시스템 분석및 설계_ChainOfResponsibility

작성자	20141640 이석원	제출일	2019.05.3
담당 교수님	권기태	제출기한	2019.5.7

클래스의 종류와 하는일

Trouble

```
package ChainOfResponsibility;

public class Trouble {///(발생한 트러블을 나타내는 클래스. 트러블 번호를 가진다.)
    private int number;     //트러블 번호 디렉토리를 방문할 때의 처리를 하위클래스에 강제한다.
    public Trouble(int number) {
        this.number = number;
    }
    public int getNumber() {     //number의 getter
        return number;
    }
    public String toString() {       //트러블의 문자열(클래스명) 표현
        return "[Trouble " + number + "]";
    }
}
```

Support

```
package ChainOfResponsibility;

public abstract class Support { //(트러블을 해결하는 추상 클래스)
    private String name;      //     트러블 해결자의 이름
    private Support next; //떠넘기는 곳
    public Support(String name) {
        this.name = name;
    }
    public Support setNext(Support next) {//떠넘기는 곳을 지정
        this.next = next;
        return next;
    }
    public final void support(Trouble trouble) { //트러블 해결 템플릿 메소드 형태
        if (resolve(trouble)) {
            done(trouble);
        } else if (next != null) {
            next.support(trouble);
        } else {
            fail(trouble);
        }
```

```
    }
    public String toString() {
        return "[" + name + "]";
    }
    protected abstract boolean resolve(Trouble trouble); //해결용 메소드
    protected void done(Trouble trouble) { //해결
        System.out.println(trouble + " is resolved by " + this + ".");
    }
    protected void fail(Trouble trouble) {// 미해결
        System.out.println(trouble + " cannot be resolved.");
    }
}
```

NoSupport

```
package ChainOfResponsibility;

public class NoSupport extends Support {
    public NoSupport(String name) {
        super(name);
    }
    protected boolean resolve(Trouble trouble) {     // 해결용 메소드를 구현한다.
        return false;                                // 해결하지 않고 다음 으로 넘긴다.
    }
}
```

LimitSupport

```
package ChainOfResponsibility;

public class LimitSupport extends Support {
    private int limit;                          //이번호 미만이면 해결할 수 있다.
    public LimitSupport(String name, int limit) {//생성자
        super(name);
        this.limit = limit;
    }
    protected boolean resolve(Trouble trouble) { //해결용 메소드
        if (trouble.getNumber() < limit) { //숫자를 호출해서 limit미만이면 해결한다.
            return true;
        } else {
            return false;
        }
    }
}
```

OddSupport

```
package ChainOfResponsibility;
```

```
public class OddSupport extends Support { //생성자
    public OddSupport(String name) {
        super(name);
    }
    protected boolean resolve(Trouble trouble) {     //해결용 메소드
        if (trouble.getNumber() % 2 == 1) { //홀수이면 해결한다.
            return true;
        } else {
            return false;
        }
    }
}
```

SpecialSupport

```
package ChainOfResponsibility;

public class SpecialSupport extends Support {
    private int number;                          // 이번호만 해결할 수 있다.
    public SpecialSupport(String name, int number) {    //생성자
        super(name);
        this.number = number;
    }
    protected boolean resolve(Trouble trouble) {
        if (trouble.getNumber() == number) {//number와 같으면 해결
            return true;
        } else {
            return false;
        }
    }
}
```

