중 간 보 고 서

[MBTI 기반 이상형 매칭 애플리케이션]

과 목 명 :	분산시스템
담당교수 :	임민규 교수님
팀 명 :	분산시스템 10 팀
팀 원 :	유학현 (201713066) 김세영 (201713049) 박용준 (201711147) 서동재 (201713060) 윤영기 (201713025)
제 출 일 :	2021.04.30

목 차

1. 팀 구성 및 역할

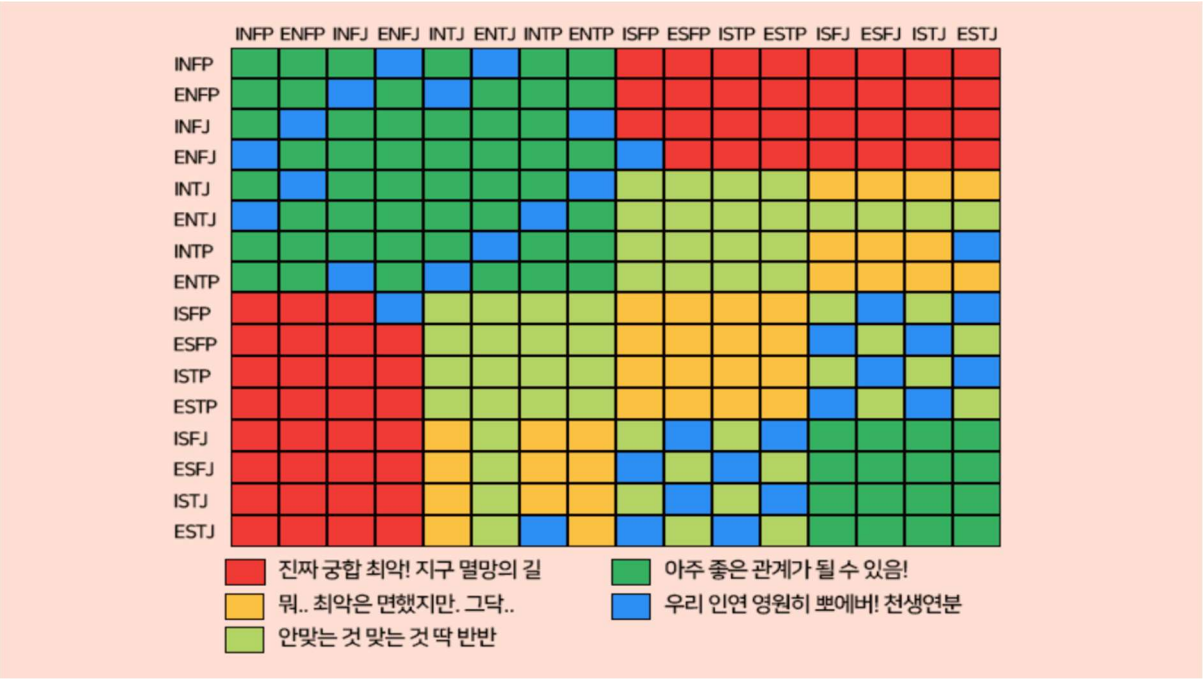
이름	역할
유학현 (팀장)	프로젝트 관리, 통합 및 개발 지원
김세영	CM 기반 이벤트 설계 및 개발
박용준	기본 프로그램, 서버 설계 및 개발
서동재	세부 프로그램 통합 및 개발 지원
윤영기	데이터베이스 설계 및 관리

2. 프로젝트 목표

2.1 프로젝트 개요

2019 년 하반기를 기점으로 전세계적으로 퍼져나간 코로나 19 바이러스의 대유행은 우리의 일상생활에 큰 변화를 가져왔다. 코로나 예방 수칙 중 하나인 '사회적 거리두기'의 장기화로 인해 **언택트 (Untact)**의 시대를 맞이하게 됨으로써 사람들은 기존 오프라인으로 이루어지던 모임들을 재택근무, 온라인 수업 그리고 화상회의 같은 온라인상의 모임으로 대체하게 되었다. 이러한 형태의 모임 증가는 이를 지원하기 위한 애플리케이션에 대한 수요 증가로 나타났으며 이는 곧 ZOOM, MS Teams 와 같은 온라인 회의 지원 애플리케이션의 사용률 증가로 이어졌다. 이렇듯 학업, 업무 등의 다양한 **오프라인 활동들을 온라인으로 대체하고자 하는 변화**는 이성과의 만남 방식 또한 바꾸어 놓았다. 기존의 오프라인 소개팅을 온라인으로 대체하기 시작한 것이다. 이는 곧, Tinder, Bumble 과 같은 **온라인 데이팅 어플리케이션의 수요 증가**로 이어지게 되었다. 따라서 우리는 이러한 수요 증가를 기반으로 기존 데이팅 애플리케이션과 **차별화된 특성을 갖는 데이팅 애플리케이션**의 개발을 목표로 하였다.

