

소프트웨어 분석 설계

12주차

20170677 오윙택

1. 연습문제 1번.

1-1 각 클래스의 기능 설명하기

Battery.java

```
package Week12.example1;

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

    public void setDisplay(BatteryLevelDisplay display) {
        this.display = display;
    }

    public void setLowBatteryWarning(LowBatteryWarning warning) {
        this.warning = warning;
    }

    public void consume(int amount) {
        level -= amount;
        display.update();
        warning.update();
    }

    public int getLevel() {
        return level;
    }
}
```

배터리 객체를 생성하는 클래스, 최댓값으로 100의 값을 갖는 Level, Display와 warning 상태를 가지고 있고, consume이 실행될 때 마다 level의 값이 차감되고, display와 warning의 상태를 업데이트한다.

BatteryLevelDisplay.java

배터리의 잔여 level을 사용자에게 보여주는 클래스

```
package Week12.example1;

public class BatteryLevelDisplay {
    private Battery battery;

    public BatteryLevelDisplay (Battery battery) {
        this.battery = battery;
    }

    public void update(){
        int level = battery.getLeve();
        System.out.println("Level: " + level);
    }
}
```

LowBatteryWarning.java

특정 조건(Battery의 잔여 level)에 따라 사용자에게 print문으로 경고 알림을 표시해주는 클래스.

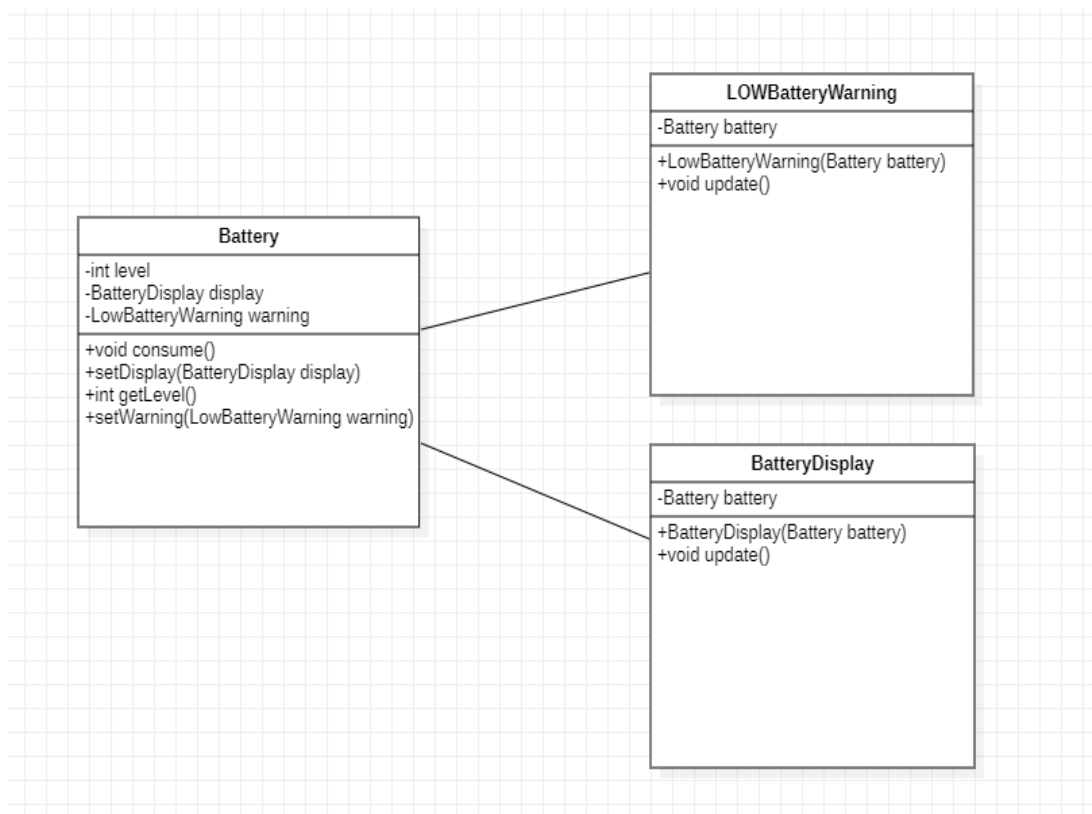
```
package Week12.example1;

public 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.getLeve();
        if( level < LOW_BATTERY){
            System.out.println("<WARNING> Low Battery: " + level + "Compared with " + LOW_BATTERY);
        }
    }
}
```

