아이템 3. private 생성자나 열거 타입으로 싱글톤임을 보증하라

생성일	@2021년 8월 19일 오후 6:32
태그	

아이템 3. private 생성자나 열거 타입으로 싱글톤임을 보증하라

- 클래스를 싱글톤으로 만들면 이를 사용하는 클라이언트 테스트가 어려워 진다. (mock으로 대체 할 수 없어서..!, 인터페이스를 구현해서 만든 싱글톤이 아니라는 전제)
- 싱글톤 만드는 방법#1

```
//public static final 필드 방식
public class Elvis {
  public static final Elvis INSTANCE = new Elvis();
  private Elvis() {...}
}
```

  - private 생성자는 Elvis.INSTANCE 를 초기화 할 때 딱 한번만 호출 된다.
  - 해당 클레스가 싱글톤임이 API 명백히 드러남. (static 필드가 final)
  - 간결함
- 싱글톤 만드는 방법#2

```
//정적 팩토리 방식

public class Elvis {
  private static final Elvis INSTANCE = new Elvis();
  private Elvis() {...}
  public static Elvis getInstance() {return INSTANCE;}
}
```

- 싱글톤 만드는 방법#3

```
//열거 타입 방식

public enum Elvis {
  INSTANCE;
}
```

- 원소가 하나뿐인 열거 타입이 싱글톤을 만드는 가장 좋은 방법이다.
- 단, 싱글톤이 Enum 외의 클래스를 상속해야 한다면 위 방법은 사용할 수 없다.

- LazyHolder

```
//책에는 나와있지 않지만 많이 쓰이는 방식

public class Elvis{
  private Elvis(){}
  public static Elvis getInstance() { return LazyHolder.INSTANCE; }

  public static class LazyHolder {
      private static final Elvis INSTANCE = new Elvis();
  }
}
```

- Enum 방식을 안전하다고 제안하고 있지만 Android 같이 Context 의존성이 있는 환경일 경우, Singleton의 초기화 과정에 Context라는 의존성이 끼어들 가능성이 있다고 한다.

  `LazyHolder` 는 그에 대한 대안으로 나온 방법이다. JVM에게 객체의 초기화를 맡기는 방식으로, 멀티스레드 환경에서도 객체의 단일성을 보장할 수 있다.