Interface in JAVA

박찬준

PANGTUDY

01

Interface?

Interface?

```java
@Repository
public interface AssessmentResultRepository extends JpaRepository<AssessmentResult, Integer> {
    List<AssessmentResult> findAssessmentResultsByResult(String result);
    List<AssessmentResult> findAssessmentResultsByHistoryAndResult(History history, String result);
    List<AssessmentResult> findAssessmentResultsByHistoryAndResourceIdAndResult(History history, String ResourceName, String result);
    List<AssessmentResult> findAssessmentResultsByResourceIdAndResult(String ResourceId, String result);
    List<AssessmentResult> findAssessmentResultsByHistory(History history);
    List<AssessmentResult> findAssessmentResultsByHistoryAndResourceId(History history, String ResourceId);
    List<AssessmentResult> findAssessmentResultsByResourceId(String ResourceId);
    void deleteByHistory(History history);
}
```

흔하게 볼 수 있는 Interface의 예시

추상 메서드들만 포함되고 포함된 인스턴스 변수는 final static으로 인식하고 …
다양한 특징들이 있고 추상 클래스와 비교되는 여러 이야기들이 있지만
이번 발표에서는 그런 개념적인 이야기보다 좀 더 재미있는 이야기를 할까 한다.

01

Interface?

Interface?

Interface를 사용함으로써 얻을 수 있는 것들?

항상 Interface를 사용하는 것이 좋은가?

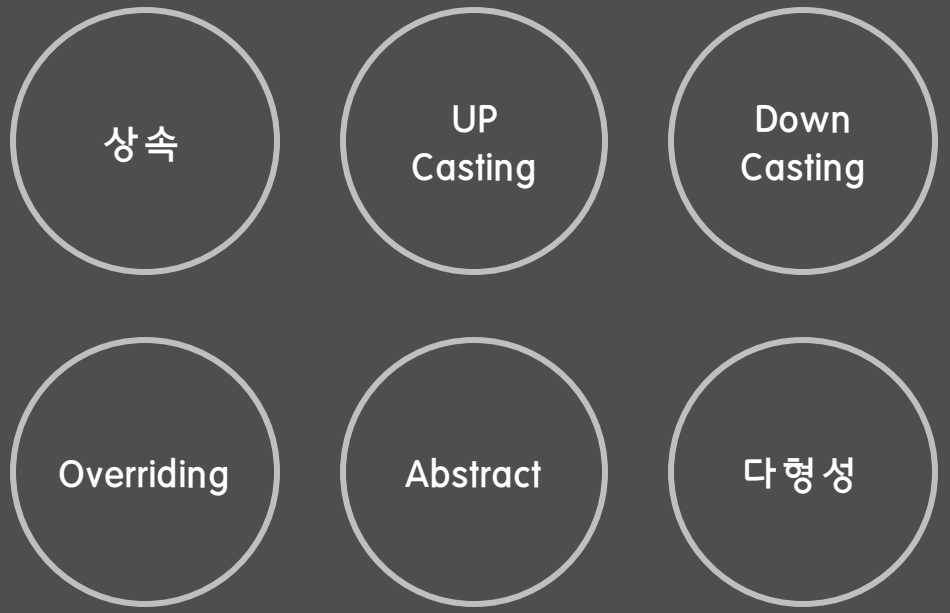
Interface는 어떤 상황에서 사용되는가?

Interface를 사용한 실제 사례엔 어떤 것들이 있을까?

02

Foundation Knowledge

Foundation Knowledge



이러한 것들을 알고 있으면
Interface를 이해하는데 도움이 된다.
물론 이외에도 많은 것들이 있지만
제 생각에 필수적인 것들

03

Interface를 사용함으로써 얻을 수 있는 것들?