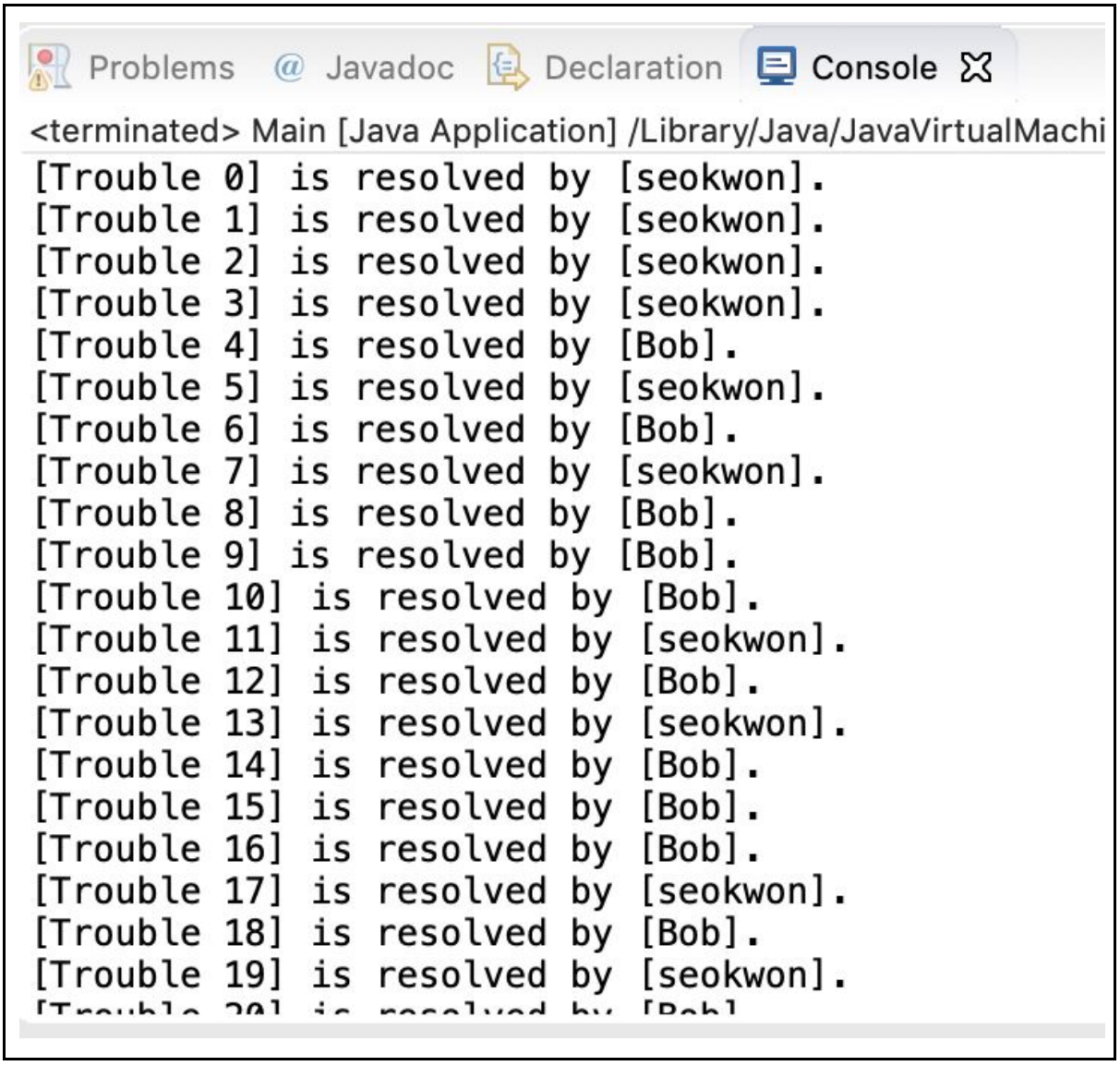
HwSupport

```
package ChainOfResponsibility;

public class HwSupport extends Support{

        public HwSupport(String name) {
                super(name);
                // TODO Auto-generated constructor stub
        }
        @Override
        protected boolean resolve(Trouble trouble) {
                // TODO Auto-generated method stub
                return calcPrimeNumber(trouble.getNumber()); //번호를 받아온다.

        }
        protected boolean calcPrimeNumber(int max) {
                int i;
                for(i = 2;i <= max; i++) {//소수인지 max까지 찾아본다.
```

```
                        if(max%i == 0) {//나눠지면 소수가 아니므로 바로 탈출
                                break;
                        }
                }
                if(i >= max) {
                        return true;
                }
                else {
                        return false;
                }
        }
}
```

Main

```
package jsvaDP;

import java.io.IOException;

import ChainOfResponsibility.*;
public class Main {

        public static void main(String[] args) {
                // TODO Auto-generated method stub

                /*Chain of Responsibility*/
                Support seokwon = new HwSupport("seokwon");
                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);

seokwon.setNext(alice).setNext(bob).setNext(charlie).setNext(diana).setNext(elmo).setNe
xt(fred);
                for (int i = 0; i < 500; i += 1) { //33의 배수로 반복문을 실행하면 소수가
걸러지지 않습니다.
                        seokwon.support(new Trouble(i));//hyeja부터 실행합니다.
                }
        }
}
```


Problems
Javadoc
Declaration
Console
<terminated> Main [Java Application] /Library/Java/JavaVirtualMachi
[Trouble 0] is resolved by [seokwon].
[Trouble 1] is resolved by [seokwon].
[Trouble 2] is resolved by [seokwon].
[Trouble 3] is resolved by [seokwon].
[Trouble 4] is resolved by [Bob].
[Trouble 5] is resolved by [seokwon].
[Trouble 6] is resolved by [Bob].
[Trouble 7] is resolved by [seokwon].
[Trouble 8] is resolved by [Bob].
[Trouble 9] is resolved by [Bob].
[Trouble 10] is resolved by [Bob].
[Trouble 11] is resolved by [seokwon].
[Trouble 12] is resolved by [Bob].
[Trouble 13] is resolved by [seokwon].
[Trouble 14] is resolved by [Bob].
[Trouble 15] is resolved by [Bob].
[Trouble 16] is resolved by [Bob].
[Trouble 17] is resolved by [seokwon].
[Trouble 18] is resolved by [Bob].
[Trouble 19] is resolved by [seokwon].