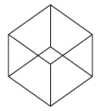
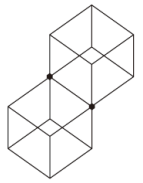
14. 컴퍼지트 패턴



JAVA 객체 지향 디자인 패턴

UML과 GoF 디자인 패턴 핵심 10가지로 배우는

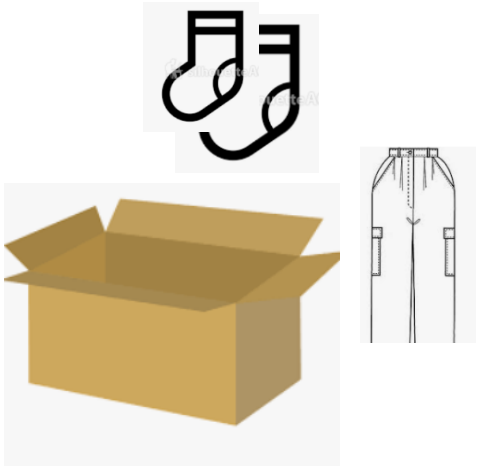


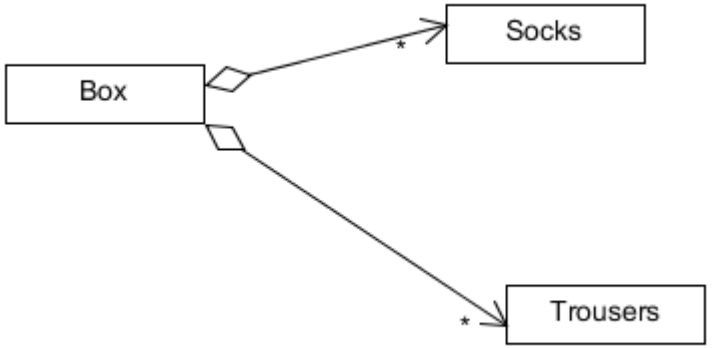
학습목표

학습목표

- 부분–전체의 관계가 있는 객체의 설계 방법 이해하기
- 컴퍼지트 패턴을 이용한 부분–전체 객체의 설계 방법 이해하기
- 사례 연구를 통한 컴퍼지트 패턴의 핵심 특징 이해하기

택배 비용 계산하기





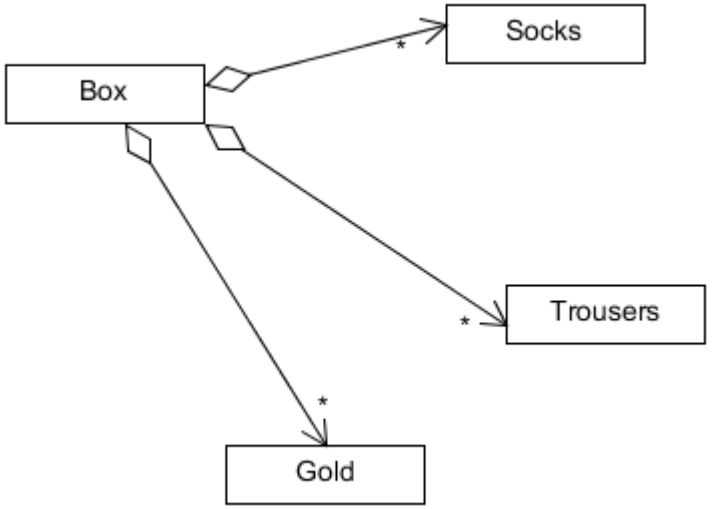
소스 코드

```
public class Box {
    private List<Trousers> trousers = new ArrayLi
    private List<Socks> socks = new ArrayList<>()


    public int price() {
        int tPrice = 0;
        int sPrice = 0;
        int gPrice = 0;

        for (Trousers t : trousers) {
            tPrice += t.price();
        }
        for (Socks s : socks) {
            sPrice += s.price();
        }


        return tPrice + sPrice + gPrice;
    }
    public void addSocks(Socks s) { socks.add(s); }
    public void addTrousers(Trousers t) { trousers.add(t); }
    public void addGolds(Gold g) { golds.add(g); }

}
```

```
public class Socks {
    private int weight;

    public Socks(int weight) {
        this.weight = weight;
    }

    public int price() {
        return this.weight/100*200;
    }
}
```

```
public class Trousers {
    private int weight;

    public Trousers(int weight) {
        this.weight = weight;
    }

    public int price() {
        return this.weight/100*200;
    }
}
```

