가짜 트레이너

- 집에서도 할 수 있는 홈 트레이닝 어플리케이션 -

수학과 201621136 이재협

수학과 201621138 허창현

소프트웨어학과 201823782 안지영

목 차

Ⅰ. 프로그램 목표 및 기능

1. 프로그램 목표

2019년 12월 중국 후베이성 우한시를 시작으로 국내에서도 코로나 19의 확진자가 급증하면서, 지역사회에서는 코로나19 감염 차단을 위해 '사회적 거리두기' 캠페인을 실시하고 있다. 이로 인해 많은 사람들이 헬스장을 대신하여 '홈 트레이닝'을 선택하고 있다.

현 상황을 반영하여 홈 트레이닝을 도울 수 있는 데스크탑 어플리케이션을 개발한다면 개인의 건강증진에 도움이 될 것이라고 생각한다. 따라서 첫째, 사용자가 선택한 난이도와 운동 부위에 적합한 운동 방법을 제안하고, 둘째, 정확한 자세와 시간을 확인할 수 있는 운동 환경을 제공하고, 셋째, 지속적인 동기부여를 위해 운동 기록을 저장할 수 있는 맞춤형 홈 트레이닝 어플리케이션을 개발하고자 한다.

2. 프로그램 기능

⑴ 운동 난이도: [상/중/하] 난이도를 선택할 수 있다.

⑵ 운동 부위: [가슴/등/코어/복근/하체] 운동 부위를 선택할 수 있다

⑶ 운동 방법: 운동 부위와 관련된 운동 중 사용자가 원하는 운동 방법을 선택할 수 있다.

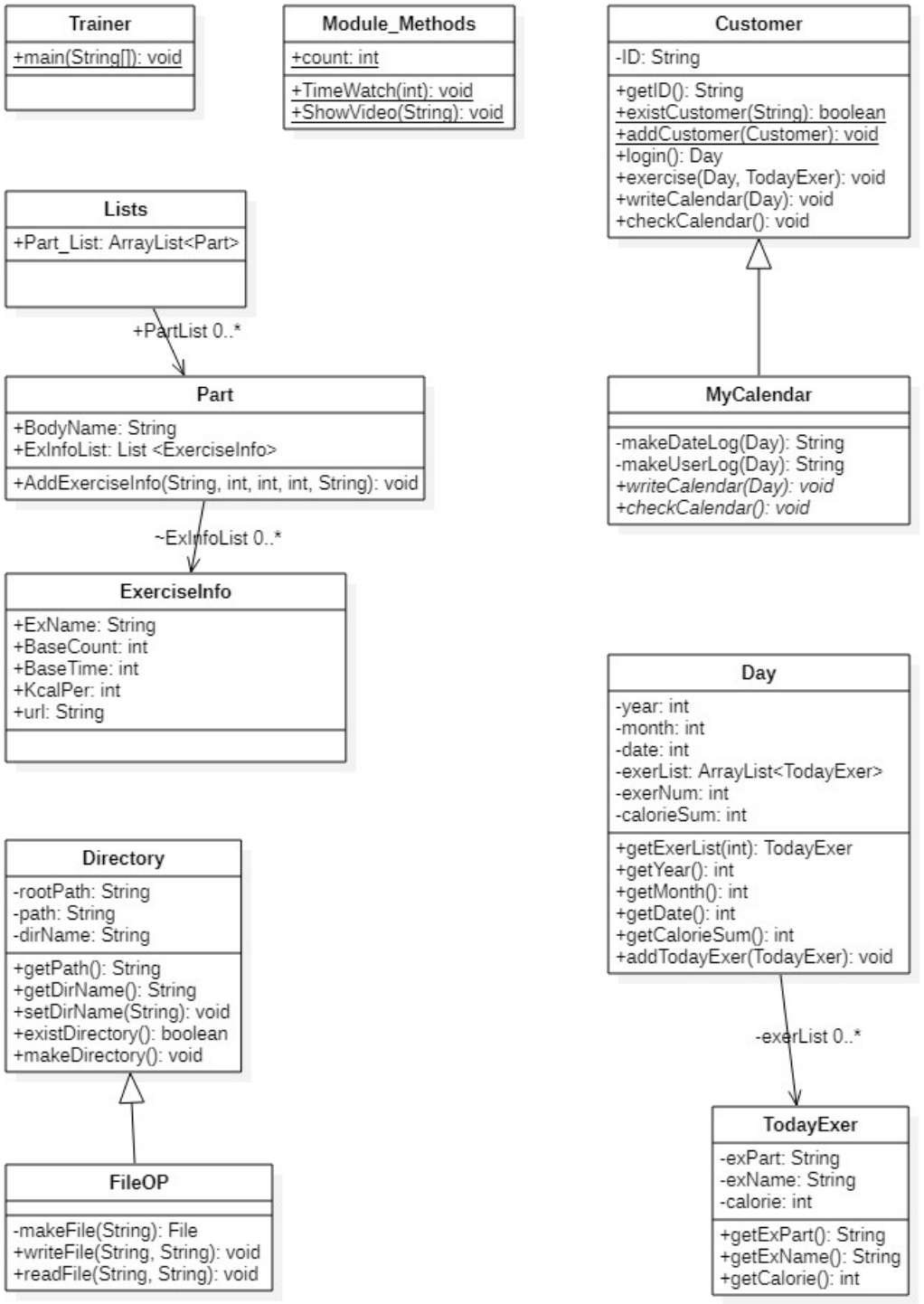
⑷ 운동 영상: 선택한 운동의 설명 영상을 보고 정확한 자세를 배울 수 있다.

