I. 프로젝트 진행 3차

프로젝트명 : 6조 타임캡슐

교과목명	고급프로그래밍설계
담당교수	윤성림
팀 장	201701780 임채원
팀 원	201502139 김리한
	201701721 김민수
	201901839 최다솜

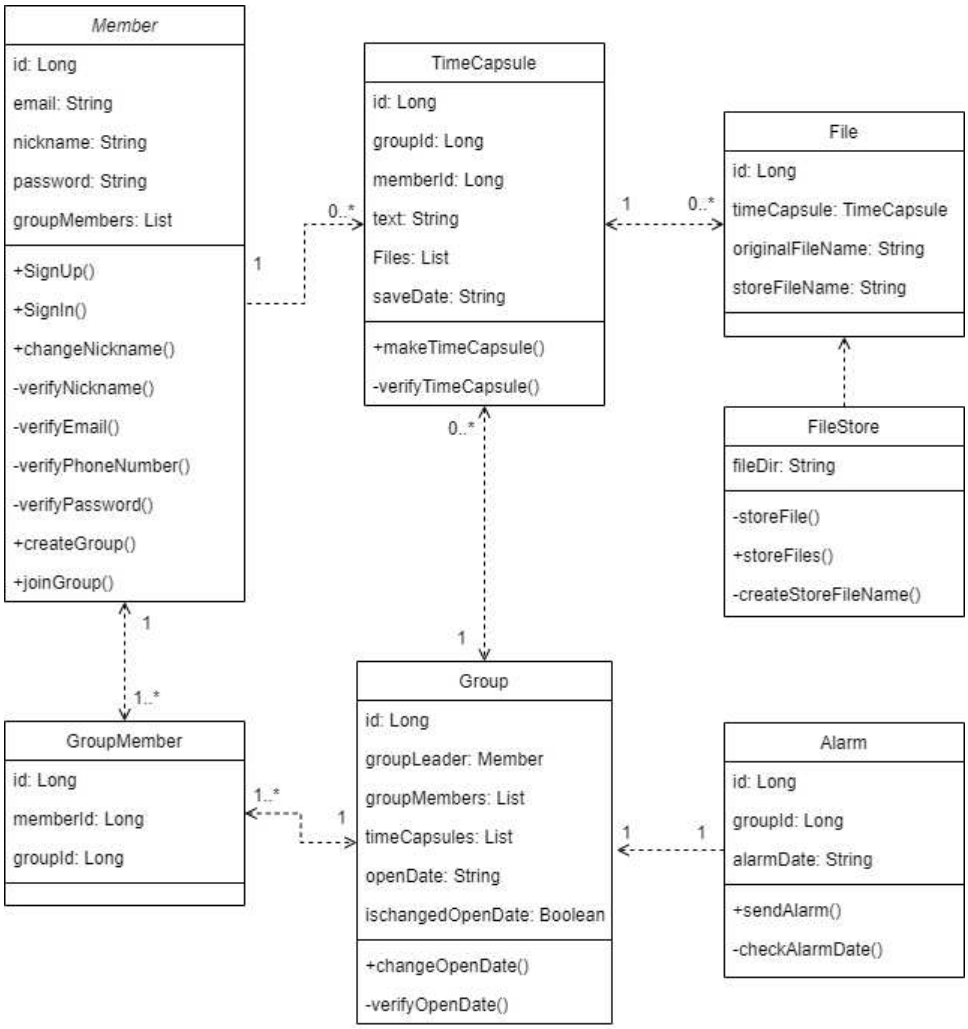
※5월 24일 프로젝트 진행 3차

- 1차에서 유스케이스 다이어그램과 클래스 다이어그램, 2차에서 시퀀스 다이어그램과 통신 다이어그램을 완료하면서 4개의 다이어그램들로 진행하였습니다.