2.2 목표 기능



[그림 1] MBTI 성격 유형별 궁합 차트

최근 젊은 층 사이에서 화제가 되고 있는 성격 유형 검사 마이어스-브릭스 유형 지표(Myers-Briggs Type Indicator), 약칭 MBTI 를 활용한 이상형 매칭 및 데이팅 애플리케이션으로 다음과 같은 서비스를 제공한다.

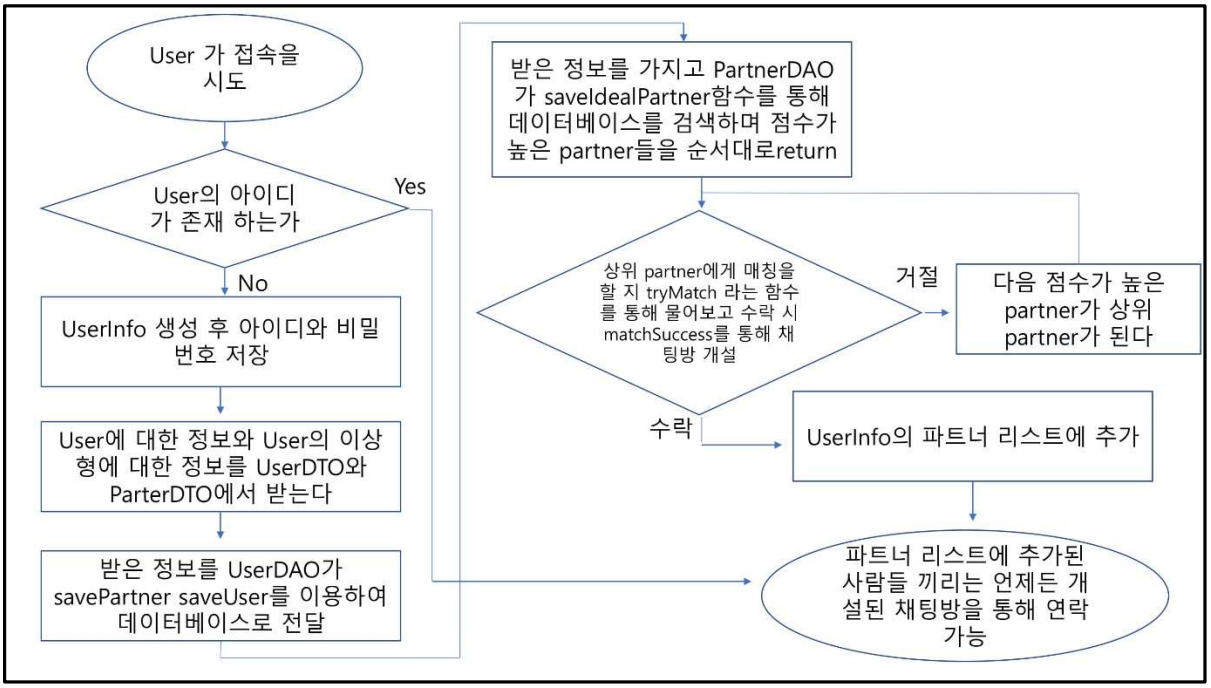
- ID 와 PW 를 사용하여 사용자 계정 생성 및 로그인한다.
- 사용자의 기본 정보 (키, 체중, MBTI 와 취미 등) 와 사용자가 원하는 이성의 조건 정보를 입력 받아 데이터베이스에 저장한다.
- 새로운 사용자가 추가되면 다른 사용자들이 이를 확인할 수 있다.
- 데이터베이스에 저장된 사용자 입력 정보를 서버의 이상형 추천 로직에 대입하여 사용자 사이의 상호 평가 점수를 산출한 뒤, 이를 기반으로 이상형을 매칭하여 사용자에게 나타낸다.
- 사용자 간의 매칭이 성사되면 CM 을 기반으로 1:1 대화가 가능하도록 한다.

2.3 개발 환경

- 개발 언어 : JAVA
- 사용 IDE : Eclipse IDE for Java Developers (includes Incubating components)
- 사용 프레임워크 : CM
- 데이터베이스 : MySQL (mysql-connector-java-5.1.31-bin.jar)

