

# 소프트웨어 분석 설계

## 15주차

20170677 오윙택

### 1. 코드 11-4 구현하기

#### 1) Motor.java

```
public MotorStatus getMotorStatus () { return motorStatus; }

private void setMotorStatus(MotorStatus motorStatus) { this.motorStatus = motorStatus; }

public void move(Direction direction) {
    MotorStatus motorStatus = getMotorStatus();
    if (motorStatus == motorStatus.MOVING)
        return;

    DoorStatus doorStatus = door.getDoorStatus();
    if (doorStatus == DoorStatus.OPENED)
        door.close();

    moveMotor(direction);
    setMotorStatus(MotorStatus.MOVING);
}

protected abstract void moveMotor(Direction direction);
}
```

## 2) LGMotor.java

```
package week15;

public class LGMotor extends Motor{

    public LGMotor(Door door) {
        super(door);
    }

    @Override
    protected void moveMotor(Direction direction) { System.out.println("LG모터 " + direction + " 움직이기"); }
}
```

## 3) HyundaiMotor.java

```
package week15;

public class HyundaiMotor extends Motor{
    public HyundaiMotor(Door door) {
        super(door);
    }

    @Override
    protected void moveMotor(Direction direction) {
        System.out.println("현대모터 " + direction + " 움직이기");
    }
}
```

## 4) Direction, DoorStatus, MotorStatus.java

```
package week15;

public enum Direction { UP, DOWN }
}
```

```
package week15;

public enum MotorStatus { MOVING, STOPPED }
}
```

```
package week15;

public enum DoorStatus { CLOSED, OPENED }
}
```

## 5) Door.java

```
package week15;

public class Door {

    private DoorStatus doorStatus;

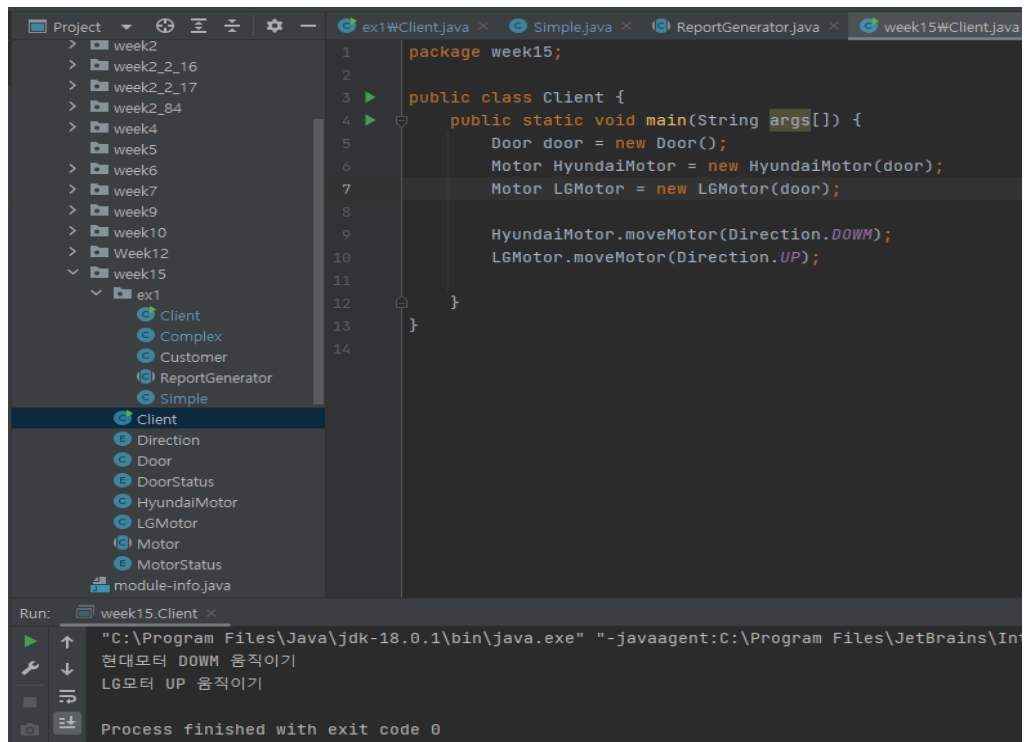
    public Door () {
        doorStatus = DoorStatus.CLOSED;
    }

    public DoorStatus getDoorStatus() { return doorStatus; }

    public void close() {
        doorStatus = DoorStatus.CLOSED;
        System.out.println("문닫기");
    }

    public void open(){
        doorStatus = DoorStatus.OPENED;
        System.out.println("문열기");
    }
}
```