⑸ 스톱 워치: 선택한 운동에 알맞게 자동 설정된 스톱워치를 활용할 수 있다.

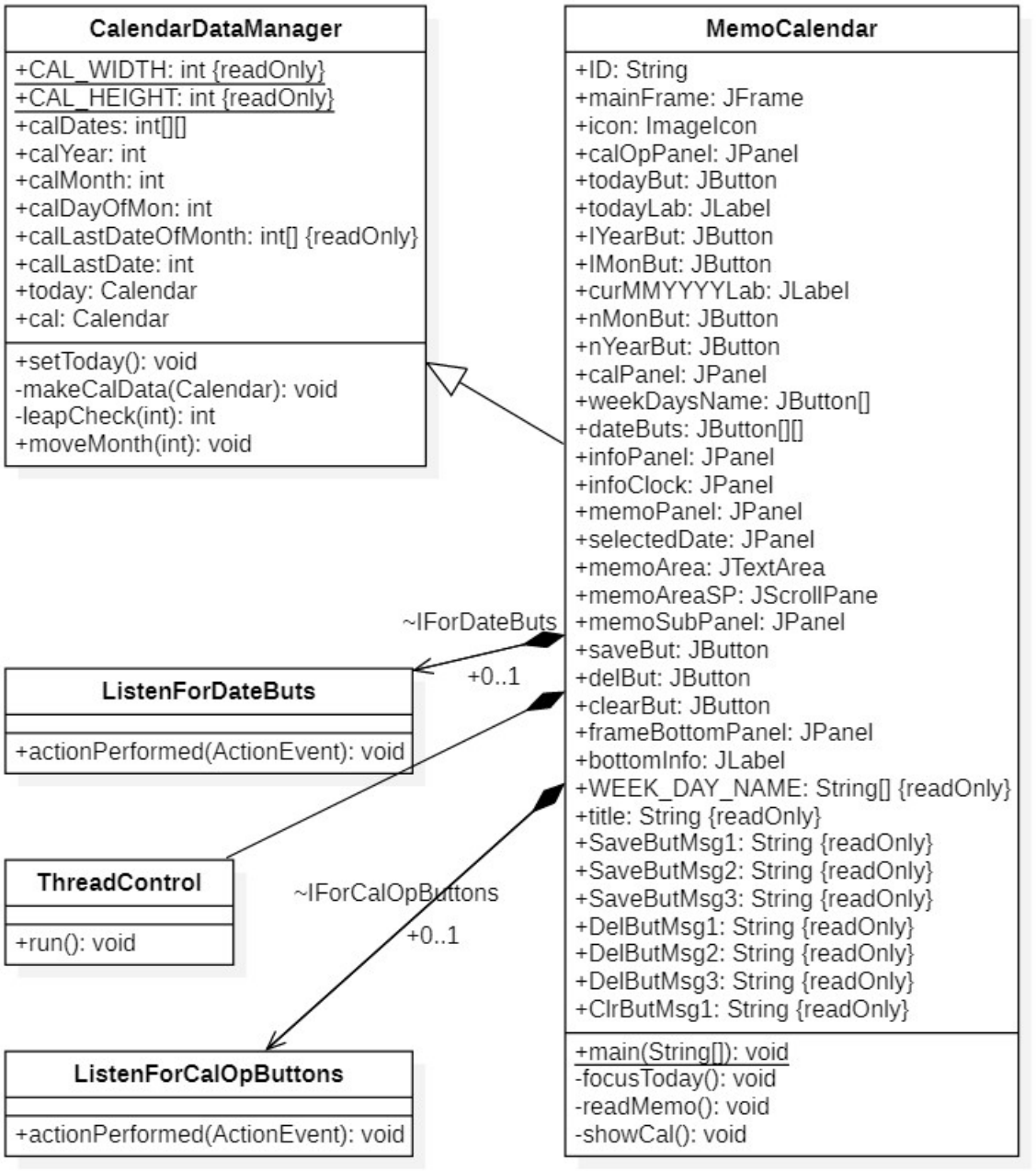
⑹ 캘린더: 사용자의 운동 기록을 달력 형태로 확인할 수 있다.

