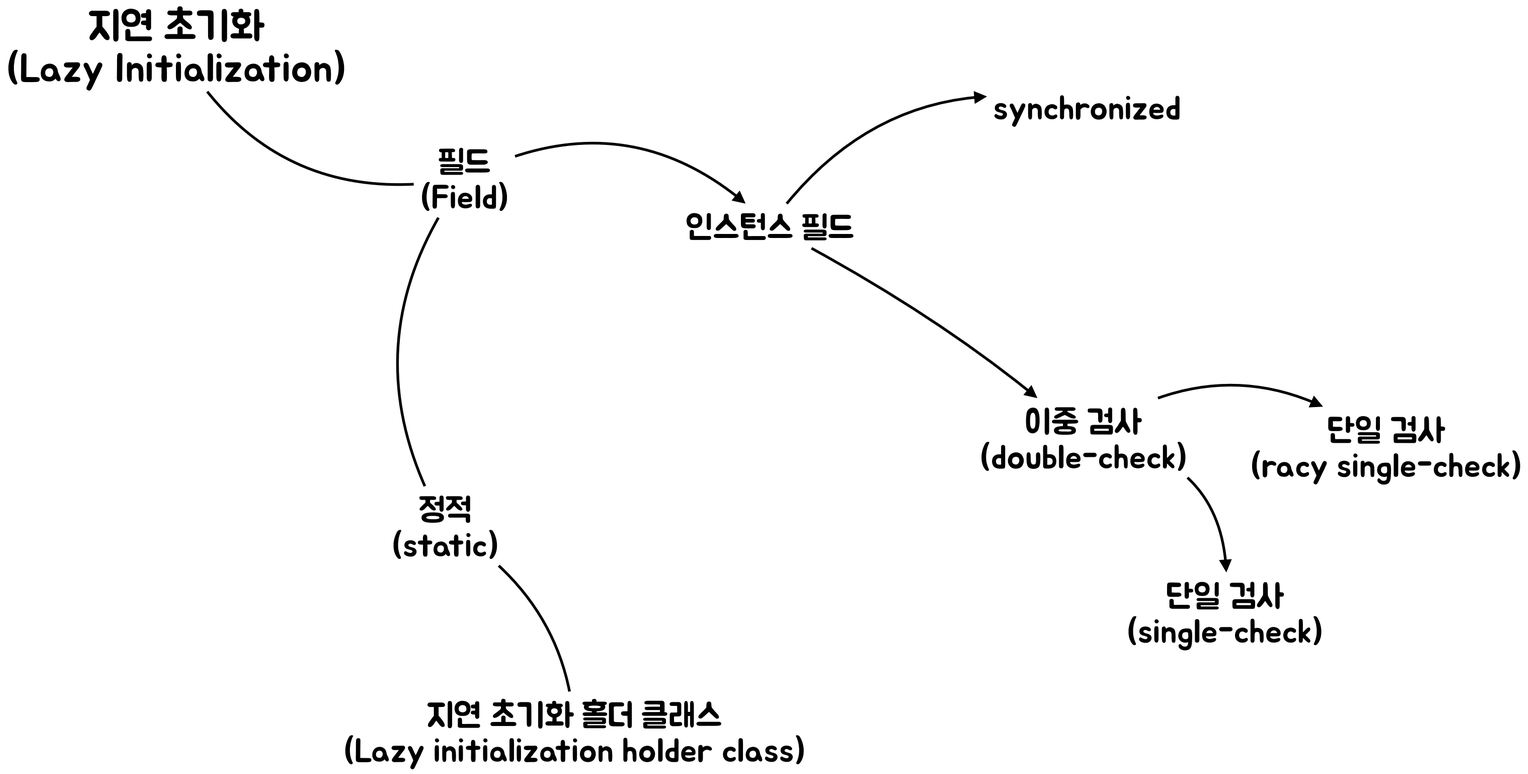
아이템 83
'지연 초기화는 신중히 사용하라'

'지연 초기화는 신중히 사용하라'

