




# 디자인 패턴

이소영(yisy0703@naver.com)



# 패턴이란

우리의 개발자 선배님들이 객체지향 언어의  
장점들을 모아 가장 효율적으로 개발을 할 수  
있게 만들어 놓은 틀

# 패턴을 통한 OOP이해 예제부분

1

**Singleton pattern**

2

**Strategy Pattern**

# Singleton P Pattern

싱글턴 패턴(Singleton pattern)이란 어떤 클래스의 객체는 오직 하나인 유일한 객체를 만들어 여러가지 상황에서 동일한 객체에 접근하기 위해 만들어진 패턴이다. 실제로 많이 쓰이는 유형이니 꼭 이해하자

# 1. Singleton Pattern

## SingletonClass

```
private static SingletonClass INSTANCE;  
private int i=10;  
private SingletonClass() {}
```

```
Static SingletonClass getInstance(){객체가없을때만생성해서 INSTANCE로}  
Int getI(){return i;}  
void setI(int){this.i=i;}
```

## FirstClass

```
FirstClass(){SingletonClass.getSingletonClass();  
setI(999);}
```

## SecondClass

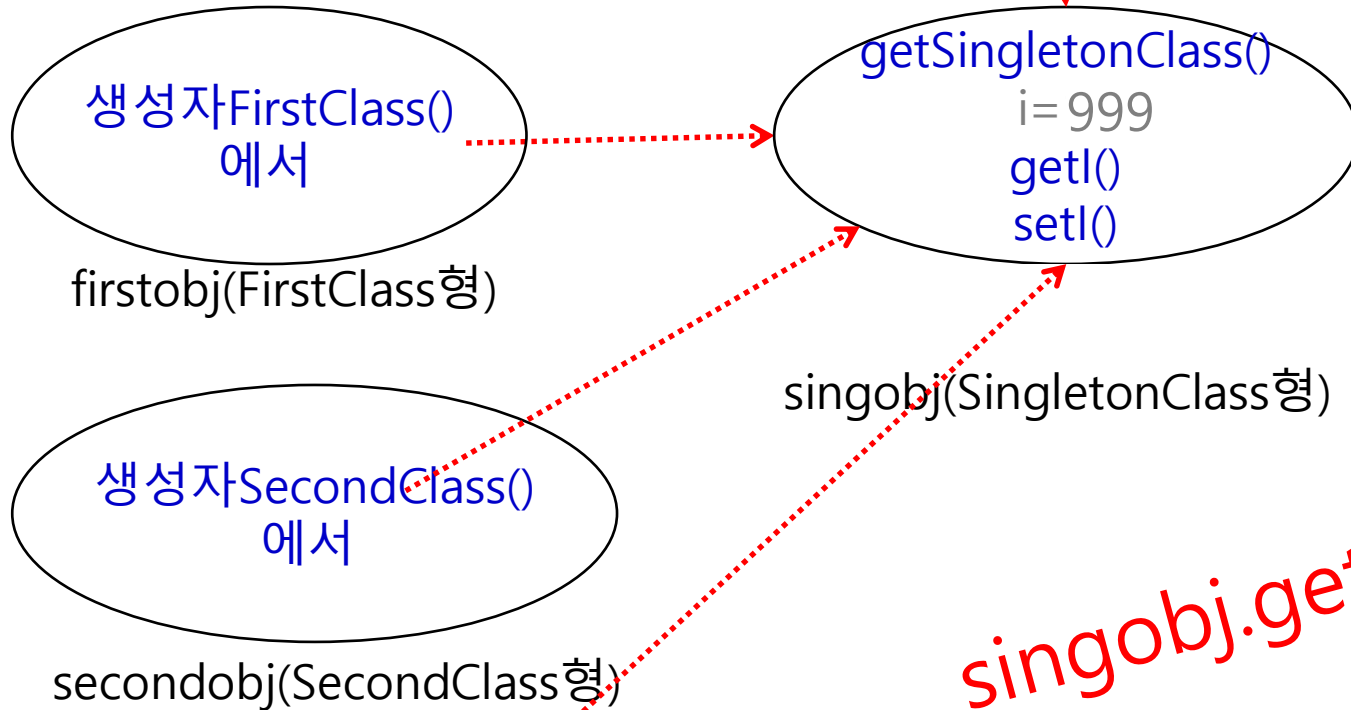
```
SecondClass(){SingletonClass.getSingletonClass();}
```

## MainSing

```
Staic void  
Main()
```

# 1-1 main()

SingletonClass.INSTANCE



singobj(SingletonClass형)

main()함수내에서도 SingletonClass형 객체 생성

singobj.getI() ?

