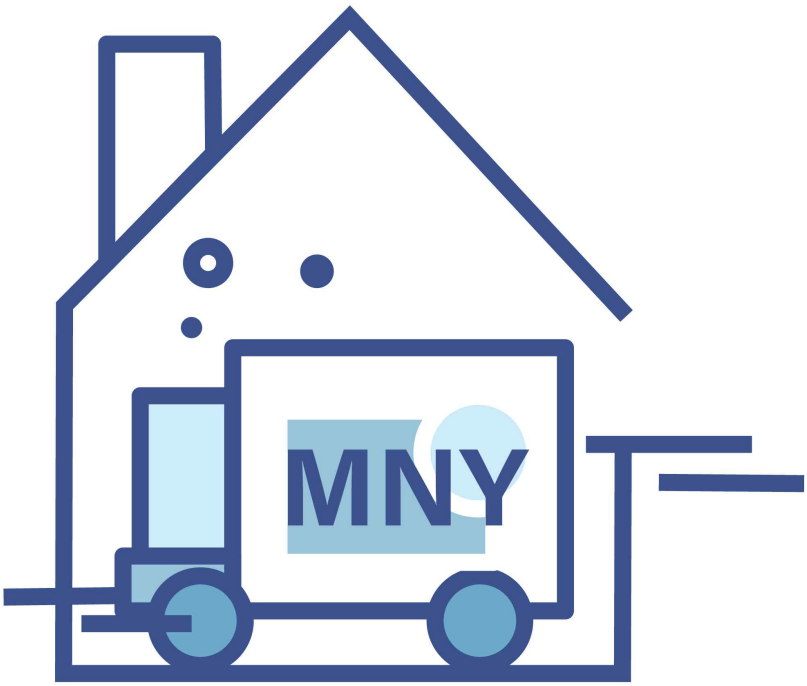
Yeungnam University

Market Nearby You

3. Design Document



[Revision history]

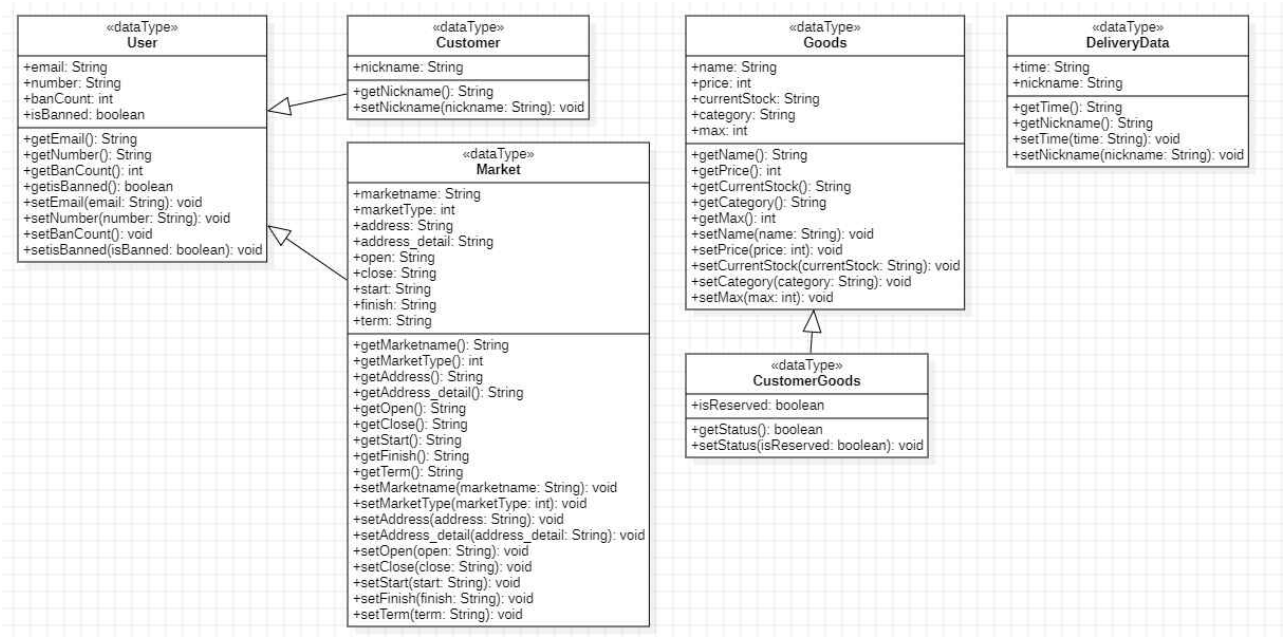
Revision date	Version #	Description	Author
	1.0	First Writing	
	1.1	seuence diagram, state machine diagram 수정	
	1.2	class diagram, state machine diagram, sequence diagram 수정	
	1.3	로고 추가	