참고

- Effective Java 3/E - item 83
- Fundamental of JVM and Class Loader in java - Java JVM과 Class Loader의 동작 과정 이해

'지연 초기화는 신중히 사용하라'



- 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항

2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구
       (double-check)

3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구

4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구
       (single-check)
    2. 짜릿한 단일 검사 관용구
       (racy single-check)

지연 초기화 (Lazy Initialization)

* 지연 초기화

- 클래스 or 인스턴스 생성 시 '초기화 비용' 감소
- 지연 초기화하는 '필드에 접근하는 비용' 증가

* 지연 초기화의 성능에 영향을 주는 요소들

- 지연 초기화하려는 필드들 중 초기화가 이루어지는 비율
- 지연 초기화하는 필드에 대해 실제 초기화에 드는 비용
- 초기화 된 필드를 얼마나 빈번하게 호출 하는지

* 지연 초기화가 필요한 경우

- 클래스의 인스턴스 중에 '필드를 사용하는 인스턴스의 비율'이 낮은 반면, 그 '필드를 초기화하는 비용'이 큰 경우 사용

```
class Example {

    // field
    private final FieldType field;

    // constructor

    // setter, getter
}
```

인스턴스를 생성하여 필드를 초기화 하는 비용 ⬆

```
Example example = new Example('field');

example.getField();
example.getField();
...
```

필드를 사용하는 인스턴스의 비율 ⬇

▸ 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항
2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구 (double-check)
3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구
4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구 (single-check)
    2. 짜릿한 단일 검사 관용구 (racy single-check)

인스턴스 필드 일반적인 초기화 방법과 지연 초기화 방법

* 일반적인 필드를 선언할 때의 일반적인 초기화 방법

- 대부분의 상황에서 일반적인 초기화가 지연 초기화보다 좋다.

```
class Example {
    // final 한정자를 통한 인스턴스 필드 생성
    private final FieldType field = computeFieldValue();
}
```

* 인스턴스 필드의 지연 초기화
(synchronized 접근자를 통한 방식)

- 초기화 순환성(initialization circularity)가 깨지는 경우 사용

```
class Example {
    private final FieldType field;

    private synchronized FieldType getField() {
        if (field == null) {
            field = computeFieldValue();
        }
        return field;
    }
}
```

‣ 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항
2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구
    (double-check)
3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구
4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구
    (single-check)
    2. 짜릿한 단일 검사 관용구
    (racy single-check)

인스턴스 필드 일반적인 초기화 방법과 지연 초기화 방법

* 인스턴스 필드의 이중 검사 관용구를 이용한 지연 초기화 (double-check)

- 초기화된 필드에 대한 동기화 비용을 없애준다.

```
class Example {
    private volatile FieldType field;

    private FieldType getField() {
        FieldType result = field; // 초기화 시 한 번만 읽도록 하기 위함
        if (result != null) {
            return result;
        }

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

필드가 초기화된 후로는 동기화하지 않으므로 해당 필드는 반드시 volatile 한정자로 선언해야 한다.

첫 번째 읽을 때는 동기화 없이 검사

두 번째 읽을 때는 동기화 하여 검사

필드의 초기화가 되지 않은 경우 필드 초기화

▸ 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항

2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구 (double-check)

3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구

4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구 (single-check)
    2. 짜릿한 단일 검사 관용구 (racy single-check)

인스턴스 필드 일반적인 초기화 방법과 지연 초기화 방법

* 인스턴스 필드의 이중 검사 관용구를 이용한 지연 초기화 (double-check)

- 초기화된 필드에 대한 동기화 비용을 없애준다.

```
class Example {
    private volatile FieldType field;

    private FieldType getField() {
        FieldType result = field; // 초기화 시 한 번만 읽도록 하기 위함
        if (result != null) {
            return result;
        }

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

result라는 지역변수를 통해 필드가 이미 초기화된 상황인 경우 그 필드를 딱 한번만 읽도록 보장하는 역할을 한다.