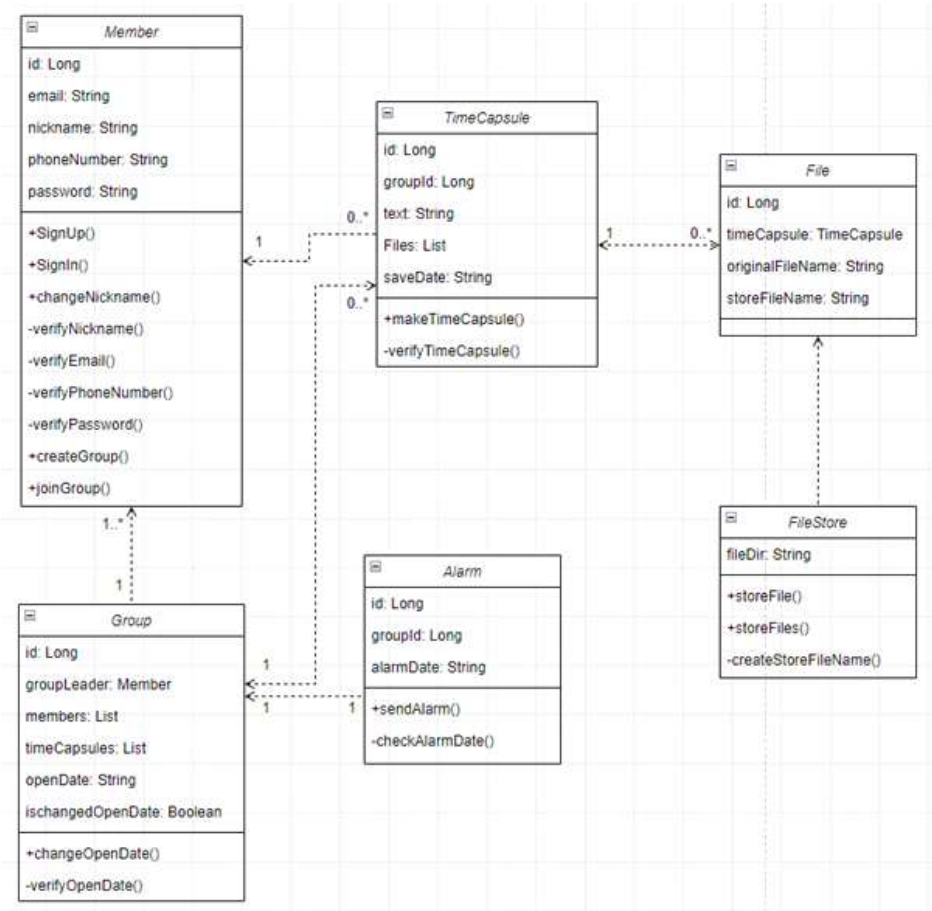
- 이번 3차를 진행하면서 클래스 다이어그램에서 두가지 변경사항이 있었습니다.

1. Member와 Group사이의 연관관계가 다대다 연관관계로 되어 있어서 GroupMember 라는 새로운 객체를 추가했습니다.
2. FileStore의 storeFile()을 private으로 변경해주었습니다.

<변경 후>



<변경 전>


Member
id: Long
email: String
nickname: String
phoneNumber: String
password: String
+SignUp()
+SignIn()
+changeNickname()
-verifyNickname()
-verifyEmail()
-verifyPhoneNumber()
-verifyPassword()
+createGroup()
+joinGroup()
TimeCapsule
id: Long
groupId: Long
text: String
Files: List
saveDate: String
+makeTimeCapsule()
-verifyTimeCapsule()
File
id: Long
timeCapsule: TimeCapsule
originalFileName: String
storeFileName: String
FileStore
fileDir: String
+storeFile()
+storeFiles()
-createStoreFileName()
Group
id: Long
groupLeader: Member
members: List
timeCapsules: List
openDate: String
ischangedOpenDate: Boolean
+changeOpenDate()
-verifyOpenDate()
Alarm
id: Long
groupId: Long
alarmDate: String
+sendAlarm()
-checkAlarmDate()
1
0..*
1..*

다음은 작성한 코드들입니다.

- Member

```
import com.timecapsule.timecapsule.domain.dto.MemberDto;
import lombok.Getter;

import javax.persistence.*;
import java.util.List;

@Entity
@Getter
public class Member {

    @Id @GeneratedValue
    @Column(name = "member_id")
    private Long id;
    private String email;
    private String password;
    private String nickname;
    private String phoneNumber;


    @OneToMany
    private List<GroupMember> groupMembers;

    protected Member(){}

    public Member(String email, String password, String phoneNumber, String nickname){
        this.email = email;
        this.password = password;
        this.phoneNumber = phoneNumber;
        this.nickname = nickname;
    }

    public void updateMemberInfo(MemberDto dto){
        this.email = dto.getEmail();
        this.password = dto.getPassword();
        this.phoneNumber = dto.getPhoneNumber();
        this.nickname = dto.getNickName();
    }
}
```

