- 다음은 배터리의 양을 관리하는 3개 클래스의 소스 코드를 보여준다.

```java
class Battery {
    private int level = 100 ;
    private BatteryLevelDisplay display ;
    private LowBatteryWarning warning ;

    public void setDisplay(BatteryLevelDisplay display) {
        this.display = display ;
    }
    public void setWarning(LowBatteryWarning warning) {
        this.warning = warning ;
    }
    public void consume(int amount) {
        level -= amount ;

        display.update() ;
        warning.update() ;
    }
    public int getLevel() { return level ; }
}
```

```java
class BatteryLevelDisplay  {
    private Battery battery ;
    public BatteryLevelDisplay(Battery battery) {
        this.battery = battery ;
    }
    public void update() {
        int level = battery.getLevel() ;
        System.out.println("Level: " + level) ;
    }
}
```

```java
class LowBatteryWarning {
    private static final int LOW_BATTERY = 30 ;
    private Battery battery ;
    public LowBatteryWarning(Battery battery) {
        this.battery = battery ;
    }
    public void update() {
        int level = battery.getLevel() ;
        if ( level < LOW_BATTERY )
            System.out.println("<Warning> Low Battery : " + level + " Compared with " + LOW_BATTERY) ;
    }
}
```

- 다음 코드의 실행 결과는?

```
public class Main {
    public static void main(String[] args) {
        Battery battery = new Battery() ;

        BatteryLevelDisplay batteryDisplay = new BatteryLevelDisplay(battery) ;
        LowBatteryWarning lowBatteryWarning = new LowBatteryWarning(battery) ;

        battery.setDisplay(batteryDisplay) ;
        battery.setWarning(lowBatteryWarning) ;

        battery.consume( amount: 20) ;
        battery.consume( amount: 50) ;
        battery.consume( amount: 10) ;
    }
}
```

- 이 코드의 문제점은?

```
class Battery {
    private int level = 100 ;
    private BatteryLevelDisplay display ;
    private LowBatteryWarning warning ;

    public void setDisplay(BatteryLevelDisplay display) {
        this.display = display ;
    }
    public void setWarning(LowBatteryWarning warning) {
        this.warning = warning ;
    }
    public void consume(int amount) {
        level -= amount ;

        display.update() ;
        warning.update() ;
    }
    public int getLevel() { return level ; }
}
```

```
class BatteryLevelDisplay  {
    private Battery battery ;
    public BatteryLevelDisplay(Battery battery) {
        this.battery = battery ;
    }
    public void update() {
        int level = battery.getLevel() ;
        System.out.println("Level: " + level) ;
    }
}
```

```
class LowBatteryWarning {
    private static final int LOW_BATTERY = 30 ;
    private Battery battery ;
    public LowBatteryWarning(Battery battery) {
        this.battery = battery ;
    }
    public void update() {
        int level = battery.getLevel() ;
        if ( level < LOW_BATTERY )
            System.out.println("<Warning> Low Battery : " + level + " Compared with " + LOW_BATTERY) ;
    }
}
```

- 옵서버 패턴을 사용해 코드의 문제점을 개선하라.