Interface(implement) I

2016년 10월 6일 목요일 오전 5:40

Preview

- Group 1의 Class A,B,C는 서로 연결되어 있다. 그리고 Group 2 의 Class D,E,F는 서로 연결되어 있다.
  Group1 과 Group2를 서로 연결시킬려면 되게 복잡해진다.

- **중간다리 역활을 하는 Interface Class 를 두어서 Group1과 Gruop2를 연결시키는 수를 최소화 시킨다.**
  Group A의 Class들은 Inteface Class만 통신을 하고, Group B의 Class들은 Inteface Class만 통신을 하게 구성한다.

Group1 (A,B,C)	⟷ Interface Class ⟷	Group2 (D,E,F)

- **Java는 둘 이상의 Class를 다중상속 받을 수 없다.**
  `Class AAA extends BBB,CCC {}` X

Case

프로젝트 담당자 A는 B사에게 특정기능을 담당하는 클래스의 설계를 요청했다.

- **프로젝트 담당자인 A의 요구사항 1**
  - 이름과 주민등록번호를 저장하는 클래스 필요
  - 이 클래스는 주민등록번호를 통해 사람의 이름을 찾는다.

- **프로젝트 담당자인 A의 요구사항 2**
  - 이름과 주민번호의 저장 : `void addPersonalInfo(String perNum, String name){};`
  - 주민번호를 이용한 검색 : `String serchName(String perNum){};`

  그런데 B사는 addPersonalInfo를 Class A에, searchName을 Class B에 **각각 정의해버렸다면?**
  B사가 몇개의 Class로 완성을 하던, A는 요구한 기능들이 하나의 Class의 Instance로 처리하고 싶다.

그래서 A는 PersonalNumberStorage Class를 하나 정의를 하고,
B사는 PersonalNumberStorage Class 를 상속해서 기능을 완성해 달라고 요청한다!

```
abstract class PersonalNumberStorage {
    abstract void addPersonalInfo(String perNum, String name);
    abstract String serchName(String perNum);
}
```

```
class AbstractInterface {
    public static void main(String[] args){
        PersonalNumberStorage storage = new(B사가 구현할 클래스);
        // B사가 어떤 인스턴스를 참조시키던 A는 PersonalInfoStorage형 참조변수만 쓰면된다!

        storage.addPersonalInfo(...);

        System.out.println("storage.serchName(...)");
    }
}
```

Interface

! **abstract로 선언하기 불편하므로 자바는 Interface 문법을 제공한다!**

```
abstract class PersonalNumberStorage {
    abstract void addPersonalInfo(String perNum, String name);
        abstract String serchName(String perNum);
```

```
}
```

↓

```
interface PersonalNumberStorage {
    public abstract void addPersonalInfo(String perNum, String name);
    public abstract String serchName(String perNum);
}
```

abstract Class 대신에 interface라고 선언하면 된다. 그러면 Method의 abstract선언도 생략가능!

- Interface에 선언된 Method는 외부에서 접근이 가능해야 되서 public이고 생략
  그리고 abstract가 생략되었다!
- Interface에 선언된 Variable은 public static final로 선언이 된다.

Use Interface

상속 = 구현

i. **Interface는 상속이란 단어 대신에 구현이라는 단어를 쓴다**
ii. **Interface를 구현할 때에는 extends 대신에 implement 선언!**
iii. **Interface는 다중 구현(상속)가능!**
iv. Interface간의 상속은 extends를 쓴다.

```
public interface MyInterface {
    public void myMethod();
}

public interface YourInterface {
    public void yourMethod();
}

class OurClass() implement MyInterface, YourInterface{
    @Override
    public void myMethod(){...};

    @Override
    public void yourMethod(){...};

}
```

- **MyInterface = new OurClass();**
