KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

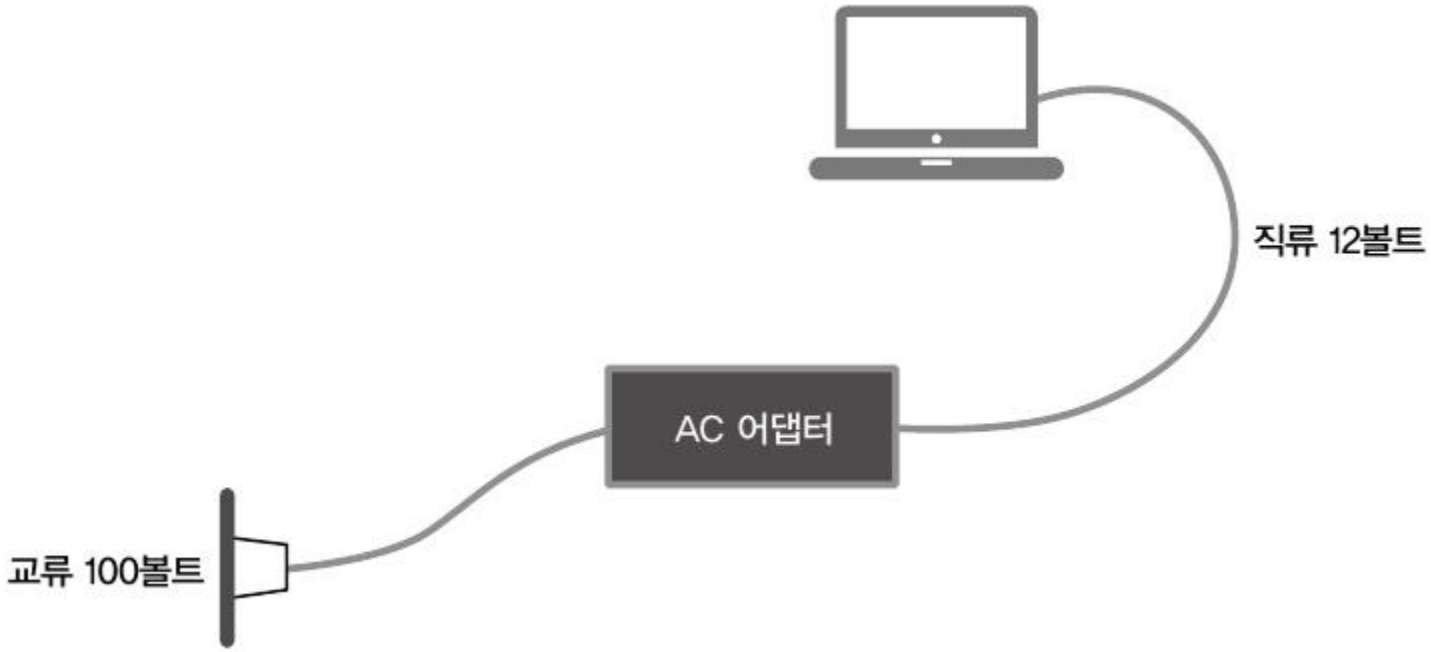
Adapter - 사이에 끼워 재사용한다

[KB] IT's Your Life

KB 국민은행

Adapter 패턴

- 이미 제공된 코드를 그대로 사용할 수 없을 때, 필요한 형태로 변환한 후 이용
- 이미 제공된 것과 필요한 것 사이의 차이를 메우는 디자인 패턴



Adapter 패턴

- Wrapper 패턴이라고 불리기도 함
- Adapter 패턴의 종류
  - 클래스에 의한 Adapter 패턴(상속을 사용한 패턴)
  - 인스턴스에 의한 Adapter 패턴(위임을 사용한 패턴)

