition을 사용하는 큐

즉시반환, 시간대기, 대기에 대한 예외 등 사용자가 필요로 할 만한 다양한 인터페이스를 제공한다.