= Contents =

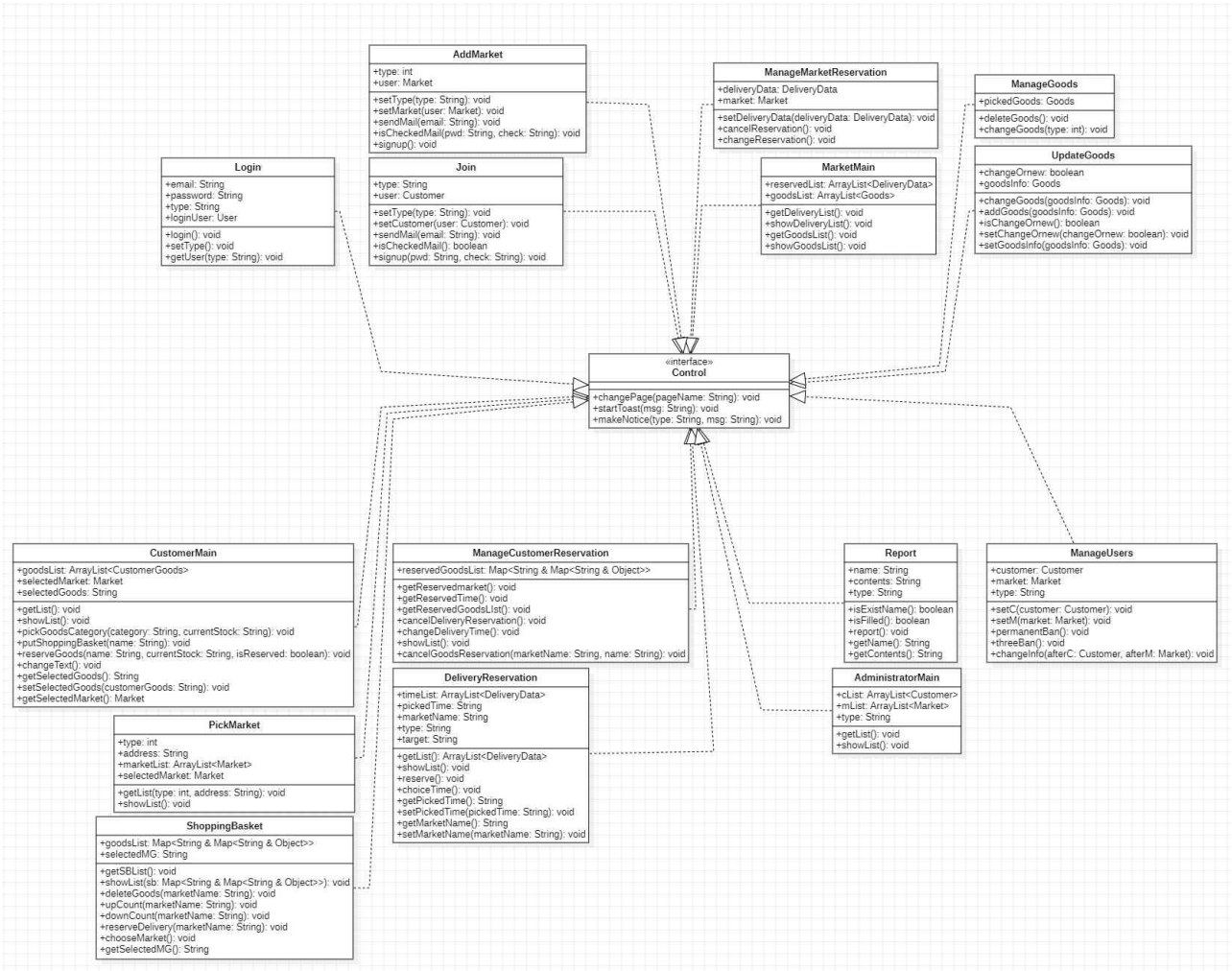
1. Introduction

일반적으로 사람들은 간단히 끼니를 해결하기 위해 장을 보러갈 때, 요깃거리를 찾고자할 때, 야근, 과제, 공부에 지쳐 시원한 음료수 한잔에 바람을 쐬고자할 때 등, 대형마트보다는 집 근처의 동네마트, 집 앞에 편의점 들을 더 찾게 된다. 그러나 문을 닫거나, 필요한 물품이 없는 등 생각보다 정보를 얻기가 매우 쉽지 않다. 반대로 대형마트들에게는 다양한 서비스가 제공되어 쉽게쉽게 정보를 얻을 수 있다. 그래서 MNY 는 대형마트들에게 맞춰져있던 초점을 잠시 돌려 사용자들에게 초점을 맞추어 자신이 필요할 때, 자신이 원할 때, 자신이 원하는 곳의 대형마트들이 아닌 슬리퍼 신고도갈 수 있는 동네마트, 집 앞 편의점, 천원마트들의 정보들을 제공받을 수 있는 서비스를 제공하고자 한다. 물론, 고객들 뿐만아니라 이는 각 가게의 관리들에게도 장점을 가져올 수 있다. 자신만의 장점들을 직접 홍보할 수 있고, '동네'라는 장점을 살려 비교적 쉬워지는 타겟팅의 고객들 등, 관리자들에게도 편의성을 제공하는 서비스가 되고자 한다. 이런 서비스들을 제공하기 위해 작성되는 이번 문서는 Design단계로 실제 MNY 시스템을 위해 구현되는 요소들을 구체화하고 세부적인 사항들, 후에 구현단계에서의 틀이 될 내용들이 작성된다. 그리고 고객들에게, 가게 관리자들에게 위의 이점들을 제공할 수 있는지 확인해볼 수 있는 것이 바로 이 단계에서의 포인트이다.

2. Class diagram

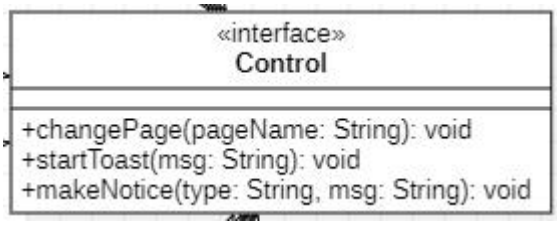


dataType(Model) 클래스



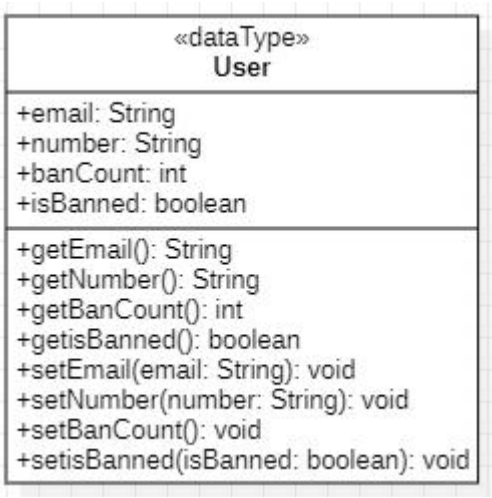
Controller 클래스

1) Control



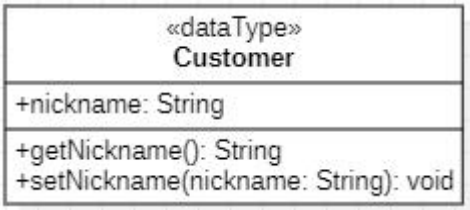
Attributes
Methods
changePage(pageName: String) : void : 화면 전환 startToast(msg: String): void : 간단한 메시지 출력 makeNotice(type: String, msg: String) : void : type에 맞는 알림창을 띄워주고 msg로 알림 메시지 지정
Description
Control 인터페이스는 MNY 시스템을 이용하면서 공통적으로 언제든 발생할 수 있는 상황들에 대해 인터페이스로 구성하여 모든 Controller 클래스들에서 구현될 것이다.