- MemberService: 회원 정보 보기, 회원 정보 수정, 회원 탈퇴 기능

```
import com.timecapsule.timecapsule.domain.Member;
import com.timecapsule.timecapsule.domain.dto.MemberDto;
import com.timecapsule.timecapsule.repository.MemberRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;

@Service
@Transactional(readOnly = true)
@RequiredArgsConstructor
public class MemberService {

    private final MemberRepository memberRepository;

    /**
     * 회원 가입
     */
    @Transactional
    public Long join(Member member){
        validateDuplicateNickname(member);
        validateDuplicateEmail(member);
        validateDuplicatePhoneNumber(member);
        memberRepository.save(member);
        return member.getId();
    }

    //중복 닉네임 검증
    private void validateDuplicateNickname(Member member){
        List<Member> findMembers =
memberRepository.findByNickname(member.getNickname());
        if(!findMembers.isEmpty()){
            throw new IllegalStateException("이미 존재하는 닉네임입니다.");
        }
    }

    //중복 이메일 검증
    private void validateDuplicateEmail(Member member){
        List<Member> findMembers = memberRepository.findByEmail(member.getEmail());
        if(!findMembers.isEmpty()){
            throw new IllegalStateException("이미 존재하는 이메일입니다.");
        }
    }
```

```
    //중복 휴대폰번호 검증
    private void validateDuplicatePhoneNumber(Member member){
        List<Member> findMembers =
memberRepository.findByPhoneNumber(member.getPhoneNumber());
        if(!findMembers.isEmpty()){
            throw new IllegalStateException("이미 존재하는 휴대폰번호입니다.");
        }
    }

    /**
     * 로그인 기능
     */
    //로그인
    public Long signIn(String email, String password){
        List<Member> findMembers = memberRepository.findByEmail(email);
        if(!findMembers.isEmpty()){
            Member targetMember = findMembers.get(0);
            if (targetMember.getPassword().equals(password)) {
                //로그인 성공
                return targetMember.getId();
            }
        }
        //로그인 실패
        throw new IllegalStateException("로그인 실패");
    }

    /**
     * 회원 조회
     */
    //회원 전체 조회
    public List<Member> findMembers(){
        return memberRepository.findAll();
    }

    //회원 개별 조회
    public Member findOne(Long memberId){
        return memberRepository.findOne(memberId);
    }

    /**
     * 회원 정보
     */
    //회원 정보 수정
    @Transactional
    public Member updateMemberInfo(Long id, MemberDto updateDTO){
```

```
        Member member = memberRepository.findOne(id);
        if(!member.equals(null)){
            member.updateMemberInfo(updateDTO);
            return member;
        }
        throw new IllegalStateException("해당 회원이 없음");
    }

    //회원 탈퇴
    public void removeMember(Long memberId){
        Member member = memberRepository.findOne(memberId);
        if(!member.equals(null)){
            memberRepository.removeMember(member);
            return;
        }
        throw new IllegalStateException("해당 회원이 없음");
    }

}
```

- MemberController

```
import com.timecapsule.timecapsule.domain.Member;
import com.timecapsule.timecapsule.service.MemberService;
import lombok.RequiredArgsConstructor;
import net.bytebuddy.implementation.bind.MethodDelegationBinder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

import javax.validation.Valid;
import java.util.List;

@Controller
@RequiredArgsConstructor
public class MemberController {

    private final MemberService memberService;

    /*
        get형식으로 /member/new에 member/createMemberForm으로 넘겨서 회원가입폼으로 가
게 함
     */
```

```
    @GetMapping("/member/new")
    public String createForm(Model model){
        model.addAttribute("memberForm", new MemberForm());
        return "member/createMemberForm";
    }

    /*
        Post형식으로 /member/new에 넘어오면 form에 입력한 값을 가지고 회원가입을 진행함
     */
    @PostMapping("/member/new")
    public String create(@Valid MemberForm form, BindingResult result){
        if(result.hasErrors()){
            return "members/createMemberForm";
        }

        Member member = new Member(form.getEmail(), form.getNickname(),
                form.getPhoneNumber(), form.getPassword());

        memberService.join(member);
        return "redirect:/";
    }
}
```

- SignController : 로그인과 로그아웃

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.validation.Valid;

@Controller
@RequiredArgsConstructor
public class SignController {

    private final MemberService memberService;

    //Get 매핑으로 signIn에 접근하면 로그인 폼을 화면에 보여줍니다.
    //로그인 정보를 받기 위한 createSignInForm 사이트가 필요합니다.
    @GetMapping("/signIn")
    public String createForm(Model model){
        model.addAttribute("form", new SignInForm());
        return "signIn/createSignInForm";
    }

    //Post 매핑으로 signIn으로 넘어오면 폼의 데이터들로 로그인을 진행합니다.
```

```
    @PostMapping("/signIn")
    public String signIn(@Valid SignInForm form, BindingResult result,
                         HttpServletRequest request){
        if(result.hasErrors()){
            return "signIn/signInForm";
        }
        HttpSession session = request.getSession();
        Long memberId = memberService.signIn(form.getEmail(), form.getPassword());
        Member findMember = memberService.findOne(memberId);
        session.setAttribute("email", findMember.getEmail());
        session.setAttribute("SIGNIN", "TRUE");
        return "redirect:/";
    }

    //Get 매핑으로 signOut으로 접근하면 로그아웃을 진행합니다.
    //세션에 저장된 속성 값을 지우고, 세션을 초기화합니다.
    @GetMapping("/signOut")
    public String signOut(HttpServletRequest request){
        HttpSession session = request.getSession();
        session.removeAttribute("email");
        session.removeAttribute("SIGNIN");
        session.invalidate();
        return "redirect:/";
    }
}
```