Ⅱ. UML 다이어그램

1. Functional implement



2. Graphical User Interface implement: 오픈소스 활용



Ⅲ. 클래스 설명

A. Trainer 클래스

Trainer 클래스는 '가짜 트레이너' 프로그램의 동작을 위한 클래스이다.

이 클래스는 main method를 가지고 있으며, 사용자가 프로그램의 기능을 선택할 수 있도록 전반적인 흐름을 제공한다.

B. Customer 클래스

Customer 클래스는 회원정보 등록 및 조회와 운동 수행을 관리하기 위한 클래스이다.

멤버변수 'ID'는 사용자의 아이디 정보를 저장하는 String 타입의 변수이다.

'회원정보 조회' 기능은 existCustomer() 메소드를 사용하여 수행한다.

'회원정보 등록' 기능은 addCustomer() 메소드를 사용하여 수행한다.

'로그인' 기능은 login() 메소드를 사용하여 수행한다.

'운동하기' 기능은 exercise() 메소드를 사용하여 수행한다.

C. MyCalendar 클래스

MyCalendar 클래스는 캘린더 작성을 위한 클래스이다.

해당 클래스는 Customer 클래스를 상속하고 있다: Customer의 서브 클래스

'날짜정보 스트링 생성' 기능은 makeDateLog() 메소드를 사용하여 수행한다.

'운동기록 스트링 생성' 기능은 makeUserLog() 메소드를 사용하여 수행한다.

'달력 작성하기' 기능은 상위 클래스의 writeCalendar() 메소드를 오버라이드 하여 수행한다.

'달력 보기' 기능은 상위 클래스의 checkCalendar() 메소드를 오버라이드 하여 수행한다.

D. Lists 클래스

Lists 클래스는 운동정보를 운동부위별로 분류하여 저장하기 위해 필요한 클래스이다.

멤버변수 'Part_List'는 부위별로 분류한 운동정보를 저장하기 위한 ArrayList<Part> 타입의 변수이다.

E. Part 클래스

Part 클래스는 특정부위의 운동끼리 묶어서 저장하기 위해 필요한 클래스이다.

멤버변수 'BodyName'는 운동부위의 이름을 저장하는 String 타입의 변수이다.

멤버변수 'ExInfoList'는 운동정보를 저장하기 위한 List<ExerciseInfo> 타입의 변수이다.

'운동정보 추가' 기능은 AddExerciseInfo() 메소드를 사용하여 수행한다.