2) User



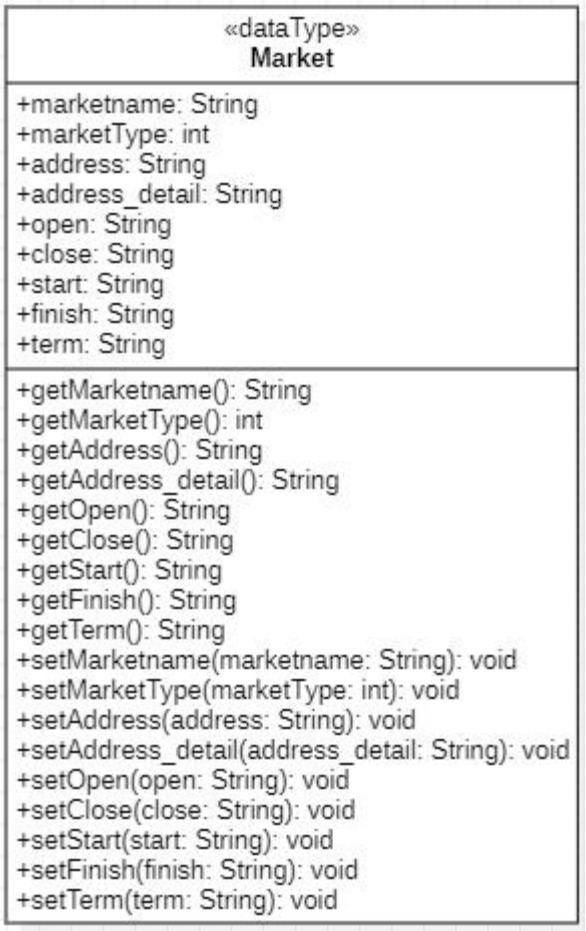
Attributes
email: String : 유저가 사용하게 될 이메일
number: String : 유저가 등록한 전화번호
banCount: int : 유저에게 누적된 정지 횟수
isBanned: boolean : 현재 정지상태인지 여부
Methods
getEmail(): String : 유저의 이메일 가져오기
getNumber(): String : 유저의 전화번호 가져오기
getBanCount(): int : 유저에게 누적된 정지 횟수 가져오기
getisBanned(): boolean : 현재 정지상태인지 확인결과 가져오기
setEmail(email: String): void : email로 유저 이메일 설정
setNumber(number: String): void : number로 유저 전화번호 설정
setBanCount(): void : 유저의 누적 정지횟수 1 증가하기
setisBanned(): void : 현재 정지상태인지 여부를 설정
Description
User 클래스는 Model의 역할을 한다. MNY 서비스를 이용하게 되는 유저들에게 공통적으로 정해지는 기본 정보들을 가지고 있는 Model 클래스이다.

3) Customer



Attributes
nickname: String : 고객의 닉네임
Methods
getNickname(): String : 닉네임 가져오기 setNickname(nickname: String): String : nickname으로 유저 닉네임 설정
Description
Customer 클래스는 역시 Model의 역할을 하는 클래스로, 2) 의 User 클래스를 상속 받는다. MNY의 유저들 중 한가지 타입인 고객의 정보를 저장하는 클래스이다.

4) Market

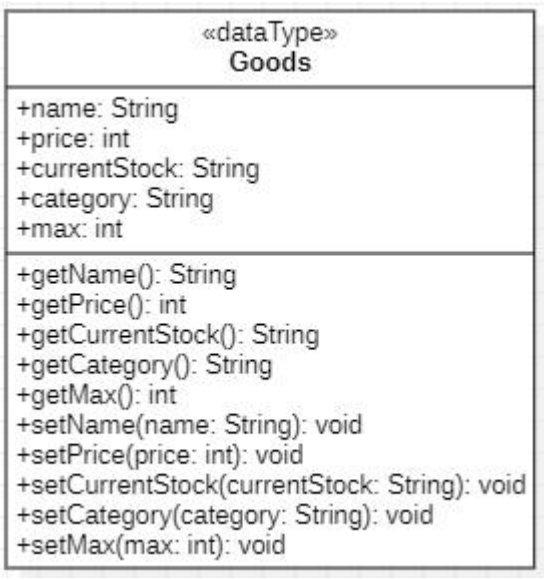


Attributes
marketname: String : 가게명 또는 지점명
marketType: int : 편의점, 동네마트와 같은 가게 유형
address: String : 가게 주소
address_detail: String : 가게 상세 주소
open: String : 영업 시작 시간
close: String : 영업 종료 시간
start: String : 배달 시작 시간
finish: String : 배달 종료 시간
term: STring : 배달 간격
Methods
getMarketname(): String : 가게명 또는 지점명 가져오기
getMarketType(): int : 가게 유형 가져오기
getAddress(): String : 가게 주소 가져오기
getAddress_detail(): String : 가게 상세 주소 가져오기
getOpen(): String : 영업 시작 시간 가져오기

getClose(): String : 영업 종료 시간 가져오기
getStart(): String : 배달 시작 시간 가져오기
getFinish(): String : 배달 종료 시간 가져오기
getTerm(): String : 배달 간격 가져오기
setMarketname(marketname: String): void : marketname으로 가게명 설정
setMarketType(marketType: int): void : marketType으로 가게 유형 설정
setAddress(address: String): void : address로 가게 주소 설정
setAddress_detail(address_detail: String): void : address_detail로 가게 상세 주소 설정
setOpen(open: String): void : open으로 영업 시작 시간 설정
setClose(close: String): void : close로 영업 종료 시간 설정
setStart(start: String): void : start로 배달 시작 시간 설정
setFinish(finish: String): void : finish로 배달 종료 시간 설정
setTerm(term: String): void : term으로 배달 간격 설정

Description
Market 클래스는 역시 Model의 역할을 하는 클래스로, 2) 의 User 클래스를 상속받는다. MNY의 유저들 중 한가지 타입인 가게의 정보를 저장하는 클래스이다.

