

# Item 83

지연 초기화는 신중히 사용하라

# 목차

1

지연 초기화란

2

팁 1~3: about 지연 초기화

3

팁 4: 지연 초기화 홀더 클래스

4

팁 5: 이중검사 & 단일검사

5

지연 초기화 필요 경우 & 결론

# 목차

1

**지연 초기화란**

2

**팁 1~3: about 지연 초기화**

3

**팁 4: 지연 초기화 홀더 클래스**

4

**팁 5: 이중검사 & 단일검사**

5

**지연 초기화 필요 경우 & 결론**

# 지연 초기화란?

**지연 초기화**

**필드의 초기화 시점**

**값 처음 필요할 때까지 늦추는 기법**

**-> 값 사용하지 않으면 초기화도 X**

# 지연 초기화란?

## 지연 초기화 용도

- 성능 최적화
- 초기화 시 발생하는 순환 문제 해결

# 지연 초기화란?



```
public class A {  
    static B b = new B();  
}  
  
public class B {  
    static A a = new A();  
}
```

# 목차

1

지연 초기화란

2

팁 1~3: **about** 지연 초기화

3

팁 4: 지연 초기화 홀더 클래스

4

팁 5: 이중검사 & 단일검사

5

지연 초기화 필요 경우 & 결론

# 팁 1: 필요할 때까지 하지 말자

**지연 초기화 필요할 때까지 하지 말자**

**장점 : 클래스 or 인스턴스 생성 시의 초기화 비용 줄어듦**

**단점 : 지연 초기화하는 필드 접근 비용 커짐**



# 팁 1: 필요할 때까지 하지 말자

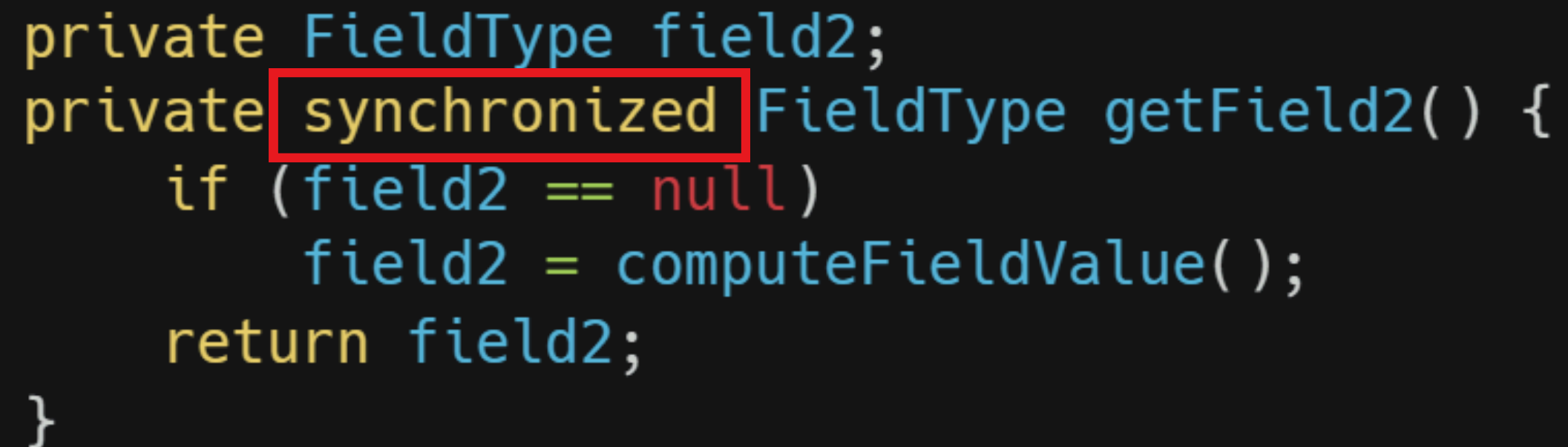
**지연 초기화 필요할 때까지 하지 말자**

**여러 고려 사항에 따라 성능 느려질 수 있음**

**대부분의 상황에서 일반적인 초기화가 나음**

# 팁 2: 초기화 순환성 해결엔 `synchronized`

자연 초기화로 초기화 순환성 해결 -> `synchronized` 사용



```
private FieldType field2;  
private synchronized FieldType getField2() {  
    if (field2 == null)  
        field2 = computeFieldValue();  
    return field2;  
}
```