3. 프로젝트 설계

3.1 프로그램 개요

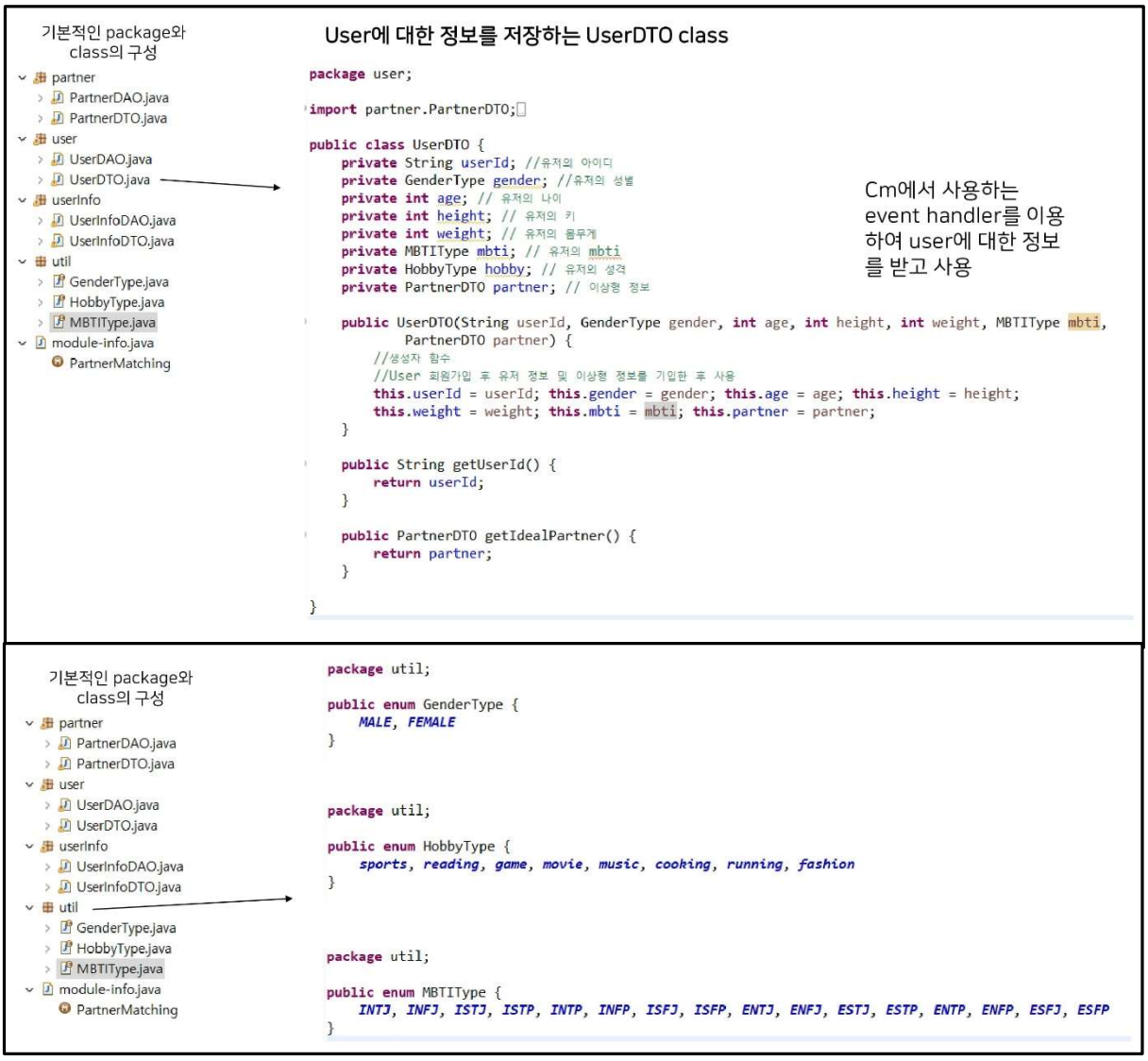


[그림 2] 이상형 매칭 애플리케이션의 간략한 구조도

- UserInfo 를 사용하여 존재하는 사용자인지 데이터베이스 조회
- UserDTO, PartnerDTO, UserDAO, PartnerDAO 사용하여 적절한 파트너 추천
- 파트너 매치 시 CM 통하여 채팅방 개설 및 대화 가능