# Strategy Pattern

알고리즘 군을 정의하고 각각의 기능을 부품처럼 캡슐화하여 교환해서 사용할 수 있도록 하는 패턴

# 시 나 리 오

- 모든 로봇은 기본적으로 걷고, 달릴 수 있어야 합니다.
- 로봇 모양은 팔, 다리, 머리, 몸통으로 이루어져 있다

	날 수 있을지	미사일 쏘지	검
SuperRobot	O	O	레이저검
StandardRobot	X	O	목검
LowRobot	X	X	없음

- 위 3가지 형태의 로봇을 만들되, 추후 다른 로봇을 만들어야 할 경우를 생각하여 유연한 프로그램을 만들어 보도록 합니다.



# 1단계. 각각 클래스 정의

## SuperRobot

Shape(){}  
actionWalk(){  
    걸을수있다}  
actionRun(){  
    뛸수있다}  
actionFly(){O}  
actionMissile(){O}  
actionKnife(){L}

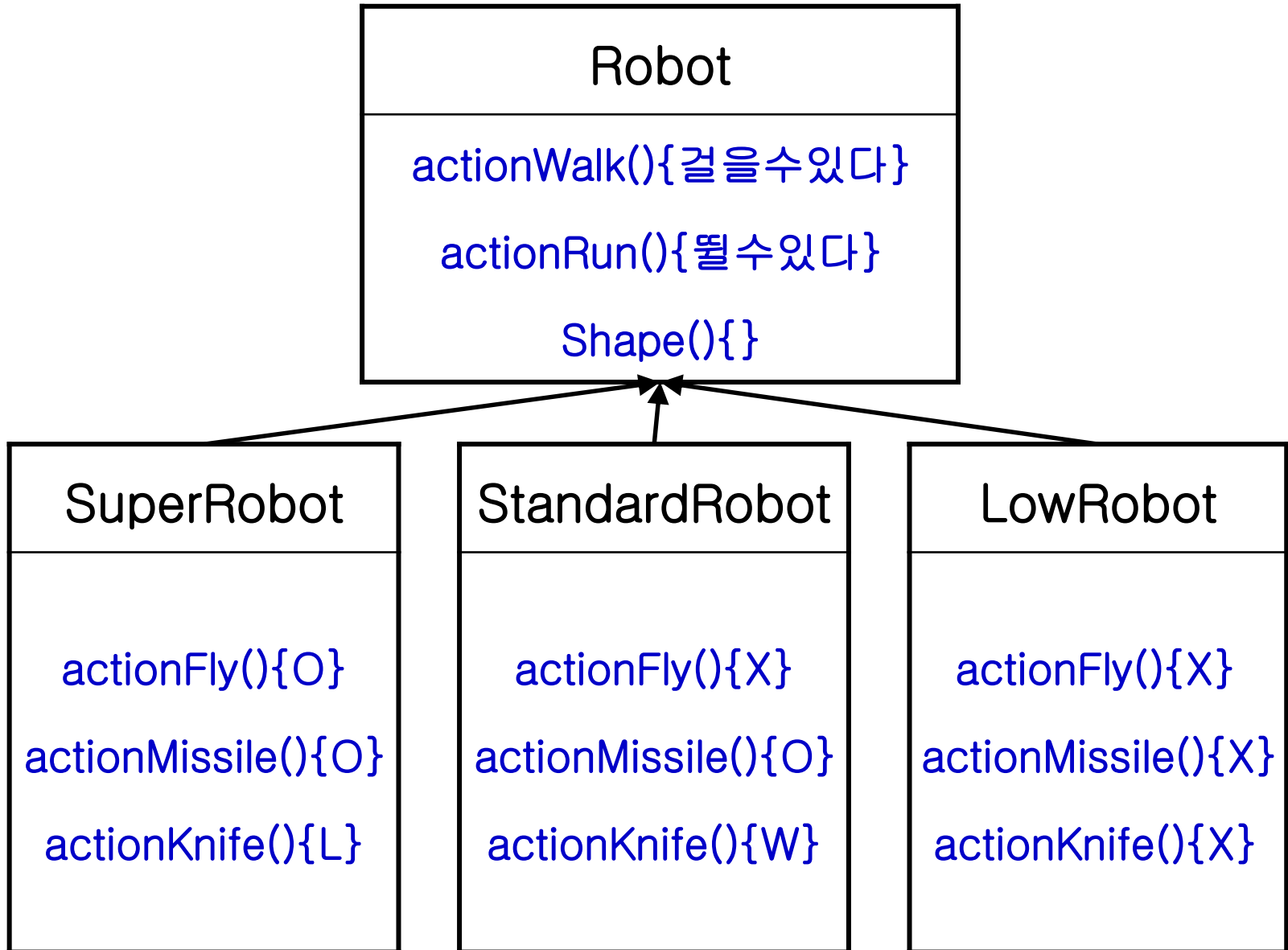
## StandardRobot

Shape(){}  
actionWalk(){  
    걸을수있다}  
actionRun(){  
    뛸수있다}  
actionFly(){X}  
actionMissile(){O}  
actionKnife(){W}