상속을 사용한 패턴

예제 프로그램

- Hello라는 주어진 문자열을 다음과 같이 표시하는 간단한 프로그램

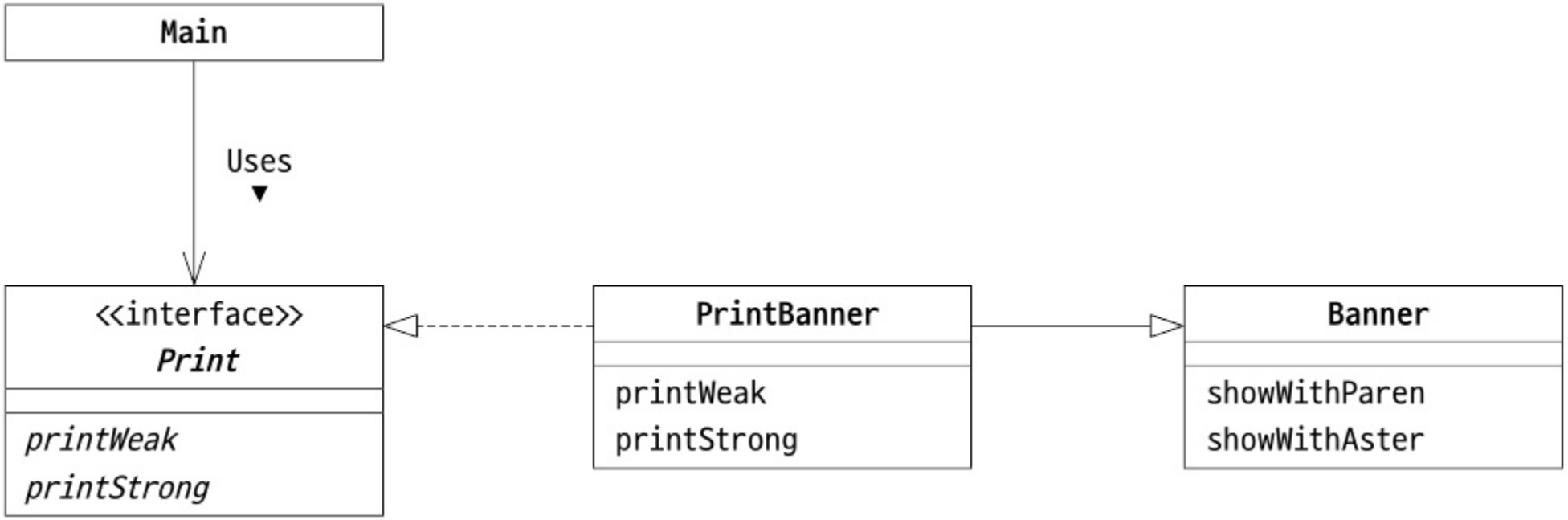
```
(Hello)
*Hello*
```

	전원의 비유	예제 프로그램
제공된 것	교류 100볼트	Banner 클래스(showWithParen, showWithAster)
변환 장치	어댑터	PrintBanner 클래스
필요한 것	직류 12볼트	Print 인터페이스(printWeak, printStrong)

상속을 사용한 패턴

예제 프로그램 클래스 다이어그램

- Banner 클래스를 사용하는 Print 인터페이스 구현체(PrintBanner) 정의
- PrintBanner가 Adapter 역할
- Banner를 상속 받아 정의



상속을 사용한 패턴

Banner.java

```
public class Banner {
    private String string;

    public Banner(String string) {
        this.string = string;
    }

    public void showWithParen() {
        System.out.println("(" + string + ")");
    }

    public void showWithAster() {
        System.out.println("*" + string + "*");
    }
}
```

상속을 사용한 패턴

Print.java

```
public interface Print {
    void printWeak();
    void printStrong();
}
```

상속을 사용한 패턴

PrintBanner.java

```
public class PrintBanner extends Banner implements Print{
    public PrintBanner(String string) {
        super(string);
    }

    @Override
    public void printWeak() {
        showWithParen();
    }

    @Override
    public void printStrong() {
        showWithAster();
    }
}
```

상속을 사용한 패턴

Main.java

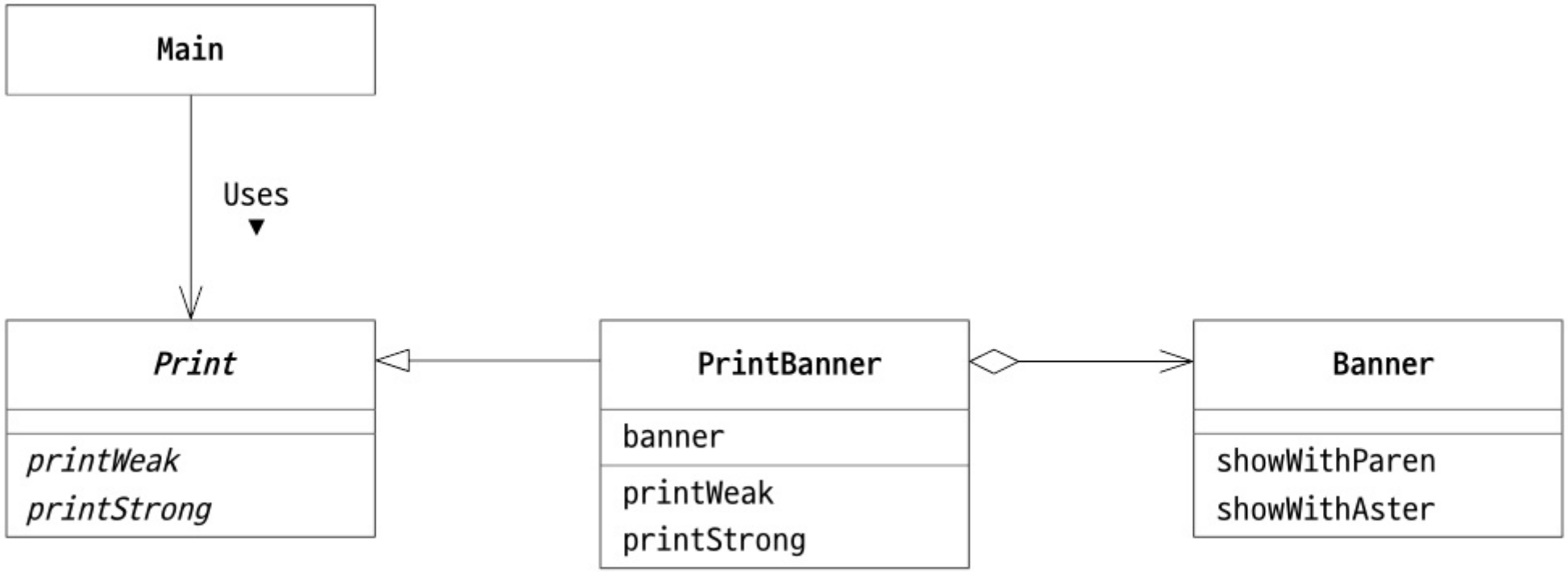
```
public class Main {
    public static void main(String[] args) {
        Print p = new PrintBanner("Hello");
        p.printWeak();
        p.printStrong();
    }
}
```

```
(Hello)
*Hello*
```

