상속의 문제

▼관련 유튜브를 보려면 사진 클릭▼

상속이라는개념은 사실 OOP 있어서 문제점이 많다.
컴포지트 패턴 (Composite Pattern)으로 해결하자.

1. Is-a Has-a

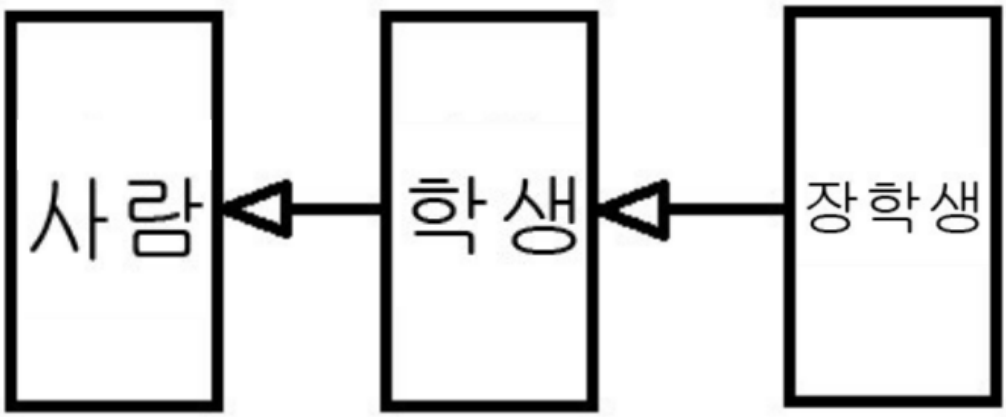
1). Is-A 관계 (상속관계)

ⓐ 학생은 사람이다, 장학생은 학생이다.

- Inheritance : class B is a Class A (or class A is extended by class B)
- ChildClass Is A ParentClass : $ChildClass \supseteq ParentClass$

ⓑ 상속관계

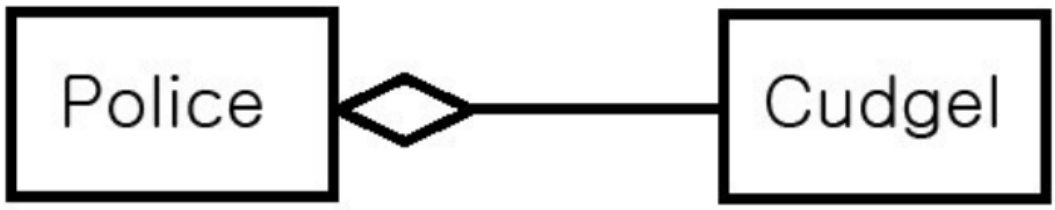
- 파생클래스로 가면 갈수록 구체화 혹은 특별화(specialization)
- 기본클래스로 가면 갈수록 일반화(generalization)

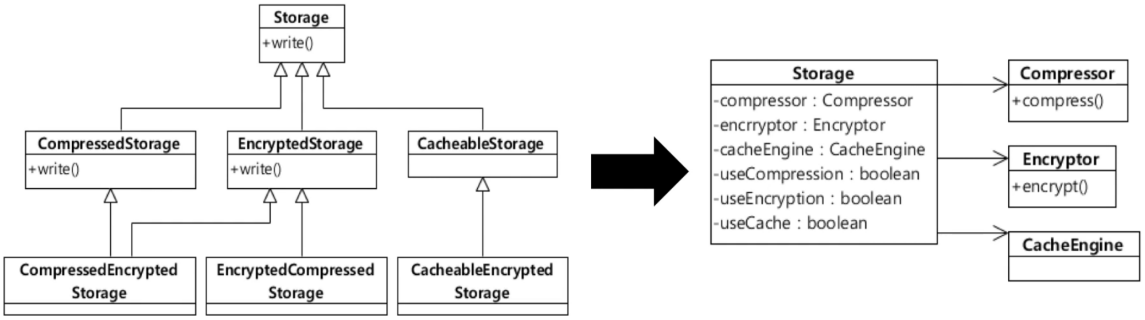


2). Has-A 관계 (포함관계)