3.2 세부 설계

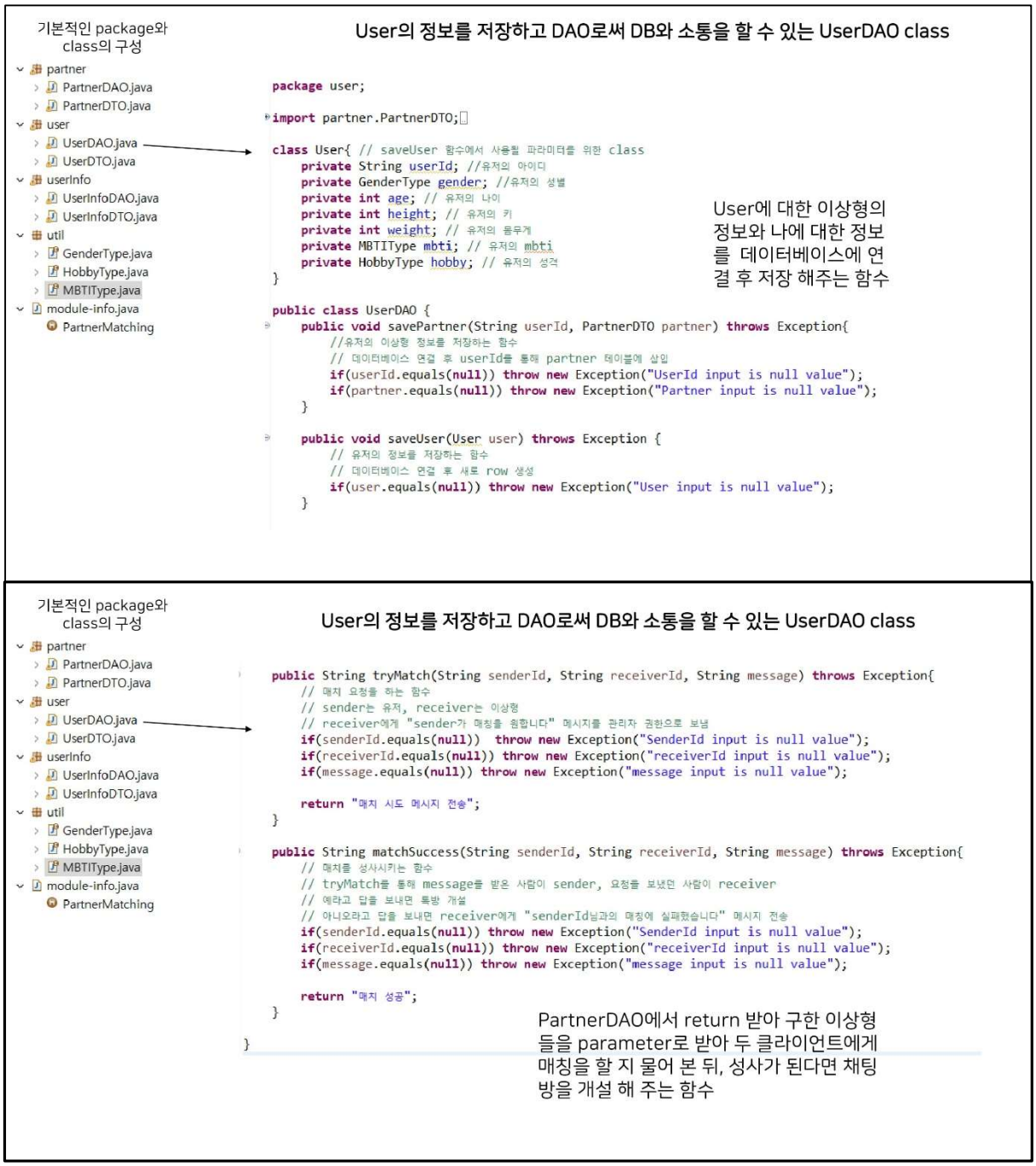
- **UserDTO**



[그림 1, 2] UserDTO Class 설계

UserDTO 는 사용자에 대한 정보를 저장하는 클래스로, CM event 를 통해 사용자의 계정 정보(ID)와 이상형 매칭을 위하여 필요한 정보 (사용자의 신체 정보, MBTI 유형 및 취미, 이상형 조건) 를 입력 받은 뒤, 새로운 사용자 정보 객체를 생성한다. 이때 사용되는 GenderType, MBTIType, HobbyType 은 enum 을 사용하여 나타낸다.

