HW3

1분반

2024년 10월 02일

32211792

박재홍

FileImporter 클래스)

```java
package template;
import java.io.*;
import java.util.*;
import java.io.BufferedReader;
// csv파일 읽고 데이터를 리스트로 반환
public class FileImporter<T> {
    private ParserStrategy<T> strategy;

    public FileImporter(ParserStrategy<T> strategy) {
        this.strategy = strategy;
    }
    // csv파일을 읽고, 파싱 후, 리스트로 반환
    public List<T> loadCSV(String fileName) {
        List<T> List = new ArrayList<>();
        try(BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            String line;
        while ((line = br.readLine()) != null) {
            T parsedObject = strategy.parse(line);
            List.add(parsedObject);
        }
        }catch (Exception e) {
            e.printStackTrace();
        }
        return List;
    }
}
```

csv파일을 읽고 데이터를 리스트로 반환할 수 있도록 선언해주었다.

ParserStrategy 인터페이스)

```java
package template;
// 데이터를 파싱하는 인터페이스. 제네릭 타입T를 사용하여 다양한 데이터 사용 가능
public interface ParserStrategy<T> {
    T parse(String line);
}

```

데이터를 파싱하는 인터페이스, 제네릭 타입T를 사용해서 다양한 데이터를 사용할 수 있도록 해주었다.

FoodDataParseStrategy 클래스)

```java
package template;
// ParserStrategy인터페이스를 구현
public class FoodDataParseStrategy implements ParserStrategy<FoodData> {
    // csv파일을 파싱, FoodData 객체로 변환시킴
    public FoodData parse(String line) {
        String[] tokens = line.split(regex:",");
        String name = tokens[0];
        double carbs = Double.parseDouble(tokens[1]);
        double calories = Double.parseDouble(tokens[2]);
        double Protein = Double.parseDouble(tokens[3]);

        return new FoodData(name, carbs, calories,Protein);
    }
}

```

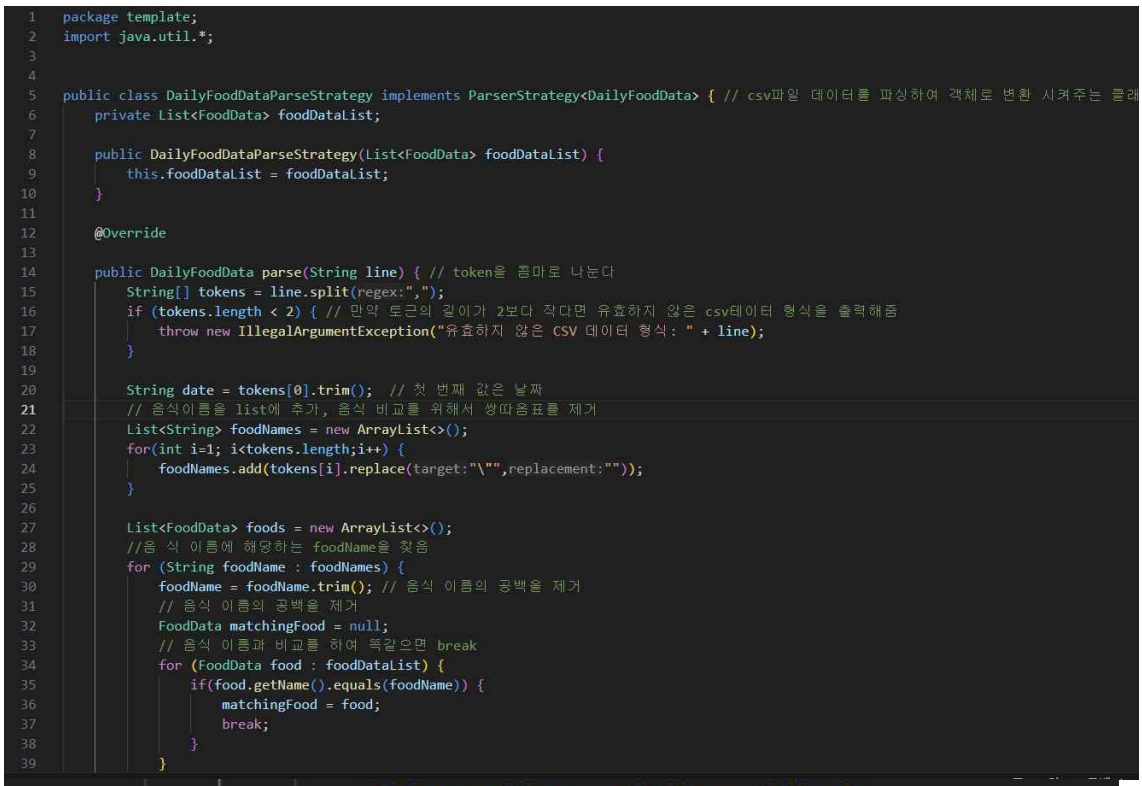
parsestrategy 인터페이스를 구현해주었고 csv파일을 파싱하고 fooddata 객체로 변환시키도록 선언해주었다.