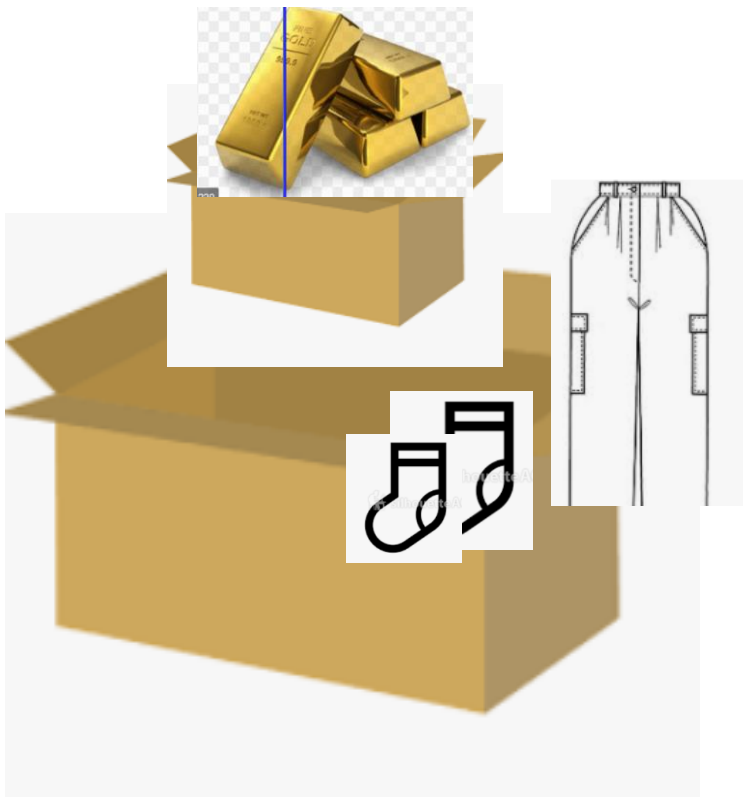
택배 아이템 추가

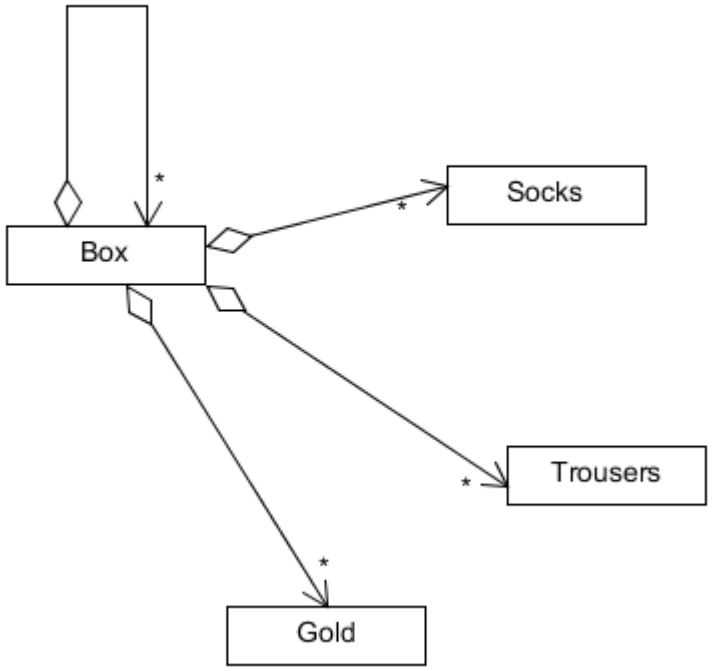


OCP 위배

```
public class Box {
    private List<Trousers> trousers = new ArrayList<>();
    private List<Socks> socks = new ArrayList<>();
    private List<Gold> golds = new ArrayList<>();

    public int price() {
        int tPrice = 0;
        int sPrice = 0;
        int gPrice = 0;

        for (Trousers t : trousers) {
            tPrice += t.price();
        }
        for (Socks s : socks) {
            sPrice += s.price();
        }
        for (Gold g : golds) {
            gPrice += g.price();
        }

        return tPrice + sPrice + gPrice;
    }
    public void addSocks(Socks s) { socks.add(s); }
    public void addTrousers(Trousers t) { trousers.add(t); }
    public void addGolds(Gold g) { golds.add(g); }

}
```

택배 아이템 추가





Box In Box

```
public class Box {
    private List<Trousers> trousers = new ArrayList<>();
    private List<Socks> socks = new ArrayList<>();
    private List<Gold> golds = new ArrayList<>();
    private List<Box> boxes = new ArrayList<>();

    public int price() {
        int tPrice = 0;
        int sPrice = 0;
        int gPrice = 0;
        int bPrice = 0;

        for (Trousers t : trousers) {
            tPrice += t.price();
        }
        for (Socks s : socks) {
            sPrice += s.price();
        }
        for (Gold g : golds) {
            gPrice += g.price();
        }
        for (Box b : boxes) {
            bPrice += b.price();
        }