위임을 사용한 패턴

예제 프로그램 클래스 다이어그램

- Print는 인터페이스가 아닌 클래스로 운영



Print.java

```
public abstract class Print {
    public abstract void printWeak();
    public abstract void printStrong();
}
```

PrintBanner.java

```
public class PrintBanner extends Print{
    private Banner banner;  // 위임 객체

    public PrintBanner(String string) {
        this.banner = new Banner(string);
    }

    @Override
    public void printWeak() {
        banner.showWithParen();  // 기능 위임
    }

    @Override
    public void printStrong() {
        banner.showWithAster();  // 기능 위임
    }
}
```

위임을 사용한 패턴

Main.java

```
public class Main {
    public static void main(String[] args) {
        Print p = new PrintBanner("Hello");
        p.printWeak();
        p.printStrong();
    }
}
```

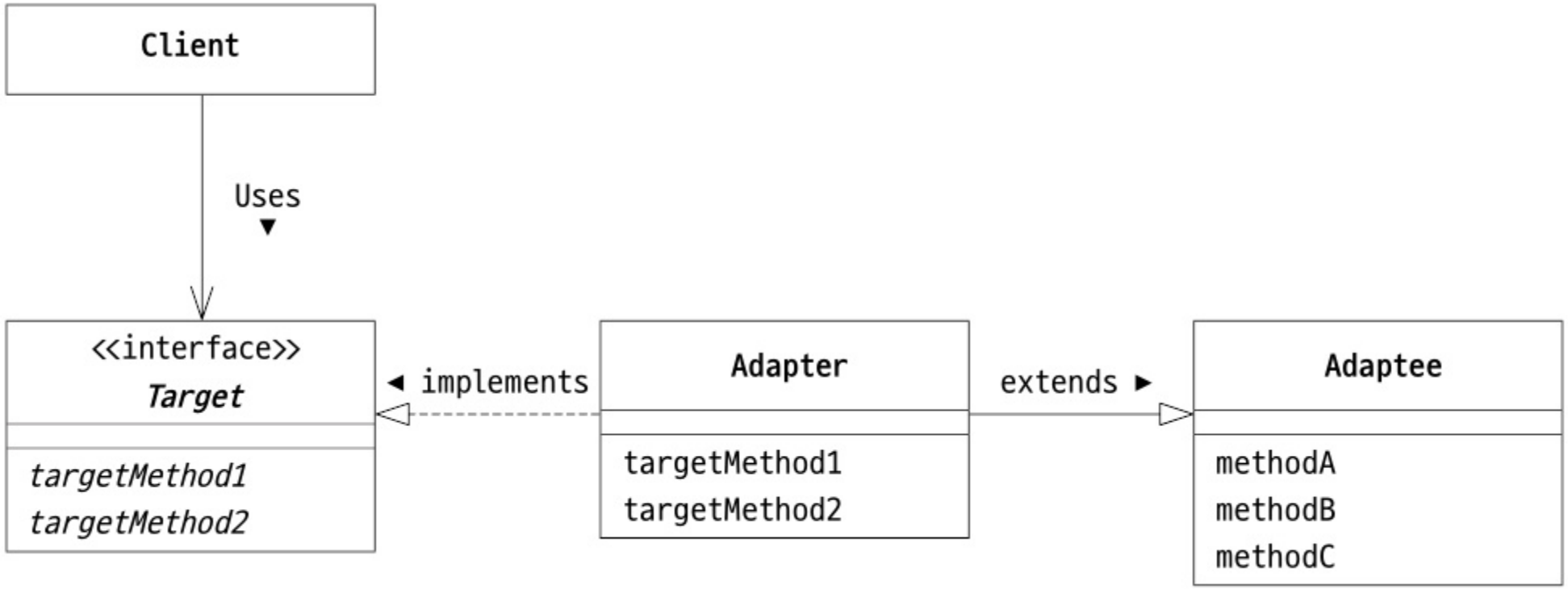
Adapter 패턴

Adapter 패턴의 등장인물

- Target
  - 지금 필요한 것
  - Print 인터페이스, Print 클래스
- Client
  - Target의 메서드를 사용
  - Main
- Adaptee
  - 이미 준비된 메서드를 가지는 역할
  - Banner
- Adapter
  - Adaptee의 메서드를 사용해서 어떻게든 Target을 만족시키는 것

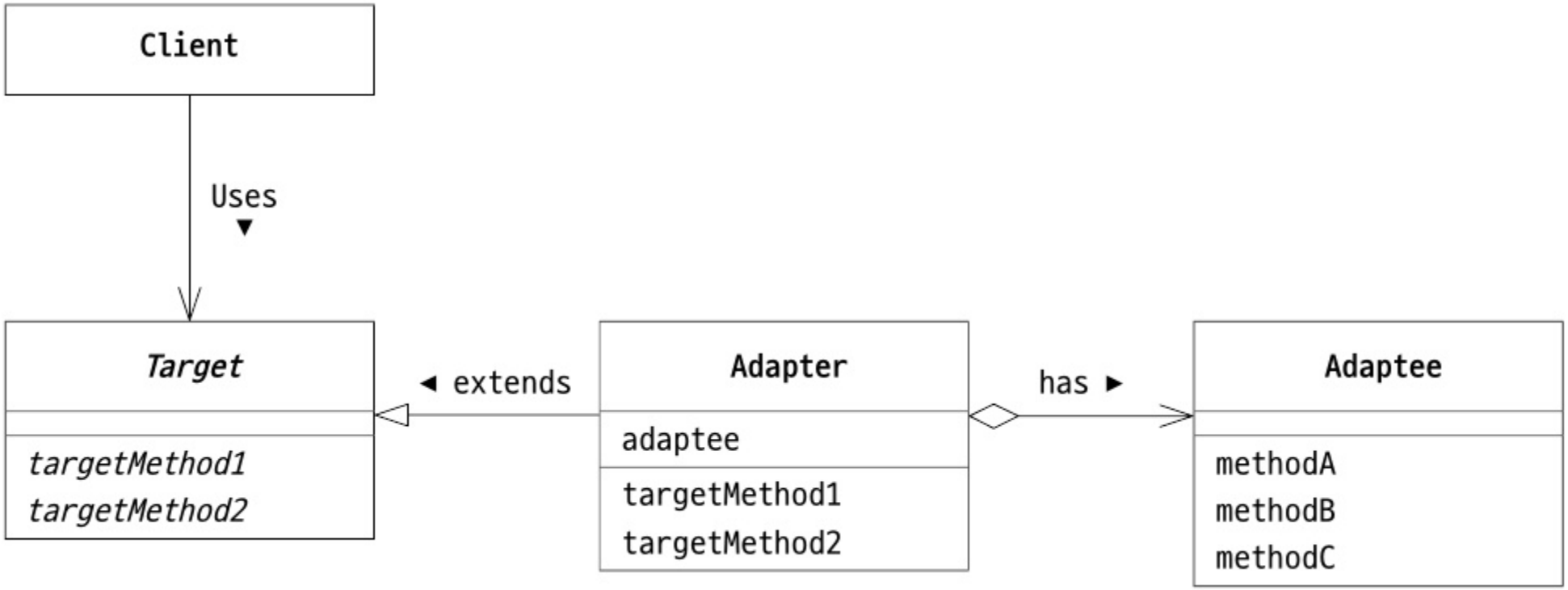
Adapter 패턴

클래스에 의한 Adapter 패턴의 클래스 다이어그램(상속)



Adapter 패턴

인스턴스에 의한 Adapter 패턴의 클래스 다이어그램(위임)



Adapter 패턴

어떤 경우에 사용하는 것일까?

- 프로그래밍할 때 늘 백지상태에서 시작하는 것은 아님
- 이미 존재하는 클래스를 이용하는 경우
- 기존 클래스에 한 겹 덧씌워 필요한 클래스를 만듦

→ 기존 클래스를 전혀 수정하지 않고 목적한 인터페이스에 맞추는 것

  - 기존 클래스의 소스 프로그램이 반드시 필요한 것이 아님