DailyHealthDataParseStrategy 클래스)

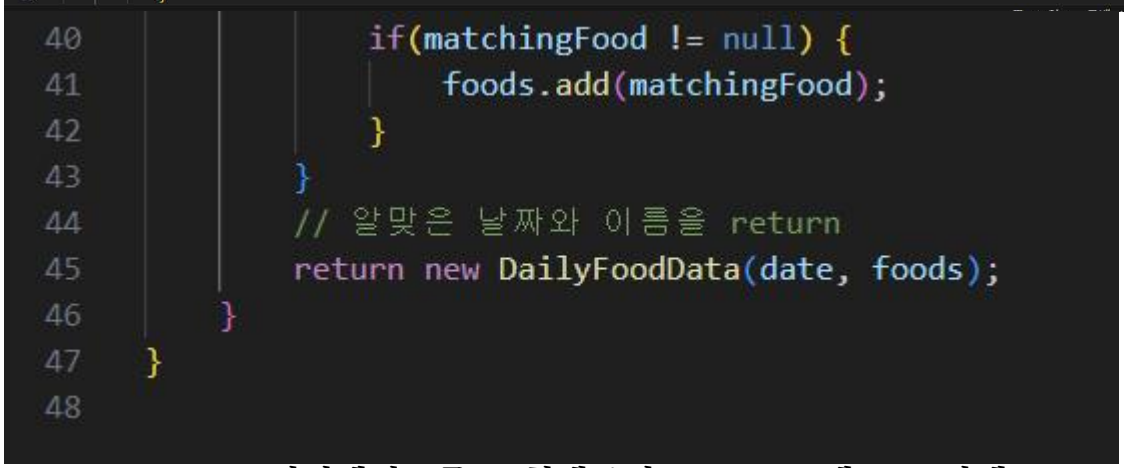
```java
package template;

public class DailyHealthDataParseStrategy implements ParserStrategy<DailyHealthData> {

    @Override
    public DailyHealthData parse(String line) {

    // pasrse메소드, token[0]부터 [4]까지 날짜, 혈당 수치, 인슐린 용량, 탄수화물, 단백질(your code) 순으로 정의
    String[] tokens = line.split(regex:",");
    String date = tokens[0];
    double bloodSugarLevel = Double.parseDouble(tokens[1]);
    double insulinDose = Double.parseDouble(tokens[2]);
    double carbsIntake = Double.parseDouble(tokens[3]);
    double Protein = Double.parseDouble(tokens[4]);

    return new DailyHealthData(date, bloodSugarLevel, insulinDose, carbsIntake, Protein);
    }
}
```

parrserstrategy 인터페이스를 구현해주었고 parser 메소드 안에 tokens 배열 안에 0번에는 날짜, 1번에는 혈당 수치, 2번에는 인슐린 용량, 3번에는 탄수화물, 4번에는 단백질(your code) 순으로 정의해주었다.

DailyFoodDataParseStrategy 클래스)



```java
package template;
import java.util.*;


public class DailyFoodDataParseStrategy implements ParserStrategy<DailyFoodData> { // csv파일 데이터를 파싱하여 객체로 변환 시켜주는 클래
    private List<FoodData> foodDataList;

    public DailyFoodDataParseStrategy(List<FoodData> foodDataList) {
        this.foodDataList = foodDataList;
    }

    @Override

    public DailyFoodData parse(String line) { // token을 콤마로 나눈다
        String[] tokens = line.split(regex:",");
        if (tokens.length < 2) { // 만약 토큰의 길이가 2보다 작다면 유효하지 않은 csv데이터 형식을 출력해줌
            throw new IllegalArgumentException("유효하지 않은 CSV 데이터 형식: " + line);
        }

        String date = tokens[0].trim();  // 첫 번째 값은 날짜
        // 음식이름을 list에 추가, 음식 비교를 위해서 쌍따옴표를 제거
        List<String> foodNames = new ArrayList<>();
        for(int i=1; i<tokens.length;i++) {
            foodNames.add(tokens[i].replace(target:"\"",replacement:""));
        }

        List<FoodData> foods = new ArrayList<>();
        //음 식 이름에 해당하는 foodName을 찾음
        for (String foodName : foodNames) {
            foodName = foodName.trim(); // 음식 이름의 공백을 제거
            // 음식 이름의 공백을 제거
            FoodData matchingFood = null;
            // 음식 이름과 비교를 하여 똑같으면 break
            for (FoodData food : foodDataList) {
                if(food.getName().equals(foodName)) {
                    matchingFood = food;
                    break;
                }
            }
```



```java
            if(matchingFood != null) {
                foods.add(matchingFood);
            }
        }
        // 알맞은 날짜와 이름을 return
        return new DailyFoodData(date, foods);
    }
}
```

parserstrategy 인터페이스를 구현해주었고 parse 메소드 안에 tokens를 콤마로 나누어주고 if문을 사용해 토큰의 길이가 2보다 작다면 유효하지 않은 csv데이터 형식을 출력해주고 tokens 첫 번째 값에는 날짜를 넣어주고 음식이름을 list에 추가해주고 음식 비교를 위해서 쌍 따옴표를 제거해주었다. 그리고 음식 이름에 해당하는 foodname을 찾아주고 음식이름의 공백을 제거해주었다. 음식이름과 비교를 하여 같으면 멈출수 있도록 break를 써주었다.