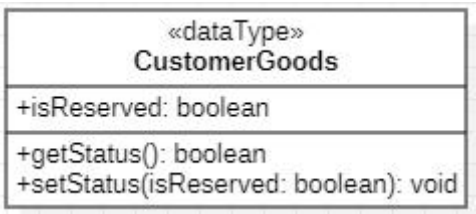
5) Goods



Attributes
name: String : 상품 이름
price: int : 상품 가격
currentStock: String : 상품 현재 재고 상황

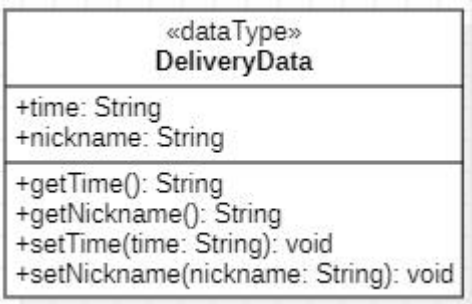
category: String : 상품 카테고리 max: int : 상품 최대 구매 수
Methods
getName(): String : 상품 이름 가져오기 getPrice(): int : 상품 가격 가져오기 getCurrentStock(): String : 상품 현재 재고 상황 가져오기 getCategory(): String : 상품 카테고리 가져오기 getMax(): int : 상품 최대 구매 수 가져오기 setName(name: String): void : name으로 상품 이름 설정 setPrice(price: int): void : price로 상품 가격 설정 setCurrentStock(currentStock: String) : currentStock으로 상품 현재 재고 상황 설정 setCategory(category: String): void : category로 상품 카테고리 설정 setMax(max: int): void : max로 상품 최대 구매 수 설정
Description
Goods 클래스는 Model의 역할을 하는 클래스이다. MNY에 등록된 상품들의 기본적인 정보들을 갖고있는 Model클래스이다.

6) CustomerGoods



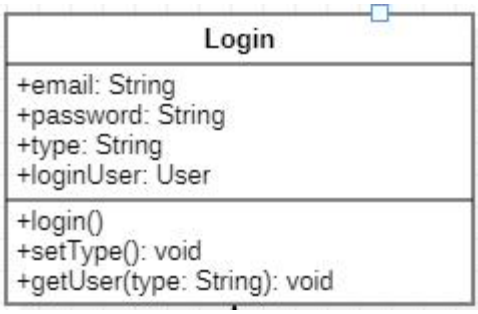
Attributes
isReserved: boolean : 고객이 상품을 예약하였는지 여부
Methods
getStatus(): boolean : 고객이 상품 예약 여부 가져오기 setStatus(isReserved: boolean): void : 해당 상품의 예약 여부 설정
Description
CustomerGoods 클래스는 역시 Model의 역할을 한다. 5)의 Goods 클래스를 상속받았으며, 고객들에게는 상품들마다 예약정보가 필요하여 따로 만들어진 클래스이다.

7) DeliveryData



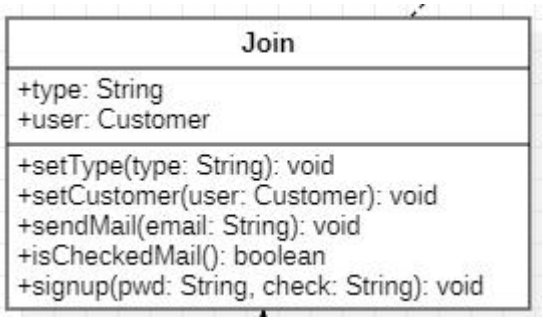
Attributes
time: String : 배달 시간 nickname: String : 예약자 닉네임
Methods
getTime(): String : 배달 시간 가져오기 getNickname(): String : 예약자 닉네임 가져오기 setTime(time: String): void : time으로 배달 시간 설정 setNickname(nickname: String): void : nickname으로 예약자 닉네임 설정
Description
DeliveryData 클래스는 Model의 역할을 한다. 고객들의 배달 예약 정보들을 다루는 클래스이며, 이 클래스는 가게 관리자들이 본인 가게의 배달 예약 정보들을 관리할 때에도 사용하는 클래스이다.

8) Login



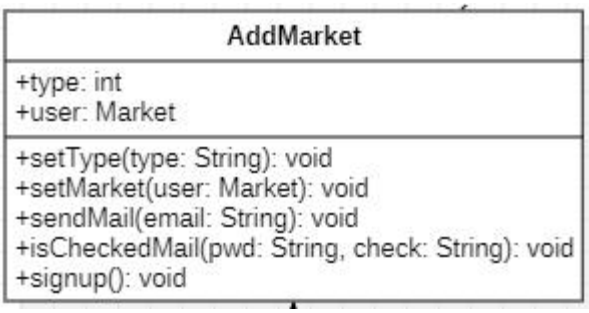
Attributes
email: String : 유저들이 입력한 이메일 password: String : 유저들이 입력한 비밀번호 type: String : 유저 타입 loginUser: User : 로그인한 유저 정보
Methods
login(): void : 유저들이 입력한 이메일과 비밀번호로 로그인 수행 setType(): void : 고객인지 가게관리자인지 시스템 관리자인지 유형 설정 getUser(type: String): void : 로그인된 유저정보 가져오기
Description
Login 클래스는 Controller의 역할을 한다. 로그인에 필요한 데이터들과 수행하게 될 기능들을 갖고 있는 클래스이다.

9) Join



Attributes
type: String : 유저의 가입 유형 user: Customer : 정보들이 저장될 Customer 클래스
Methods
setType(type: String) : void : 고객으로 유형 지정 setCustomer(user: Customer): void : 고객정보 입력 sendMail(email: String): void : 인증메일 전송 isCheckedMail(): boolean : 인증하였는지 확인 signup(pwd: String, check: String): void : 최종적으로 정보들을 종합하여 고객으로 MNY에 가입
Description
Join 클래스는 Controller의 역할을 한다. 유저들이 고객으로 MNY에 가입할 때 입력한 정보들과 유형이 저장되며, 이메일 인증, 회원가입 기능들을 수행하는 클래스이다.

