KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

객체지향 설계원칙-SOLID

[KB] IT's Your Life

KB 국민은행

객체 지향 원리

SOLID 설계 원칙

- 단일 책임 원칙(Single Responsibility Principle: SRP)
- 개방-폐쇄 원칙(Open-Closed Principle: OCP)
- 리스코프 치환 원칙(Liskov Substitution Principle: LSP)
- 인터페이스 분리 원칙(Interface Segregation Principle: ISP)
- 의존 역전 원칙(Dependency Inversion Principle: DIP)

단일 책임 원칙

- **객체 지향의 기본**
  - 책임을 객체에게 할당

- **클래스는 단 한 개의 책임을 가져야 한다.**
  - 클래스가 여러 책임을 갖게 되면 그 클래스는 각 책임마다 변경되는 이유가 발생
  - 클래스가 한 개의 이유로만 변경되려면 클래스는 한 개의 책임만 가져야 함

→ 클래스를 변경하는 이유는 단 한 개여야 한다.

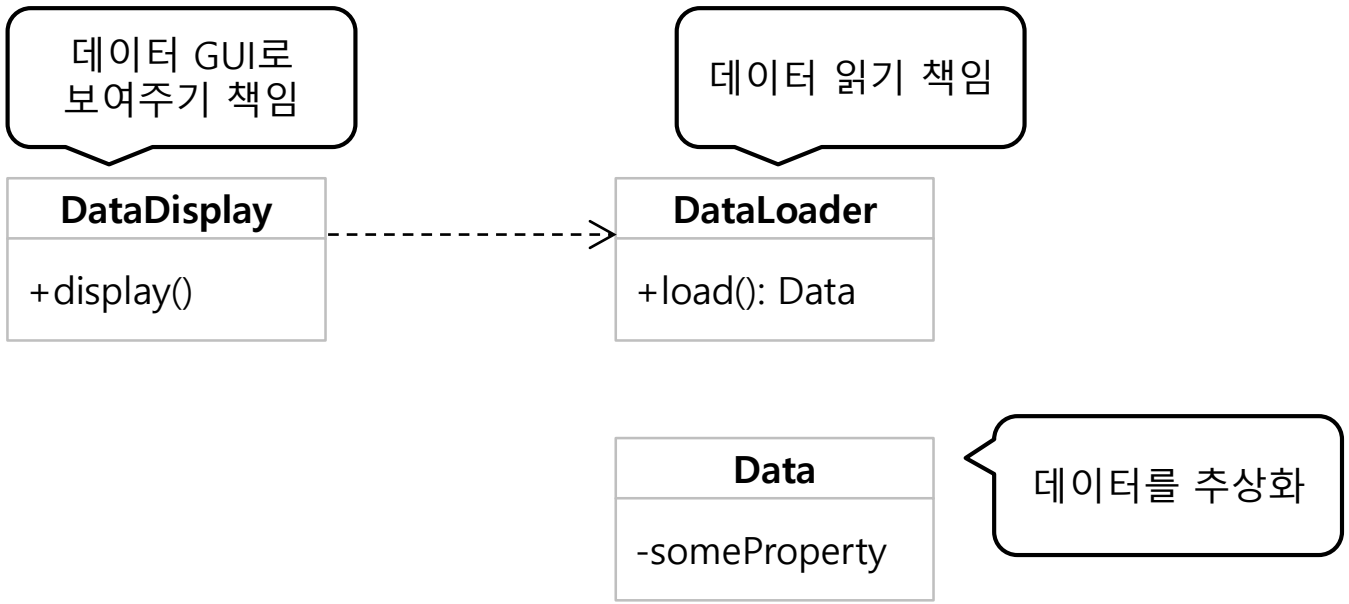
단일 책임 원칙

단일 책임 원칙 위반이 불러오는 문제점

```
public class DataViewer {

    public void display() {
        String data = loadHtml();
        updateGui(data);
    }

    public String loadHtml() {
        HttpClient client = new HttpClient();
        client.connect(url);
        return client.getResponse();
    }

    private void updateGui(String data) {
        GuiData guiModel = parseDataToGuiData(data);
        tableUI.changeData(guiModel);
    }
    private GuiData parseDataToGuiData(String data) {
        … // 파싱 처리 코드
    }
    … // 기타 필드 등 다른 코드
}
```

