Bsidesoft co.

CODE SPITZ

84

OBJECT 2

1 2 3 4 5 6

상속과 확장

상속(inherit)

상속(inherit)

유산을 물려받음.

상속(inherit)

유산을 물려받음.

상속해 준 사람은 이미 사망

상속(inherit)

유산을 물려받음.

상속해 준 사람은 이미 사망

살아있으면서 물려주는 경우는 증여

상속(inherit)

유산을 물려받음.

상속해 준 사람은 이미 사망

살아있으면서 물려주는 경우는 증여

유산은 있으나 망자와 교감할 수 없다

증여(bestowal)

증여(bestowal)

증여를 해준 쪽과 상호작용할 수 있다.

증여(bestowal)

증여를 해준 쪽과 상호작용할 수 있다.

증여받고도 괴롭히면 싫다.

증여(bestowal)

증여를 해준 쪽과 상호작용할 수 있다.

증여받고도 괴롭히면 싫다.

하지만 재산 외적인 교감은 환영.

확장(extend)

확장(extend)

유산을 물려받지 말 것

확장(extend)

유산을 물려받지 말 것

대리역할을 하지 말 것

확장(extend)

유산을 물려받지 말 것

대리역할을 하지 말 것

오히려 확장하는 쪽이 부분 책임만 질 것

나쁜 확장

나쁜 확장 1

super는 나쁘다.

나쁜 확장 1

super는 나쁘다.

result =

나쁜 확장 1

super는 나쁘다.

result = base

나쁜 확장 1

super는 나쁘다.

result = base

result2 =

나쁜 확장 1

super는 나쁘다.

result = base

result2 = base + extend

나쁜 확장 1

super는 나쁘다.

result3 = base

result2 = base + extend

나쁜 확장 1

super는 나쁘다.

result3 = base′

result2 = base + extend

나쁜 확장 1

super는 나쁘다.

result3 = base′

result2 = base′ + extend

나쁜 확장 1

super는 나쁘다.

result3 = base′

~~result2~~ = base′ + extend

나쁜 확장 2

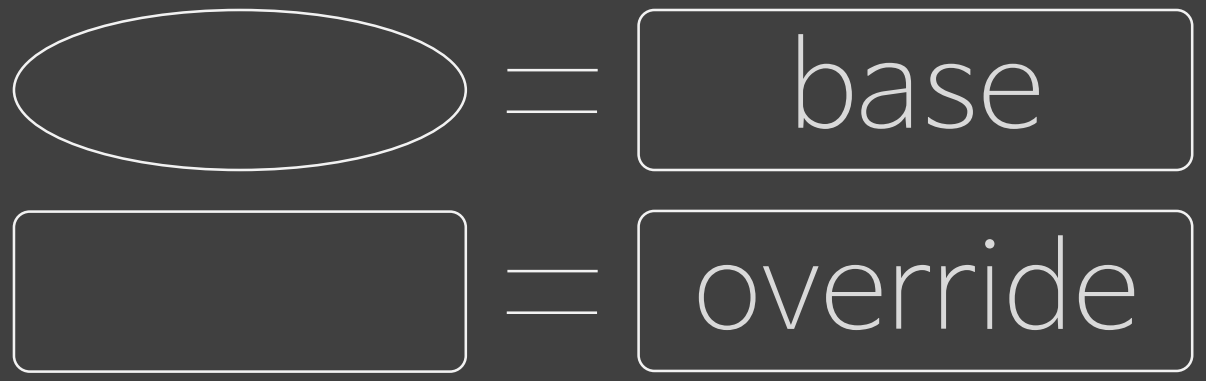
override는 나쁘다.

나쁜 확장 2

override는 나쁘다.