F. ExerciseInfo 클래스

ExerciseInfo 클래스는 하나의 운동정보(운동이름, 기본횟수, 기본시간, 칼로리정보, URL)를 저장하기 위해 필요한 클래스이다.

멤버변수 'ExName'는 운동이름을 저장하는 String 타입의 변수이다.

멤버변수 'BaseCount', 'BaseTime', 'KcalPer'는 각각 기본횟수, 기본시간, 칼로리 정보를 저장하는 int 타입의 변수이다.

멤버변수 'url'는 운동영상의 URL을 저장하는 String 타입의 변수이다.

G. Day 클래스

Day 클래스는 운동기록을 저장하기 위한 클래스이다.

멤버변수 ‘year’, ‘month’, ‘date’는 각각 날짜정보(년도/월/일)를 저장하는 int 타입의 변수이다.

멤버변수 ‘exerList’는 하루동안 수행한 운동정보를 저장하기 위한 ArrayList<TodayExer> 타입의 변수이고, ‘exerNum’은 exerList의 사이즈를 나타내기 위한 int 타입의 변수이다.

멤버변수 ‘calorieSum’은 하루동안 소모한 총 칼로리를 저장하기 위한 int 타입의 변수이다.

'운동기록 저장' 기능은 addTodayExer() 메소드를 사용하여 수행한다.

H. TodayExer 클래스

TodayExer 클래스는 수행한 운동정보(운동부위, 운동이름, 칼로리정보)를 표현하는 클래스이다.

멤버변수인 ‘exPart’, ‘exName’, ‘calorie’는 각각 운동부위, 운동이름, 칼로리정보를 저장하기 위한 String 타입 또는 int 타입의 변수이다.

I. Module_Methods 클래스

Module_Methods 클래스는 영상 시청 및 스탑워치 실행을 위해 필요한 클래스이다.

멤버변수 ‘count’는 남은 시간을 계산하기 위한 int 타입의 변수이다.

'안내영상 시청' 기능은 ShowVideo() 메소드를 사용하여 수행한다.

'스탑워치' 기능은 TimeWatch() 메소드를 사용하여 수행한다.

J. Directory 클래스

Directory 클래스는 디렉토리 확인 및 생성을 위한 클래스이다.

멤버변수 ‘rootPath’는 상위 디렉토리의 경로를 저장하는 String 타입의 변수이다.

멤버변수 ‘path’는 생성할 디렉토리의 경로를 저장하는 String 타입의 변수이다.

멤버변수 ‘dirName’는 생성할 디렉토리의 이름을 저장하는 String 타입의 변수이다.

'디렉토리 확인' 기능은 existDirectory() 메소드를 사용하여 수행한다.

'디렉토리 생성' 기능은 makeDirectory() 메소드를 사용하여 수행한다.

K. FileOP 클래스

FileOP 클래스는 텍스트 파일 생성과 쓰기 및 읽기를 위한 클래스이다.

해당 클래스는 Directory 클래스를 상속하고 있다: Directory의 서브 클래스

'파일 생성' 기능은 makeFile() 메소드를 사용하여 수행한다.

'파일 쓰기' 기능은 writeFile() 메소드를 사용하여 수행한다.

'파일 읽기' 기능은 readFile() 메소드를 사용하여 수행한다.

Ⅳ. OOP 컨셉

1. Encapsulation(캡슐화)와 정보은닉(Information Hiding)

- 멤버변수의 Access modifier: private

```
private int year;
private int month;
private int date;

private ArrayList<TodayExer> exerList = new ArrayList<TodayExer>(); // 하루동안 수행한 운동정보를 저장
private int exerNum;     // 하루동안 수행한 운동종목의 개수
private int calorieSum; // 하루동안 소모한 총 칼로리
```

[그림1] Day 클래스의 멤버변수

```
public TodayExer getExerList(int index) {
    return exerList.get(index);
}

public int getYear() {
    return year;
}

public int getMonth() {
    return month;
}

public int getDate() {
    return date;
}

public int getExerNum() {
    return exerNum;
}

public int getCalorieSum() {
    return calorieSum;
}
```

[그림2] Day 클래스의 Getter

Day 객체의 멤버 변수의 Access modifier를 private로 선언함으로써, 날짜 정보와 운동 기록을 외부에서 직접적으로 접근하지 못하게 제어한다. 따라서 Day 객체의 멤버 변수 값에 접근하기 위해서는 Getter 함수를 사용하여 간접적으로 접근해야 한다. 이를 통해 외부 클래스와의 결합도를 낮추고 해당 클래스가 독립적으로 기능할 수 있도록 하며, 코드 변경 시 해당 클래스만 수정함으로써 코드 수정양을 줄일 수 있다.