# 팁 3: 멀티 스레드 환경에서 지연 초기화

**멀티 스레드 환경 -> 지연 초기화 까다로움**

**멀티 스레드 환경에서  
지연 초기화 하는 필드 스레드가 공유**

**-> 반드시 동기화 필요**

# 목차

1

지연 초기화란

2

팁 1~3: about 지연 초기화

3

**팁 4: 지연 초기화 홀더 클래스**

4

팁 5: 이중검사 & 단일검사

5

지연 초기화 필요 경우 & 결론

# 팁 4: 지연 초기화 홀더 클래스

성능 때문에 정적 필드 지연 초기화  
-> 지연 초기화 홀더 클래스 사용

# 팁 4: 지연 초기화 홀더 클래스


## 지연 초기화 홀더 클래스

지연 초기화 구현하는 패턴

클래스 초기화 관련 특성 사용

**thread safe, 객체 초기화 성능 동시 확보 가능**

## 팁 4: 지연 초기화 홀더 클래스



```
private static class FieldHolder {  
    static final FieldType field = computeFieldValue();  
}
```

3

```
private static FieldType getField() {  
    return FieldHolder.field;  
}
```

1

2

# 팁 4: 지연 초기화 홀더 클래스

JVM **클래스 초기화 시에만 필드 접근 동기화**  
= 클래스 초기화 후 동기화 X

-> **초기화 이후 필드 접근 시 동기화 필요 X**

-> 성능 느려지는 것 없음



# 팁 4: 지연 초기화 홀더 클래스

정적 필드 지연 초기화

-> 지연 초기화 홀더 클래스 사용

# 목차

1

지연 초기화란

2

팁 1~3: about 지연 초기화

3

팁 4: 지연 초기화 홀더 클래스

4

**팁 5: 이중검사 & 단일검사**

5

지연 초기화 필요 경우 & 결론

# 팁 5: 이중검사 & 단일검사

성능 때문에 **인스턴스 필드 지연 초기화**  
-> **이중검사** 사용

# 팁 5: 이중검사 & 단일검사

## 이중검사

멀티 스레드 환경 지연 초기화

-> 불필요한 동기화 비용 발생

멀티 스레드 환경에서

객체 안전 초기화 & 성능 최적화 위한

설계 기법

# 팁 5: 이중검사 & 단일검사

## 이중검사

**두번 확인**해 동기화 이용 최소화 & thread safe

첫번째 확인 : 동기화 없이 검사

두번째 확인 : 필드가 아직 초기화 되지 않았다면 동기화 해 검사

# 팁 5: 이중검사 & 단일검사

```
private volatile FieldType field4;

private FieldType getField4() {
    FieldType result = field4;
    if (result != null) // 첫 번째 검사 (락 사용 안 함)
        return result;

    synchronized(this) {
        if (field4 == null) // 두 번째 검사 (락 사용)
            field4 = computeFieldValue();
        return field4;
    }
}
```