협업

- 설계자와 개발자의 협업 (인터페이스는 명세서 역할)
- 특정 기능에 대한 강제성
- 표준화

다중상속

- 클래스 상속과 인터페이스 구현의 차이
- 예시) Thread 클래스와 Runnable 인터페이스

교체 용이

- A-B-C 관계를 가지는 프로그램이 있고,
  B는 부서별로 다르게 동작해야 하는 상황
- A에 B의 인스턴스 변수가 포함되어 있고,
  몇 가지 상황에 따라서 B의 클래스 타입을 변경해야 하는 상황
- 결합도↓
- 각종 디자인 패턴에서 사용되는 이유

기타

- Interface를 여러 클래스에서 구현 -> **다형성**
- 추상 메서드를 정의 -> **캡슐화**
  (인터페이스 수준에서 코드가 보이지 않음)
- 설계 관점에서 메서드의 바디 없이 인자와 메스드를 정의
  -> **추상화**

04

항상 Interface를 사용하는 것이 좋은가?

1. 다들 인터페이스를 사용하니까!
2. 인터페이스를 쓰는게 좋다고 하니까!

➡ 정확한 디자인적 관점에서 이해를 한 후 사용

05

Interface는 어떤 상황에서 사용되는가?

1. 특정 위치에서 사용되는 객체의 종류가 자주 바뀔 필요가 있을 때
2. 많은 클래스에서 상속 받는 상황에서 다중 상속을 우회하고 싶을 때
3. 반드시 구현해야 하는 메서드를 정의하거나 메서드 이름을 수정하지 못하도록 표준화하고 싶을 때
4. 완전한 요구사항이 도출되지 않은 상황에서 빠르게 개발하고 싶을 때 (애자일 프로세스)
5. 인스턴스 참조 변수가 있을 때 해당 변수가 특정 메서드를 가지고 있음을 보장받고 싶을 때
6. …

06

Interface를 사용한 실제 사례엔 어떤 것들이 있을까?

1. Thread 클래스와 Runnable 인터페이스

- Thread 클래스와 Runnable 인터페이스는 동일한 목적을 가진다.
- Runnable 인터페이스는 왜 필요할까?
- public Thread(Runnable target) 생성자가 존재하는 이유.