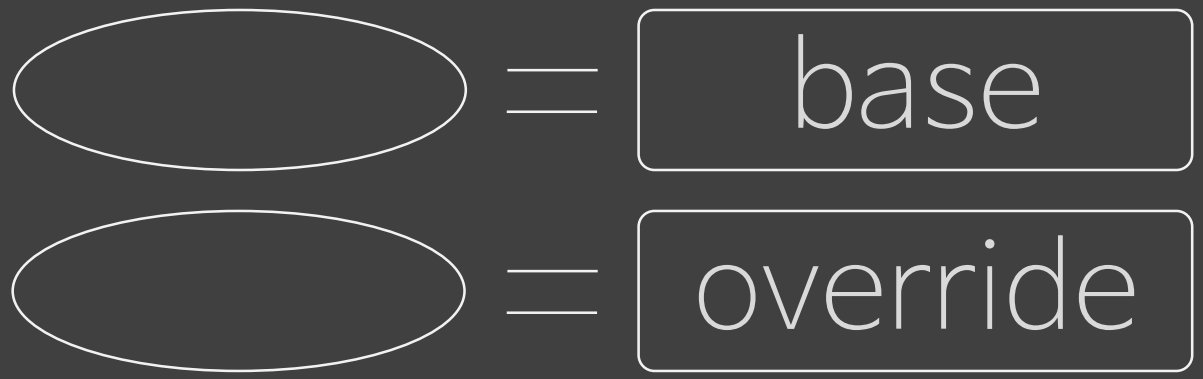
나쁜 확장 2

override는 나쁘다.



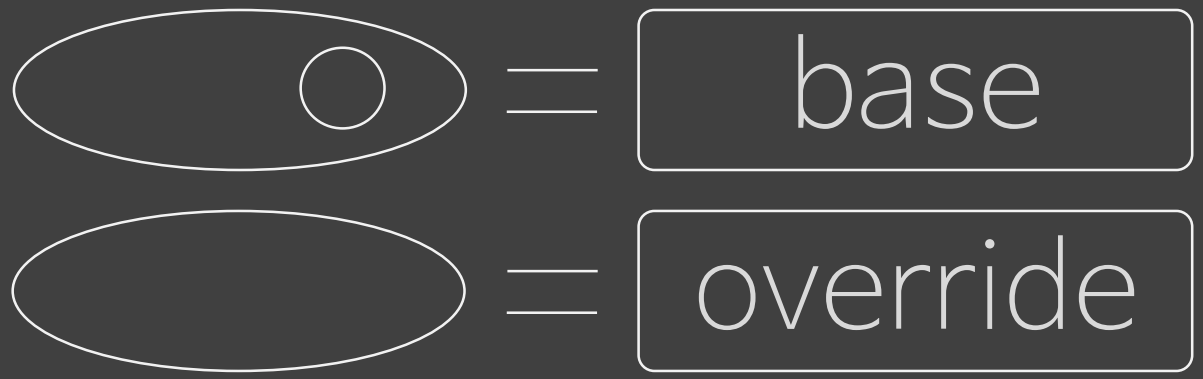
나쁜 확장 2

override는 나쁘다.



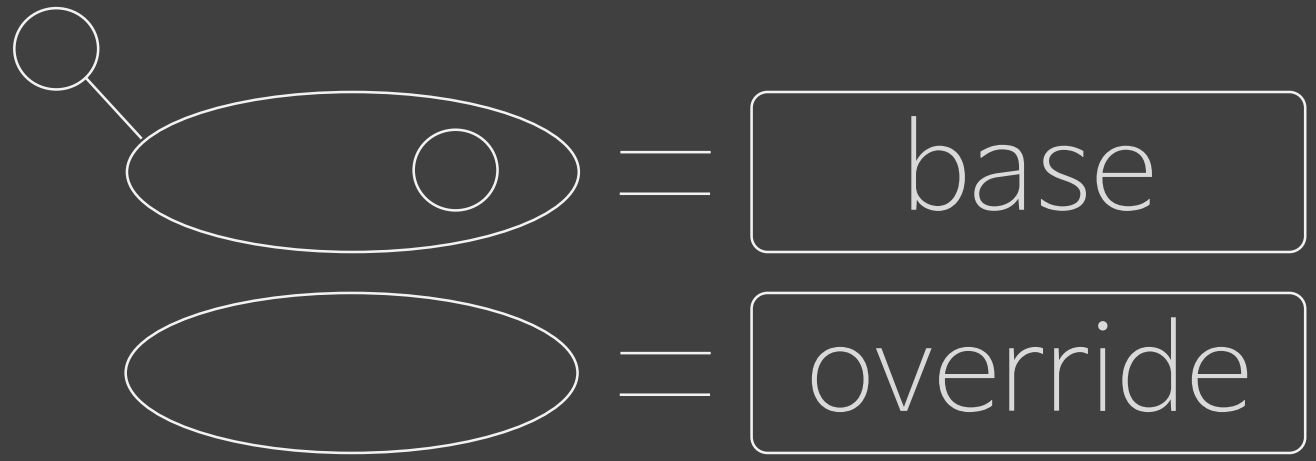
나쁜 확장 2

override는 나쁘다.