1-2 3개 클래스를 클래스 다이어그램으로 표현하기



1-3 Client.java의 실행 결과 작성하기

```
package Week12.example1;

public class Client {
    public static void main(String[] args) {
        Battery battery = new Battery();
        BatteryLevelDisplay display = new BatteryLevelDisplay(battery);
        LowBatteryWarning warning = new LowBatteryWarning(battery);

        battery.setDisplay(display);
        battery.setLowBatteryWarning(warning);

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

The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure, with `Client` selected under `Week12.example1`. The main editor shows the `Client.java` code. The bottom panel shows the execution output:

```
Run: Week12.example1.Client
"C:\Program Files\Java\jdk-18.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commu
Level: 80
Level: 30
Level: 20
<WARNING> Low Battery: 20Compared with 30
Process finished with exit code 0
```

1-4 Battery클래스의 설계 취약점 OCP측면에서 위반한 부분을 명시하고 설명하기.

```
package Week12.example1;

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

    public void setDisplay(BatteryLevelDisplay display) {
        this.display = display;
    }

    public void setLowBatteryWarning(LowBatteryWarning warning) {
        this.warning = warning;
    }

    public void consume(int amount) {
        level -= amount;
        display.update();
        warning.update();
    }

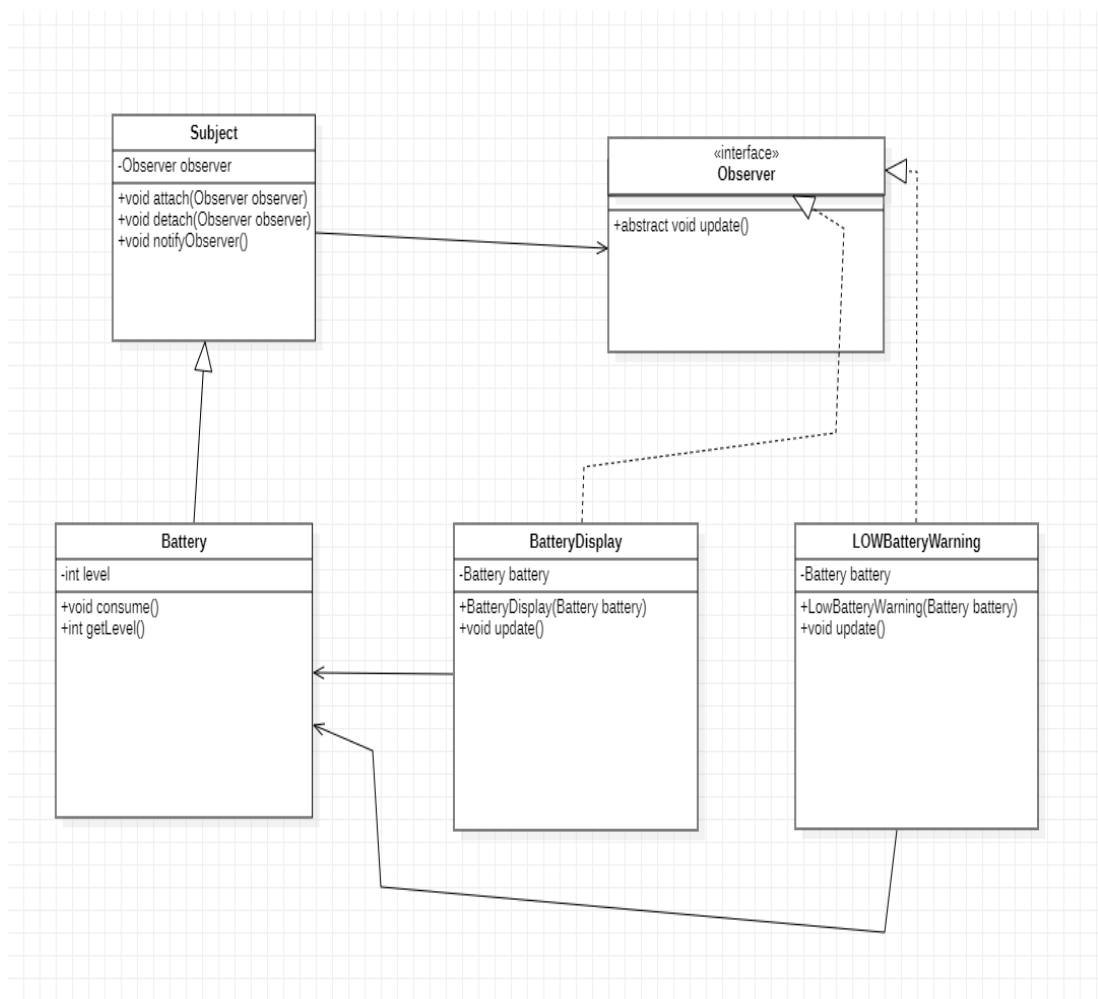
    public int getLevel() {
        return level;
    }
}
```

우선 Battery코드는 BatteryLevelDisplay 클래스와 LowBatteryWarning클래스와 직접 연관되어 level속성을 update하고 있다. 이는 의존 역전 원칙을 위반하는것이면서 동시에 OCP의 위반이기도 하다. 만약 display나 warning이 아닌 새로운 기능 (EmptyBatteryWarning)과 같은 기능이 추가된다면 consume메서드에 Empty.update()와 같은 수정사항이 필요해 지게 된다.

1-5 옵서버 패턴을 활용해 Battery클래스의 설계 취약점을 개선하기