10) AddMarket

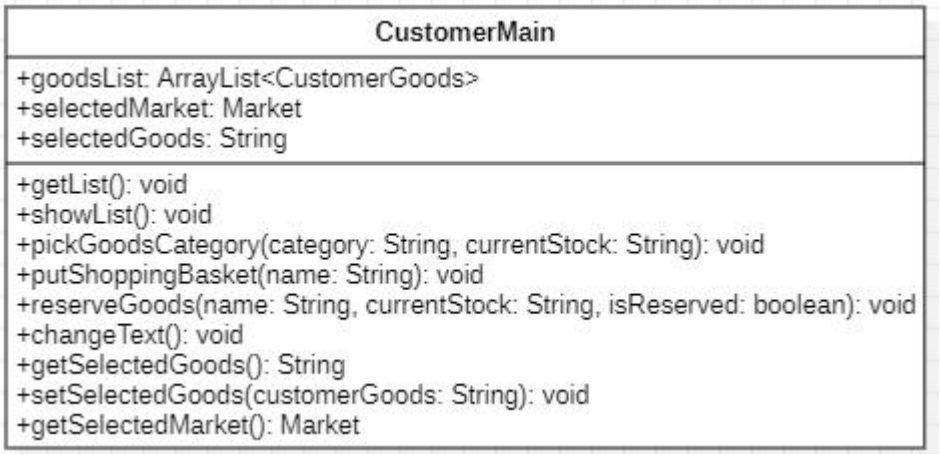


Attributes
type: String : 유저의 가입 유형 user: Market : 정보들이 저장될 Market 클래스
Methods
setType(type: String) : void : 유저의 유형 지정 setMarket(user: Market): void : 가게정보 입력 sendMail(email: String): void : 인증메일 전송 isCheckedMail(pwd: String, check: String): void : 인증됐는지 확인 후 1차 정보 입력절차 진행 signup(): void : 최종적으로 정보들을 종합하여 가게로 MNY에 가입
Description
AddMarket 클래스는 Controller의 역할을 한다. 유저들이 가게로 MNY에 가입할 때 입력한 정보들과 유형이 저장되며, 이메일 인증, 회원가입 기능들을 수행하는 클래스이다.

11) PickMarket

Attributes
type: int : 찾고자하는 가게 유형 address: String : 선택하고자하는 위치 marketList: ArrayList〈Market〉 : 조건에 맞는 가게 목록 selectedMarket: Market : 선태된 가게
Methods
getList(type: int, address: String): void : 조건에 맞는 가게 목록 가져오기 showList(): void : 가져온 가게 목록들을 보여준다
Description
PickMarket 클래스는 Controller의 역할을 한다. 유저들은 직접 원하는 가게의 유형과 위치를 지정한다. 그러면 조건에 맞는 가게 목록을 데이터베이스에서 가져와 보여주는 기능을 하는 클래스이다.

12) CustomerMain

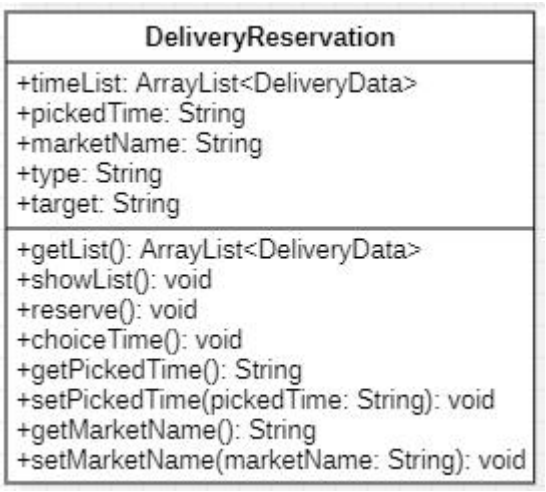


Attributes
goodsList: ArrayList〈CustomerGoods〉 : 고객전용 상품정보 목록 selectedMarket: Market : 선택된 가게 selectedGoods: String : 선택된 상품 이름
Methods
getList(): void : 선택된 가게의 상품들을 고객전용 상품정보 클래스에 담아서 목록을 가져온다 showList(): void : 가져온 상품 목록들을 보여준다 pickGoodsCategory(category: String, currentStock: String) : void : 고객이 직접 카테고리를 정하면 그에 맞는 상품들 목록 가져오기 putShoppingBasket(name: String): void : 선택된 상품 장바구니에 저장하기 reserveGoods(name: String, currentStock: String, isReserved: boolean): void : 예약이 가능한 상품인지 확인 후 예약목록에 추가 changeText(): void : 카테고리에 맞는 상품이 없음을 텍스트로 알려주기 getSelectedGoods(): String : 선택된 상품 이름 가져오기 setSelectedGoods(customerGoods: String): void : 선택된 상품 이름 설정 getSelectedMarket(): Market : 선택된 가게 가져오기
Description
CustomerMain 클래스는 Controller의 역할을 한다. 고객들이 장바구니 저장, 상품 카테고리 선택, 상품예약과 같이 주요 기능들을 할 수 있는 클래스이다.

13) ShoppingBasket

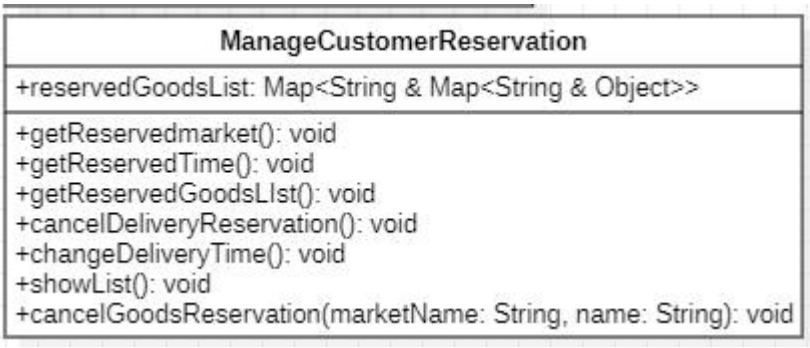
Attributes
goodsList: Map〈String, Map〈String, Object〉〉 : 장바구니의 상품 목록 selectedMG: String : 선택된 가게와 상품 이름
Methods
getSBList(): void : 장바구니의 상품 목록 가져오기 getSelectedMG(): String : 선택된 가게와 상품 이름 가져오기 showList(sb: Map〈String, Map〈String, Object〉〉): void : 가져온 상품 목록 보여주기 deleteGoods(marketName: String): void : 장바구니의 선택된 상품 삭제 upCount(marketName: String): void : 장바구니 저장 개수 증가 downCount(marketName: String): void : 장바구니 저장 개수 감소 reserveDelivery(marketName: String): void : 저장된 장바구니의 상품들 배달 예약하기 chooseMarket(): void : 배달받을 가게 정하기
Description
ShoppingBasket 클래스는 Controller의 역할을 한다. 고객들이 장바구니에 저장한 상품들을 관리할 수 있는 기능들이 포함된 클래스이다. 특히, 배달을 예약할 수 있는 중요한 기능도 포함되어있다.