# 팁 5: 이중검사 & 단일검사

## 단일검사

가끔 반복해서 초기화해도 상관없는  
인스턴스 필드 지연 초기화

-> 단일검사

# 팁 5: 이중검사 & 단일검사

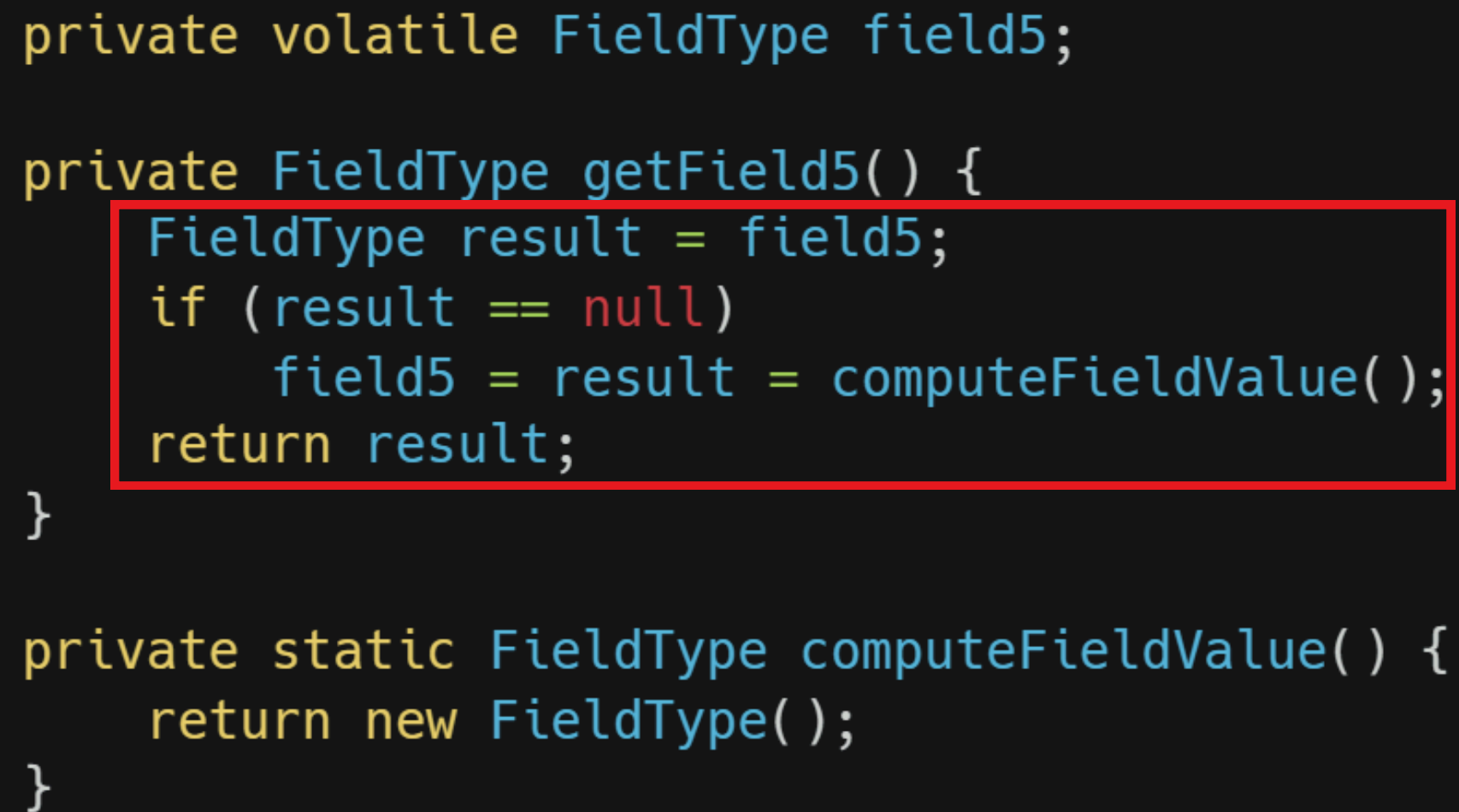
**단일검사**

이중검사에서 **두 번째 검사 생략**한 것

**이중검사의 변형**



# 팁 5: 이중검사 & 단일검사



```
private volatile FieldType field5;

private FieldType getField5() {
    FieldType result = field5;
    if (result == null)
        field5 = result = computeFieldValue();
    return result;
}

private static FieldType computeFieldValue() {
    return new FieldType();
}
```

# 목차

1

지연 초기화란

2

팁 1~3: about 지연 초기화

3

팁 4: 지연 초기화 홀더 클래스

4

팁 5: 이중검사 & 단일검사

5

지연 초기화 필요 경우 & 결론

# 지연 초기화 필요 경우

지연 초기화 필요할 때까지 X  
대부분 상황에서 일반적인 초기화 O

but 지연 초기화 필요할 때 있음

# 지연 초기화 필요 경우

1. 필드를 사용하는 인스턴스의 비율이 낮다.
2. 그 필드를 초기화하는 비용이 크다.

# 결론

대부분 필드는 지연 초기화 X, **바로 초기화**

지연 초기화 **사용 시 올바른 지연 초기화 기법 사용**

정적 필드 - 지연 초기화 홀더 클래스

인스턴스 필드 - 이중검사

반복 초기화 해도 괜찮은 인스턴스 필드 - 단일검사