나쁜 확장 2

override는 나쁘다.



좋은 확장

좋은 확장

super는 나쁘다.

override는 나쁘다.

좋은 확장

super는 나쁘다.

부모의 모든 메소드는 final, private, abstract protected
부모의 생성자는 인자를 받지 않음

override는 나쁘다.

좋은 확장

super는 나쁘다.

부모의 모든 메소드는 final, private, abstract protected
부모의 생성자는 인자를 받지 않음

override는 나쁘다.

부모의 모든 메소드는 final, private, abstract protected

좋은 확장

부모의 모든 메소드는 다음 중 한 가지
final or private or abstract protected

부모의 생성자는 인자를 받지 않음

좋은 확장

```
abstract class Plan {
    private Set<Call> calls = new HashSet<>();
    public final void addCall(Call call){
        calls.add(call);
    }
    public final Money calculateFee(){
        Money result = Money.ZERO;
        for(Call call:calls) result = result.plus(calcCallFee(call));
        return result;
    }
    abstract protected Money calcCallFee(Call call);
}
```

좋은 확장

```
abstract class Plan {
    private Set<Call> calls = new HashSet<>();
    public final void addCall(Call call){
        calls.add(call);
    }
    public final Money calculateFee(){
        Money result = Money.ZERO;
        for(Call call:calls) result = result.plus(calcCallFee(call));
        return result;
    }
    abstract protected Money calcCallFee(Call call);
}
```

좋은 확장

```
public class PricePerTime extends Plan {
    private final Money price;
    private final Duration second;
    public PricePerTime(Money price, Duration second){
        this.price = price;
        this.second = second;
    }
    @Override
    protected Money calcCallFee(Call call) {
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

좋은 확장

```
public class PricePerTime extends Plan {
    private final Money price;
    private final Duration second;
    public PricePerTime(Money price, Duration second){
        this.price = price;
        this.second = second;
    }
    @Override
    protected Money calcCallFee(Call call) {
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

좋은 확장

```
public class NightDiscount extends Plan {
    private final Money dayPrice;
    private final Money nightPrice;
    private final Duration second;
    public NightDiscount(Money dayPrice, Money nightPrice, Duration second){
        this.dayPrice = dayPrice;
        this.nightPrice = nightPrice;
        this.second = second;
    }
    @Override
    protected Money calcCallFee(Call call) {
        Money price = call.getFrom().getHour() >= 22 ? nightPrice : dayPrice;
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

합성

```java
abstract class Plan {
    private Set<Call> calls = new HashSet<>();
    public final void addCall(Call call){
        calls.add(call);
    }
    public final Money calculateFee(){
        Money result = Money.ZERO;
        for(Call call:calls) result = result.plus(calcCallFee(call));
        return result;
    }
    abstract protected Money calcCallFee(Call call);
}
```

```java
public class Plan{
    private Calculator calc;
    private Set<Call> calls = new HashSet<>();
    public final void addCall(Call call){
        calls.add(call);
    }
    public final void setCalculator(Calculator calc){
        this.calc = calc;
    }
    public final Money calculateFee(){
        Money result = Money.ZERO;
        for(Call call:calls) result = result.plus(calc.calcCallFee(call));
        return result;
    }
}
```

```java
public interface Calculator {
    Money calcCallFee(Call call);
}
```

```java
public interface Calculator {
    Money calcCallFee(Call call);
}

public class PricePerTime extends Plan {
    private final Money price;
    private final Duration second;
    public PricePerTime(Money price, Duration second){
        this.price = price;
        this.second = second;
    }
    @Override
    protected Money calcCallFee(Call call) {
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

```java
public interface Calculator {
    Money calcCallFee(Call call);
}

public class PricePerTime implements Calculator {
    private final Money price;
    private final Duration second;
    public PricePerTime(Money price, Duration second){
        this.price = price;
        this.second = second;
    }
    @Override
    public Money calcCallFee(Call call) {
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

