



# Singleton Pattern

구본아 김예지 이성호 이충무

# CONTENTS

- 1 Singleton Pattern 이란?  
✓ Singleton pattern 정의, 용도
- 2 Singleton Pattern 구현  
✓ 클래스 다이어그램, 구현방법
- 3 문제점  
✓ 멀티 스레드 상황에서 발생하는 문제점
- 4 보완 구현방법  
✓ 멀티 스레드 환경을 고려한 구현 방법

# 1 Singleton Pattern 이란?

✓ Singleton pattern 정의, 용도

## 1) Singleton Pattern 이란?

해당클래스의 인스턴스가 하나만 만들어지고,  
어디에서나 그 인스턴스에 접근할 수 있도록 하기 위한 패턴

## 2) Singleton Pattern 용도

- 특정 클래스에 하나의 인스턴스만 필요할 때

Ex) 로그 찍는 객체, 프린터 드라이버

## 2 Singleton Pattern 구현

✓ 클래스 다이어그램, 구현방법

### 1) Class Diagram

SingletonClass
Static instance  // ...
Static getInstance()  // ...

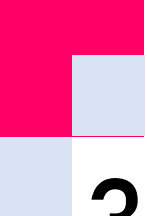
## 2 Singleton Pattern 구현

✓ 클래스 다이어그램, 구현방법

### 2) 구현방법 : Lazy initialization

```
public class Singleton {  
    private static Singleton instance;  
  
    private Singleton() {}  
  
    public static Singleton getInstance() {  
        if(instance == null) {  
            instance = new Singleton();  
        }  
        return instance;  
    }  
}
```

인스턴스 요청  
Singleton.getInstance();

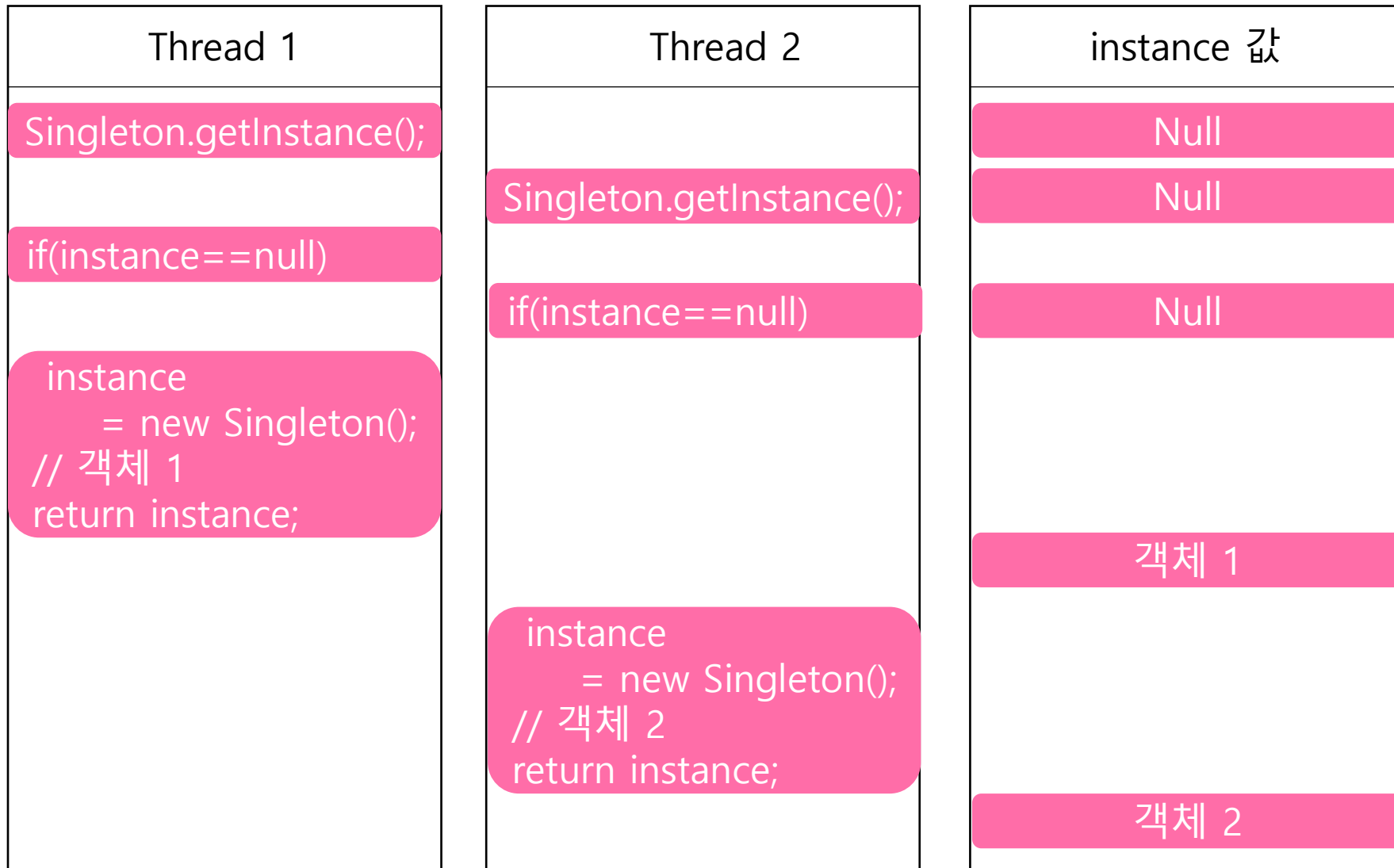


## 3 문제점

✓ 멀티 스레드 상황에서 발생하는 문제점

---

### 1) 멀티 스레드



## 4 보완 구현방법

✓ 멀티 스레드 환경을 고려한 구현 방법

### 1) Synchronized

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



## 4

## 보완 구현방법

✓ 멀티 스레드 환경을 고려한 구현 방법

## 2) Eager Initialization

```
public class Singleton {  
    private static Singleton instance = new Singleton();  
  
    private Singleton() {}  
  
    public static Singleton getInstance() {  
        return instance;  
    }  
}
```

## 4 보완 구현방법

✓ 멀티 스레드 환경을 고려한 구현 방법

### 3) Double-Checking Locking

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



---

Thank You  
for watching