- 메소드의 Access modifier: private

```
private File makeFile(String date) {
    String fileName = date + ".txt";
    String filePath = super.getPath() + "\\" + fileName;
    File file = new File(filePath);
    return file;
}
```

```java
public void writeFile(String date, String userLog) {
    File file = makeFile(date);
    try(FileWriter fw = new FileWriter(file, true)) {
        fw.write(userLog);
        fw.flush();
    } catch(IOException e) {
        e.printStackTrace();
    }
}
```

[그림3] FileOP 클래스의 makeFile()와 writeFile()

FileOP 클래스의 makeFile() 메소드는 writeFile() 메소드 내부에서만 사용되기 때문에 makeFile()의 Access modifier를 private로 선언함으로써, writeFile() 메소드의 내부 로직을 감추고 외부에는 '파일 쓰기' 기능만을 제공하여 메소드 사용의 편의를 높인다.

```java
private String makeDateLog(Day date_records) {
private String makeUserLog(Day exer_records) {
```

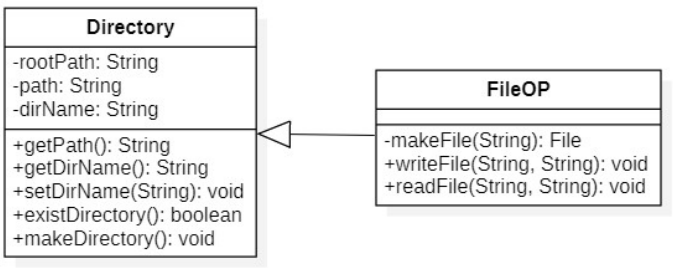
```java
public void writeCalendar(Day exer_records) {
    FileOP file_op = new FileOP(getID());
    String dateLog = makeDateLog(exer_records);
    String userLog = makeUserLog(exer_records);
    file_op.writeFile(dateLog, userLog);
}
```

[그림4] MyCalendar 클래스의 makeDateLog() / makeUserLog()와 writeCalendar()

MyCalendar의 makeDateLog()와 makeUserLog() 메소드 또한 writeCalendar() 메소드 내부에서만 사용되기 때문에, 두 메소드의 Access modifier를 private로 선언하여 외부에는 '달력 작성하기' 기능만을 제공한다.

2. Inheritance(상속)

- Directory 클래스와 FileOP 클래스의 상속관계



[그림5] Directory 클래스와 FileOP 클래스의 상속관계: UML 다이어그램

```java
public class Directory {

    private String rootPath; // 현재 작업 경로
    private String path;     // 디렉토리 경로
    private String dirName;  // 디렉토리 이름

    public Directory() {

    }

    public Directory(String dirName) {
        File file = new File(".");
        rootPath = file.getAbsolutePath();
        path = rootPath + "\\" + dirName;
        this.dirName = dirName;
    }
```

```java
public class FileOP extends Directory {

    public FileOP() {

    }

    public FileOP(String name) {
        super(name);
    }
```

[그림6] Directory 클래스와 FileOP 클래스의 생성자

```java
private File makeFile(String date) {
    String fileName = date + ".txt";
    String filePath = super.getPath() + "\\" + fileName;
    File file = new File(filePath);
    return file;
}
```

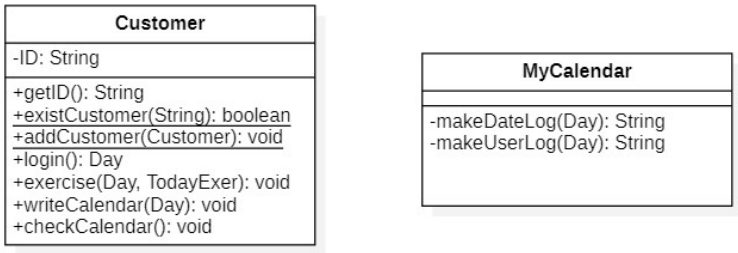
[그림7] FileOP 클래스의 makeFile()

FileOP 클래스는 Directory 클래스가 가지고 있는 속성(멤버변수)을 그대로 사용하기 때문에, Directory 클래스를 상속함으로써 상위 클래스의 속성을 그대로 물려받아 자신의 속성으로 사용한다.

