KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

Chain of Responsibility - 책임을 떠넘긴다

[KB] IT's Your Life

KB 국민은행

Chain of Responsibility 패턴

- 어떤 요청이 있을 때, 그 요청을 처리할 객체(오브젝트)를 고정적으로 결정할 수 없는 경우
- 여러 객체를 사슬(chain) 처럼 연쇄적으로 묶고, 객체 사슬을 차례대로 돌면서 원하는 객체를 결정하는 방법
- 요청하는 쪽과 처리하는 쪽의 결합을 약하게 할 수 있고, 각각 부품으로 독립시킬 수 있음
- 상황에 따라 요청을 처리할 객체가 변하는 프로그램에도 대응 가능

- 어떤 사람에게 요청이 들어온 경우, 그 사람이 요청을 처리할 수 있으면 처리하고, 처리할 수 없을 때는 그 요청을 다음 사람에게 떠넘김. 이 처리를 반복 → 책임 사슬(Chain of Responsibility) 패턴

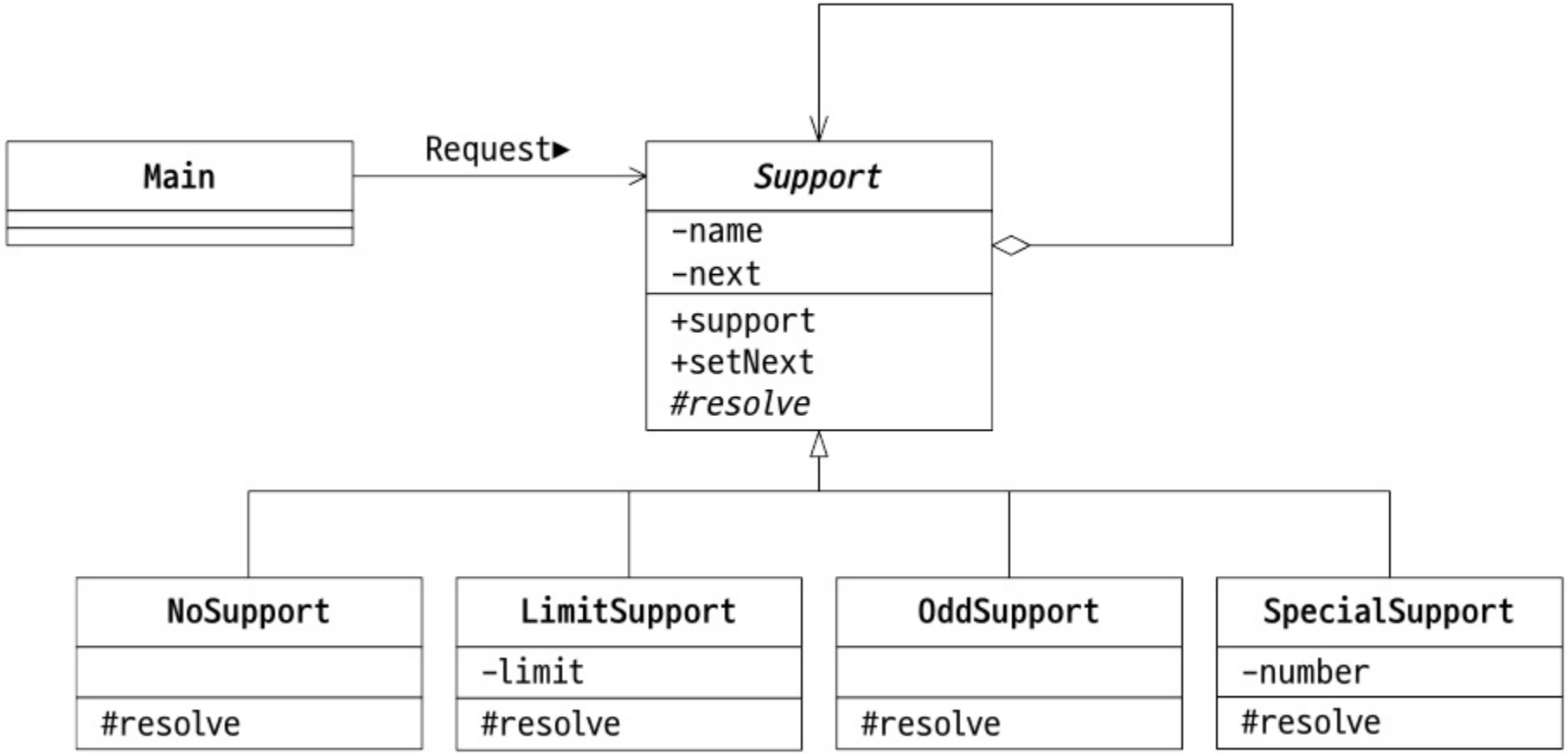
예제 프로그램

- 트러블이 발생했을 때 누군가가 해결해야 하는 상황

이름	설명
Trouble	발생한 트러블을 나타내는 클래스. 트러블 번호(number)를 갖는다
Support	트러블을 해결하는 추상 클래스
NoSupport	트러블을 해결하는 구상 클래스(항상 '처리하지 않음')
LimitSupport	트러블을 해결하는 구상 클래스(지정한 번호 미만의 트러블 해결)
OddSupport	트러블을 해결하는 구상 클래스(홀수 번호 트러블 해결)
SpecialSupport	트러블을 해결하는 구상 클래스(특정 번호 트러블 해결)
Main	Support의 연쇄를 만들어 트러블을 일으키는 동작 테스트용 클래스

예제 프로그램

예제 프로그램 클래스 다이어그램



Trouble.java

```
public class Trouble {
    private int number; // 트러블 번호

    public Trouble(int number) {
        this.number = number;
    }

    public int getNumber() {
        return number;
    }

    @Override
    public String toString() {
        return "[Trouble " + number + "]";
    }
}
```

Support.java

```
public abstract class Support {
    private String name;    // 이 트러블 해결자 이름
    private Support next;   // 떠넘길 곳

    public Support(String name) {
        this.name = name;
        this.next = null;
    }

    // 떠넘길 곳을 설정한다.
    public Support setNext(Support next) {
        this.next = next;
        return next;
    }

    // 트러블 해결 절차를 결정한다.
    public void support(Trouble trouble) {
        if(resolve(trouble)) {
            done(trouble);
        } else if(next != null) {
            next.support(trouble);
        } else {
            fail(trouble);
        }
    }
```

Support.java

```
    @Override