필수는 아니지만 성능을 높여주고, 저수준 동시성 프로그래밍에 표준적으로 적용되는 방법이다. ★

'정적 필드'에 대한 경우에도 이중 검사 관용구를 적용할 수 있지만 지연 초기화 홀더 클래스 방식이 더 좋다.

▸ 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항
2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구 (double-check)
3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구
4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구 (single-check)
    2. 짜릿한 단일 검사 관용구 (racy single-check)

정적 필드 지연 초기화 방법

* 정적 필드용 지연 초기화 홀더 클래스
(Lazy Initialization Holder Class)

- 성능면에서 정적 필드의 지연 초기화가 필요한 경우 사용
- 클래스는 '클래스가 가장 처음 쓰일 때 비로소 초기화된다는 특성'을 이용

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

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

이때 FieldHolder 클래스 초기화를 수행한다.

getField() 처음 호출하는 순간 FieldHolder.field 를 처음 읽는다.

Example example = new Example();

example.getField();

- 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항
2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구 (double-check)
3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구
4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구 (single-check)
    2. 짜릿한 단일 검사 관용구 (racy single-check)

정적 필드 지연 초기화 방법

* 정적 필드용 지연 초기화 홀더 클래스
(Lazy Initialization Holder Class)

- 성능면에서 정적 필드의 지연 초기화가 필요한 경우 사용
- 클래스는 '클래스가 가장 처음 쓰일 때 비로소 초기화된다는 특성'을 이용

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

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

필드에 접근하면서 동기화를 하지 않아서
성능이 느려질 이유가 없다.

getField() 처음 호출하는 순간 FieldHolder.field 를 처음 읽는다.

Example example = new Example();

example.getField();

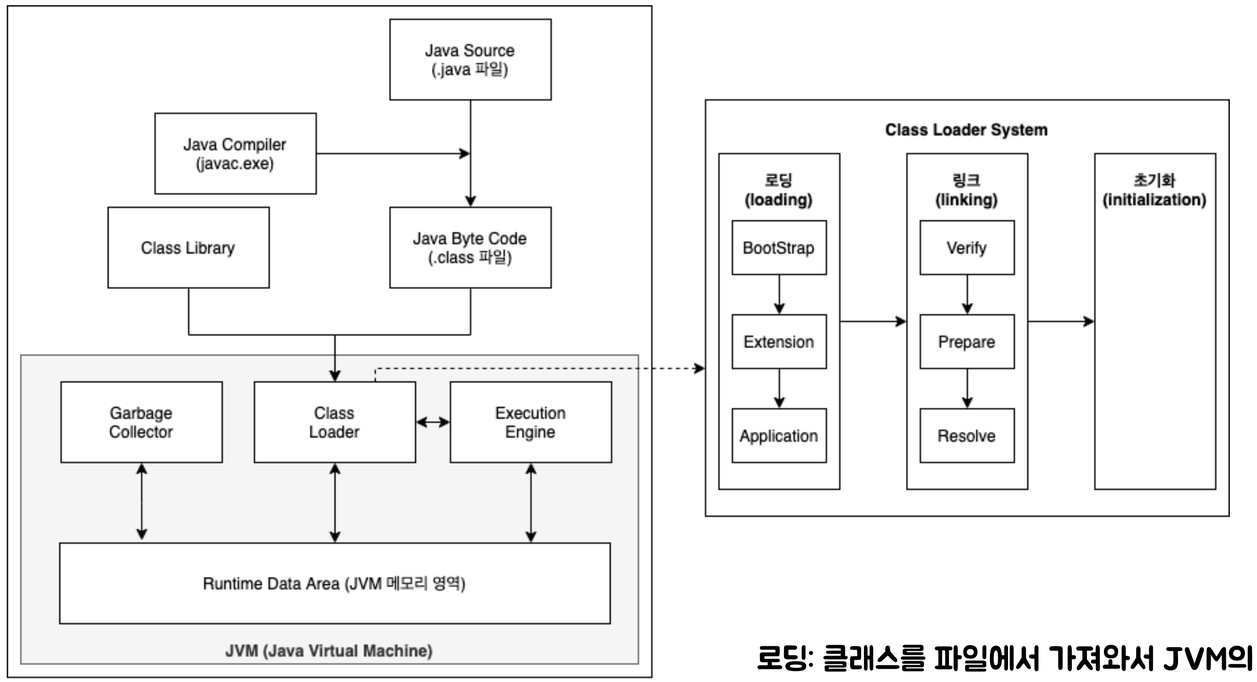
- 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항
2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구 (double-check)
3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구
4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구 (single-check)
    2. 짜릿한 단일 검사 관용구 (racy single-check)