14) DeliveryReservation



Attributes
timeList: ArrayList〈DeliveryData〉 : 배달 시간 간격마다의 예약 현황 정보 목록 pickedTime: String : 선택된 배달 예약 시간 marketName: String : 배달받을 가게 이름 type: String : 현재 배달예약 진행 중인 유저 타입 target: String : 해당 일정의 고객 닉네임
Methods
getList(): void : 배달 시간 간격마다의 예약 현황 정보 목록 가져오기 showList(): void : 가져온 예약 현황 정보 보여주기 getPickedTime(): String : 선택된 배달 예약 시간 가져오기 setPickedTime(pickedTime: String): void : 배달 예약 시간 설정 getMarketName(): String : 선택된 가게 이름 가져오기 setMarketName(marketName: String): void : 가게 이름 설정 reserve(): void : pickedTime 시간에 배달을 예약 choiceTime(): void : 선택된 시간정보 수정
Description
DeliveryReservation 클래스는 Controller의 역할을 한다. 고객들이 실제 가게에게 배달을 예약할 때의 기능들이 포함된 클래스이다. 가게에서 정한 시간을 간격마다 나누어 예약 현황 정보들을 갖고 있다.

15) ManageCustomerReservation

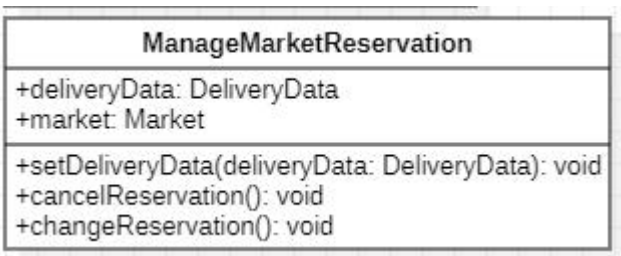


Attributes
reservedGoodsList: Map〈String & Map〈String & Object〉〉 : 예약 상품 목록
Methods
getReservedmarket(): void : 예약된 가게 이름 가져오기 getReservedTime(): void : 예약된 시간 가져오기 getReservedGoodsList() : void : 예약 상품 목록 가져오기 cancelDeliveryReservation(): void : 예약된 배달 일정 취소 changeDeliveryTime(): void : 배달예약 일정 변경 showList(): void : 가져온 예약상품 목록 보여주기 cancelGoodsReservation(marketName: String, name: String): void : 예약했던 상품 예약취소
Description
ManageCustomerReservation 클래스는 Controller의 역할을 한다. 고객들 예약한 배달 일정, 상품들에 대한 정보를 관리할 수 있는 기능들을 포함하는 클래스이다.

16) MarketMain

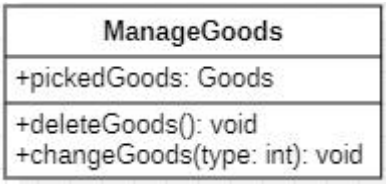
Attributes
reservedList: ArrayList〈DeliveryData〉 : 가게의 등록된 배달 일정 목록 goodsList: ArrayList〈Goods〉 : 가게의 등록된 상품 목록
Methods
getReservedList(): void : 가게의 등록된 배달 일정 목록 가져오기 showDeliveryList(): void : 가져온 배달 일정 목록 보여주기 getGoodsList(): void : 가게의 등록된 상품 목록 가져오기 showGoodsLlst(): void : 가져온 상품 목록 보여주기
Description
MarketMain 클래스는 Controller의 역할을 한다. 가게가 등록한 배달 일정들과 상품들에 대한 정보들을 보여주는 기능을 하는 클래스이다.

17) ManageMarketReservation



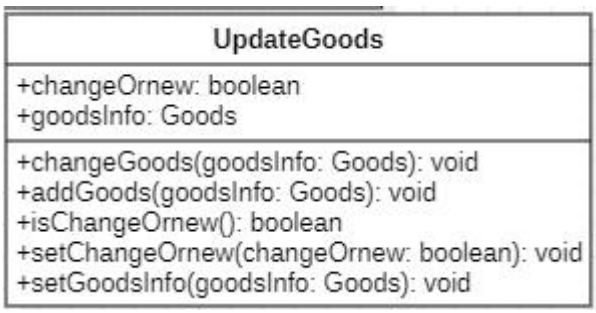
Attributes
deliveryData: DeliveryData : 관리하고자 하는 배달 일정의 데이터 market: Market : 현재 진행 중인 가게 정보
Methods
setDeliveryData(deliveryData: DeliveryData): void : 배달 일정 설정 cancelReservation(): void : 해당 배달 일정 취소하기 changeReservation(): void : 해당 배달 일정 정보 변경하기
Description
ManageMarketReservation 클래스는 Controller의 역할을 한다. 가게가 등록된 배달 일정들 중에서 관리하고자 하는 특정 배달 일정에 대해 컨트롤 할 수 있는 기능들이 포함되어있는 클래스이다.

18) ManageGoods



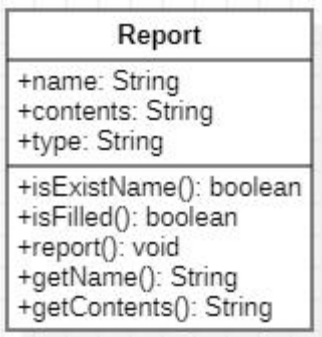
Attributes
pickedGoods: Goods : 관리하고자 선택한 상품 데이터
Methods
deleteGoods(): void : 특정 상품 삭제하기 changeGoods(type: int): void : 새로운 제품인지 기존 제품의 변경인지 type으로 확인하여 새로운 또는 변경된 상품 정보 적용하기
Description
ManageGoods 클래스는 Controller의 역할을 한다. 가게가 등록된 상품들 중에서 관리하고자 하는 상품에 대해 컨트롤을 지정할 수 있는 기능들이 수행되는 클래스이다.