Observer 인터페이스)

```java
package template;

//옵저버 패턴에 사용되는 인터페이스. 객체 갱신될 때 호출되는 update메소드 정의
public interface Observer {
    void update(DailyHealthData d);
}

```

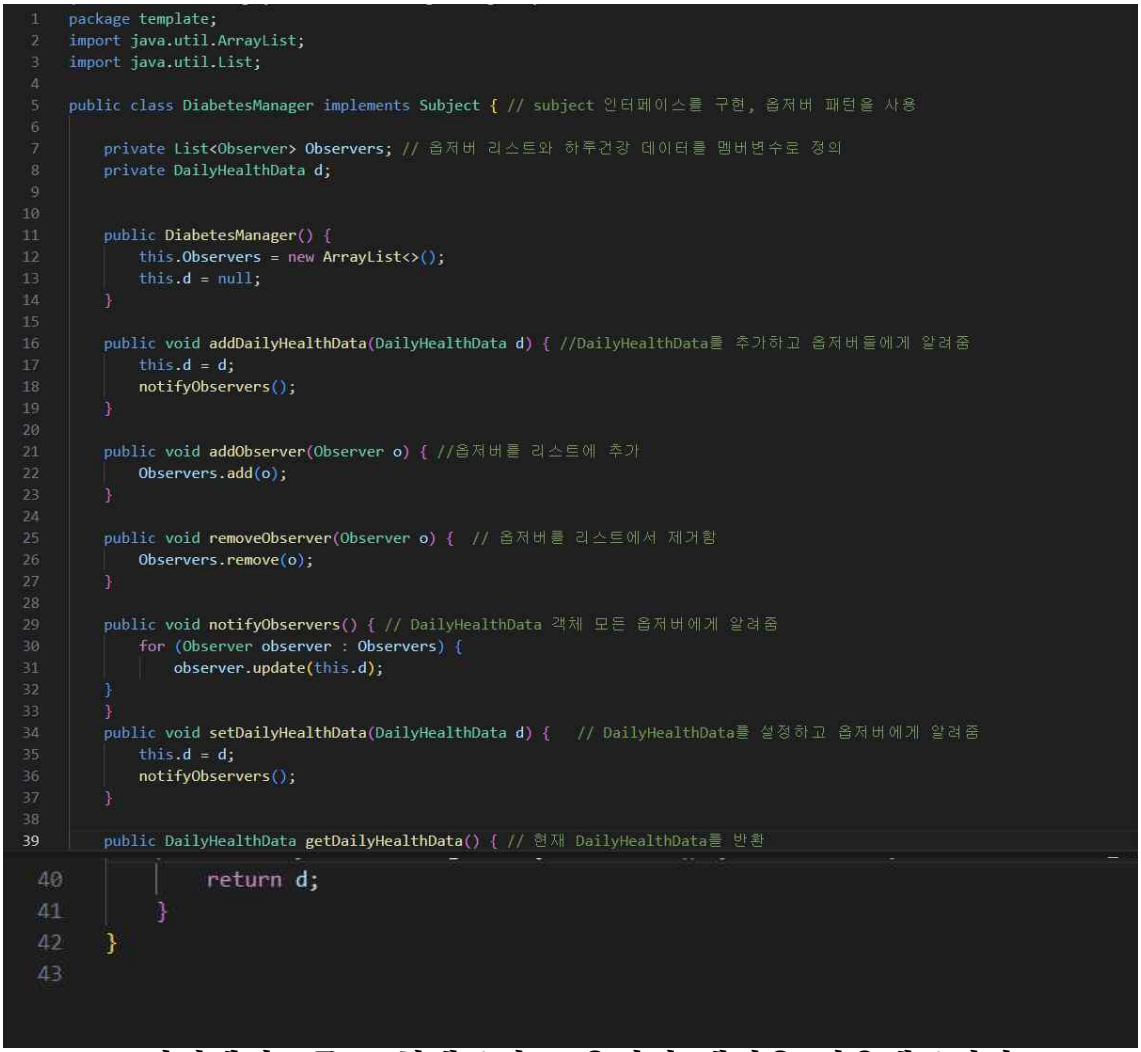
옵저버 패턴에 사용되는 인터페이스, 객체가 갱신될 때 호출되는 update 메소르를 정이해주었다.

Subject 인터페이스)

```java
package template;
// 옵저버 패턴에서 사용하는 subject. 옵저버를 추가, 제거, 알리는 메소드 정의
public interface Subject {
    void addObserver(Observer o);

    void removeObserver(Observer o);

    void notifyObservers();
}

```

옵저버 패턴에서 사용하는 subject, 옵저버를 추가해주고 옵저버를 제거해주고 알려주는 메소드를 정의해주었다.