- **UserDAO**



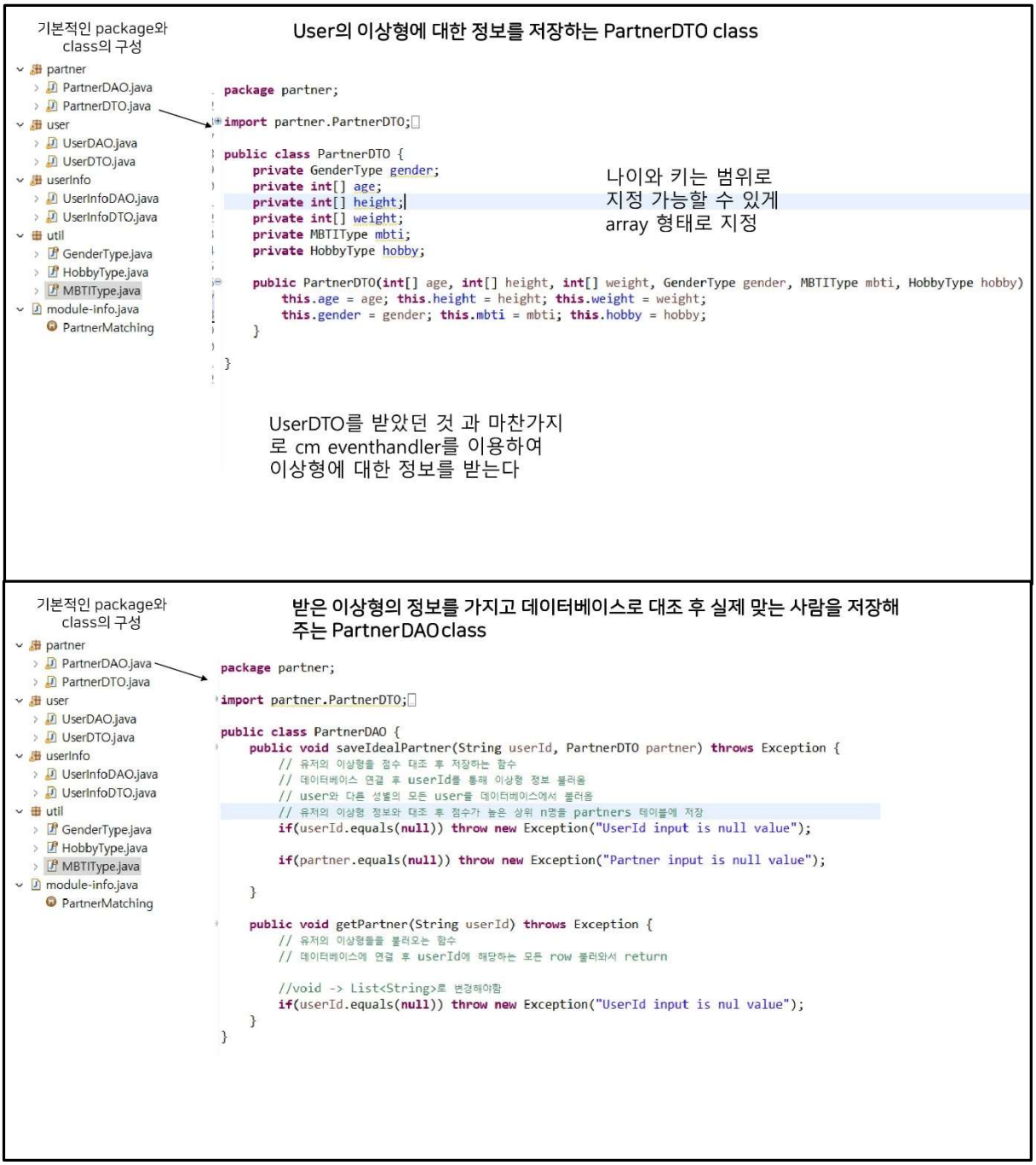

```java
package user;

import partner.PartnerDTO;

class User{ // saveUser 함수에서 사용될 파라미터를 위한 class
    private String userId; //유저의 아이디
    private GenderType gender; //유저의 성별
    private int age; // 유저의 나이
    private int height; // 유저의 키
    private int weight; // 유저의 몸무게
    private MBTIType mbti; // 유저의 mbti
    private HobbyType hobby; // 유저의 성격
}

public class UserDAO {
    public void savePartner(String userId, PartnerDTO partner) throws Exception{
        //유저의 이상형 정보를 저장하는 함수
        // 데이터베이스 연결 후 userId를 통해 partner 테이블에 삽입
        if(userId.equals(null)) throw new Exception("UserId input is null value");
        if(partner.equals(null)) throw new Exception("Partner input is null value");
    }

    public void saveUser(User user) throws Exception {
        // 유저의 정보를 저장하는 함수
        // 데이터베이스 연결 후 새로 row 생성
        if(user.equals(null)) throw new Exception("User input is null value");
    }
```

```java
    public String tryMatch(String senderId, String receiverId, String message) throws Exception{
        // 매치 요청을 하는 함수
        // sender는 유저, receiver는 이상형
        // receiver에게 "sender가 매칭을 원합니다" 메시지를 관리자 권한으로 보냄
        if(senderId.equals(null))  throw new Exception("SenderId input is null value");
        if(receiverId.equals(null)) throw new Exception("receiverId input is null value");
        if(message.equals(null)) throw new Exception("message input is null value");

        return "매치 시도 메시지 전송";
    }

    public String matchSuccess(String senderId, String receiverId, String message) throws Exception{
        // 매치를 성사시키는 함수
        // tryMatch를 통해 message를 받은 사람이 sender, 요청을 보냈던 사람이 receiver
        // 예라고 답을 보내면 특방 개설
        // 아니오라고 답을 보내면 receiver에게 "senderId님과의 매칭에 실패했습니다" 메시지 전송
        if(senderId.equals(null)) throw new Exception("SenderId input is null value");
        if(receiverId.equals(null)) throw new Exception("receiverId input is null value");
        if(message.equals(null)) throw new Exception("message input is null value");

        return "매치 성공";
    }
}
```

[그림 3, 4] UserDAO Class 설계

사용자 정보를 저장하고 데이터베이스와의 커뮤니케이션을 담당하는 클래스로, UserDTO 를 통해 생성된 사용자 정보 객체를 데이터베이스에 저장한다. 또한, 후술할 PartnerDAO 를 통해 매칭된 사용자들에게 매칭 확정 또는 거절 의사를 물어본 뒤, 매칭 확정 시 CM 을 통해 채팅할 수 있도록 한다.