    public String toString() {
        return "[" + name + "]";
    }

    // 해결하려고 한다.
    protected abstract boolean resolve(Trouble trouble);


    // 해결했다.
    private void done(Trouble trouble) {
        System.out.println(trouble + " is resolved by " + this + ".");
    }

    // 해결되지 않았다.
    private void fail(Trouble trouble) {
        System.out.println(trouble + " cannot be resolved.");
    }
}
```

NoSupport.java

```
public class NoSupport extends Support {

    public NoSupport(String name) {
        super(name);
    }

    @Override
    protected boolean resolve(Trouble trouble) {
        return false;   // 자신은 아무 것도 해결하지 않는다.
    }
}
```

LimitSupport.java

```
public class LimitSupport extends Support {
    private int limit;  // 이 번호 미만이면 해결할 수 있다.

    public LimitSupport(String name, int limit) {
        super(name);
        this.limit = limit;
    }

    @Override
    protected boolean resolve(Trouble trouble) {
        if(trouble.getNumber() < limit) {
            return true;
        } else {
            return false;
        }
    }
}
```

OddSupport.java

```
public class OddSupport extends Support {

    public OddSupport(String name) {
        super(name);
    }

    @Override
    protected boolean resolve(Trouble trouble) {
        if(trouble.getNumber() % 2 == 1) {
            return true;
        } else {
            return false;   // 자신은 아무 것도 해결하지 않느다.
        }
    }
}
```

SpecialSupport.java

```
public class SpecialSupport extends Support {
    private int number;  // 이 번호만 해결할 수 있다.

    public SpecialSupport(String name, int number) {
        super(name);
        this.number = number;
    }

    @Override
    protected boolean resolve(Trouble trouble) {
        if(trouble.getNumber() == number) {
            return true;
        } else {
            return false;
        }
    }
}
```