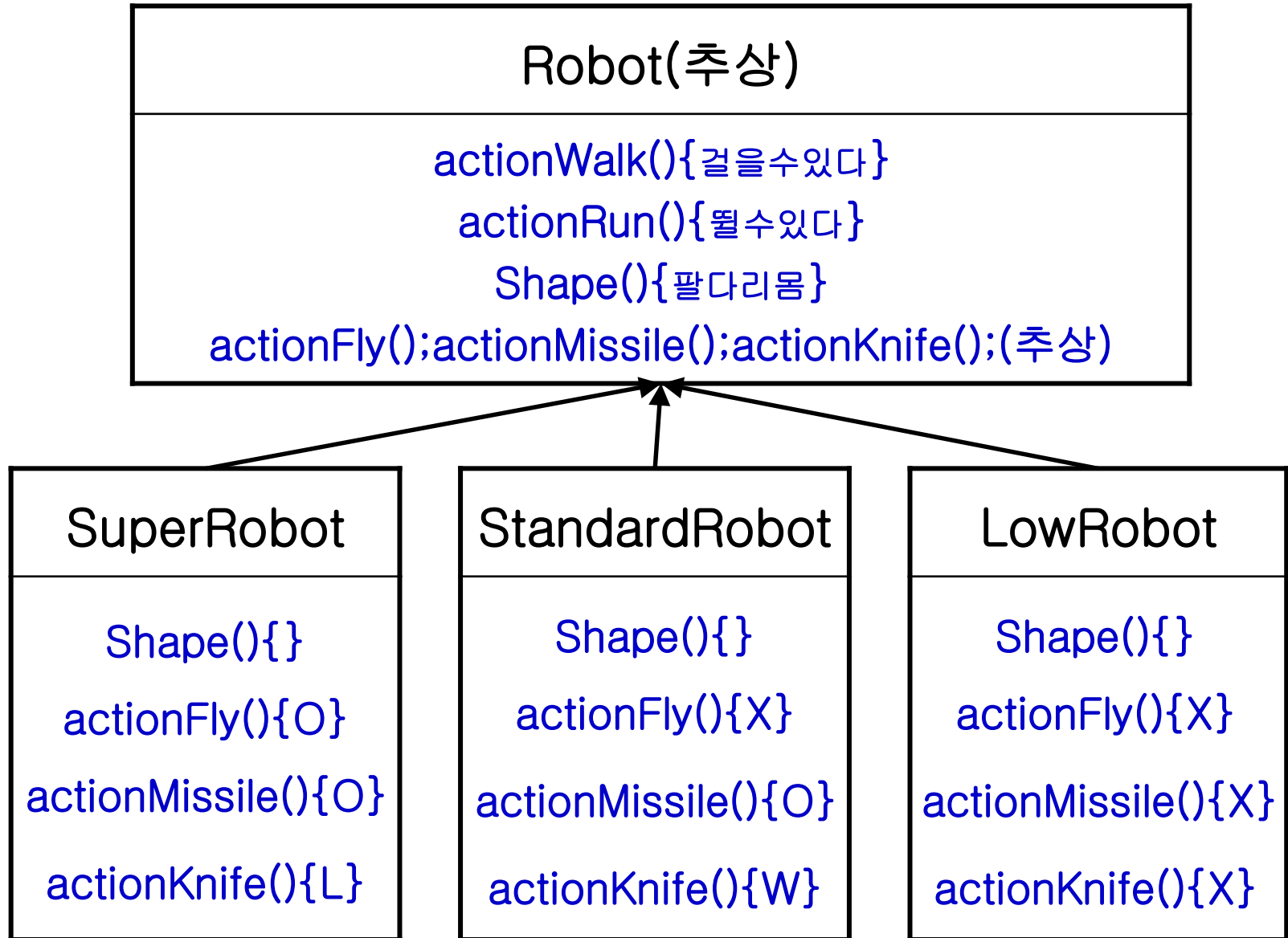
## LowRobot

Shape(){}  
actionWalk(){  
    걸을수있다}  
actionRun(){  
    뛸수있다}  
actionFly(){X}  
actionMissile(){X}  
actionKnife(){X}