옵서버 패턴에서는 인스턴스가 가지고 있는 속성들의 변경을 감지하고, 인터페이스를 통해 만들어진 특정 기능을 가진 클래스에 통보하는 기능을 가지고 있는 추상클래스 subject를 통해 간접적으로 인스턴스의 속성들을 관리한다. 옵서버에서는 attach와 detach를 통해 기능들을 붙이거나 떼어내어 관리할 수 있는 장점이 있고, subject에서 인터페이스에게 속성의 변경을 통보하기 때문에 attach되어 있는 클래스들에 일괄적으로 통보할 수 있다. 여기서 attach되어 있는 클래스들에 일괄통보가 가능한 이유는 interface를 통해 상속을 받아 구현되었기 때문에 부모-자식관계가 형성되어 상속되어 같은 메시지를 반드시 가져야 하기 때문이다.

1-6 옵서버 패턴을 통해 개선된 설계를 클래스 다이어그램으로 표현하기



1-7 클래스 명시하기

Observer : BatteryDisplay, LowBatteryWarning클래스를 명시하기 위한 interface

BatteryDisplay/LowBatteryWarning : Observer를 상속받아 특정한 기능을 수행하는 구체화된 class

Subject : Battery클래스에서 생성된 객체가 Interface를 통해 구현된 BatteryDisplay, LowBatteryWarning 클래스 사이에 직접적으로 상호작용하는 것을 피하고, attach, detach 메서드를 통해 클래스들을 동적으로 추가하거나 제거하기 위한 추상클래스

Battery : Subject를 Extends 받아 생성된 Battery 인스턴스를 생성하는 구체화된 클래스

1-8 옵서버 패턴을 위한 클래스 코드 작성

Observer.interface

```
package Week12.example1;

public interface Observer {
    public abstract void update();
}
```

Subject.java

```
package Week12;

import java.util.ArrayList;
import java.util.List;

public abstract class Subject {
    private List<Observer> observers = new ArrayList<Observer>();

    public void attach (Observer observer) { observers.add(observer); }

    public void detach (Observer observer){
        observers.remove(observer);
    }

    public void notifyObservers(){
        for (Observer o: observers) {
            o.update();
        }
    }
}
```

Battery.java

```
package Week12.example1;

public class BatteryLevelDisplay implements Observer{
    private Battery battery;

    public BatteryLevelDisplay (Battery battery) { this.battery = battery; }

    public void update(){
        int level = battery.getLeve();
        System.out.println("Level: " + level);
    }
}
```

BatteryLevelDisplay.java

```
1 package Week12.example1;
2
3 public class Battery extends Subject {
4     private int level = 100;
5
6     public void consume(int amount) {
7         level -= amount;
8         notifyObservers();
9     }
10
11     public int getLeve() { return level; }
12
13 }
14
15
```

LowBatteryWarning.java

```
package Week12.example1;

