1) Proxy 패턴

```
class 클라이언트 {
    세금계산  tax

    public 클라이언트(세금계산  tax) {
        this.tax = tax;
    }

    public void run() {
        doSomething1(tax);
        doSomething2(tax);
    }

    public void doSomething1(세금계산 tax) {
        tax.계산1();
        tax.계산2();
    }

    public void doSomething2(세금계산 tax) {
        tax.계산1();
        tax.계산2();
    }
}

interface 세금계산 {
    int 계산1();
    int 계산2();
}

class 세금계산A  implements 세금계산 {
    public int 계산1() {
        ...계산...
    }

    public int 계산2() [
        ...계산...
    }
}

class 세금계산B  implements 세금계산 {
    public int 계산1() {
        ...계산...
    }

    public int 계산2() [
        ...계산...
    }
}

class 세금계산_권한통제A implements 세금계산 {
    세금계산 tax;

    public 세금계산_권한통제A(세금계산 tax) {
        this.tax = tax;
    }

    public int 계산1() {
        ... 권한조회A...
        return tax.계산1();
    }

    public int 계산2() [
        ... 권한조회A...
        return tax.계산2();
    }
}

class 세금계산_권한통제B implements 세금계산 {
    세금계산 tax;
```

```
    public 세금계산_권한통제B(세금계산 tax) {
        this.tax = tax;
    }

    public int 계산1() {
        ... 권한조회B...
        return tax.계산1();
    }

    public int 계산2() [
        ... 권한조회B...
        return tax.계산2();
    }
}


///// 객체 조립 사례#1
클라이언트 client1 = new 클라이언트(new 세금계산A());
client1.run();

클라이언트 client2 = new 클라이언트(new 세금계산_권한통제A(new 세금계산B()));
client2.run();

클라이언트 client3 = new 클라이언트(new 세금계산_권한통제B(new 세금계산B()));
client3.run();
```

2) Proxy 패턴 + Adpater 패턴

```
class 클라이언트 {
    세금계산  tax

    public 클라이언트(세금계산  tax) {
        this.tax = tax;
    }

    public void run() {
        doSomething1(tax);
        doSomething2(tax);
    }

    public void doSomething1(세금계산 tax) {
        tax.계산1();
        tax.계산2();
    }

    public void doSomething2(세금계산 tax) {
        tax.계산1();
        tax.계산2();
    }
}

interface 세금계산 {
    int 계산1();
    int 계산2();
}

class 세금계산A  implements 세금계산 {
    public int 계산1() {
        ...계산...
    }

    public int 계산2() [
        ...계산...
    }
}

class 세금계산B  implements 세금계산 {
    public int 계산1() {
        ...계산...
    }

    public int 계산2() [
        ...계산...
    }
}

class 세금계산_권한통제A implements 세금계산 {
    세금계산 tax;

    public 세금계산_권한통제A(세금계산 tax) {
        this.tax = tax;
    }

    public int 계산1() {
        ... 권한조회A...
        return tax.계산1();
    }

    public int 계산2() [
        ... 권한조회A...
        return tax.계산2();
    }
}

class 세금계산_권한통제B implements 세금계산 {
    세금계산 tax;
```

```
    public 세금계산_권한통제B(세금계산 tax) {
        this.tax = tax;
    }

    public int 계산1() {
        ... 권한조회B...
        return tax.계산1();
    }

    public int 계산2() [
        ... 권한조회B...
        return tax.계산2();
    }
}
```

```
class 세금계산C {
    public int 계산a1() {
        ...계산....
    }

    public int 계산a2() {
        ...계산....
    }

    public int 계산b1() {
        ...계산....
    }
}

class 세금계산Adapter implements 세금계산 {
    세금계산C tax = new 세금계산C();

    public int 계산1() {
        tax.계산a1();
        tax.계산a2();
    }

    public int 계산2() [
        tax.계산b1();
    }
}
```

```
///// 객체 조립 사례
클라이언트 client1 = new 클라이언트(new 세금계산Adapter(new 세금계산C()));
client1.run();

클라이언트 client2 = new 클라이언트(new 세금계산_권한통제A(new 세금계산Adapter(new 세금계산C())));
client2.run();

클라이언트 client3 = new 클라이언트(new 세금계산_권한통제B(new 세금계산Adapter(new 세금계산C())));
client3.run();
```