예제 프로그램

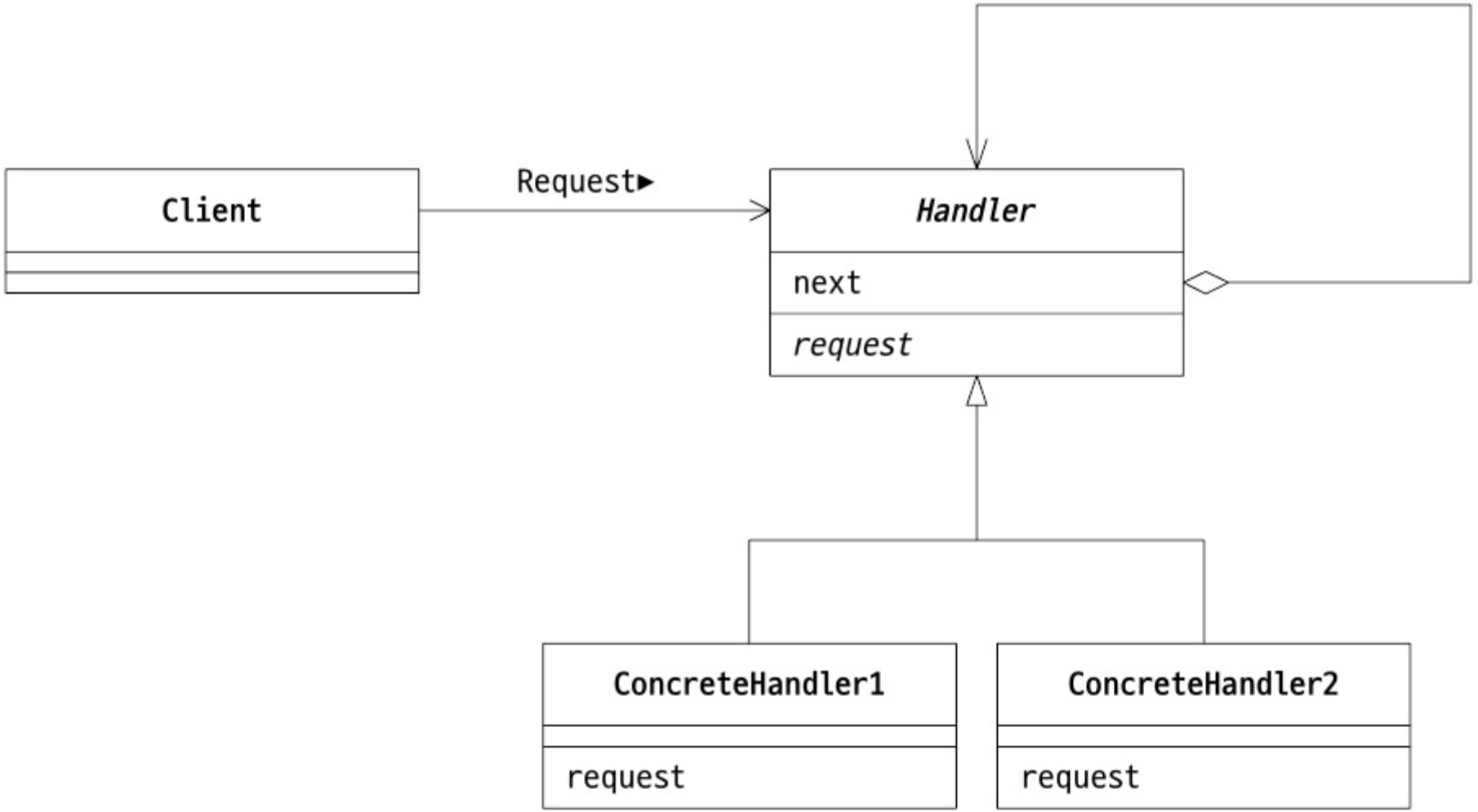
Main

```
public class Main {
    public static void main(String[] args) {
        Support alice = new NoSupport("Alice");
        Support bob = new LimitSupport("Bob", 100);
        Support charlie = new SpecialSupport("Charlie", 429);
        Support diana = new LimitSupport("Diana", 200);
        Support elmo = new OddSupport("Elmo");
        Support fred = new LimitSupport("Fred", 300);

        // 사슬 형성
        alice.setNext(bob)
                .setNext(charlie)
                .setNext(diana)
                .setNext(elmo)
                .setNext(fred);

        // 다양한 트러블 발생
        for (int i = 0; i < 500; i += 33) {
            alice.support(new Trouble(i));
        }
    }
}
```

```
[Trouble 0] is resolved by [Bob].
[Trouble 33] is resolved by [Bob].
[Trouble 66] is resolved by [Bob].
[Trouble 99] is resolved by [Bob].
[Trouble 132] is resolved by [Diana].
[Trouble 165] is resolved by [Diana].
[Trouble 198] is resolved by [Diana].
[Trouble 231] is resolved by [Elmo].
[Trouble 264] is resolved by [Fred].
[Trouble 297] is resolved by [Elmo].
[Trouble 330] cannot be resolved.
[Trouble 363] is resolved by [Elmo].
[Trouble 396] cannot be resolved.
[Trouble 429] is resolved by [Charlie].
[Trouble 462] cannot be resolved.
[Trouble 495] is resolved by [Elmo].
```

Chain of Responsibility 패턴

Chain of Responsibility 패턴 클래스 다이어그램



예제 프로그램

시퀀스 다이어그램