단일 책임 원칙

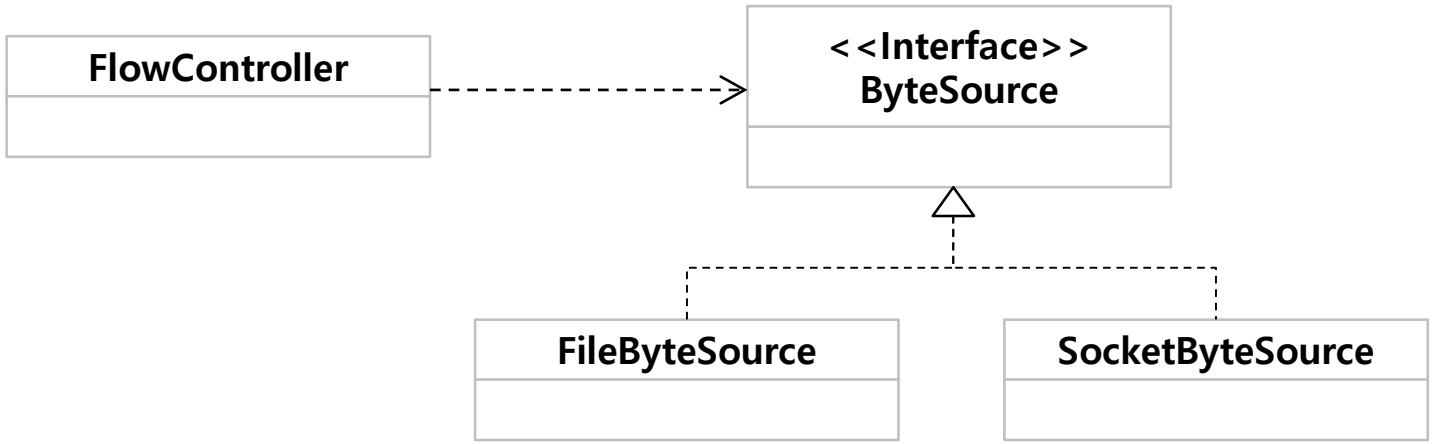
책임의 분리



개방 폐쇄 원칙

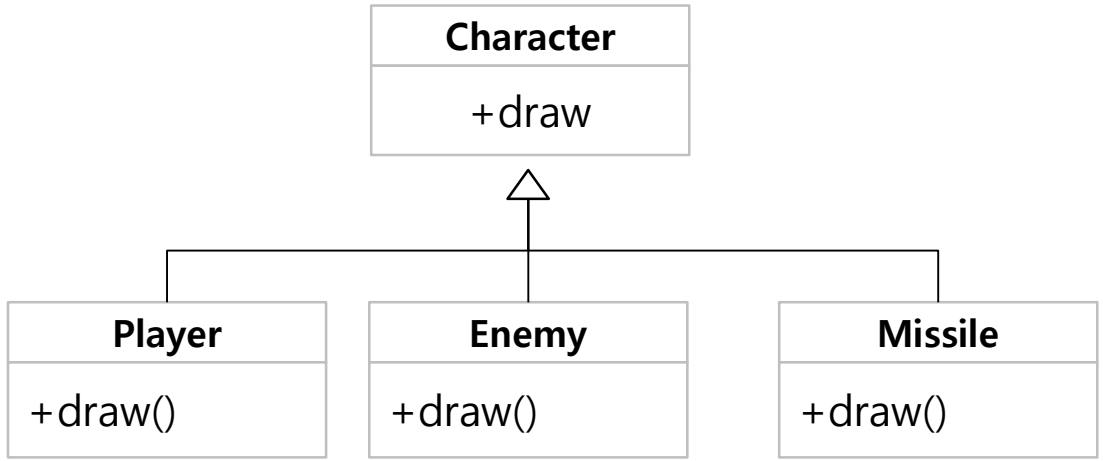
개방 폐쇄 원칙(Open-Closed Principle)

- 기능확장에는 열려 있고, 변화에는 닫혀 있어야 함
- 추상화와 다형성(상속)을 이용해서 구현



개방 폐쇄 원칙

개방 폐쇄 원칙(Open-Closed Principle)




```
public void drawCharacter(Character character) {
   character.draw();
}
```

리스코프 치환 원칙(Liskov Substitution Principle)

- 상위 타입의 객체를 하위 타입의 객체로 치환해도 상위 타입을 사용하는 프로그램은 정상적으로 동작해야 한다.
  - 개방 폐쇄 원칙을 받쳐 주는 다형성에 관한 원칙을 제공

```
public void someMethod(SuperClass sc) {
   sc.someMethod();
}
```