성능이 떨어질 이유가 없다는 말이 무슨 말?

* VM은 오직 클래스를 초기화할 때만 필드에 대한 접근을 동기화한다.

- 클래스 초기화가 끝난 후에는 VM이 동기화 코드를 제거 -> 그 이후로는 검사나 동기화 없이 필드에 접근하게 된다.



로딩: 클래스를 파일에서 가져와서 JVM의 메모리에 로드한다.

링크: 레퍼런스를 연결

초기화: static 한 값들을 초기화 한다.

- 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항

2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구
    (double-check)

3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구

4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구
    (single-check)
    2. 짜릿한 단일 검사 관용구
    (racy single-check)

이중 검사의 2가지 특이 케이스

* 인스턴스 필드의 단일 검사 관용구
(single-check)

- 반복해서 사용해도 상관없는 인스턴스 필드를 지연 초기화 해야 하는 경우,
이중 검사에서 두 번째 검사를 생략할 수 있다.

```
class Example {
    private volatile FieldType field;

    private FieldType getField() {
        FieldType result = field;

        if(result == null) {
            field = result = computeFieldValue();
        }
        return result;
    }
}
```

인스턴스 필드를 volatile로 선언
초기화가 중복해서 일어날 수 있다.

result라는 지역변수를 통해 필드가 이미 초기화된 상황인 경우
그 필드를 딱 한번만 읽도록 보장하는 역할을 한다.

필드의 초기화가 되지 않은 경우 필드 초기화

이중 검사의 2가지 특이 케이스

* 인스턴스 필드의 짜릿한 단일검사
  (racy single-check)

- 모든 스레드가 필드의 값을 다시 계산해도 되고, 필드의 타입이 long과 double을 제외한 수치 기본 타입인 경우

  단일 검사의 필드 선언에서 volatile 한정자를 없애도 된다.

- 어떤 환경에서는 필드 접근 속도를 높여주지만, 초기화가 스레드당 최대 한 번 더 이루어질 수 있다.

- 이는 이례적인 기법으로, 보통은 쓰이지 않는다.

정리

정리

- 지금까지 언급한 모든 초기화 기법은 기본 타입 필드와 객체 참조 필드에 모두 적용이 가능하다.
- 이중검사와 단일검사 관용구를 수치 기본 타입 필드에 적용한다면 필드의 값을 null 대신 0과 비교하면 된다.
- 대부분의 필드는 지연시키지 말고 곧바로 초기화 해야 한다.
- 성능 때문에 혹은 위험한 초기화 순환을 막기 위해 꼭 지연 초기화를 써야 하는 경우 올바른 지연 초기화 기법을 사용한다.
- 인스턴스 필드에는 이중검사 관용구, 정적 필드에는 지연 초기화 홀더 클래스 관용구를 사용한다.
- 반복해 초기화해도 괜찮은 인스턴스 필드에는 단일 검사 관용구도 고려 대상이다.

▸ 지연 초기화는 신중히 사용하라

1. 지연 초기화
    1. 지연 초기화가 필요한 경우
    2. 지연 초기화의 주의 사항

2. 인스턴스 필드의 지연 초기화 방법
    1. synchronized
    2. 이중 검사 관용구 (double-check)

3. 정적 필드의 지연 초기화 방법
    1. 지연 초기화 홀더 클래스 관용구

4. 이중 검사의 특이 케이스 2가지
    1. 단일 검사 관용구 (single-check)
    2. 짜릿한 단일 검사 관용구 (racy single-check)