```java
public class NightDiscount extends Plan {
    private final Money dayPrice;
    private final Money nightPrice;
    private final Duration second;
    public NightDiscount(Money dayPrice, Money nightPrice, Duration second){
        this.dayPrice = dayPrice;
        this.nightPrice = nightPrice;
        this.second = second;
    }
    @Override
    protected Money calcCallFee(Call call) {
        Money price = call.getFrom().getHour() >= 22 ? nightPrice : dayPrice;
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

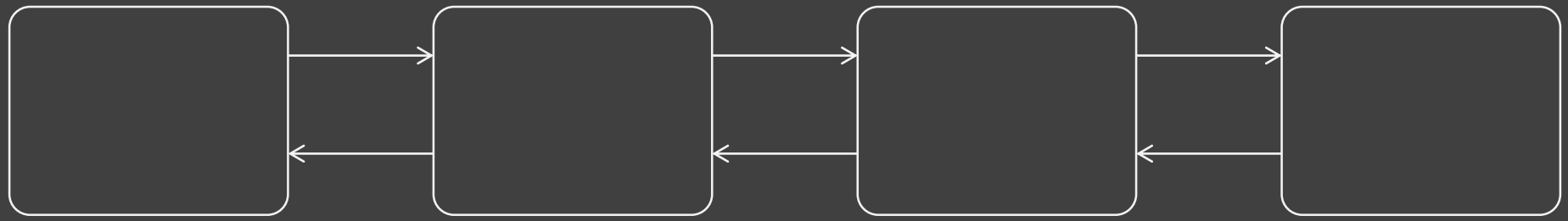
```java
public class NightDiscount implements Calculator {
    private final Money dayPrice;
    private final Money nightPrice;
    private final Duration second;