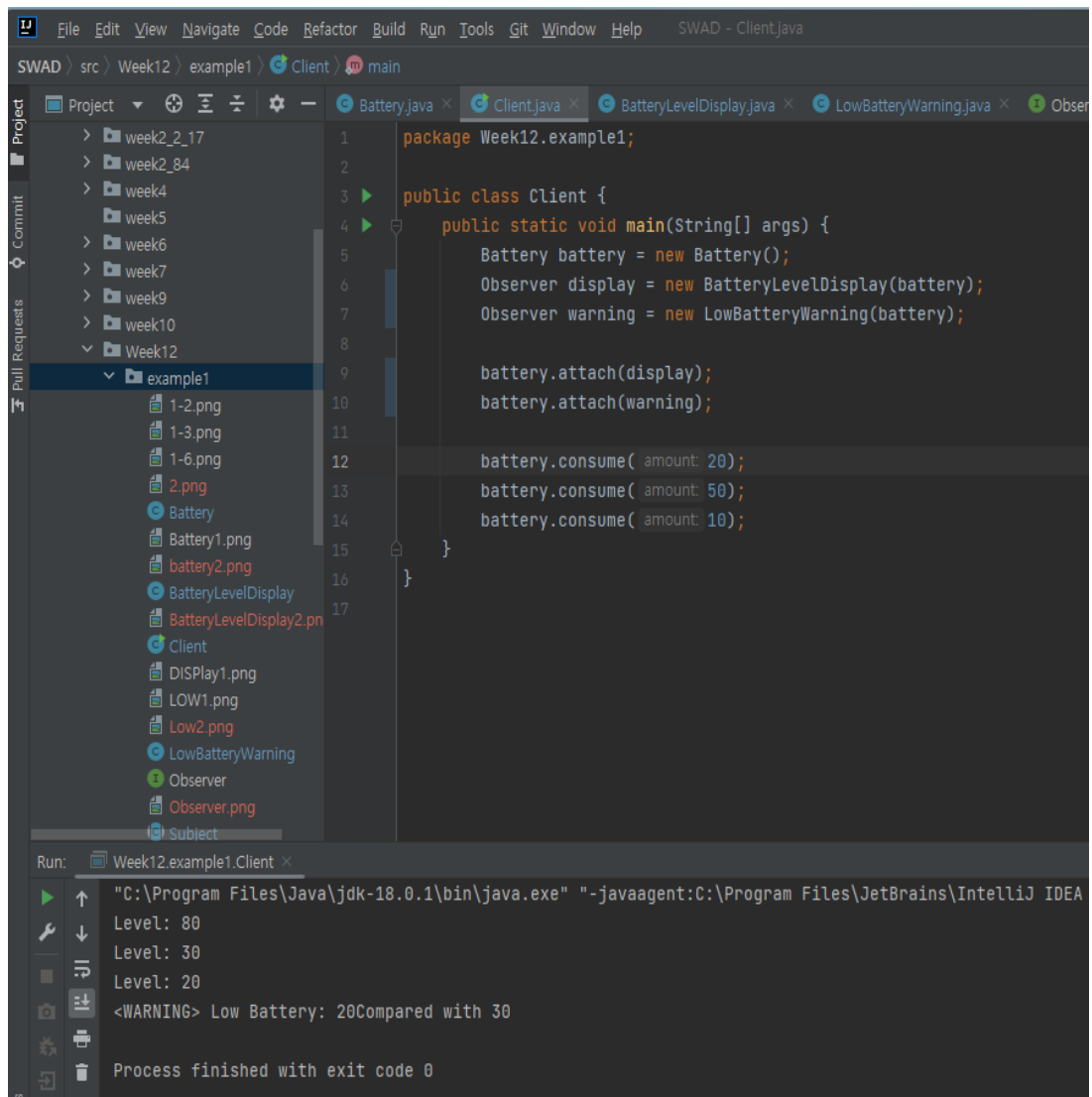
public class LowBatteryWarning implements Observer{
    private static final int LOW_BATTERY = 30;
    private Battery battery;

    public LowBatteryWarning(Battery battery) {
        this.battery = battery;
    }