19) UpdateGoods



Attributes
changeOrnew: boolean : 변경된 정보인지 새로운 제품의 정보인지 여부 goodsInfo: Goods : 상품의 정보를 저장
Methods
changeGoods(goodsInfo: Goods): void : 저장된 정보로 상품의 정보 변경하기 addGoods(goodsInfo: Goods): void : 저장된 정보로 새로운 상품 등록하기 isChangeOrnew(): boolean : 기존상품 변경인지 새로운 상품인지 여부 가져오기 setChangeOrnew(changeOrnew: boolean): void : 기존상품 변경인지 새로운 상품인지 결정 setGoodsInfo(goodsInfo: Goods): void : 변경된 또는 새로운 상품 정보 설정
Description
UpdateGoods 클래스는 Controller의 역할을 한다. 18) 클래스에서 정보가 변경되고자 선택된 경우 기능을 시작하는 클래스로 기존의 상품인지 새로운 상품인지에 따라 다르게 처리되는 기능을 가지고 있는 클래스이다.

20) Report

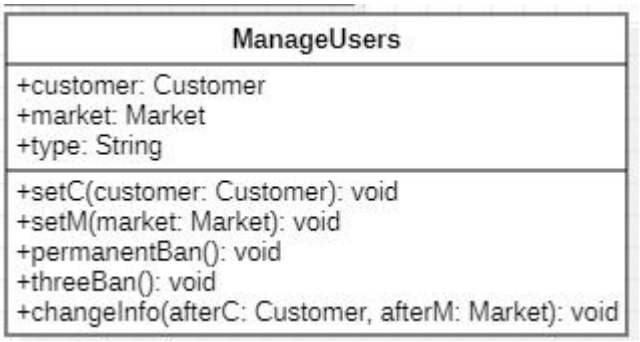


Attributes
name: String : 신고하고자 하는 고객의 닉네임 또는 가게명(지점명) contents: String : 신고하고자 하는 내용 type: String : 현재 유저가 고객인지 가게인지 판단
Methods
isExistName(): boolean : 존재하는 고객 또는 가게인지 확인하기 isFilled(): boolean : 입력되어야 할 정보들이 모두 입력되었는지 확인하기 report(): void : 신고대상과 신고내용들을 관리자에게 보내주기 getName(): String : 작성된 이름 가져오기 getContents(): String : 작성된 내용 가져오기
Description
Report 클래스는 Controller의 역할을 한다. 고객 또는 가게관리자들이 MNY 서비스를 이용하면서 특정 고객 또는 가게를 신고하고자 하는 경우에 필요한 기능들을 수행하는 클래스이다.

21) AdministratorMain

Attributes
cList: ArrayList〈Customer〉 : 시스템 관리자가 보고자 하는 고객 목록 mList: ArrayList〈Market〉 : 시스템 관리자가 보고자 하는 가게 목록 type: String : 고객 또는 가게의 타입
Methods
getList(): void : 시스템 관리자가 보고자 하는 고객 또는 가게 목록 가져오기 showList(): void : 가져온 고객 또는 가게 목록을 보여주기
Description
AdministratorMain 클래스는 Controller의 역할을 한다. 시스템 관리자가 메인화면에서 원하는 타입의 유저리스트를 보고 특정 유저에게 접근할 수 있는 기능들을 수행하는 클래스이다.