        return tPrice + sPrice + gPrice + bPrice;
    }
    public void addSocks(Socks s) { socks.add(s); }
    public void addTrousers(Trousers t) { trousers.add(t); }
    public void addGolds(Gold g) { golds.add(g); }
    public void addBox(Box b) { boxes.add(b); }

}
```

클라이언트 코드

```
public class Main {
    public static void main(String[] args) {
        Box box1 = new Box();
        Socks s1 = new Socks(200);
        Socks s2 = new Socks(300);
        Trousers t1 = new Trousers(600);
        box1.addSocks(s1);
        box1.addSocks(s2);
        box1.addTrousers(t1);
        System.out.println(box1.price());

        Box box2 = new Box();
        Gold g1 = new Gold(800);
        box2.addBox(box1);
        box2.addGolds(g1);
        System.out.println(box2.price());

    }
}
```

OCP 위배

```
public class Box {
    private List<Trousers> trousers = new ArrayList<>();
    private List<Socks> socks = new ArrayList<>();
    private List<Gold> golds = new ArrayList<>();

    public int price() {
        int tPrice = 0;
        int sPrice = 0;
        int gPrice = 0;

        for (Trousers t : trousers) {
            tPrice += t.price();
        }
        for (Socks s : socks) {
            sPrice += s.price();
        }
        for (Gold g : golds) {
            gPrice += g.price();
        }

        return tPrice + sPrice + gPrice;
    }
    public void addSocks(Socks s) { socks.add(s); }
    public void addTrousers(Trousers t) { trousers.add(t); }
    public void addGolds(Gold g) { golds.add(g); }

}
```