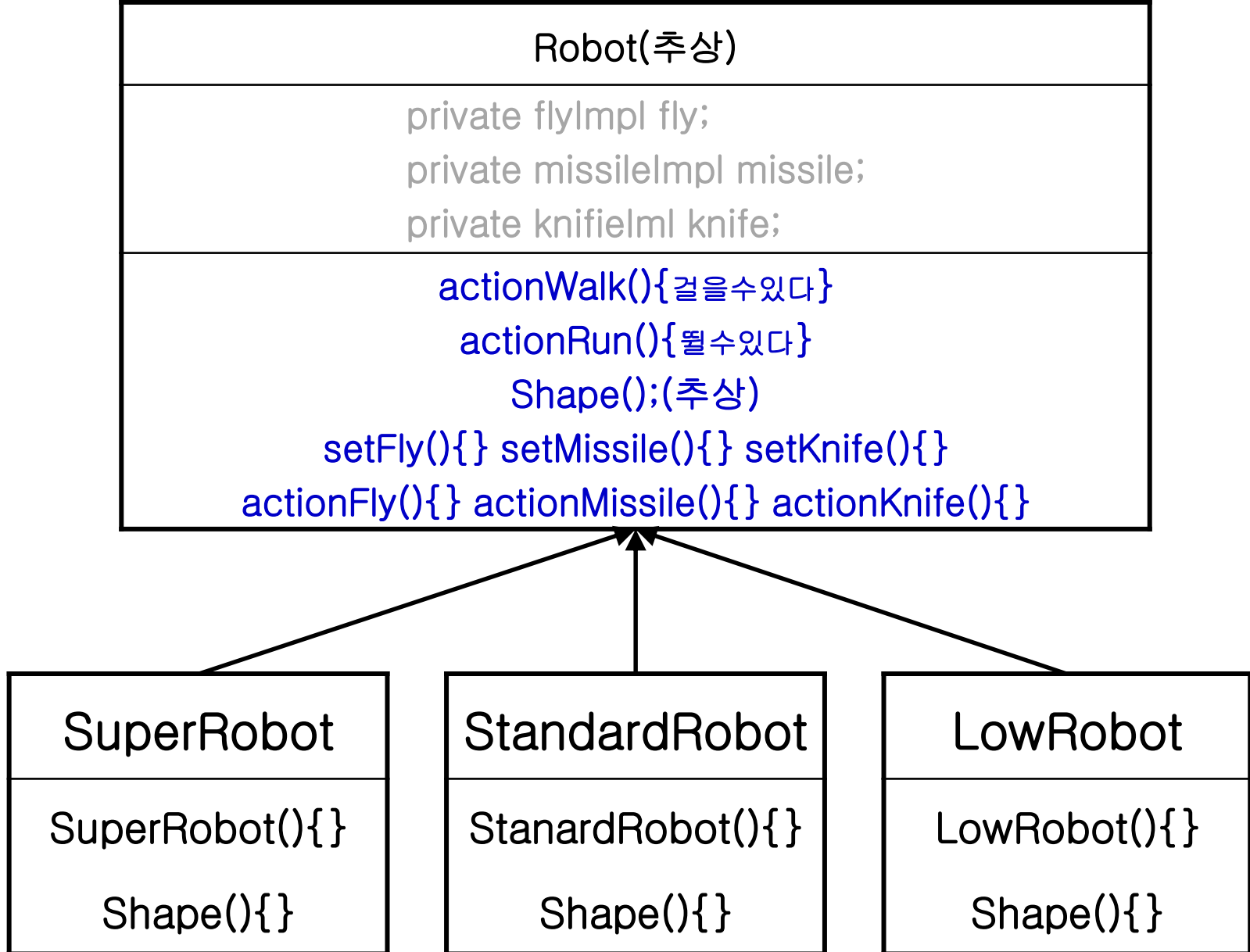
## 2단계. 공통점을 슈퍼클래스로