## 6) Client.java

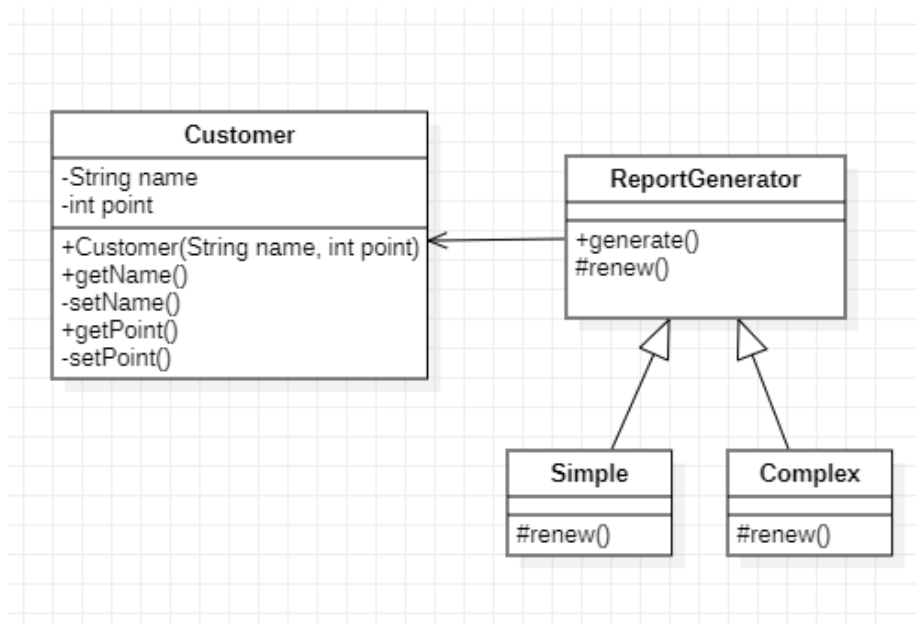


The screenshot shows an IDE with the following components:

- Project Explorer:** A tree view on the left showing a project structure with folders 'week2' through 'week15' and a sub-folder 'ex1' containing files like 'Client', 'Complex', 'Customer', 'ReportGenerator', 'Simple', and 'week15.Client'.
- Code Editor:** The main window displays the code for 'Client.java'. It includes package declarations, class declarations, and a static 'main' method that creates a 'Door' object, instantiates 'HyundaiMotor' and 'LGMotor', and calls their 'moveMotor' methods with 'DOWN' and 'UP' directions respectively.
- Run Console:** At the bottom, the output of the program execution is shown. It indicates the Java command used, the execution of '현대모터 DOWN 움직이기' and 'LG모터 UP 움직이기', and a final message 'Process finished with exit code 0'.

## 2. 연습문제

- 1) 템플릿 메서드 패턴을 사용해 클래스 다이어그램으로 표현하고 설명하기



**Customer** : Report제너레이터가 항상 가지고 있어야 하는 인스턴스를 책임지는 클래스

**ReportGenerator** : Customer 리스트 인스턴스를 가지고 있는 추상 클래스, generate메소드 안에서 renew()메소드를 호출하여 보고서의 출력 대상이 될 Customer인스턴스 리스트를 다시 연산한다.

이 중 renew()는 추상 메소드 이므로 자식 클래스에서 구현해야하는 책임을 준다.