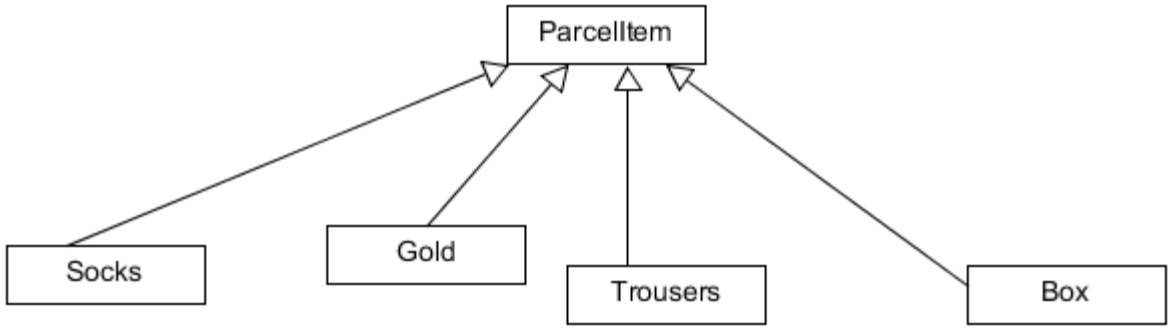
해결책 1

```
public class Box {

    private List<Object> items = new ArrayList<>();

    public int price() {
        int tPrice = 0;

        for (Object o : items) {
          //  totalPrice += o.price();
            if (o instanceof Trousers) tPrice += ((Trousers)
o).price();
            else if (o instanceof Socks) tPrice += ((Socks)
o).price();
            else if (o instanceof Gold) tPrice += ((Gold)
o).price();
            else if (o instanceof Shirts) tPrice += ((Shirts)
o).price();
            else tPrice += ((Box) o).price();
        }
        return tPrice;


    public void addItems(Object o) { items.add(o); }

}
```

OCP 위배

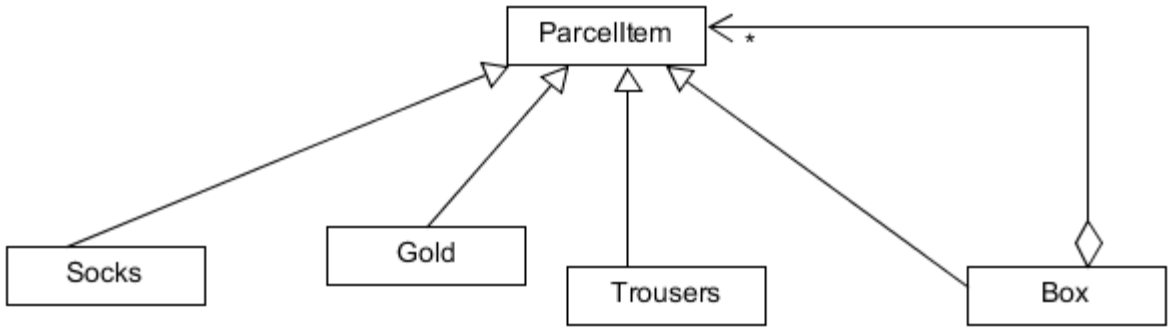
해결책 2

- 택배로 보낼 수 있는 항목을 포함하는 개념(ParcelItem) 생성



해결책 2

- “한 Box는 택배로 보낼 수 있는 항목(ParcelItem)을 여러 개 가질 수 있다” 는 사실을 표현.



소스코드

```java
public class Box extends ParcelItem{

    private List<ParcelItem> items = new ArrayList<>();

    public Box(int weight) {
        super(weight);
    }

    public int price() {
        int totalPrice = 0;


        for (ParcelItem item : items) {
            totalPrice += item.price();
        }
        return totalPrice;
    }

    public void addItems(ParcelItem item) { items.add(item); }

}
```

```java
public abstract class ParcelItem {
    protected int weight;
    public ParcelItem(int weight) {
        this.weight = weight;
    }
    public abstract int price();
}
```

```java
public class Socks extends ParcelItem{
    public Socks(int weight) {
       super(weight);
    }
    public int price() {
        return weight/100*200;
    }
}
```