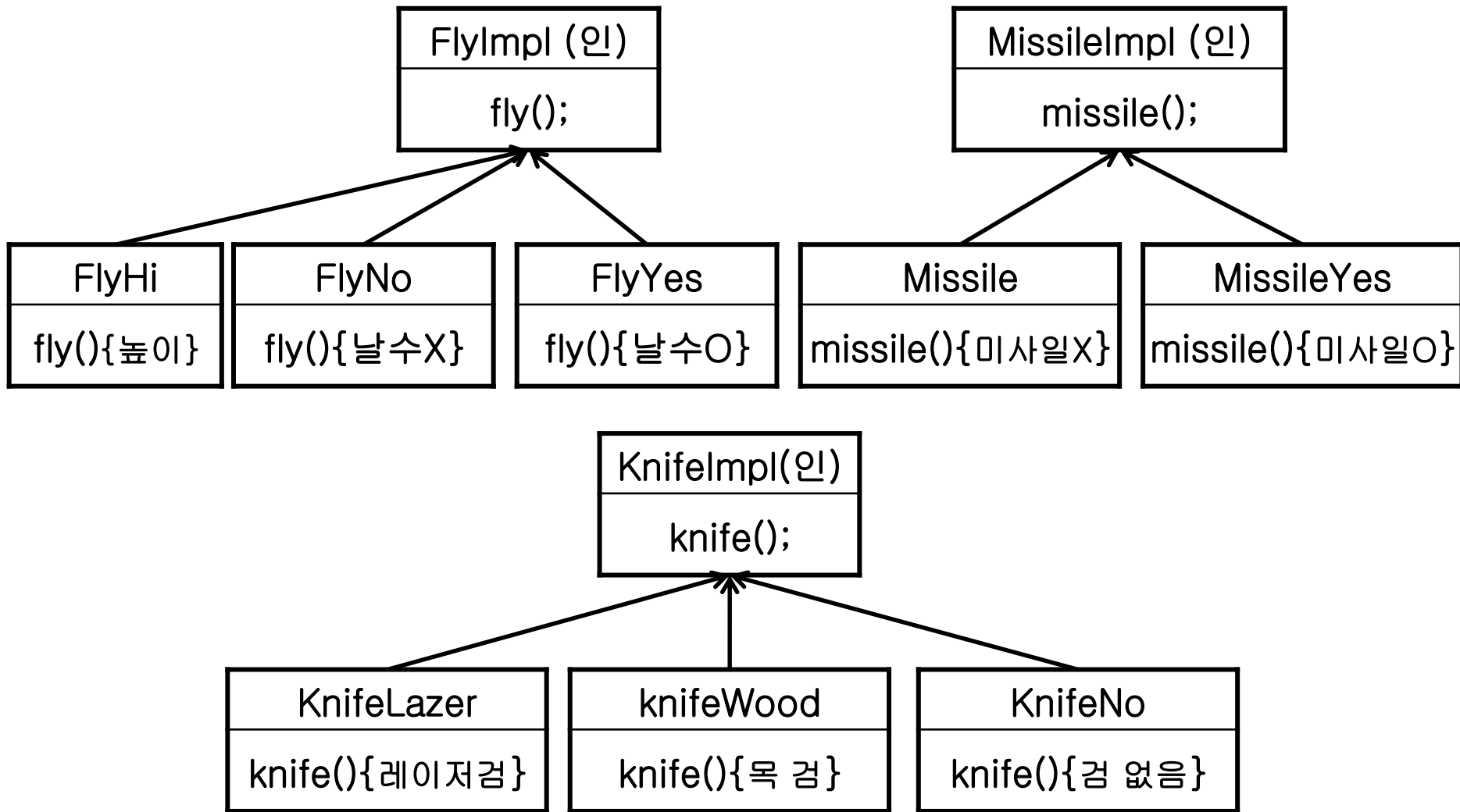
# 3단계. 공통점을 추상클래스로