**Simple/Complex** : ReportGenerator클래스를 상속받는 구체화 클래스, renew()메소드를 반드시 재정의해줘야 한다.

- 2) 템플릿 패턴을 이용해 개선한 설계에서 각 구성 요소에 대응하는 클래스와 메서드를 구분하라.

Customer : 보고서 생성의 대상이 되는 데이터를 가지고 있는 인스턴스

ReportGenerator : Customer 인스턴스를 속성으로 가지고 보고서를 출력하는 추상클래스

Generate() : 보고서를 출력하는 public 메서드

Renew() : 아직 구현되지 않은 protected List<Customer> 추상 메서드

Simple / Complex : ReportGenerator을 상속받는 자식클래스

Renew() : Generate() 메소드의 실행시 보고서 출력 되상이 될

List<Customer> Customers 에서 조건에 맞는 Customer만 뽑아 새로운 Customers리스트를 리턴함

- 3) 코드 작성하기

Customer.java

```
package week15.ex1;

public class Customer {
    private String name;
    private int point;

    public Customer(String name, int point) {
        this.setName(name);
        this.setPoint(point);
    }

    public int getPoint() {
        return this.point;
    }

    public String getName() { return this.name; }
    private void setPoint(int point) { this.point = point; }
    private void setName(String name) { this.name = name; }
}
```

## ReportGenerator.java

```
package week15.ex1;

import java.util.List;

public abstract class ReportGenerator {
    public String generate(List<Customer> customers) {
        customers = renew(customers);
        String report = String.format("고객 수: %d명\n", customers.size());
        for (int i = 0; i < customers.size(); i++) {
            Customer customer = customers.get(i);
            report += String.format("%s: %d\n", customer.getName(), customer.getPoint());
        }
        return report;
    }

    protected abstract List<Customer> renew(List<Customer> customers);
}
```

## Simple.java

```
package week15.ex1;

import java.util.List;

public class Simple extends ReportGenerator {
    @Override
    protected List<Customer> renew(List<Customer> customers) {
        return customers;
    }
}
```

