Singleton Pattern

인스턴스 하나만 만들기

Architecture, Modeling, Design Pattern, Creational

Double-Checking Locking

```
public class Singleton {
    private volatile static Singleton instance;

    private Singleton() {
    }

    public static Singleton getInstance() {
        if (instance == null) {
            synchronized (Singleton.class) {
                if (instance == null) {
                    instance = new Singleton();
                }
            }
        }

        return instance;
    }
}
```

Initialization on demand holder idiom

※ java.lang.ExceptionInInitializerError 발생 주의.

```
public class Singleton {
    private Singleton(){
    }
    private static class SingletonHolder{
        static final Singleton instance = new Singleton();
    }
    public static Singleton getInstatnce(){
        return SingletonHolder.instance;
    }
}
```

Enum initialization

Effective Java에서 소개된 방법.

- Thread-safe
- 단 한번의 인스턴스 생성 보장.
- 사용 간편
- enum value는 전역에서 접근 가능

```
public enum Singleton {
    INSTANCE;
    public static Singleton getInstance() {
        return INSTANCE;
    }
}
```