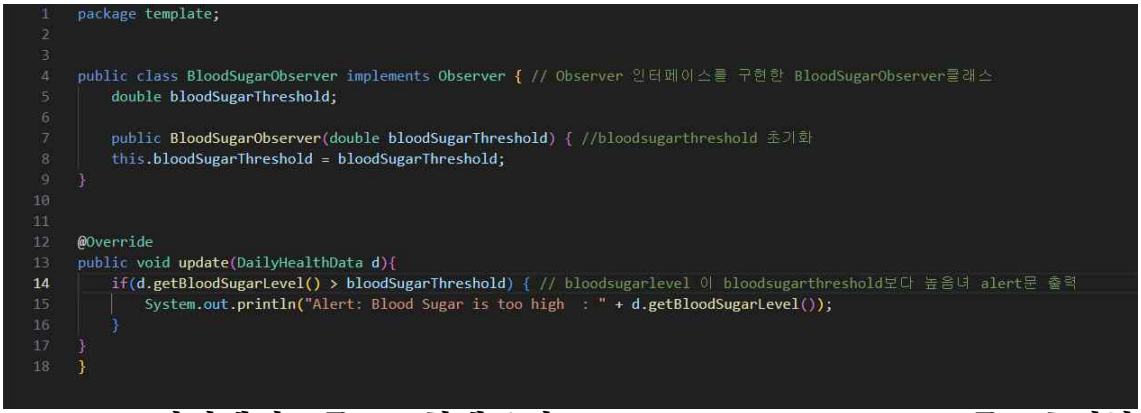
DiabeteManager 클래스)



```java
package template;
import java.util.ArrayList;
import java.util.List;

public class DiabetesManager implements Subject { // subject 인터페이스를 구현, 옵저버 패턴을 사용

    private List<Observer> Observers; // 옵저버 리스트와 하루건강 데이터를 멤버변수로 정의
    private DailyHealthData d;


    public DiabetesManager() {
        this.Observers = new ArrayList<>();
        this.d = null;
    }

    public void addDailyHealthData(DailyHealthData d) { //DailyHealthData를 추가하고 옵저버들에게 알려줌
        this.d = d;
        notifyObservers();
    }

    public void addObserver(Observer o) { //옵저버를 리스트에 추가
        Observers.add(o);
    }

    public void removeObserver(Observer o) {  // 옵저버를 리스트에서 제거함
        Observers.remove(o);
    }

    public void notifyObservers() { // DailyHealthData 객체 모든 옵저버에게 알려줌
        for (Observer observer : Observers) {
            observer.update(this.d);
    }
    }
    public void setDailyHealthData(DailyHealthData d) {   // DailyHealthData를 설정하고 옵저버에게 알려줌
        this.d = d;
        notifyObservers();
    }

    public DailyHealthData getDailyHealthData() { // 현재 DailyHealthData를 반환
        return d;
    }
}
```

subject 인터페이스를 구현해주었고 옵저버 패턴을 사용해주었다.
옵저버 리스트와 하루건강 데이터를 멤버변수로 정의해주었고 adddailyhealthdata 메소드를 통해 DailyHealthdata를 추가해주고 옵저버에게 알려줄 수 있도록 선언해주었고 addobserver 메소드를 통해 옵저버를 리스트에 추가시켜주었고 removeobserver 메소드를 통해 옵저버를 리스트에서 제거해줄 수 있게 해주었고 notiyyobserver 메소드를 통해 dailyhealthdata 객체 모든 옵저버에게 알려줄 수 있도록 해주었고 setdailyhealthdata 메소드를 통해 dailyhealthdata를 설정하고 옵저버에게 알려줄 수 있도록 선언해주었고 getdailhealthdata 메소드를 통해 현재 dailyhealthdata를 반환할 수 있도록 해주었다.

BloodSugarObserver 클래스)



```java
package template;


public class BloodSugarObserver implements Observer { // Observer 인터페이스를 구현한 BloodSugarObserver클래스
    double bloodSugarThreshold;

    public BloodSugarObserver(double bloodSugarThreshold) { //bloodsugarthreshold 초기화
    this.bloodSugarThreshold = bloodSugarThreshold;
}


@Override
public void update(DailyHealthData d){
    if(d.getBloodSugarLevel() > bloodSugarThreshold) { // bloodsugarlevel 이 bloodsugarthreshold보다 높을녀 alert문 출력
        System.out.println("Alert: Blood Sugar is too high  : " + d.getBloodSugarLevel());
    }
}
}
```

