객체 공유

Date. 1-11 No.

메모리 가시성. (Memory visibility.)

Synchronized를 잘 사용하자.
명시적 동기화 // 적절한 동기화 기능 내장.
메모리에 공유된 변수는 동기화 기능을 구현해야 한다.
적절한 동기화 기법.

적재적소에 Volatile을 활용하자.
Getter, Setter 모두 동기화가 필요함. Getter에서 읽지 못할수 있기 때문.
Volatile - synchronized 보다 약한 변수.

캡슐화의 중요성.
락이나 동기화를 사용하지 않음
Volatile 변수를 읽으면 synchronized 블록에 진입할 때와 비슷한 상태에 해당한다. 하지만 synchronized 보다 약한 효과.

Thread Local의 활용.
객체 내부를 캡슐화 해야한다. 예상치 못한 동작을 예방한다.

final을 잘쓰자.
생성자 스레드 조심할것.
스택한정. 지역변수.

Java Concurrency Library를 잘 활용하면 동기화 처리를 잘할수 있음.
ThreadLocal의 활용. 잘써야한다. 전역변수가 아니지만 전역변수처럼 쓰는 일이 때문에 프로그램 전체 영향이 있다. 잘알고 써야한다.
'불변 객체는 언제라도 스레드에 안전하다.' Effective 부터 스레드 안전. 상태 '객체'가 불변 '참조'가 불변.
* final 키워드는 초기화 안전성을 보장한다. (적절하게 사용하면)
그래서 동기화 작업 없이 불변객체를 자유롭게 사용하고 공유할수 있다.
private 처럼 final 사용하기.
자바에서 제공하는 컨커런시 라이브러리를 잘 사용하기.
Static 변수와 new 연산자. 안전한 객체 공개 방식

스레드 한정. 읽기 전용 객체를 공유. 스레드에 안전한 객체를 활용.
동기화 방법 적용.

Synchronized. Volatile. 캡슐화.
ThreadLocal. final. Java Concurrency library 동기화 처리를 잘하자.