## Complex.java

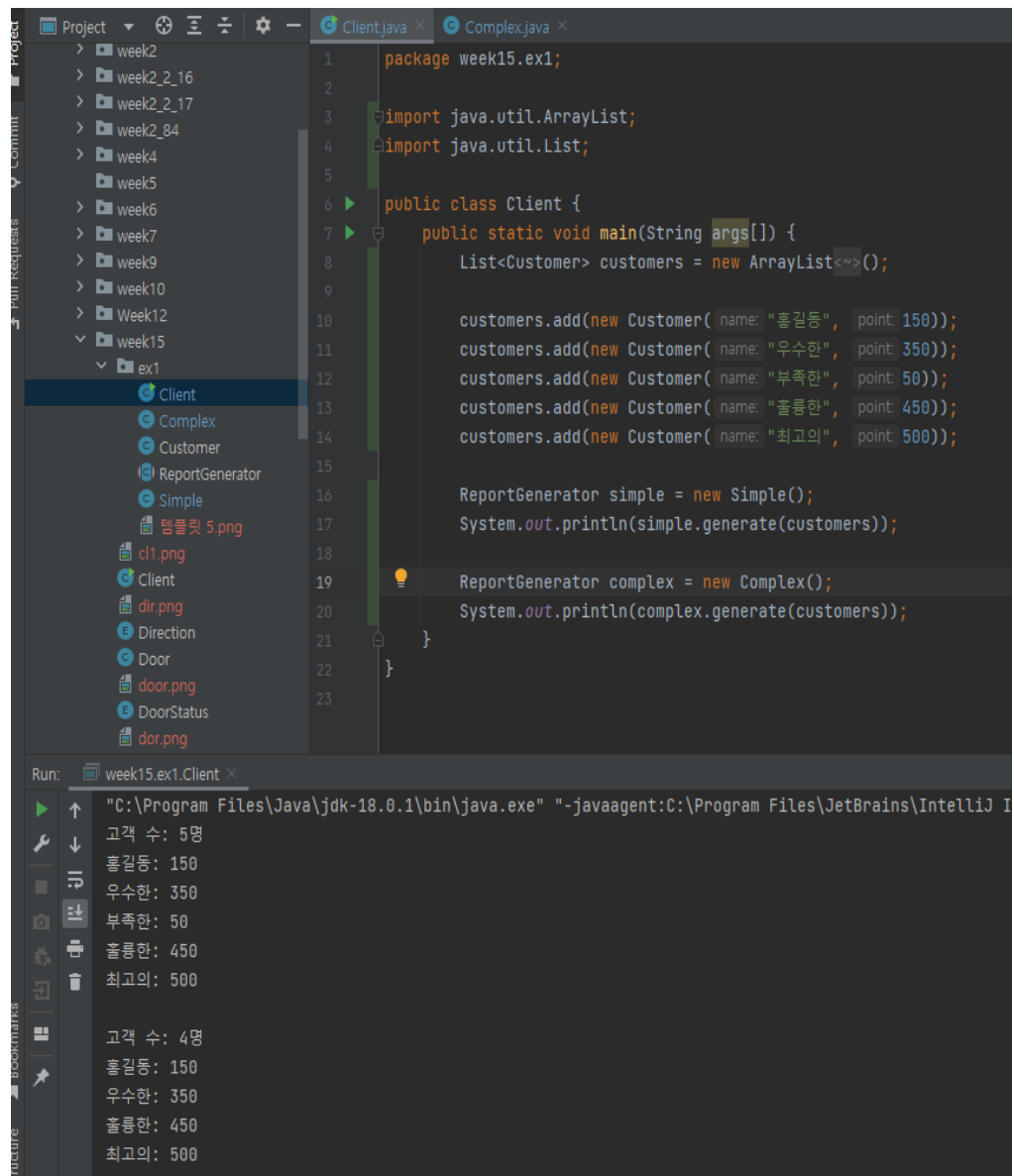
```
package week15.ex1;

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

public class Complex extends ReportGenerator {
    @Override
    protected List<Customer> renew(List<Customer> customers) {
        List<Customer> renewedCustomers = new ArrayList<>();
        for (int i = 0; i < customers.size(); i++) {
            Customer customer = customers.get(i);
            if (customer.getPoint() >= 100) {
                renewedCustomers.add(customer);
            }
        }
        return renewedCustomers;
    }
}
```

#### 4) Client클래스 재작성하고 실행 결과 기록하기

Client.java



The screenshot displays the IntelliJ IDEA IDE. On the left, the Project tool window shows a directory structure with folders like week2, week2\_2\_16, week2\_2\_17, week2\_84, week4, week5, week6, week7, week9, week10, week12, and week15. Under week15, there is a folder 'ex1' containing files like Client, Complex, Customer, ReportGenerator, Simple, and various image files (e.g., cl1.png, dir.png, Direction, Door, door.png, DoorStatus, dor.png). The main editor shows the Client.java file with the following code:

```
1 package week15.ex1;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Client {
7     public static void main(String args[]) {
8         List<Customer> customers = new ArrayList<>();
9
10        customers.add(new Customer( name: "홍길동", point: 150));
11        customers.add(new Customer( name: "우수한", point: 350));
12        customers.add(new Customer( name: "부족한", point: 50));
13        customers.add(new Customer( name: "훌륭한", point: 450));
14        customers.add(new Customer( name: "최고의", point: 500));
15
16        ReportGenerator simple = new Simple();
17        System.out.println(simple.generate(customers));
18
19        ReportGenerator complex = new Complex();
20        System.out.println(complex.generate(customers));
21    }
22 }
23
```

At the bottom, the Run tool window shows the execution of 'week15.ex1.Client'. The command line is: "C:\Program Files\Java\jdk-18.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ I...". The output is as follows:

```
고객 수: 5명
홍길동: 150
우수한: 350
부족한: 50
훌륭한: 450
최고의: 500

고객 수: 4명
홍길동: 150
우수한: 350
훌륭한: 450
최고의: 500
```