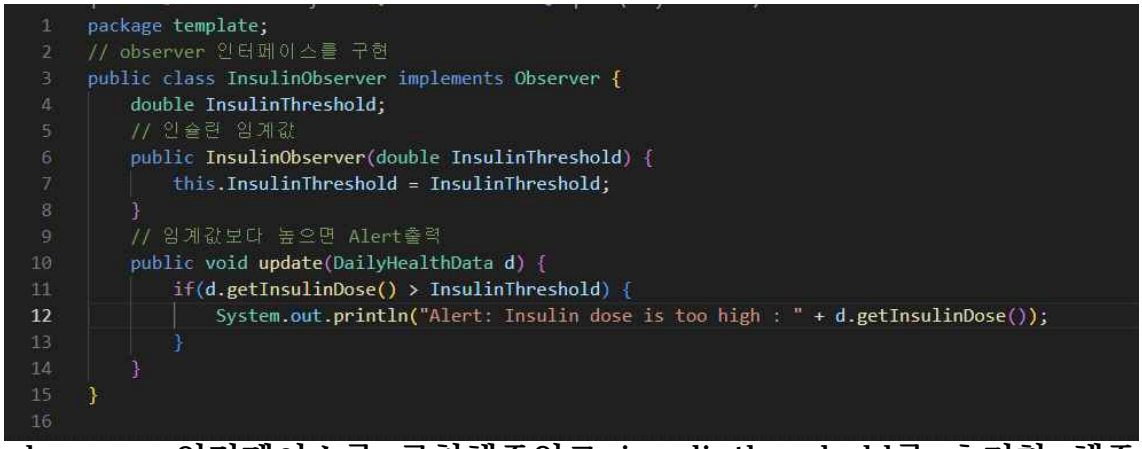
observer인터페이스를 구현해주었고 bloodsugarthreshold를 초기화 해주고 bloodsugarlevel이 bloodsugarthreshold 보다 높으면 alert문이 출력 되도록 해주었다

CarbsIntakeObserver 클래스)

```java
package template;

public class CarbsIntakeObserver implements Observer {  // Observer 인터페이스를 구현한 CarbsIntakeObserver클래스
        double CarbsIntakeThreshold;

    public CarbsIntakeObserver(double CarbsIntakeThreshold ) { // carbsintakethreshold를 초기화하는 생성자
        this.CarbsIntakeThreshold = CarbsIntakeThreshold;
    }

    public void update(DailyHealthData d) { // getcarbsintake가 carbsintakethreshold 보다 높으면 alert문 출력
        if(d.getCarbsIntake() > CarbsIntakeThreshold) {
            System.out.println("Alert: Carb intake is too high : " + d.getCarbsIntake());
        }
    }


}
```

observer 인터페이스를 구현해주었고 carbsintakethreshold를 초기화 해주고 getcarbsintake가 carbsintakethreshold 보다 높으면 alert 문을 출력 하도록 해주었다.

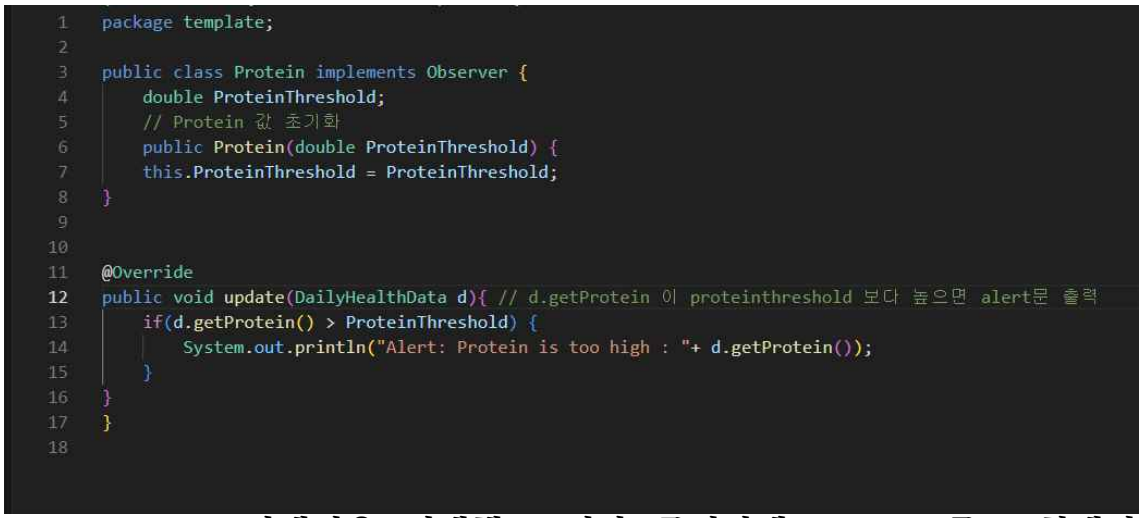
InsulinObserver 클래스)



```java
package template;
// observer 인터페이스를 구현
public class InsulinObserver implements Observer {
    double InsulinThreshold;
    // 인슐린 임계값
    public InsulinObserver(double InsulinThreshold) {
        this.InsulinThreshold = InsulinThreshold;
    }
    // 임계값보다 높으면 Alert출력
    public void update(DailyHealthData d) {
        if(d.getInsulinDose() > InsulinThreshold) {
            System.out.println("Alert: Insulin dose is too high : " + d.getInsulinDose());
        }
    }
}
```

observer 인터페이스를 구현해주었고 insulinthreshold를 초기화 해주고 getinsulindose가 insulinthreshold보다 높으면 alert문을 출력할 수 있도록 해주었다.