ⓐ 경찰은 몽둥이를 포함한다

Aggregation : class A has a class B





ⓑ 코드

- ```
  //포함관계
  public class 몽둥이 {
   public virtual void Swing(){Console.WriteLine("때리 맥이고~");}
  }

  public class Police //몽둥이를 소유하는 경찰
  {
   public 몽둥이 mong; ✨
   public void UseWeapon(){ mong.Swing(); ✨ }
  };

  static void Main(String argv) {
   Police p;
   p.UseWeapon();
   return 0;
  }
  ```

---

2. 상속 문제점

1). 상속이 가지는 문제점

상속을 잘못 사용하면 변화에 유연하지 않고, 오류를 내기 쉬운 소프트웨어가 된다.
  ```

1. 부모 클래스의 불필요한 메소드가 하위클래스에 노출되거나 같이 상속될 위험이 있다. LSP
2. **tightly coupled relationship** : 상속은 부모 자식들 간에 코드의 의존과 결합도가 높아져 강하게 때문에 변화에 유연하게 대처하기 어려워진다.
3. 부모가 가지는 문제점을 자식도 물려받게 된다.
    - 부모 로직을 바꾸면 하위 클래스에서 일일이 수정을 해주는 설계할 수 밖에 없다.

---

3. 오버라이딩 문제점

```
class A {
    void func1() { System.out.println("A: func1"); }

        *클래스 B 에서 func1() 을 오버라이드 하고 있기 때문에 부작용 발생 가능*
    void func2() { func1(); System.out.println("A: func2");}
}

class B extends A {
    @Override
    void func1() { System.out.println("B: func1"); }
    void run() { func2(); System.out.println("B: run"); }
}

class Main {
    public static void main(String[] args) {
        B b = new B(); b.run();
    }
}

클래스 A 의 func2() 구현에는 func1() 을 호출하는 코드가 있는데,
클래스 B 가 func1() 을 오버라이드 했기 때문에 B 의 func1() 이 호출되는 상황이다.

> A: func2
> B: func1
> B: run
```

. Dynamic method binding

Virtual Methods

- Methods that can be overridden are called
  virtual methods
- 자바에서는 모든 매서드가 Virtual 이다 C# 이랑 다른 점이다.

Abstract Methods

- Base class가 되는 매서드가 생략된 상황
- Abstract Class는 꼭 하나의 Abstract Method를 포함해야한다.

```
abstract class person {
        public abstract void printLabel();
        ...
}
```

- 여담으로 C++에서는 *assignment to 0*를 하여 추상매서드를 만들 수 있다.

```
class person {
public:
        virtual void printLabel() = 0;
}
```

동적 바인딩 Dynamic Binding

- 다형성을 사용하여 메소드를 호출할 때, 발생하는 현상이다.
-
- **실행 시간(Runtime) 즉, 파일을 실행하는 시점에 성격이 결정**된다.
  실제 참조하는 객체는 서브 클래스이니 서브 클래스의 메소드를 호출한다.
- Runtime 시점에 해당 메소드를 구현하고 있는 실제 객체 타입을 기준으로 찾아가서 실행될 함수를 호출한다.

런타임중에 어떤 객체가 들어왔는지에 따라. 함수가 바뀜

. 해결법

1. final, private 키워드 : 상속이 안되는 멤버라는것을 명시

2. 주요기능은 abstract로 만들어 상속
3. **컴포지트 패턴**을 사용하면 된다.
    - 컴포지트은 유연성을 높이고 불필요한 메소드를 노출시키지 않는다
    - 자바스크립트에서도 비슷한 느낌이 있다.
    - __ proto__ 를 여러개 붙인 느낌이다.

. 참조

https://smilejsu.tistory.com/1034
https://tecoble.techcourse.co.kr/post/2020-05-18-inheritance-vs-composition/
https://8iggy.tistory.com/238

오버라이딩의 부작용