    public void update(){
        int level = battery.getLeve();
        if( level < LOW_BATTERY){
            System.out.println("<WARNING> Low Battery: " + level + "Compared with " + LOW_BATTERY);
        }
    }
}
```

1-9 Client코드 재작성하기

Client.java



The screenshot displays the IntelliJ IDEA IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, and Help. The breadcrumb navigation shows the path: SWAD > src > Week12 > example1 > Client > main. The left sidebar shows the Project view with a tree structure of folders (week2_2_17, week2_84, week4, week5, week6, week7, week9, week10, Week12) and files (1-2.png, 1-3.png, 1-6.png, 2.png, Battery, Battery1.png, battery2.png, BatteryLevelDisplay, BatteryLevelDisplay2.png, Client, Display1.png, LOW1.png, Low2.png, LowBatteryWarning, Observer, Observer.png, Subject). The main editor window shows the code for Client.java, which is part of the package Week12.example1. The code defines a public class Client with a static void main method. Inside main, a Battery object is created, and two Observer objects (BatteryLevelDisplay and LowBatteryWarning) are created and attached to the battery. The battery then consumes energy in three steps: 20, 50, and 10 units. The bottom status bar shows the Run configuration for Week12.example1.Client. The Run output window displays the execution results, including the battery level (80, 30, 20) and a warning message: <WARNING> Low Battery: 20Compared with 30. The process finished with exit code 0.

```
package Week12.example1;

public class Client {
    public static void main(String[] args) {
        Battery battery = new Battery();
        Observer display = new BatteryLevelDisplay(battery);
        Observer warning = new LowBatteryWarning(battery);

        battery.attach(display);
        battery.attach(warning);

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

Run: Week12.example1.Client x

"C:\Program Files\Java\jdk-18.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA

Level: 80
Level: 30
Level: 20
<WARNING> Low Battery: 20Compared with 30
Process finished with exit code 0

1-10 개인 문제 정의하기

배터리가 0이 되었을 때 경고를 표시하는 EmptyBatteryWarning 클래스,
배터리 충전기능을 구현하기 위한 Battery.java – charge()메서드

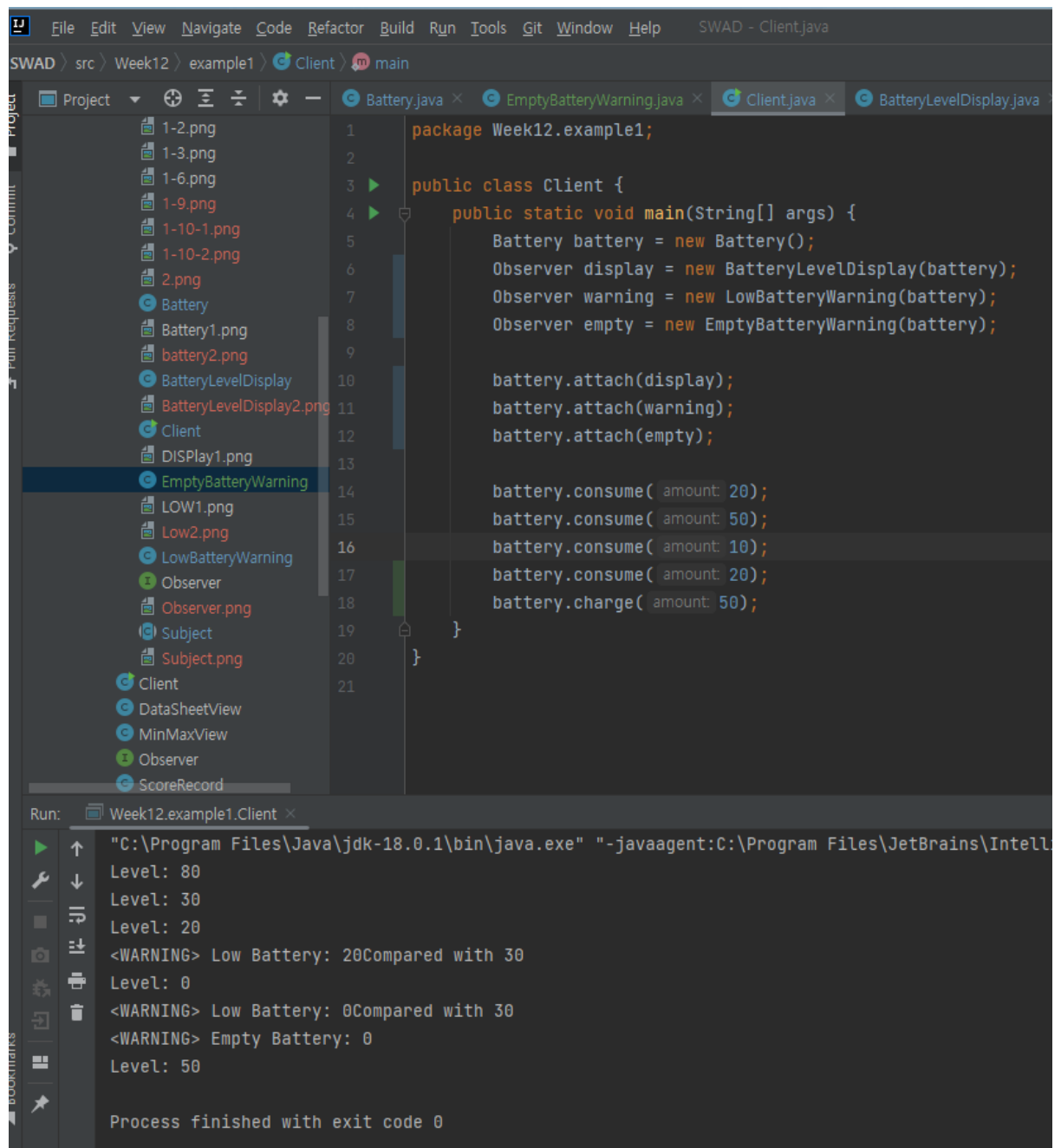
Battery.java

```
1 package Week12.example1;
2
3 public class Battery extends Subject {
4     private int level = 100;
5
6     public void consume(int amount) {
7         level -= amount;
8         notifyObservers();
9     }
10
11     public void charge(int amount) {
12         level += amount;
13         notifyObservers();
14     }
15
16     public int getLeve() { return level; }
17
18 }
```