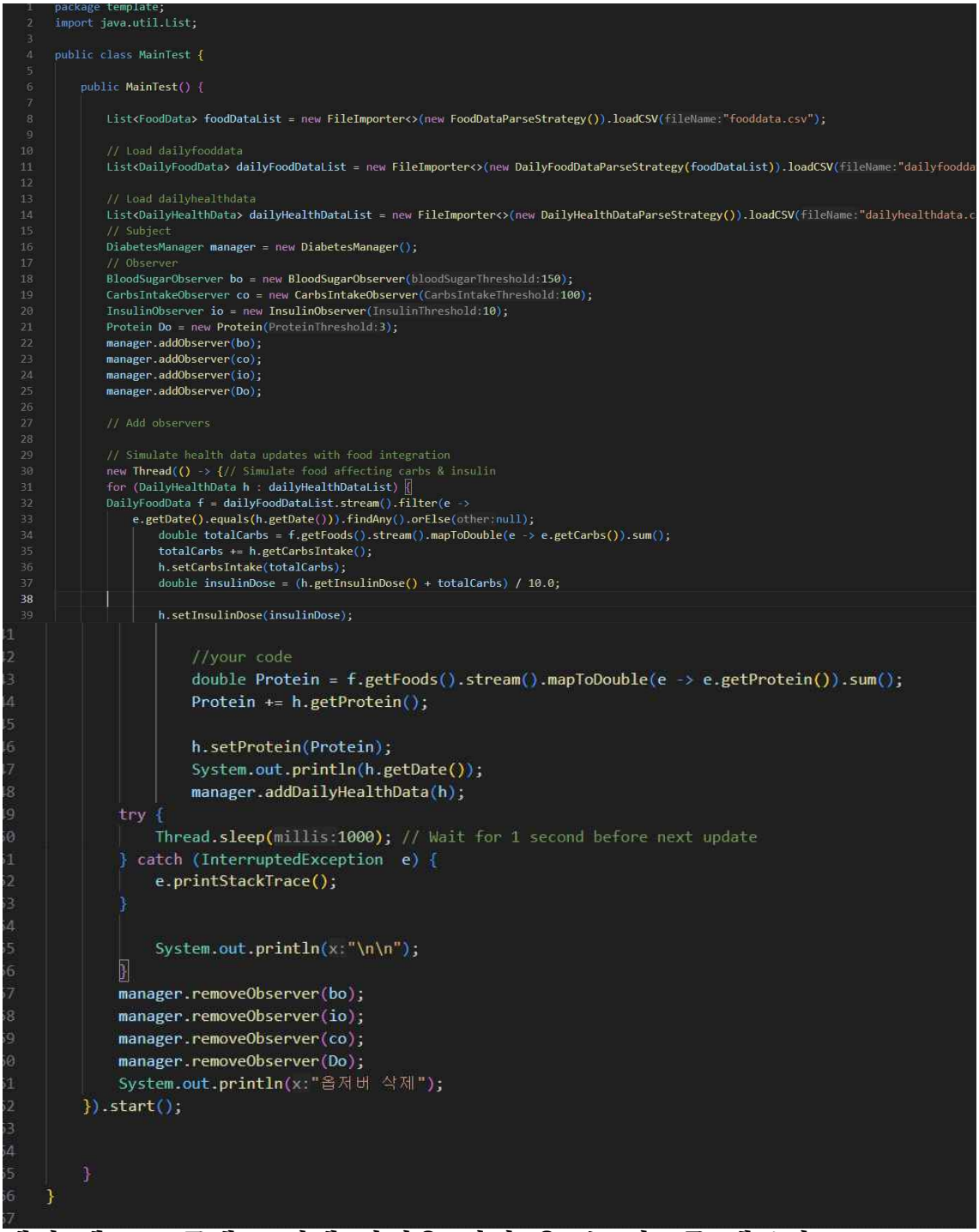
Your Code)



```java
package template;

public class Protein implements Observer {
    double ProteinThreshold;
    // Protein 값 초기화
    public Protein(double ProteinThreshold) {
    this.ProteinThreshold = ProteinThreshold;
}


@Override
public void update(DailyHealthData d){ // d.getProtein 이 proteinthreshold 보다 높으면 alert문 출력
    if(d.getProtein() > ProteinThreshold) {
        System.out.println("Alert: Protein is too high : "+ d.getProtein());
    }
}
}
```

your code로 단백질을 선택했고 위와 동일하게 obsever를 구현해서 protein 값을 먼저 초기화 해준 후 getprotein이 proteinthreshold 보다 높으면 alert문을 출력할 수 있도록 해주었다.