2. Serializable 인터페이스, Cloneable 인터페이스

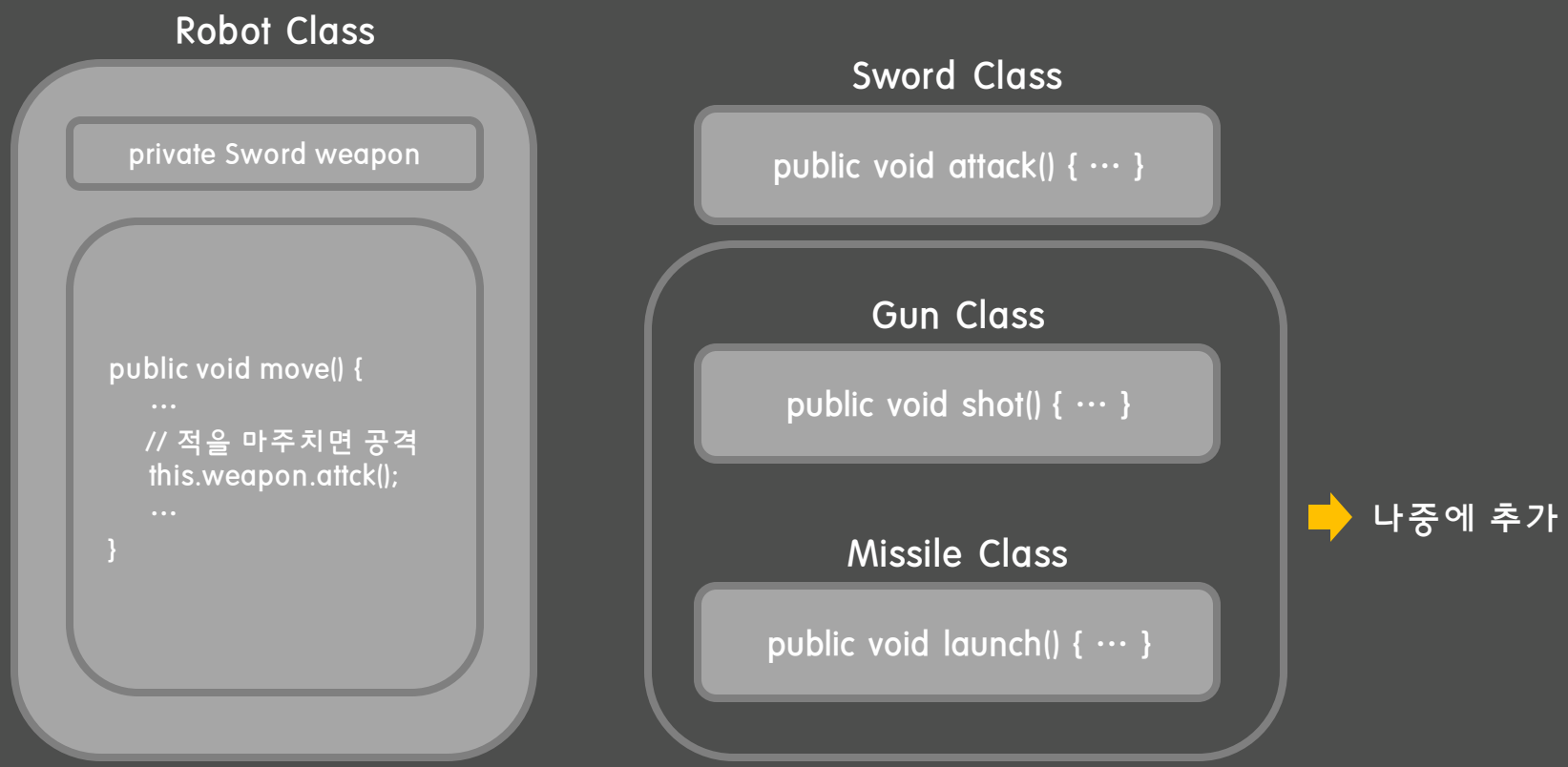
- 이 두 가지 인터페이스는 비어있다?
- 마커 인터페이스의 목적은 무엇일까.
- Cloneable 인터페이스와 Object 클래스의 clone() 메서드의 동작 방식에 대해서.

3. 다양한 Design Pattern

- Abstract Factory Pattern
- Strategy Pattern
- Observer Pattern
- 등 다양한 Design Pattern에서 인터페이스를 사용한다.

06

Interface를 사용한 실제 사례엔 어떤 것들이 있을까?



현재 Robot에는 Sword라는 무기가 장착되어 있다. 하지만 나중에 이 무기를 Gun 혹은 Missile로 변경하고 싶다.
이 과정을 위해서는 Robot 클래스를 수정해야만 한다.
(만약 Robot 클래스가 내가 만든 것이 아니라 누군가 만든 것을 사용하고 있다면?)