EmptyBatteryWarning.java

```
1 package Week12.example1;
2
3 public class EmptyBatteryWarning implements Observer{
4     private static final int empty = 0;
5     private Battery battery;
6
7     public EmptyBatteryWarning (Battery battery) { this.battery = battery; }
8
9
10
11     public void update(){
12         int level = battery.getLeve();
13         if (level == empty){
14             System.out.println("<WARNING> Empty Battery: " + level);
15         }
16     }
17 }
18
```

Client.java



The screenshot shows an IDE window with the following components:

- Project Explorer (Left):** Lists files in the 'Week12' project, including 'Client' and 'EmptyBatteryWarning'.
- Editor (Center):** Displays the code for 'Client.java'.
- Run Console (Bottom):** Shows the execution output of the 'Client' class.

Client.java Code:

```
1 package Week12.example1;
2
3 public class Client {
4     public static void main(String[] args) {
5         Battery battery = new Battery();
6         Observer display = new BatteryLevelDisplay(battery);
7         Observer warning = new LowBatteryWarning(battery);
8         Observer empty = new EmptyBatteryWarning(battery);
9
10        battery.attach(display);
11        battery.attach(warning);
12        battery.attach(empty);
13
14        battery.consume( amount: 20);
15        battery.consume( amount: 50);
16        battery.consume( amount: 10);
17        battery.consume( amount: 20);
18        battery.charge( amount: 50);
19    }
20 }
21
```

Run Console Output:

```
Run: Week12.example1.Client
"C:\Program Files\Java\jdk-18.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA\bin\idea-agent.jar" -Didea.config.path=C:\Program Files\JetBrains\IntelliJ IDEA\config -Didea.system.path=C:\Program Files\JetBrains\IntelliJ IDEA\lib -Didea.class.path=C:\Program Files\JetBrains\IntelliJ IDEA\lib\idea.jar -Didea.platform.prefix=JDK -Didea.platform.classpath=C:\Program Files\JetBrains\IntelliJ IDEA\lib\platform.jar -Didea.class.path=C:\Program Files\JetBrains\IntelliJ IDEA\lib\idea.jar -Didea.class.path=C:\Program Files\JetBrains\IntelliJ IDEA\lib\idea.jar -Didea.class.path=C:\Program Files\JetBrains\IntelliJ IDEA\lib\idea.jar -Didea.class.path=C:\Program Files\JetBrains\IntelliJ IDEA\lib\idea.jar
Level: 80
Level: 30
Level: 20
<WARNING> Low Battery: 20Compared with 30
Level: 0
<WARNING> Low Battery: 0Compared with 30
<WARNING> Empty Battery: 0
Level: 50
Process finished with exit code 0
```