    public NightDiscount(Money dayPrice, Money nightPrice, Duration second){
        this.dayPrice = dayPrice;
        this.nightPrice = nightPrice;
        this.second = second;
    }
    @Override
    public Money calcCallFee(Call call) {
        Money price = call.getFrom().getHour() >= 22 ? nightPrice : dayPrice;
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

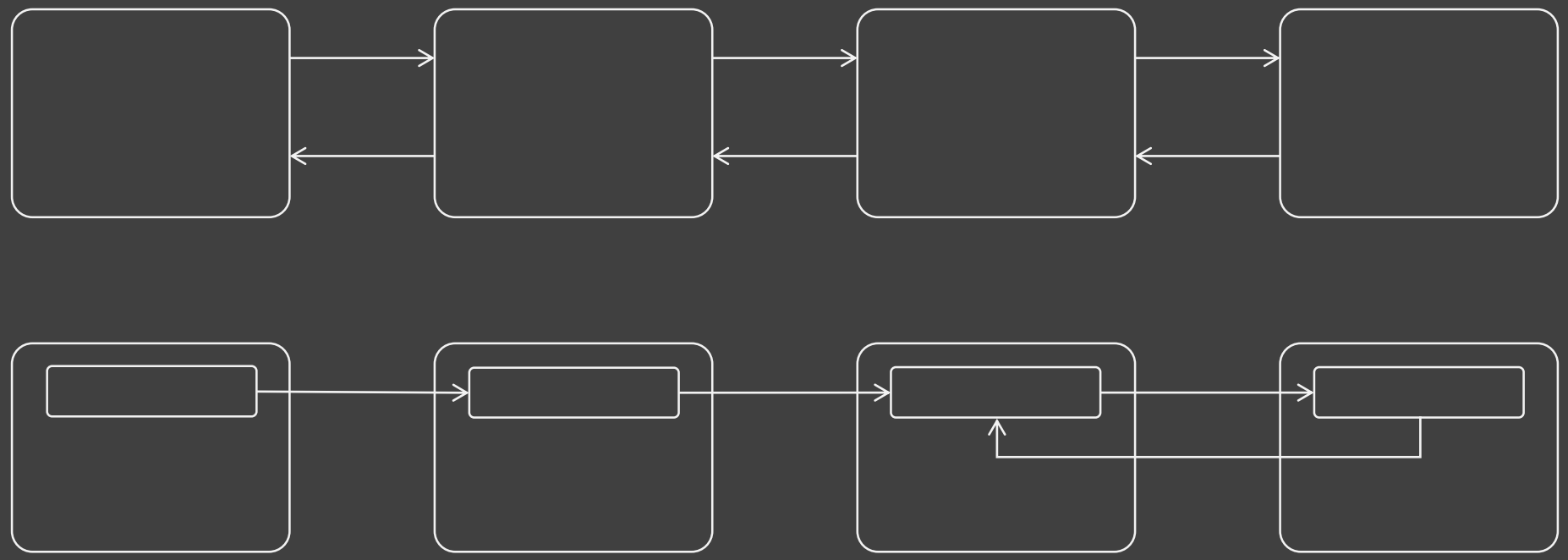
포워딩(forwarding)

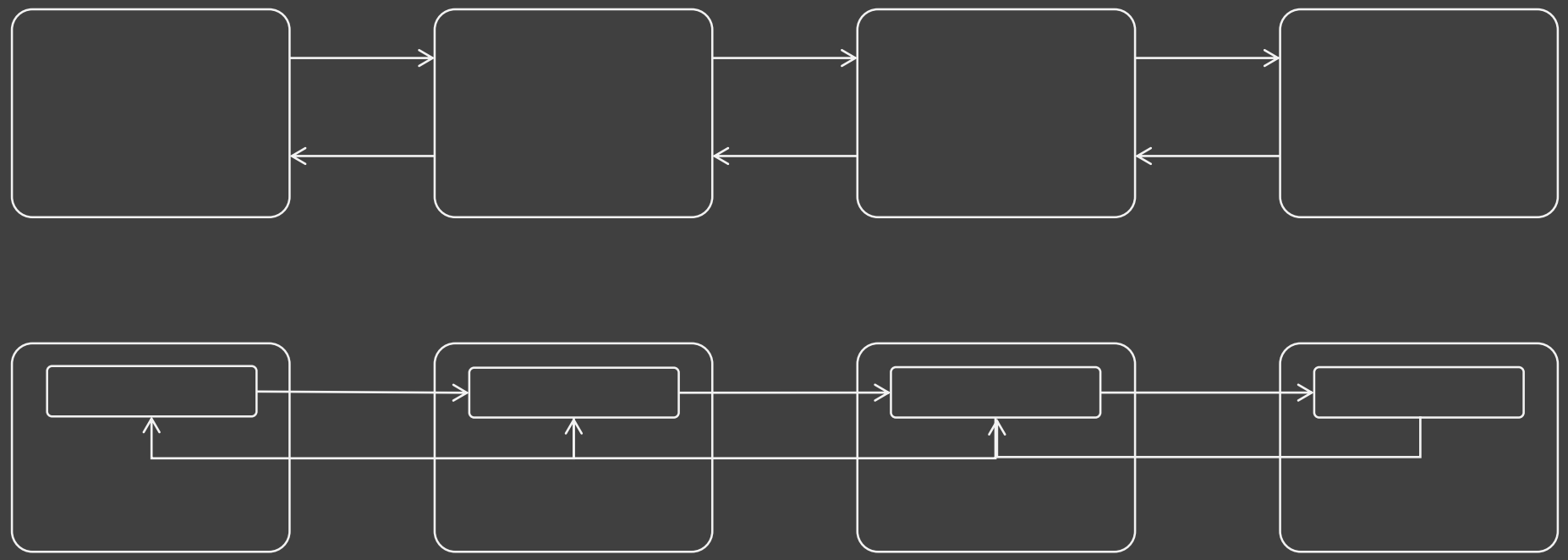
합성한 객체에게 일부의 책임을 요청하지만
컨텍스트의 공유가 없는 경우

```
public class Plan{
    ...
    public final Money calculateFee(){
        Money result = Money.ZERO;
        for(Call call:calls) result = result.plus(calc.calcCallFee(call));
        return result;
    }
}
```

연결되는 합성객체







