자바 프로그래밍 2 1분반

Lab3

32210608 김민준

제출일 2024월 10월 1일

Source Code Explain

DailyHealthData.java

package template;

public class DailyHealthData {

private String date;

private double bloodSugarLevel;

private double insulinDose;

private double carbsIntake;

private boolean healthCheck; // HealthCheckObserver에 세용될 불리안 변수

public DailyHealthData() {

this.bloodSugarLevel = 0.0;

this.carbsIntake = 0.0;

this.insulinDose = 0.0;

this.healthCheck = false;

}

public DailyHealthData(String date, double bloodSugarLevel, double insulinDose, double carbsIntake, boolean healthCheck) {

this.date = date;

this.bloodSugarLevel = bloodSugarLevel;

this.insulinDose = insulinDose;

this.carbsIntake = carbsIntake;

this.healthCheck = healthCheck;

}…(이하 생략)

healthCheck는 My code에 사용되는 HealthCheckObserver의 기능을 구현하기 위안 불리안 변수이다.

HealthCheckObserver는 BloodSugarObserver, CarbIntakeObserver, InsulinObserver의 기능 중 해당자료값이 특정 값을 넘어는 경우 Alert를 해주는 경우가 있을 때, 즉 자료상 건강이 있는 경우에 몸이 건강한 상태인지 아닌지를 표시해주는 기능을 가진다.

HealthCheckObserver.java (My code)

public class HealthCheckObserver implements Observer{

boolean healthCaution; // 경고 상태 판별을 위한 불리안값

@Override

public boolean update(DailyHealthData d) {

// 하나라도 true이면 경고 상태

if (BloodSugarObserver.bloodSugarCaution == true ||

CarbIntakeObserver.carbIntakeCaution == true ||

InsulinObserver.insulinCaution == true) {

healthCaution = true;

System.out.print("\nBe cautious about your health...");

} else {

healthCaution = false;

System.out.print("\nYou are healthy today!");

}

return healthCaution;

}

}

기능의 구현을 위해 새로운 불리안 변수 healthCaution을 설정하였고, 추상메서드를 구현함에 있어서 보이는 bloodSugarCaution, carbIntakeCaution, InsulinCaution은 각각 BloodSugarObserver, CarbIntakeObserver, InsulinObserver에서 리턴된 값을 가져온다.

BloodSugarObserver.java

public class BloodSugarObserver implements Observer{

double bloodSugarThreshold;

static boolean bloodSugarCaution; //HealthCheckObserver에 사용할 트리거

public BloodSugarObserver(double bloodSugarThreshold){

this.bloodSugarThreshold = bloodSugarThreshold;

}

@Override

public boolean update(DailyHealthData d) {

if(d.getBloodSugarLevel() > bloodSugarThreshold){

System.out.print("BloodSugar Level Alert! ");

bloodSugarCaution = true;

}

else{

bloodSugarCaution = false;

}

return bloodSugarCaution; // HealthCheckObserver에 사용할 리턴값

}

}

CarbIntakeObserver.java

public class CarbIntakeObserver implements Observer{

double carbThreshold;

static boolean carbIntakeCaution; // HealthCheckObserver에 사용할 트리거

public CarbIntakeObserver(double carbThreshold) {

this.carbThreshold = carbThreshold;

}

@Override

public boolean update(DailyHealthData d) {

if(d.getCarbsIntake() > carbThreshold){

System.out.print("CarbsIntake Alert! ");

carbIntakeCaution = true;

}

else{

carbIntakeCaution = false;

}

return carbIntakeCaution; // HealthCheckObserver에 사용할 리턴값

}

}

InsulinObserver.java

public class CarbIntakeObserver implements Observer{

double carbThreshold;

static boolean carbIntakeCaution; // HealthCheckObserver에 사용할 트리거

public CarbIntakeObserver(double carbThreshold) {

this.carbThreshold = carbThreshold;

}

@Override

public boolean update(DailyHealthData d) {

if(d.getCarbsIntake() > carbThreshold){

System.out.print("CarbsIntake Alert! ");

carbIntakeCaution = true;

}

else{

carbIntakeCaution = false;

}

return carbIntakeCaution; // HealthCheckObserver에 사용할 리턴값

}

}

이때 Observer인터페이스에 있는 update메서드는 원래 void를 리턴값으로 가질 예정이었지만 HealthCheckObserver의 구현을 위해 불리안 값을 리터하기 위하여 리턴값을 조정하였다.

Observer.java

public interface Observer {

boolean update(DailyHealthData d); // 제네릭을 없애고 DailyHealthData를 직접 받도록 수정

}

FileImporter.java

// T는 파싱할 데이터 타입을 의미 (FoodData)

public class FileImporter<T> {

private final ParseStrategy<T> parseStrategy;

// 생성자에서 파싱 전략을 주입받음

public FileImporter(ParseStrategy<T> parseStrategy) {

this.parseStrategy = parseStrategy;

}

// loadCSV 메서드

public List<T> loadCSV(String filePath) {

List<T> dataList = new ArrayList<>(); // 파싱된 데이터를 담을 리스트

String line = "";

try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {

Boolean firstLine = true; // CSV파일 첫줄은 자료에 대한 설명 + 첫줄 읽으면 발생하는 오류로 인해 첫줄 건너뛰기위한 불리안값

while ((line = br.readLine()) != null) {

if (firstLine) {

firstLine = false;

continue;

}

// 파싱 전략을 통해 각 줄을 객체로 변환

T data = parseStrategy.parse(line);

dataList.add(data); // 변환된 객체를 리스트에 추가

}

} catch (IOException e) {

e.printStackTrace(); // 파일을 읽는 중 예외 처리

}

return dataList; // 리스트 반환

}

}

FileImporter에서는 CSV파일을 읽어서 리스트를 반환하는 기능을 가지는데 이때 CSV파일을 읽어오는 과정에서 첫째 줄의 #을 포함하는 부분까지 읽어올 경우 에러가 발생하였다. 따라서 firstLine이라는 불리안 변수를 주어 첫째줄을 continue하여 에러를 처리하였다. 또한 while문을 통해 줄별로 읽어옴에 따라 아래와 같이 ParseStrategy 인터페이스를 구성하였다.

ParseStrategy.java

public interface ParseStrategy<T>{

T parse(String line); // CSV 파일의 한 줄을 파싱하는 메서드

}

MainTest.java

public class MainTest {

int count = 0;

…(이하 중략)

Count 변수는 결과값 출력 중 15일 이후에는 혈당 문제가 해결되었다는 가정을 하여 Observer를 삭제하기 위해 추가하였다.

…(이하 중략)

manager.addDailyHealthData(h);

try {

Thread.sleep(1000); // Wait for 1 second before next update

} catch (InterruptedException e) {

e.printStackTrace();

}

count++;

if(count > 15){

manager.removeObserver(bo); // 15일 이후 혈당문제가 해결되었다고 가정(혈당 경고 무시) -> BloodSugar Level 표시하는 옵저버 삭제

System.out.print("(Without Blood sugar level)");

}

System.out.println("\n\n\n");

}

}).start();

}

}

Result (보고서 길이상 13일 ~ 18일 첨부)

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명