Main Test 클래스)



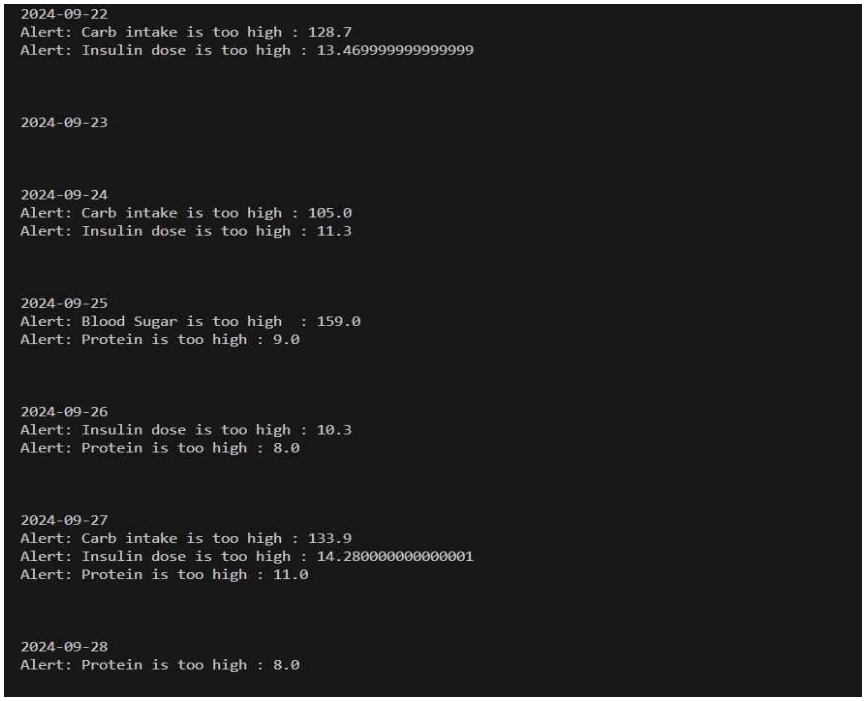
```
package template;
import java.util.List;

public class MainTest {

    public MainTest() {

        List<FoodData> foodDataList = new FileImporter<>(new FoodDataParseStrategy()).loadCSV(fileName:"fooddata.csv");

        // Load dailyfooddata
        List<DailyFoodData> dailyFoodDataList = new FileImporter<>(new DailyFoodDataParseStrategy(foodDataList)).loadCSV(fileName:"dailyfoodda

        // Load dailyhealthdata
        List<DailyHealthData> dailyHealthDataList = new FileImporter<>(new DailyHealthDataParseStrategy()).loadCSV(fileName:"dailyhealthdata.c
        // Subject
        DiabetesManager manager = new DiabetesManager();
        // Observer
        BloodSugarObserver bo = new BloodSugarObserver(bloodSugarThreshold:150);
        CarbsIntakeObserver co = new CarbsIntakeObserver(CarbsIntakeThreshold:100);
        InsulinObserver io = new InsulinObserver(InsulinThreshold:10);
        Protein Do = new Protein(ProteinThreshold:3);
        manager.addObserver(bo);
        manager.addObserver(co);
        manager.addObserver(io);
        manager.addObserver(Do);

        // Add observers

        // Simulate health data updates with food integration
        new Thread(() -> {// Simulate food affecting carbs & insulin
        for (DailyHealthData h : dailyHealthDataList) {
        DailyFoodData f = dailyFoodDataList.stream().filter(e ->
            e.getDate().equals(h.getDate())).findAny().orElse(other:null);
                double totalCarbs = f.getFoods().stream().mapToDouble(e -> e.getCarbs()).sum();
                totalCarbs += h.getCarbsIntake();
                h.setCarbsIntake(totalCarbs);
                double insulinDose = (h.getInsulinDose() + totalCarbs) / 10.0;

                h.setInsulinDose(insulinDose);

                    //your code
                    double Protein = f.getFoods().stream().mapToDouble(e -> e.getProtein()).sum();
                    Protein += h.getProtein();

                    h.setProtein(Protein);
                    System.out.println(h.getDate());
                    manager.addDailyHealthData(h);
            try {
                Thread.sleep(millis:1000); // Wait for 1 second before next update
            } catch (InterruptedException  e) {
                e.printStackTrace();
            }

                System.out.println(x:"\n\n");
            }
            manager.removeObserver(bo);
            manager.removeObserver(io);
            manager.removeObserver(co);
            manager.removeObserver(Do);
            System.out.println(x:"옵저버 삭제");
        }).start();


    }
}
```