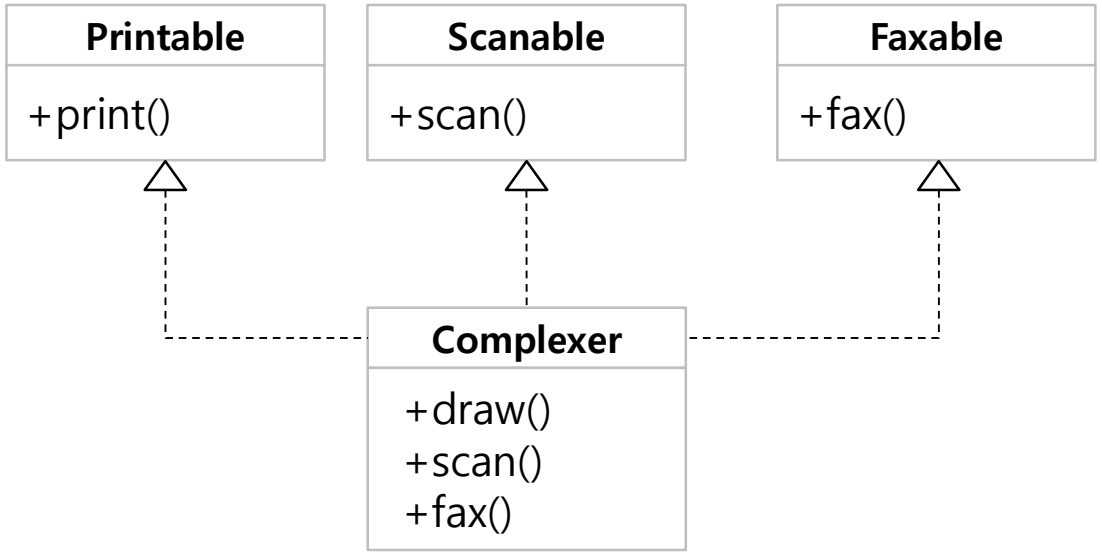
```
someMethod(new SubClass());
```

인터페이스 분리 원칙

인터페이스 분리 원칙(Interface Segregation Principle: ISP)

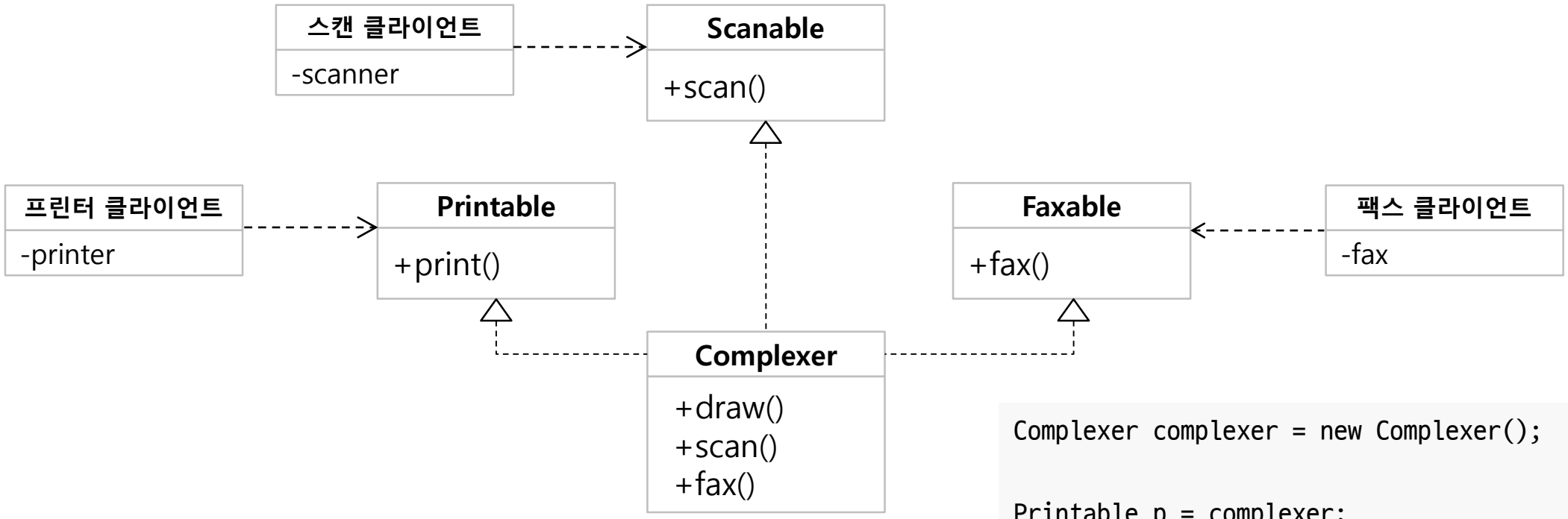
- 클라이언트는 자신이 사용하는 메서드에만 의존해야 한다.

→ 인터페이스는 그 인터페이스를 사용하는 클라이언트를 기준으로 분리해야 한다.



인터페이스 분리 원칙

인터페이스 분리 원칙(Interface Segregation Principle: ISP)




```
public class Complexer implements Printable, Scanable, Faxable {
    …
}
```

```
Complexer complexer = new Complexer();

Printable p = complexer;
p.print();

Scanable s = complexer;
s.scan();

Faxable f = complexer;
f.fax();
```

의존 역전 원칙

의존 역전 원칙(Dependency Inversion Principle)

- 고수준 모듈은 저수준 모듈의 구현에 의존해서는 안된다. 저수준 모듈이 고수준 모듈에서 정의한 추상 타입에 의존해야 한다.
  - 고수준 모듈: 어떤 의미 있는 단일 기능을 제공하는 모듈
  - 저수준 모듈: 고수준 모듈의 기능을 구현하기 위해 필요한 하위 기능의 실현 구현으로 정의

고수준 모듈

바이트 데이터를 읽어와 암호화하고 결과 바이트 데이터를 쓴다.

저수준 모듈

파일에서 바이트 데이터를 읽어온다.

AES 알고리즘으로 암호화한다.

파일에 바이트 데이터를 쓴다.

의존 역전 원칙

의존 역전 원칙(Dependency Inversion Principle)

- 고수준 모듈로 프래그램을 만들어 두고,
- 런타임시에 구체적인 저수준의 모듈을 결정해서 전달할 수 있음.