3. 객체 공유 – 특정객체를 명시적으로 동기화하거나, 객체 내부에 적절한 동기화 기능을 내장시키자.

3.1 가시성 – 특정 변수의 값을 가져갈 때 다른스레드가 작성한 값을 가져갈수 있다는 보장도 없고, 심지어 값을 읽지 못할수도 있다.

메모리상의 공유된 변수를 여러 스레드에서 서로 공유할 수 있게 하려면 반드시 동기화 기능을 구현해야한다.

Ex3\_1\_NoVisibility

\*\* 동기화 되지않은 상황에서, 메모리상의 변수를 대상으로 작성해둔 코드가 반드시 이런순서로 동작할것이라고 단정할수 없다.

즉, 여러스레드에서 공동으로 사용하는 변수에는 항상 적절한 동기화 기법을 적용해야한다.

3.1.1 스테일데이터 – 동기화되지않은 프로그램을 실행하며 일어나는 변수데이터 오류현상. Ex3\_1\_NoVisibility

- 프로그램이 멈추지않아, 시스템 자원을 점유 가능성.

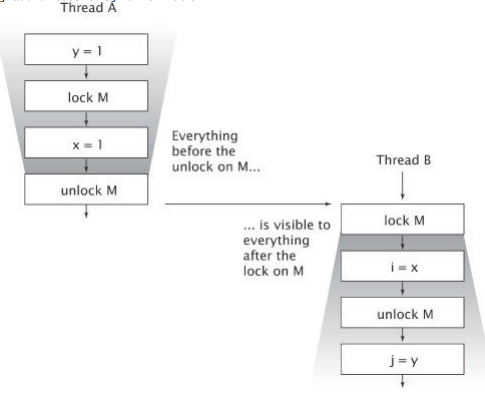
**Ex3\_2\_MutableInteger** - Get과 set의 각 스레드의 대상 객체가 다르다면.

**Ex3\_3\_SynchronizedInteger -** Get과 set의 각 스레드의 대상 객체를 동기화할 때(@GuardBy)

3.1.2 단일하지않은 64비트 연산 – long, double (64bits)데이터형태. 두번의 32비트 연산을 사용필요.

3.1.3 락과 가시성 – 내장된 락을 잘 사용해, 이전 스레드A가 해당 변수를 사용후 결과를 예측할수 잇는 상태에서

B스레드에서 계속 사용할수 있다.

쓰레드A = synchronized 블락.

쓰레드B = 같은 락을 사용하며, synchronized 블락

* 스레드 안정화.

\*\* 락은 상호 배제뿐만 아니라 정상적인 메모리 가시성을 확보하기 위해서도 사용한다. 변경 가능하면서 여러 스레드가 공유해 사용하는 변수를 각 스레드에서 각자 최신의 정상값으로 활용하려면, 동일한 락을 사용해 모두 동기화해야한다.

3.1.4 volatile(휘발성) 변수 – 컴파일러와 런타임 모두 ‘이 변수는 공유해 사용하고, 따라서 실행순서를 재배치해서는 안된다’고 이해한다.

Volatile변수는 synchronized나 락이 작용되지 않는다. 근데도 volatile변수와 synchronized블록에 진입한 변수와 비슷한 상태에 있다.

락 – 가시성과 연산의 단일성 보장. volatile변수 – 연산의 단일성은 보장못하고 가시성만 보장된다.

3.2 공개와 유출

Ex3\_5\_Secrets

특정 객체를 현재 코드의 스코프 범위 밖에서 사용할수 있도록 만들면 ‘공개상태’라고 한다.

Ex3\_6\_UnsafeStates

클래스 내부의 상태변수를 외부에 공개하거나, 객체가 안정적이지 않은 상태에서 의도적으로 공개하지 않았지만,

외부에서 사용할 수 있게 공개된 경우가 ‘유출상태’라고 한다.

Ex3\_7\_ThisEscape

객체를 공개하면 모든 변수 속성에 연결되어있는 객체 내부의 변수나 메소드를 불러오는 객체를 함께 공개된다.

3.2.1. 생성 메소드 안정성

Ex3\_7\_ThisEscape의 이슈를 방지하려면, 생성 메소드를 실행하는 도중에는 this 변수가 외부에 유출되지 않게 해야한다.