**용어 정리**

-임계영역(Critical Section) : 둘 이상의 스레드가 공유데이터에 동시에 접근해서 문제가 될 수 있는 코드영역

-동기화 : 스레드간 동시 접근 가능한 공유영역에 대해서 접근제어 및 접근 순서 컨트롤 해주는 작업

1) 상호배제 (Mutual Exclusion) : 둘 이상의 스레드가 임계영역에 동시에 접근하는 것을 막는 것

2) 협력 (Cooperation) : 접근 순서 컨트롤

-일반적인 동기화 처리 방식(OS) : 뮤텍스(Mutex), Semaphore(세마포어)

**-자바는?**

-동기화 코드 영역(synchronized)에 진입하기 위해 모니터락을 획득한다.



-EntrySet : 모니터락을 기다리는 스레드를 담는 자료구조

-WaitSet : 모니터가 notify 해줄 때까지 기다리는 스레드를 담는 자료구조



**스레드 상태**

package java.lang;  
public class Object {  
public final native void notify();  
public final native void notifyAll();  
public final native void wait(long timeout) throws InterruptedException;  
public final void wait(long timeout, int nanos) throws InterruptedException {  
 if (timeout < 0) {  
 throw new IllegalArgumentException("timeout value is negative");  
 }  
  
 if (nanos < 0 || nanos > 999999) {  
 throw new IllegalArgumentException(  
 "nanosecond timeout value out of range");  
 }  
  
 if (nanos >= 500000 || (nanos != 0 && timeout == 0)) {  
 timeout++;  
 }  
  
 wait(timeout);  
 }  
public final void wait() throws InterruptedException {  
 wait(0);  
 }  
}

**자바에서 동기화**

1) 상호배제 (Mutual Exclusion) : synchronized (keyword)

2) 협력 (Cooperation) : wait, notify, notifyAll (object)

**ITEM 78 공유 중인 가변 데이터는 동기화해 사용하라**

**synchronized**

1) 배타적 실행 (Mutual Exclusion) : 하나의 스레드만 접근 (객체의 일관된 상태)

2) 가시성 (Visibility) 문제 (한 스레드에서 변경한 특정 메모리 값을 다른 스레드에서 제대로 읽지 못하는 경우) 해결

**가시성 문제 원인**

**1) Java Memory Model**



**2) reoreder**

reoreder란 최적화를 위해 컴파일이나 Java 가상머신이 프로그램의 처리순서를 바꾸는 것을 말한다.

**volatile**

1) 배타적 실행 (X)

2) 가시성 문제 해결 (O)



🡺하나의 Thread에서 write 하고 나머지 Thread에서 read 할 때 적합

long stat = 324L;  
volatile int val = stat+10;

🡺Thread1은 324를 가지고 갈 수도 있고, Thread2는 334를 가지고 갈 수도 있다. (안전 실패)

🡺이럴 때는 synchronized를 사용해서 작업 자체를 원자화 시킨다.

**Atomic (넌 블로킹)**

1) 배타적 실행 (O) / lock 아님

2) 가시성 문제 해결 (O)

락 대신 저수준의 하드웨어에서 제공하는 CAS(compare-and-swap) 방식 사용

대상 메모리, 기존 값, 새로 설정할 값

public final long getAndAddLong(Object o, long offset, long delta) {  
 long v;  
 do {  
 v = getLongVolatile(o, offset);  
 } while (!weakCompareAndSetLong(o, offset, v, v + delta));  
 return v;  
}

**가장 좋은 방법?**

불변 데이터만 공유 하거나, 가변 데이터는 단일 스레드에서만 사용하자

가변 데이터를 공유 할꺼면 반드시 동기화를 해야 한다.

**ITEM79 과도한 동기화는 피하라**

과도한 동기화는 성능을 떨어뜨린다.