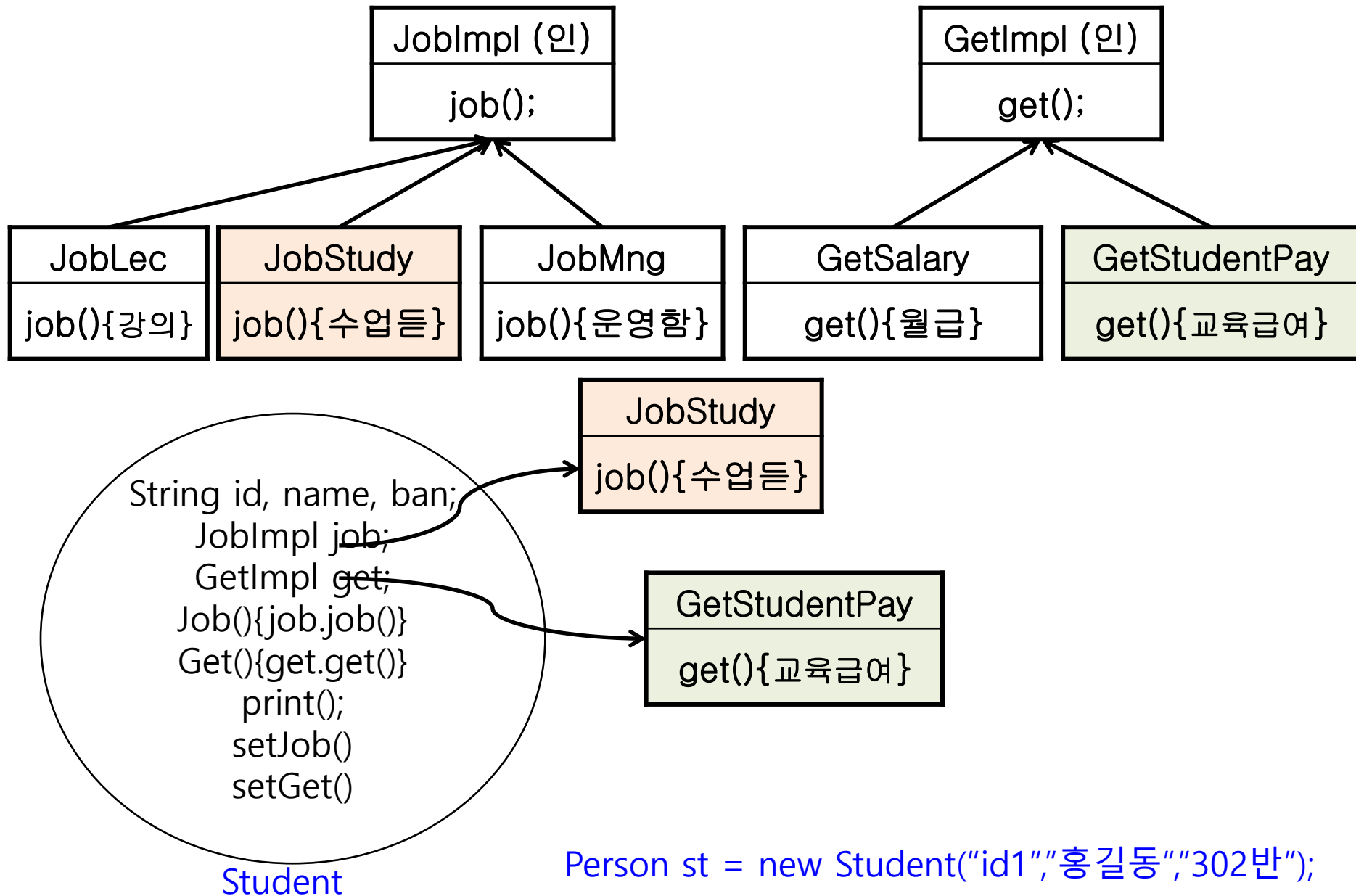
## 5단계. 각기능을 객체(부품)화 ; object modularization