또한 FileOP 클래스의 makeFile() 메소드는 내부적으로 파일경로를 만들어야 하는데, 이때 자신의 상위 클래스인 Directory 클래스의 getPath() 메소드를 활용하여 파일경로를 생성한다. 즉, 파일경로 생성에 관한 연산을 다시 정의하지 않고 상위 클래스의 메소드를 사용함으로써 중복 코드를 없애고 유지 보수성을 개선한다.

3. Polymorphism(다형성)

- Customer 클래스와 MyCalendar 클래스의 관계



[그림7] Customer 클래스와 MyCalendar 클래스의 초기구상: UML 다이어그램

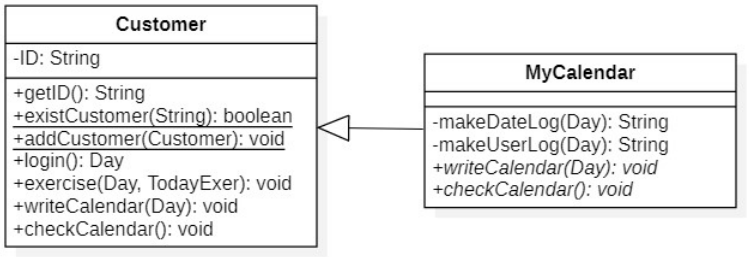
Customer 클래스와 MyCalendar 클래스의 초기구상은 위 그림과 같고, 아래의 요구사항을 만족함.

(1) Customer 클래스는 writeCalendar() 메소드를 통해 '달력 작성하기' 기능을 수행한다.

(2) Customer 클래스는 checkCalendar() 메소드를 통해 '달력 보기' 기능을 수행한다.

즉, Customer 객체는 '달력 작성하기'와 '달력 보기'의 행위를 수행한다.

하지만 실제로 writeCalendar() 메소드를 구현하려면 MyCalendar 클래스의 makeDateLog()와 makeUserLog() 메소드가 필요하다. 따라서 두 클래스 간의 유사성보다는 '오버라이딩' 기법을 활용하기 위해 두 클래스의 관계를 상속관계로 변경하고, MyCalendar 클래스가 Customer 클래스의 writeCalendar()와 checkCalendar() 메소드를 오버라이드 하도록 코드를 수정하였다.



[그림8] Customer 클래스와 MyCalendar 클래스의 상속관계: UML 다이어그램

```
System.out.println(ID + "님 환영합니다.");
Customer cus = new MyCalendar(ID); // polymorphism!
Day exer_records = cus.login();
cus.exercise(exer_records, new TodayExer(Base.Part_
cus.writeCalendar(exer_records);
```

[그림9] Trainer 클래스의 main method: Polymorphism(다형성)

이러한 상속관계를 통해 [그림9]와 같은 동작을 가능하게 할 수 있다. 예를 들어, Customer 타입의 레퍼런스 변수 'cus'가 있다고 가정하자. 이때 변수 'cus'는 Customer 타입의 객체뿐 만이 아니라 Customer 클래스를 상속하고 있는 서브 클래스 타입의 객체 또한 가리킬 수 있다.