```java
public interface Calculator {
    Money calcCallFee(Set<Call> calls, Money result);
}
```

```java
public interface Calculator {
    Money calcCallFee(Set<Call> calls, Money result);
}

public class Plan{
    private Calculator calc;
    private Set<Call> calls = new HashSet<>();
    public final void addCall(Call call){
        calls.add(call);
    }
    public final void setCalculator(Calculator calc){
        this.calc = calc;
    }
    public final Money calculateFee(){
        Money result = Money.ZERO;
        for(Call call:calls) result = result.plus(calc.calcCallFee(call));
        return result;
    }
}
```

```java
public interface Calculator {
    Money calcCallFee(Set<Call> calls, Money result);
}

public class Plan{
    private Calculator calc;
    private Set<Call> calls = new HashSet<>();
    public final void addCall(Call call){
        calls.add(call);
    }
    public final void setCalculator(Calculator calc){
        this.calc = calc;
    }
    public final Money calculateFee(){
        return calc.calcCallFee(calls, Money.ZERO);
    }
}
```

```java
public class PricePerTime implements Calculator {
    private final Money price;
    private final Duration second;
    public PricePerTime(Money price, Duration second){
        this.price = price;
        this.second = second;
    }
    @Override
    public Money calcCallFee(Call call) {
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

```java
public class PricePerTime implements Calculator {
    private final Calculator next;
    private final Money price;
    private final Duration second;
    public PricePerTime(Calculator next, Money price, Duration second){
        this.next = next;
        this.price = price;
        this.second = second;
    }
    @Override
    public Money calcCallFee(Set<Call> calls, Money result) {
        for(Call call:calls){
          result = result.plus(price.times((call.getDuration().getSeconds() / second.getSeconds())));
        }
        return next == null ? result : next.calcCallFee(calls, result);
    }
}
```

```java
public class PricePerTime implements Calculator {
    private final Calculator next;
    private final Money price;
    private final Duration second;
    public PricePerTime(Calculator next, Money price, Duration second){
        this.next = next;
        this.price = price;
        this.second = second;
    }
    @Override
    public Money calcCallFee(Set<Call> calls, Money result) {
        for(Call call:calls){
          result = result.plus(price.times((call.getDuration().getSeconds() / second.getSeconds())));
        }
        return next == null ? result : next.calcCallFee(calls, result);
    }
}
```

```java
public class NightDiscount extends Plan {
    private final Money dayPrice;
    private final Money nightPrice;
    private final Duration second;
    public NightDiscount(Money dayPrice, Money nightPrice, Duration second){
        this.dayPrice = dayPrice;
        this.nightPrice = nightPrice;
        this.second = second;
    }
    @Override
    protected Money calcCallFee(Call call) {
        Money price = call.getFrom().getHour() >= 22 ? nightPrice : dayPrice;
        return price.times((call.getDuration().getSeconds() / second.getSeconds()));
    }
}
```

```java
public class NightDiscount implements Calculator {
    private final Calculator next;
    private final Money dayPrice;
    private final Money nightPrice;
    private final Duration second;
    public NightDiscount(Calculator next, Money dayPrice, Money nightPrice, Duration second){
        this.next = next;
        this.dayPrice = dayPrice;
        this.nightPrice = nightPrice;
        this.second = second;
    }
    @Override
    public Money calcCallFee(Set<Call> calls, Money result) {
        for(Call call:calls){
            Money price = call.getFrom().getHour() >= 22 ? nightPrice : dayPrice;
            result = result.plus(price.times((call.getDuration().getSeconds() / second.getSeconds())));
        }
        return next == null ? result : next.calcCallFee(calls, result);
    }
}
```

```java
public class Tax implements Calculator {
    private final Calculator next;
    private final double ratio;
    public Tax(Calculator next, double ratio){
        this.next = next;
        this.ratio = ratio;
    }
    @Override
    public Money calcCallFee(Set<Call> calls, Money result) {
        result = result.plus(result.times(ratio));
        return next == null ? result : next.calcCallFee(calls, result);
    }
}
```

```java
public class AmountDiscount implements Calculator {
    private final Calculator next;
    private final Money amount;
    public RateDiscount(Calculator next, Money amount){
        this.next = next;
        this.amount = amount;
    }
    @Override
    public Money calcCallFee(Set<Call> calls, Money result) {
        result = result.minus(amount);
        return next == null ? result : next.calcCallFee(calls, result);
    }
}
```

```java
public static void main(String[] args) {
    Plan plan = new Plan();
    plan.setCalculator(
        new PricePerTime(
            new AmountDiscount(
                new Tax(null, 0.1),
                Money.of(10000)
            ),
            Money.of(18),
            Duration.ofSeconds(60)
        )
    );
}
```

```java
public static void main(String[] args) {
    Plan plan = new Plan();
    plan.setCalculator(
        new PricePerTime(
            new AmountDiscount(
                new Tax(null, 0.1),
                Money.of(10000)
            ),
            Money.of(18),
            Duration.ofSeconds(60)
        )
    );
}
```