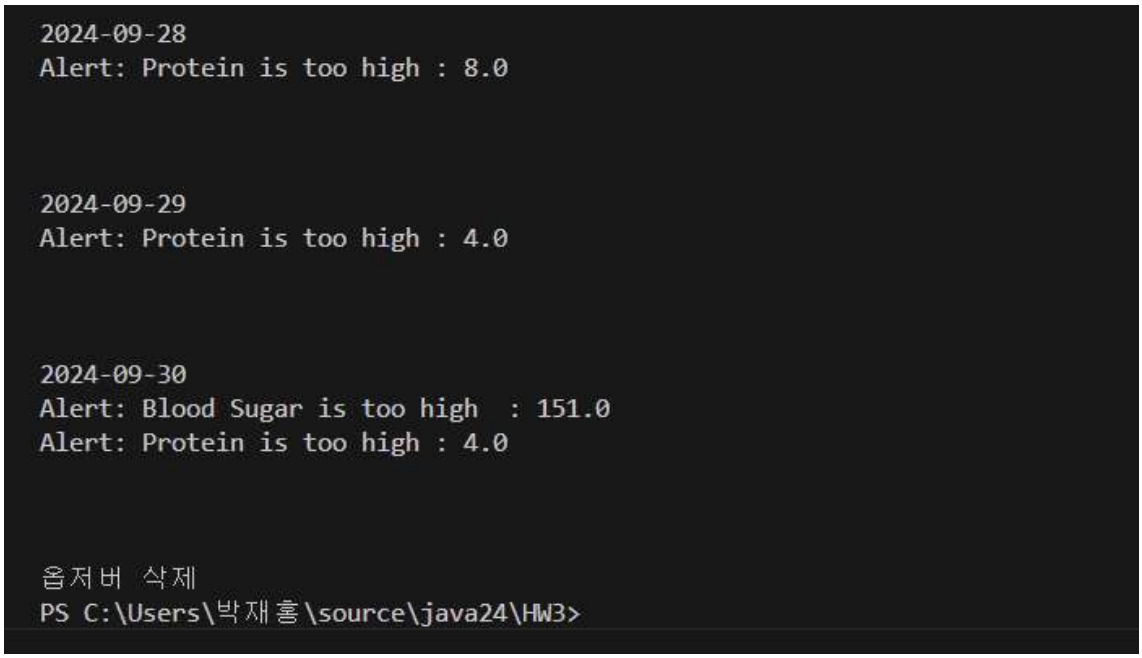
메인 테스트 클래스 안에 파일을 읽어 올 수 있도록 해주었고 subject 객체, observer 객체를 생성해주었고, 그 후 옵저버를 추가 시켜주고 아래에서 new thread 문을 통해 날짜, 혈당 수치, 인슐린 용량, 단백질의 값이 출력 되도록 해주고 그 후 옵저버를 삭제시켜주었다.

출력문)

```
2024-09-01
Alert: Protein is too high : 8.0



2024-09-02
Alert: Carb intake is too high : 107.5
Alert: Insulin dose is too high : 11.35
Alert: Protein is too high : 11.0



2024-09-03
Alert: Carb intake is too high : 138.3
Alert: Insulin dose is too high : 14.530000000000001



2024-09-04



2024-09-05
Alert: Protein is too high : 6.0



2024-09-06
Alert: Protein is too high : 4.0



2024-09-07
Alert: Blood Sugar is too high  : 152.0
Alert: Carb intake is too high : 103.6
Alert: Insulin dose is too high : 11.33
Alert: Protein is too high : 8.0
 2024-09-08



 2024-09-09
 Alert: Protein is too high : 7.0



 2024-09-10
 Alert: Blood Sugar is too high  : 200.0
 Alert: Protein is too high : 6.0



 2024-09-11
 Alert: Protein is too high : 12.0



 2024-09-12



 2024-09-13
 Alert: Carb intake is too high : 139.4
 Alert: Insulin dose is too high : 14.64



 2024-09-14
 Alert: Blood Sugar is too high  : 154.0
 Alert: Carb intake is too high : 118.3
 Alert: Insulin dose is too high : 12.629999999999999
```

```
2024-09-15



2024-09-16
Alert: Carb intake is too high : 102.7
Alert: Insulin dose is too high : 11.16
Alert: Protein is too high : 5.0



2024-09-17
Alert: Carb intake is too high : 101.5
Alert: Insulin dose is too high : 11.1
Alert: Protein is too high : 6.0



2024-09-18



2024-09-19
Alert: Protein is too high : 8.0



2024-09-20
Alert: Protein is too high : 12.0



2024-09-21
Alert: Blood Sugar is too high  : 154.0
```


2024-09-22
Alert: Carb intake is too high : 128.7
Alert: Insulin dose is too high : 13.469999999999999
2024-09-23
2024-09-24
Alert: Carb intake is too high : 105.0
Alert: Insulin dose is too high : 11.3
2024-09-25
Alert: Blood Sugar is too high : 159.0
Alert: Protein is too high : 9.0
2024-09-26
Alert: Insulin dose is too high : 10.3
Alert: Protein is too high : 8.0
2024-09-27
Alert: Carb intake is too high : 133.9
Alert: Insulin dose is too high : 14.280000000000001
Alert: Protein is too high : 11.0
2024-09-28
Alert: Protein is too high : 8.0


2024-09-28
Alert: Protein is too high : 8.0
2024-09-29
Alert: Protein is too high : 4.0
2024-09-30
Alert: Blood Sugar is too high : 151.0
Alert: Protein is too high : 4.0
옵저버 삭제
PS C:\Users\박재홍\source\java24\HW3>