Customer cus = new MyCalendar();

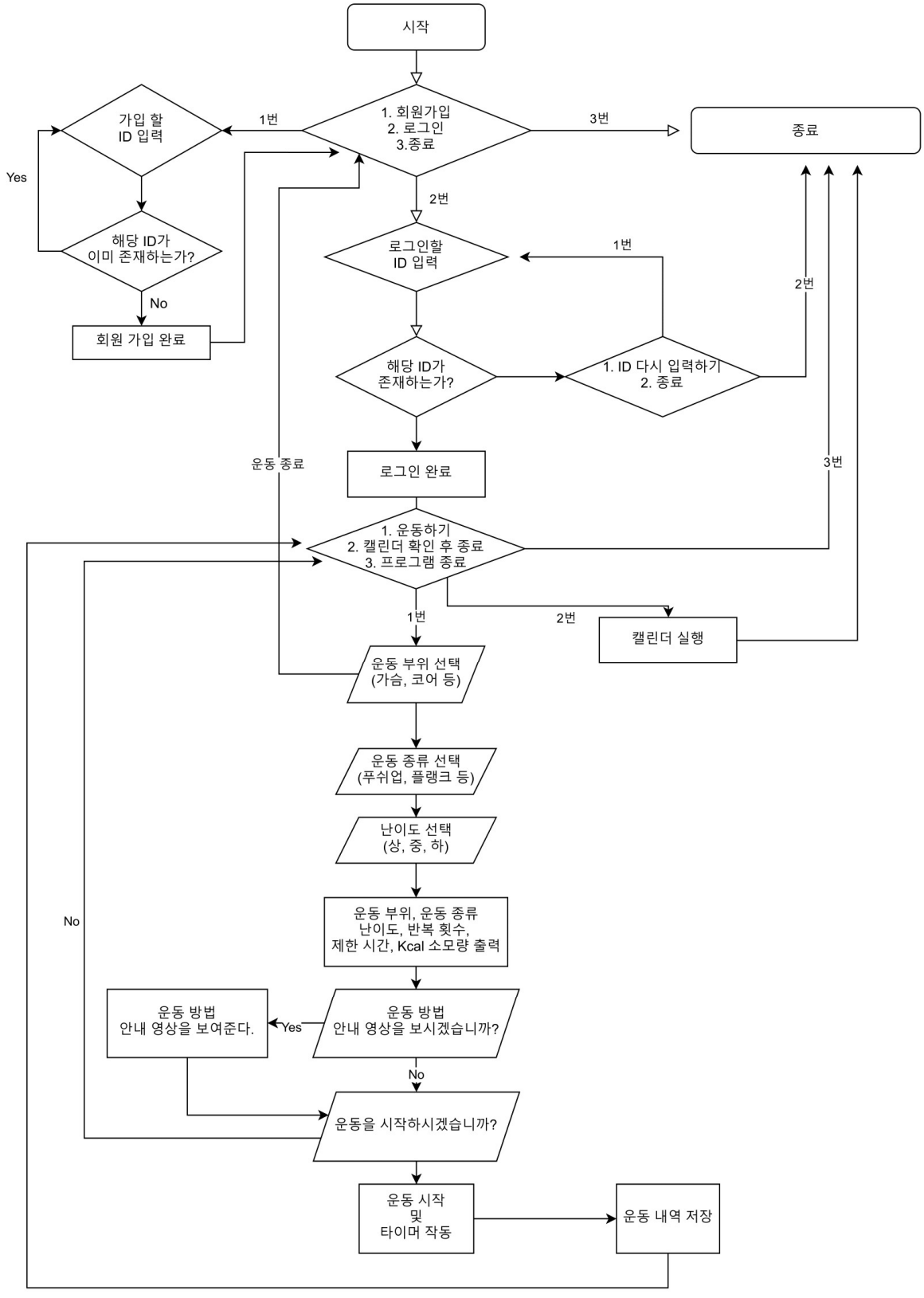
cus.writeCalendar();

cus.checkCalendar();

이때, 변수 'cus'가 가리키는 객체의 타입이 MyCalendar이므로, 호출되는 writeCalendar() 메소드와 checkCalendar() 메소드 모두 MyCalendar 클래스에서 오버라이드한 메소드이다.

Ⅴ. 알고리즘 설명

- Trainer 클래스 내 main method의 알고리즘



Ⅵ. 팀원 명단

수학과	201621136	이재협
수학과	201621138	허창현
소프트웨어학과	201823782	안지영