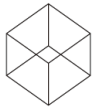
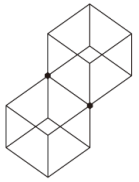
Adapter 패턴

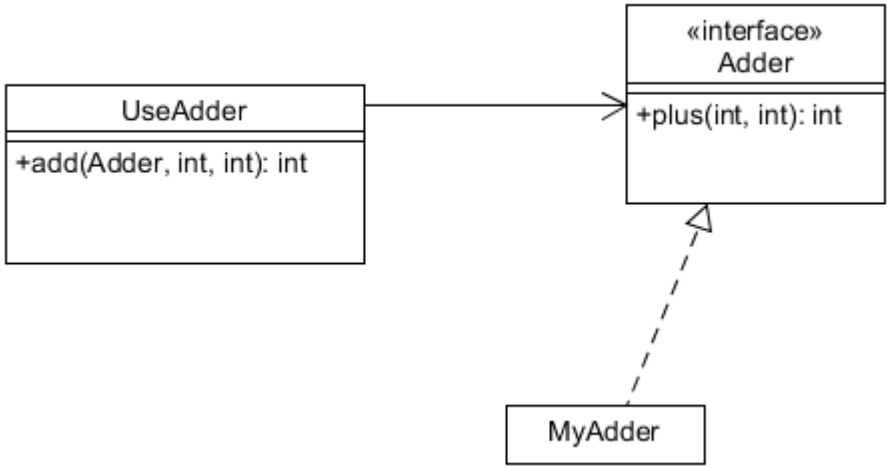


JAVA 객체 지향 디자인 패턴

UML과 GoF 디자인 패턴 핵심 10가지로 배우는



예제 프로그램 설계도


«interface»
Adder
+plus(int, int): int
UseAdder
+add(Adder, int, int): int
MyAdder

소스코드

```java
public interface Adder {
    public int plus(int x, int y);
}
```

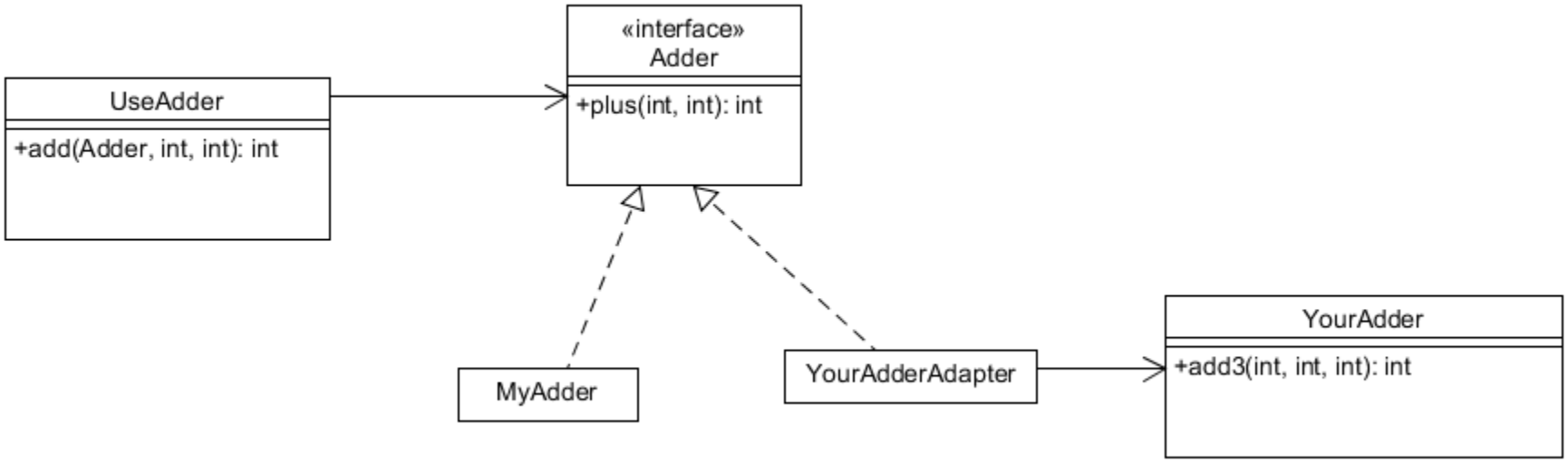
```java
public class MyAdder implements Adder {
    @Override
    public int plus(int x, int y) {
        return x + y;
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Adder adder = new MyAdder();
        UseAdder use = new UseAdder();
        System.out.println(use.add(adder, 10, 20));
    }
}
```

```java
public class UseAdder {
    public int add(Adder adder, int x, int y) {
        int r = 0;
        r = adder.plus(x, y);
        return r;
    }
}
```