중복제거 및 고도화

```java
public class AmountDiscount implements Calculator {
    private final Calculator next;
    private final Money amount;
    public RateDiscount(Calculator next, Money amount){
        this.next = next;
        this.amount = amount;
    }
    @Override
    public Money calcCallFee(Set<Call> calls, Money result) {
        result = result.minus(amount);
        return next == null ? result : next.calcCallFee(calls, result);
    }
}
```

```java
public interface Calculator {
    Money calcCallFee(Set<Call> calls, Money result);
}
```

```java
public interface Calculator {
    Money calcCallFee(Set<Call> calls, Money result);
}
```

```java
public abstract class Calculator {
    private Calculator next;
    public final Calculator setNext(Calculator next){
        this.next = next;
        return this;
    }
    public final Money calcCallFee(Set<Call> calls, Money result){
        result = calc(calls, result);
        return next == null ? result : next.calcCallFee(calls, result);
    }
    abstract protected Money calc(Set<Call> calls, Money result);
}
```

```java
public class Plan{
    private Calculator calc;
    private Set<Call> calls = new HashSet<>();
    public final void addCall(Call call){
        calls.add(call);
    }
    public final void setCalculator(Calculator calc){
        this.calc = calc;
    }
    public final Money calculateFee(){
        return calc.calcCallFee(calls, Money.ZERO);
    }
}
```

```java
public class PricePerTime extends Calculator {
    private final Money price;
    private final Duration second;
    public PricePerTime(Money price, Duration second){
        this.price = price;
        this.second = second;
    }
    @Override
    public Money calc(Set<Call> calls, Money result) {
        for(Call call:calls){
          result = result.plus(price.times((call.getDuration().getSeconds() / second.getSeconds())));
        }
        return result;
    }
}
```

```java
public class PricePerTime implements Calculator {
    private final Calculator next;
    private final Money price;
    private final Duration second;
    public PricePerTime(Calculator next, Money price, Duration second){
        this.next = next;
        this.price = price;
        this.second = second;
    }
    @Override
    public Money calcCallFee(Set<Call> calls, Money result) {
        for(Call call:calls){
          result = result.plus(price.times((call.getDuration().getSeconds() / second.getSeconds())));
        }
        return next == null ? result : next.calcCallFee(calls, result);
    }
}
```

```java
public class NightDiscount extends Calculator {
    private final Money dayPrice;
    private final Money nightPrice;
    private final Duration second;
    public NightDiscount(Money dayPrice, Money nightPrice, Duration second){
        this.dayPrice = dayPrice;
        this.nightPrice = nightPrice;
        this.second = second;
    }
    @Override
    public Money calc(Set<Call> calls, Money result) {
        for(Call call:calls){
            Money price = call.getFrom().getHour() >= 22 ? nightPrice : dayPrice;
            result = result.plus(price.times((call.getDuration().getSeconds() / second.getSeconds())));
        }
        return result;
    }
}
```

```java
public class Tax extends Calculator {
    private final double ratio;
    public Tax(double ratio){this.ratio = ratio;}
    @Override
    public Money calc(Set<Call> calls, Money result) {
        return result.plus(result.times(ratio));
    }
}

public class AmountDiscount extends Calculator {
    private final Money amount;
    public AmountDiscount(Money amount){this.amount = amount;}
    @Override
    public Money calc(Set<Call> calls, Money result) {
        return result.minus(amount);
    }
}
```

```java
public static void main(String[] args) {
    Plan plan = new Plan();
    plan.setCalculator(
        new PricePerTime(Money.of(18), Duration.ofSeconds(60))
        .setNext(new AmountDiscount(Money.of(10000)))
        .setNext(new Tax(0.1))
    );
}
```

```java
public static void main(String[] args) {
    Plan plan = new Plan();
    plan.setCalculator(
        new PricePerTime(
            new AmountDiscount(
                new Tax(null, 0.1),
                Money.of(10000)
            ),
            Money.of(18),
            Duration.ofSeconds(60)
        )
    );
}
```