# 인터페이스 관련

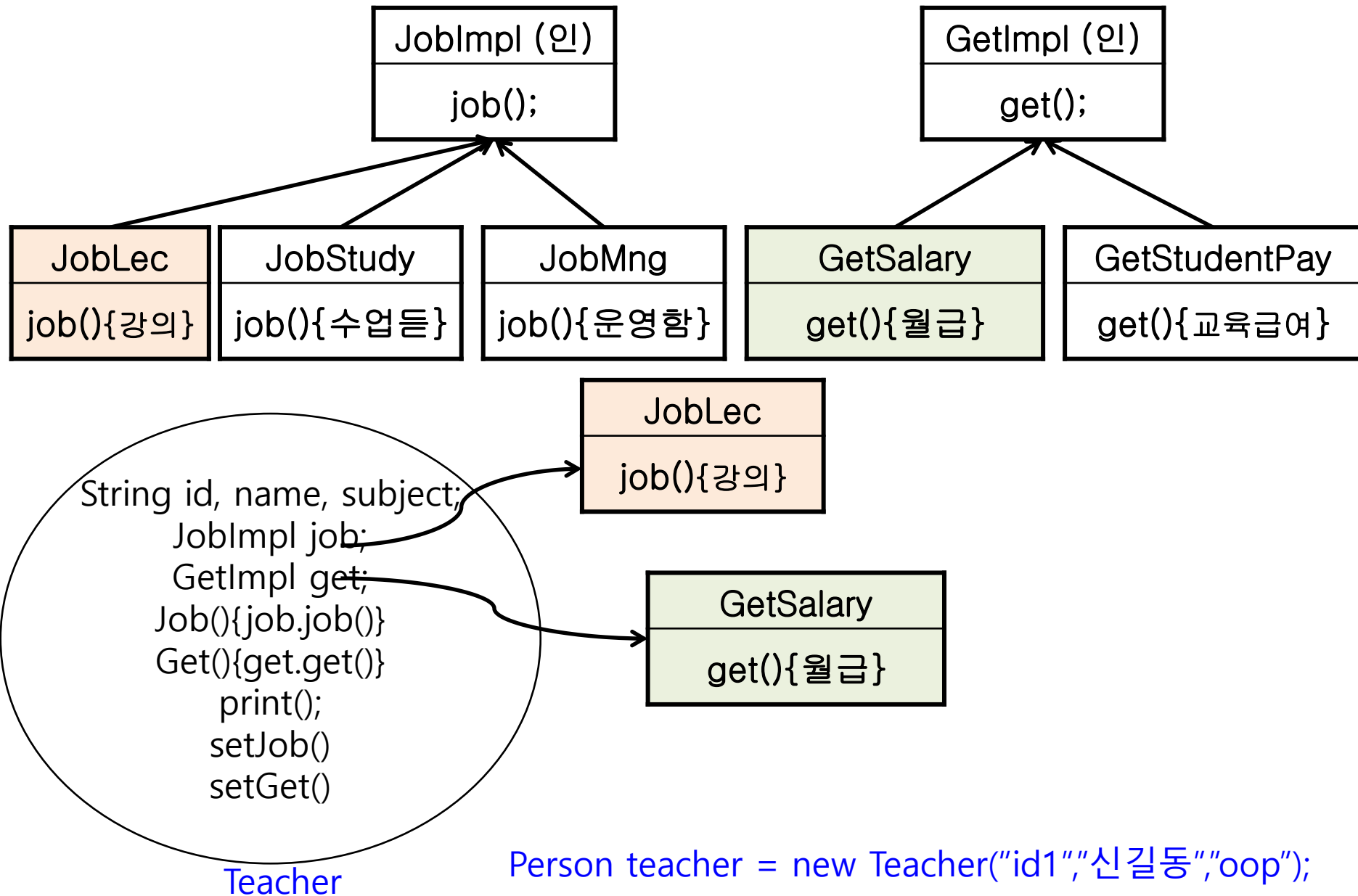


# 인터페이스 관련





# 인터페이스 관련



# 인터페이스 관련