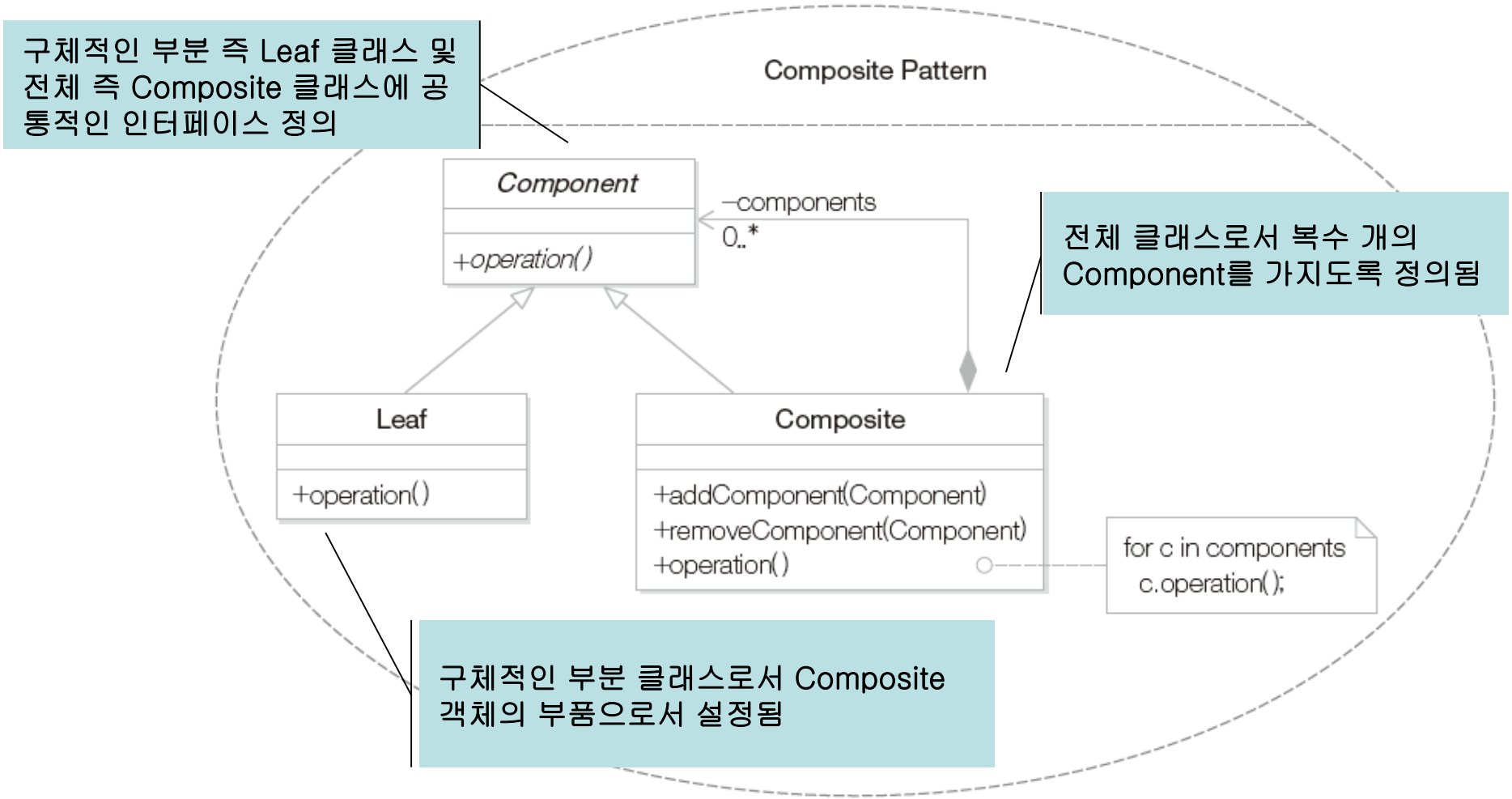
컴퍼지트 패턴

- 부분(part)-전체(whole)의 관계를 가지는 객체들을 정의할 때 유용

컴퍼지트 패턴은 전체-부분의 관계를 가지는 객체들 간의 관계를 정의할 때 유용하다. 그리고 클라이언트는 전체와 부분을 구분하지 않고 동일한 인터페이스를 사용할 수가 있다.