- **PartnerDTO/DAO**

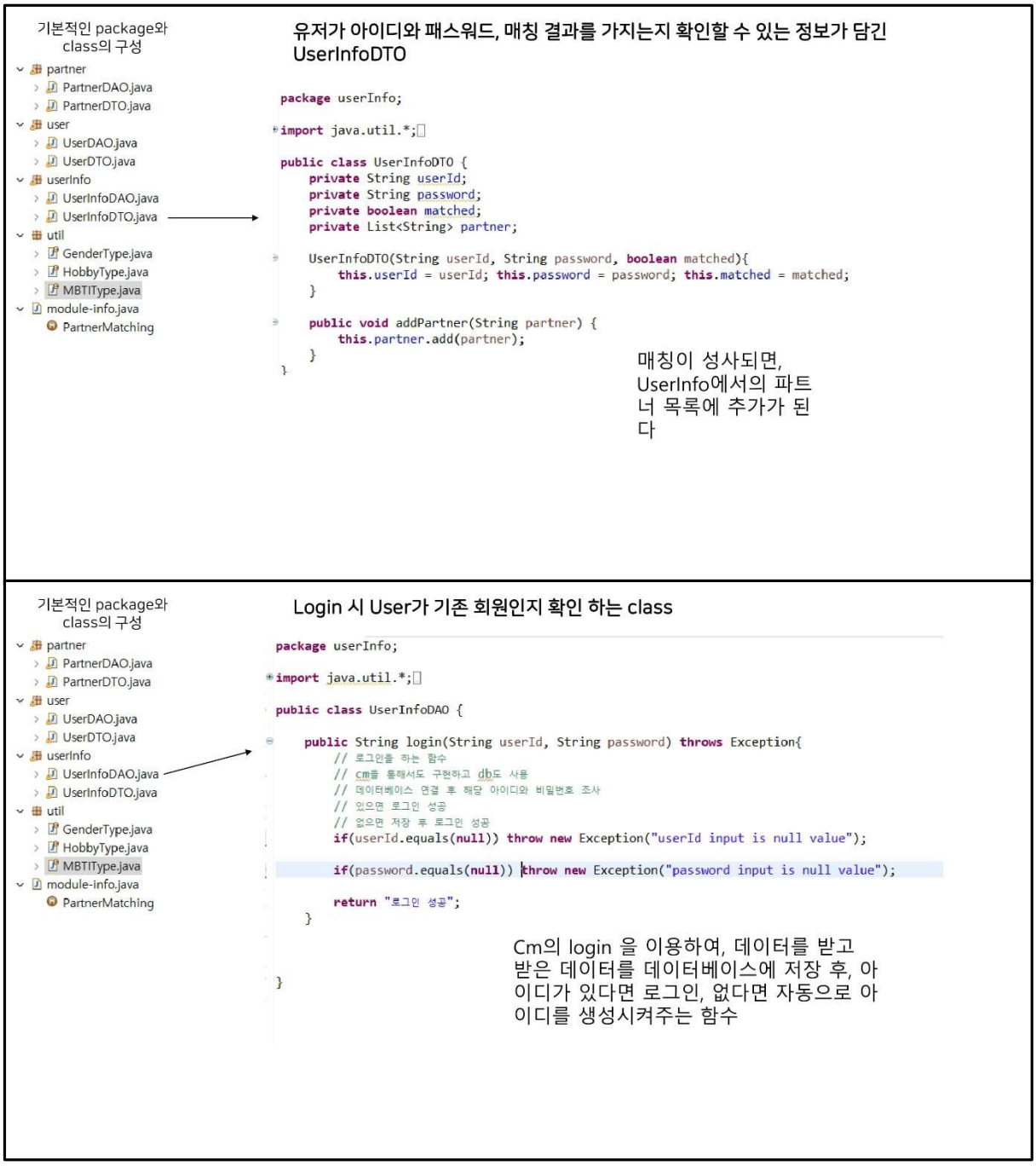


[그림 5, 6] PartnerDTO/DAO Class 설계

PartnerDTO 는 UserDTO 에서의 이상형 조건 정보를 저장하기 위한 클래스로, UserDTO 와 마찬가지로 CM event 를 통해 원하는 조건 정보 (신체 조건, MBTI 유형, 취미) 를 입력 받는다.

PartnerDAO 는 UserDTO 의 PartnerDTO 를 기반으로 이상형 점수를 산출하여, 이를 기반으로 생성/정렬된 이상형 리스트를 사용자에게 나타낸다.