제약

- 특정 회사에서 제공하는 Adder(eg. YourAdder)를 사용해야 하고 라이브러리 형태로 제공되어 수정 할 수 없다.
- 클라이언트 코드도 수정할 수 없다.
- Adapter 생성



소스 코드

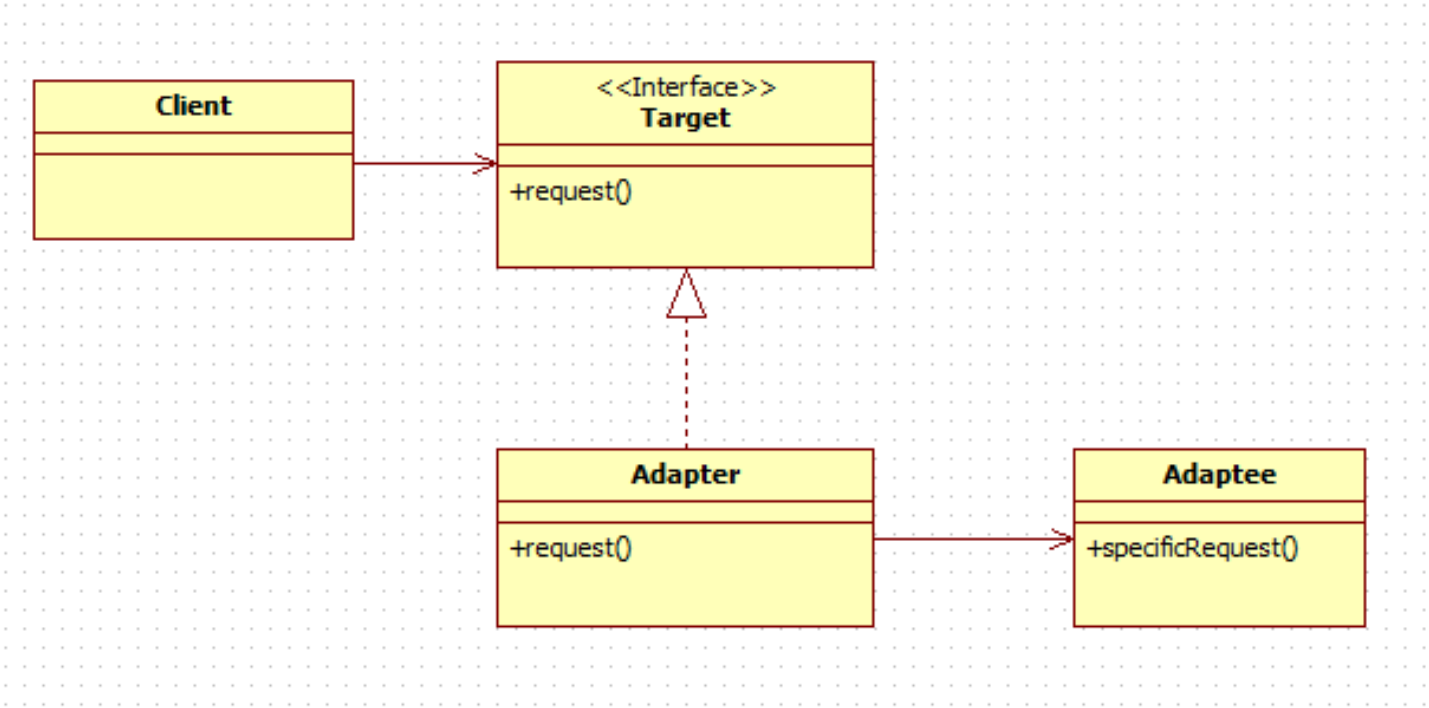
```java
public class YourAdderAdapter implements Adder {
    private YourAdder yourAdder;
    public YourAdderAdapter(YourAdder yourAdder) {
        this.yourAdder = yourAdder;
    }
    @Override
    public int plus(int x, int y) {
        return yourAdder.add3(x, y, 0);
    }
}
```

```java
    public int add3(int x, int y, int z) {
        return x + y + z;
    }
}
```

Main

```
public class Main {
    public static void main(String[] args) {
        Adder adder = new MyAdder();
        UseAdder use = new UseAdder();
        System.out.println(use.add(adder, 10, 20));
        Adder adder1 = new YourAdderAdapter(new YourAdder());
        System.out.println(use.add(adder1, 10, 20));
    }
}
```

Adapter 패턴


Client
<<Interface>>
Target
+request()
Adapter
+request()
Adaptee
+specificRequest()