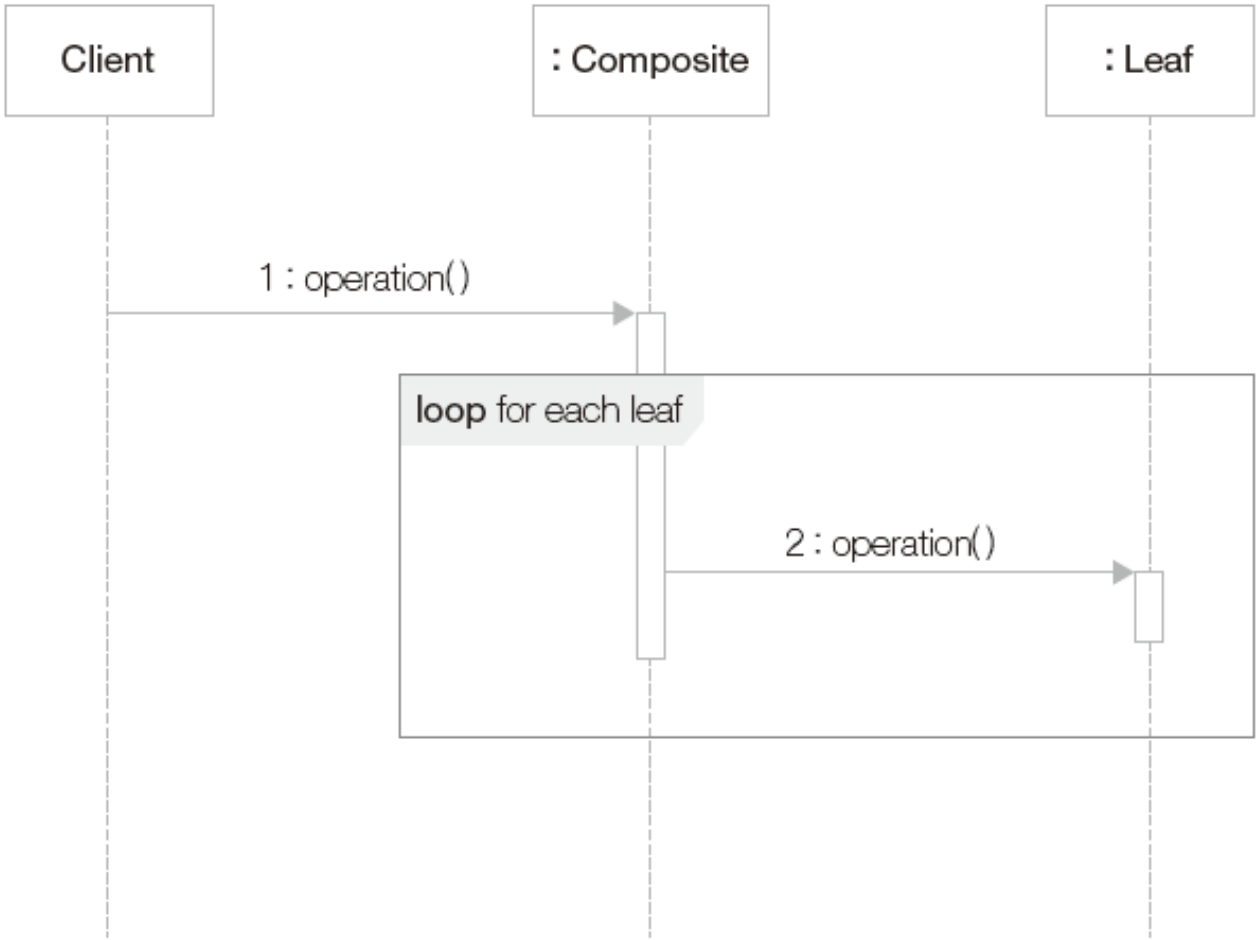
14.4 컴퍼지트 패턴

그림 14-7 컴퍼지트 패턴의 컬레보레이션



14.4 컴퍼지트 패턴

그림 14-8 컴퍼지트 패턴의 순차 다이어그램



컴퍼지트 패턴의 적용

그림 14-9 컴퍼지트 패턴을 컴퓨터 추가 장치 예제에 적용한 경우