- **UserInfoDTO/DAO**



[그림 7, 8] UserInfoDTO/DAO Class 설계

UserInfoDTO 는 사용자의 계정 정보와 매칭 여부를 저장하기 위한 클래스로, 사용자의 ID/PW 와 매칭이 성사된 사용자의 정보를 저장한다.

UserInfoDAO 는 사용자의 로그인 시 사용되는 클래스로, CM 의 로그인 기능을 통해 전달받은 ID/PW 가 데이터베이스 내에 존재하면 해당 계정 정보를 불러오며 그렇지 않을 시, 입력된 ID/PW 로 계정을 자동 생성한다.

3.3 CM 프로토콜 정의

Event type		
Event ID		
Event field	Field datatype	Field definition

위의 제시된 차트를 사용하여 새롭게 정의된 CM 프로토콜에 대해 설명한다.

- Event type : 사용되는 CM Event type
- Event ID : 해당 Event ID
- Event field : Event에 포함되는 각 데이터의 Field
- Field datatype : 해당 Field의 데이터 타입
- Field definition : 해당 Field 설명

예시)

Event type		CMInfo.CM_USER_EVENT
Event ID		"SEND_ACCOUNT"
Event field	Field datatype	Field definition
USER_ID	CMInfo.CM_STR	사용자의 ID
USER_PW	CMInfo.CM_STR	사용자의 PW

- CMInfo.CM_USER_EVENT을 이용하며 "SEND_ACCOUNT"로 정의된 프로토콜.
  데이터 타입으로 CMInfo.CM_STR을 갖는 USER_ID와 USER_PW를 보내며 각각 사용자의 ID, PW를 의미한다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"inputUserProfile"
Event field	Field datatype	Field definition
userID	CMInfo.CM_STR	사용자의 이름 정보
gender	CMInfo.CM_INT	사용자의 성별 정보
age	CMInfo.CM_INT	사용자의 나이 정보
height	CMInfo.CM_INT	사용자의 신장 정보
weight	CMInfo.CM_INT	사용자의 체중 정보
mbti	CMInfo.CM_INT	사용자의 mbti 타입 정보
hobby	CMInfo.CM_STR	사용자의 취미

- 사용자가 자신의 정보를 입력할 때 클라이언트가 서버에게 전송하는 유저 이벤트이다. 입력받은 정보를 이벤트에 담아 서버가 받아 볼 수 있다. 서버는 이 이벤트를 받으면 saveUser()에서 사용자의 프로필 정보를 DB에 저장한다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"inputPartnerProfile"
Event field	Field datatype	Field definition
senderID	CMInfo.CM_STR	사용자의 이름 정보
gender	CMInfo.CM_INT	사용자가 원하는 파트너의 성별 정보
age	CMInfo.CM_INT	사용자가 원하는 파트너의 나이 정보
height	CMInfo.CM_INT	사용자가 원하는 파트너의 신장 정보
weight	CMInfo.CM_INT	사용자가 원하는 파트너의 체중 정보
mbti	CMInfo.CM_INT	사용자가 원하는 파트너의 mbti 타입 정보
hobby	CMInfo.CM_STR	사용자가 원하는 파트너의 취미

- 사용자가 자신의 이상형 정보를 입력할 때 클라이언트가 서버에게 전송하는 유저 이벤트이다. 입력받은 정보를 이벤트에 담아 서버가 받아 볼 수 있다. 서버가 이 이벤트를 받으면 savePartner()에서 이 정보를 DB에 저장한다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"tryMatch"
Event field	Field datatype	Field definition
sender ID	CMInfo.CM_STR	매칭을 시도하는 사용자의 ID 정보
receiver ID	CMInfo.CM_STR	매칭을 희망하는 파트너의 ID 정보
message	CMInfo.CM_STR	사용자가 보내고 싶은 메시지

- 사용자가 원하는 사람과 매칭을 시도할 때 클라이언트가 서버에게 전송하는 유저 이벤트이다. 서버는 사용자가 원하는 대상의 ID를 받아보고 tryMatch()에서 대상에게 매칭 수락 여부를 물어본다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"askMatch"
Event field	Field datatype	Field definition
sender ID	CMInfo.CM_STR	해당 사용자와 매칭을 희망하는 사용자의 ID 정보
message	CMInfo.CM_STR	서버가 전달하는 메시지

- 서버가 유저 이벤트 "tryMatch"를 받고 나서 해당하는 대상에 매칭 수락 여부를 물어보려 할 때 서버가 클라이언트에게 전송하는 유저 이벤트이다. 대상 클라이언트는 어느 사용자가 자신과 매칭을 원하는지 정보를 받는다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"requestProfile"
Event field	Field datatype	Field definition
sender ID	CMInfo.CM_STR	프로필을 요청하는 사용자의 ID 정보
receiver ID	CMInfo.CM_STR	프로필 요청 대상이 되는 사용자의 ID 정보