다시 합성으로

```java
public abstract class Calculator {
    private Calculator next;
    public final Calculator setNext(Calculator next){
        this.next = next;
        return this;
    }
    public final Money calcCallFee(Set<Call> calls, Money result){
        result = calc(calls, result);
        return next == null ? result : next.calcCallFee(calls, result);
    }
    abstract protected Money calc(Set<Call> calls, Money result);
}
```

```java
public class Calculator {
    private Set<Calc> calcs = new HashSet<>();
    public Calculator(Calc calc){
        calcs.add(calc);
    }
    public final Calculator setNext(Calc next){
        calcs.add(next);
        return this;
    }
    public Money calcCallFee(Set<Call> calls, Money result){
        for(Calc calc:calcs) result = calc.calc(calls, result);
        return result;
    }
}
```

```java
public interface Calc{Money calc(Set<Call> calls, Money result);}

public class Tax implements Calc{
    private final double ratio;
    public Tax(double ratio){
        this.ratio = ratio;
    }
    @Override
    public Money calc(Set<Call> calls, Money result) {
        return result.plus(result.times(ratio));
    }
}

public class AmountDiscount implements Calc{
    private final Money amount;
    public AmountDiscount(Money amount){this.amount = amount;}
    @Override
    public Money calc(Set<Call> calls, Money result) {
        return result.minus(amount);
    }
}
```

```java
public class PricePerTime implements Calc{
    private final Money price;
    private final Duration second;
    public PricePerTime(Money price, Duration second){
        this.price = price;
        this.second = second;
    }
    @Override
    public Money calc(Set<Call> calls, Money result) {
        for(Call call:calls){
          result = result.plus(price.times((call.getDuration().getSeconds() / second.getSeconds())));
        }
        return result;
    }
}
```

```java
public class NightDiscount implements Calc{
    private final Money dayPrice;
    private final Money nightPrice;
    private final Duration second;
    public NightDiscount(Money dayPrice, Money nightPrice, Duration second){
        this.dayPrice = dayPrice;
        this.nightPrice = nightPrice;
        this.second = second;
    }
    @Override
    public Money calc(Set<Call> calls, Money result) {
        for(Call call:calls){
            Money price = call.getFrom().getHour() >= 22 ? nightPrice : dayPrice;
            result = result.plus(price.times((call.getDuration().getSeconds() / second.getSeconds())));
        }
        return result;
    }
}
```

```java
public static void main(String[] args) {
    Plan plan = new Plan();
    plan.setCalculator(
        new Calculator(new PricePerTime(Money.of(18), Duration.ofSeconds(60)))
            .setNext(new AmountDiscount(Money.of(10000)))
            .setNext(new Tax(0.1))
    );
}
```

```java
public class Plan{
    private Calculator calc;
    private Set<Call> calls = new HashSet<>();
    public final void addCall(Call call){
        calls.add(call);
    }
    public final void setCalculators(Calc... calcs){
        boolean isFirst = true;
        for(Calc calc:calcs){
            if(isFirst){
                isFirst = false;
                this.calc = new Calculator(calcs[0]);
            }else{
                this.calc.setNext(calc);
            }
        }
    }
    public final Money calculateFee(){
        return calc.calcCallFee(calls, Money.ZERO);
    }
}
```

```java
public static void main(String[] args) {
    Plan plan = new Plan();
    plan.setCalculator(
        new PricePerTime(Money.of(18), Duration.ofSeconds(60)),
        new AmountDiscount(Money.of(10000)),
        new Tax(0.1)
    );
}
```