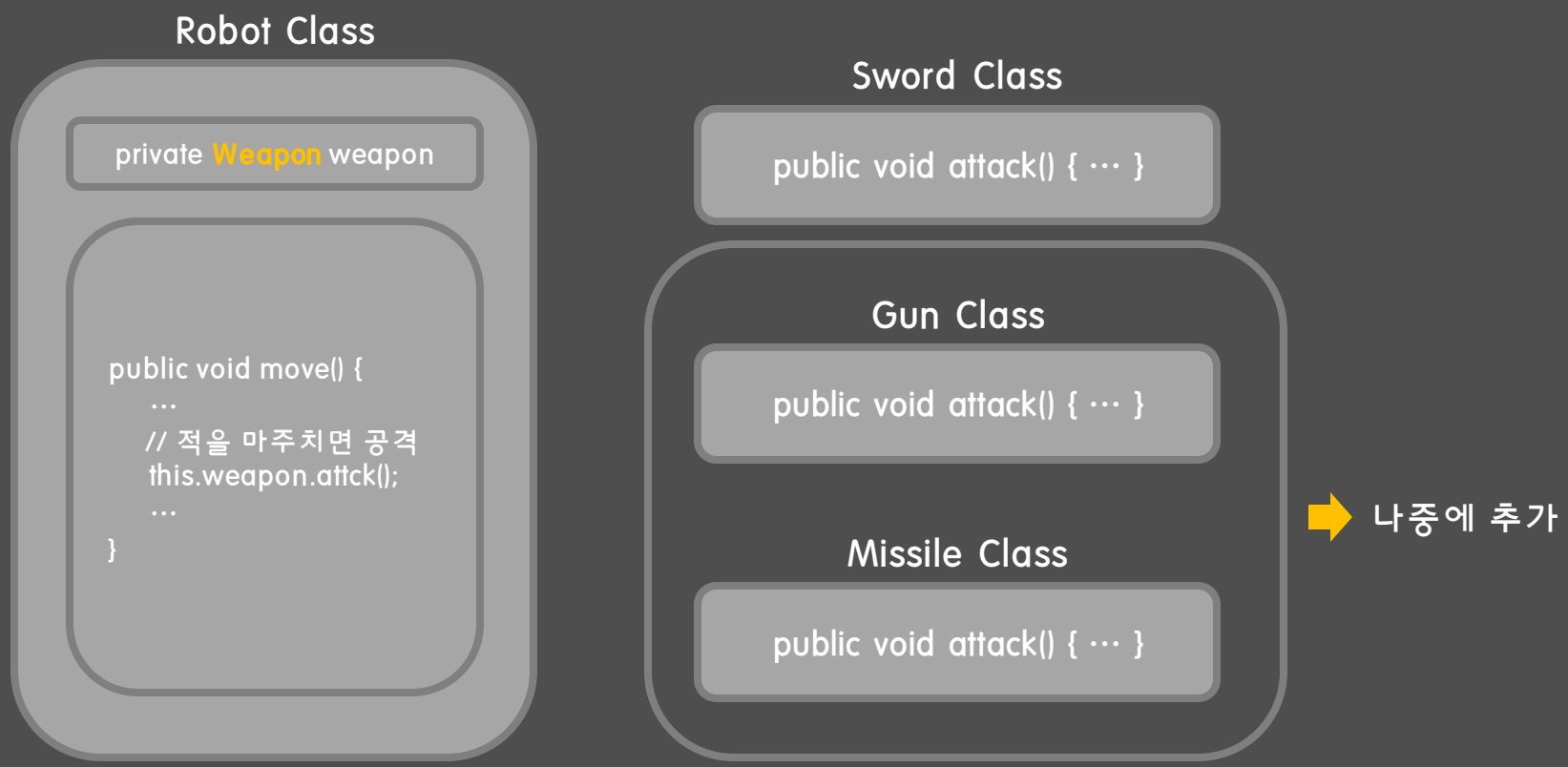
06

Interface를 사용한 실제 사례엔 어떤 것들이 있을까?

표준화 : Weapon 인터페이스를 구현한 하위 클래스는 반드시 attack() 이라는 메서드를 가지고 있음을 보장한다.
강제성 : 실수 혹은 고의로 attack() 이라는 메서드를 구현하지 않는 것을 방지할 수 있다. (컴파일 에러)
다형성 : Weapon 에 대한 참조 변수에는 Sword, Gun, Missile 심지어는 Weapon의 익명함수까지도 들어갈 수 있다.

06

Interface를 사용한 실제 사례엔 어떤 것들이 있을까?



이러한 패턴을 Strategy Pattern 이라고 한다.

감사합니다.

THANK YOU.

Ch4njun's PORTFOLIO