- 매칭 요청을 받은 사용자가 매칭을 요청한 사용자의 프로필 정보를 서버에 요청할 때 클라이언트가 서버에게 전송하는 유저 이벤트이다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"sendProfile"
Event field	Field datatype	Field definition
userID	CMInfo.CM_STR	사용자의 이름 정보
gender	CMInfo.CM_INT	사용자의 성별 정보
age	CMInfo.CM_INT	사용자의 나이 정보
height	CMInfo.CM_INT	사용자의 신장 정보
weight	CMInfo.CM_INT	사용자의 체중 정보
mbti	CMInfo.CM_INT	사용자의 mbti 타입 정보
hobby	CMInfo.CM_STR	사용자의 취미

- 서버가 요청받은 프로필 정보를 보낼 때 서버가 클라이언트에게 전송하는 유저 이벤트이다. 서버는 DB에 저장된 사용자의 정보를 열람하여 그 정보를 이벤트에 담아 보낸다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"acceptMatch"
Event field	Field datatype	Field definition
senderID	CMInfo.CM_STR	매칭을 수락한 사용자의 ID 정보
partnerID	CMInfo.CM_STR	매칭이 수락된 사용자의 ID 정보
message	CMInfo.CM_INT	매칭 수락 여부

- 매칭 요청을 받은 사용자가 매칭을 수락할 때 클라이언트가 서버에게 전송하는 유저 이벤트이다. 서버는 메시지 내용을 보고 친구 관계 성립 여부를 판단한다. 만일 메시지 내용이 수락을 의미하는 1이라면 서버는 matchSucess()에서 매칭된 사용자 간 대화를 할 수 있도록 별도의 채팅방을 개설하는 등의 작업을 실행한다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"requestFriend"
Event field	Field datatype	Field definition
senderID	CMInfo.CM_STR	친구를 요청한 사용자의 ID 정보
partnerID	CMInfo.CM_STR	친구를 희망하는 사용자의 ID 정보

- 사용자가 마음에 드는 사용자와 친구 관계를 성립하고 싶을 때 클라이언트가 서버에게 전송하는 유저 이벤트이다. 서버는 사용자가 원하는 대상의 ID를 받아보고 대상에게 친구 요청 수락 여부를 물어본다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"askRequestFriend"
Event field	Field datatype	Field definition
senderID	CMInfo.CM_STR	친구 요청을 보낸 사용자의 ID 정보
partnerID	CMInfo.CM_STR	친구 요청을 받을 사용자의 ID 정보

- 서버가 사용자에게 대상과 친구 요청을 바라는 사용자가 있음을 알리고 수락 여부를 물어볼 때 서버가 클라이언트에게 전송하는 유저 이벤트이다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"responseFriendRequest"
Event field	Field datatype	Field definition
senderID	CMInfo.CM_STR	친구 요청을 처리하는 사용자의 ID 정보
partnerID	CMInfo.CM_STR	친구 요청을 시도한 사용자의 ID 정보
message	CMInfo.CM_INT	요청 수락 여부

- 사용자가 다른 사용자가 보낸 친구 요청을 수락할 때/혹은 거절할 때 클라이언트가 서버에게 전송하는 유저 이벤트이다. 서버는 메시지 내용을 보고 친구 관계 성립 여부를 판단한다. 만일 메시지 내용이 수락을 의미하는 1이라면 서버는 DB에 있는 친구 관계가 성

립된 두 사용자의 친구 테이블 정보를 업데이트하는데 이때 CM에서 기본 제공하는 addNewFriend() 메소드를 사용할 수 있다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"reportUser"
Event field	Field datatype	Field definition
senderID	CMInfo.CM_STR	신고하는 사용자의 ID 정보
repoertedID	CMInfo.CM_STR	신고받은 사용자의 ID 정보
message	CMInfo.CM_INT	신고 내용

- 사용자가 다른 사용자를 신고/차단하기를 원할 때 클라이언트가 서버에게 전송하는 유저 이벤트이다. 서버는 메시지 내용을 보고 부적절한 사용자를 신고한 것인지 차단을 요청한 것인지 판단한다. 메시지 내용이 부적절한 사용자 신고라면 신고받은 사용자에게 경고 메시지를 보낸다. 사용자가 차단을 원하거나 신고받은 사용자와 신고한 사용자가 친구 관계라면 DB에 있는 친구 관계를 정리한다. 이때 CM에서 기본 제공하는 removeFriend() 메소드를 사용할 수 있다.

Event type		CMInfo.CM_USER_EVENT
Event ID		"requestSecession"
Event field	Field datatype	Field definition
senderID	CMInfo.CM_STR	탈퇴를 희망하는 사용자의 ID 정보

- 사용자가 해당 서비스를 탈퇴하려 할 때 클라이언트가 서버에게 전송하는 유저 이벤트이다. 서버가 이 요청을 받으면 DB 에 있는 사용자와 관련된 정보들을 삭제한다.