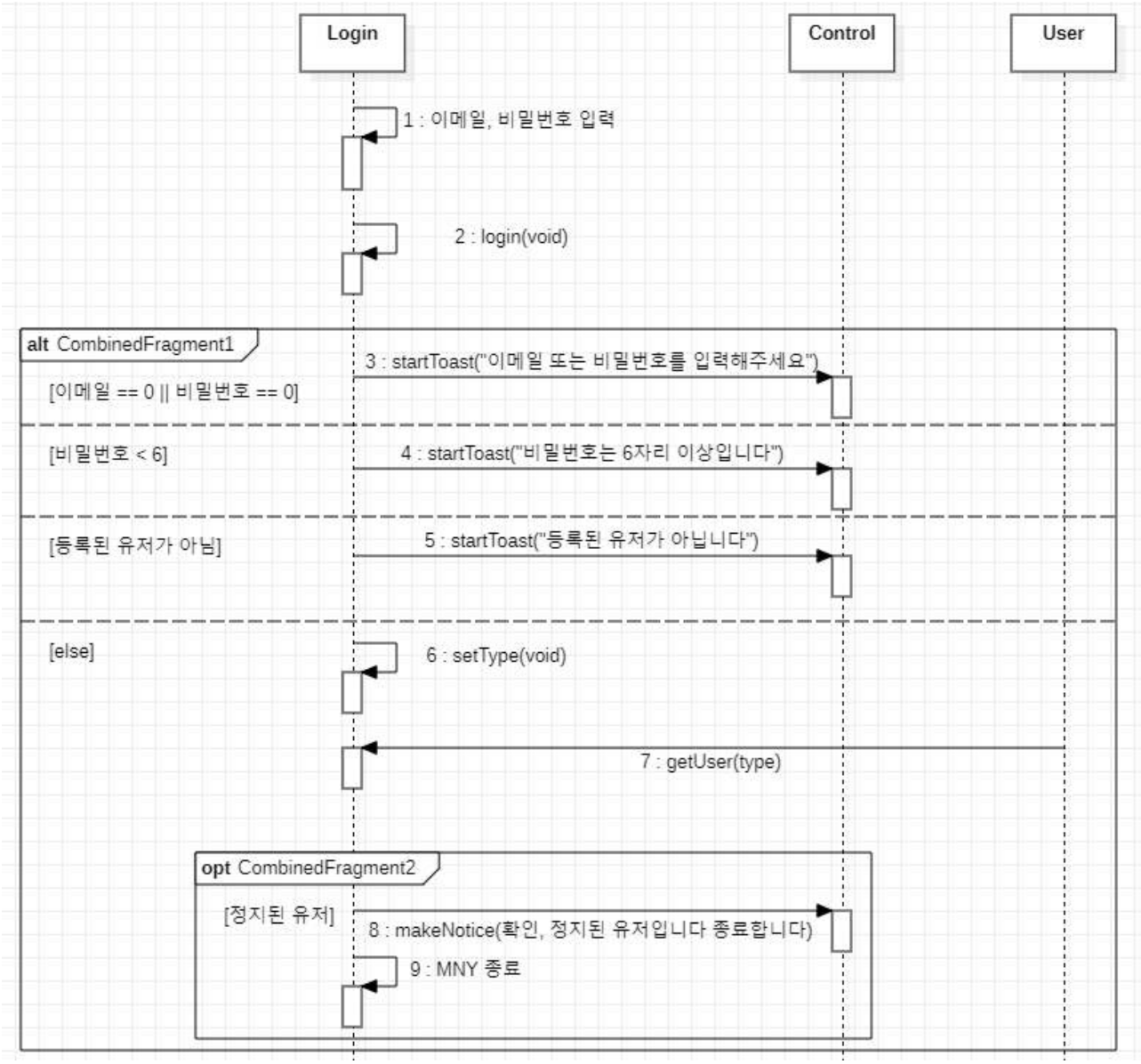
22) ManageUsers



Attributes
customer: Customer : 선택된 고객 market: Market : 선택된 가게 type: String : 고객인지 가게인지
Methods
setC(customer: Customer): void : 선택된 고객 설정 setM(market: Market): void : 선택된 가게 설정 permanentBan(): void : 선택한 고객 또는 가게에게 영구 정지 적용하기 threeBan() : void : 선택한 고객 또는 가게에게 3일 정지 적용하기 changeInfo(afterC: Customer, afterM: Market): void : 선택한 고객 또는 가게의 정보 변경하기
Description
ManageUsers 클래스는 Controller의 역할을 한다. 시스템 관리자가 접수받은 신고내용이나 특정 이유로 고객 또는 가게에게 제재를 가하거나 특정 정보에 접근해야할 이유가 생겼을 때의 기능들이 포함된 클래스이다.

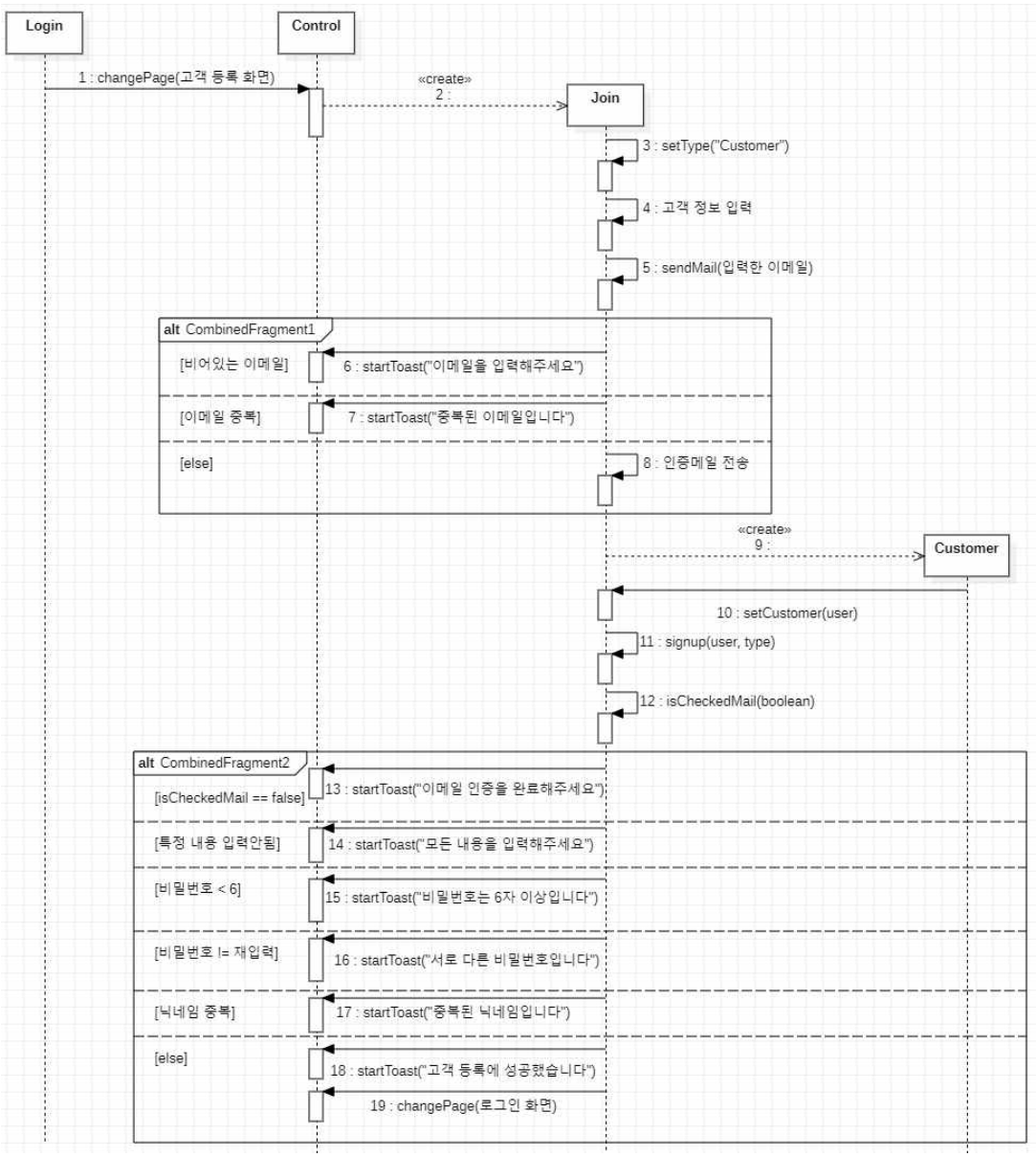
3. Sequence diagram

1) Login



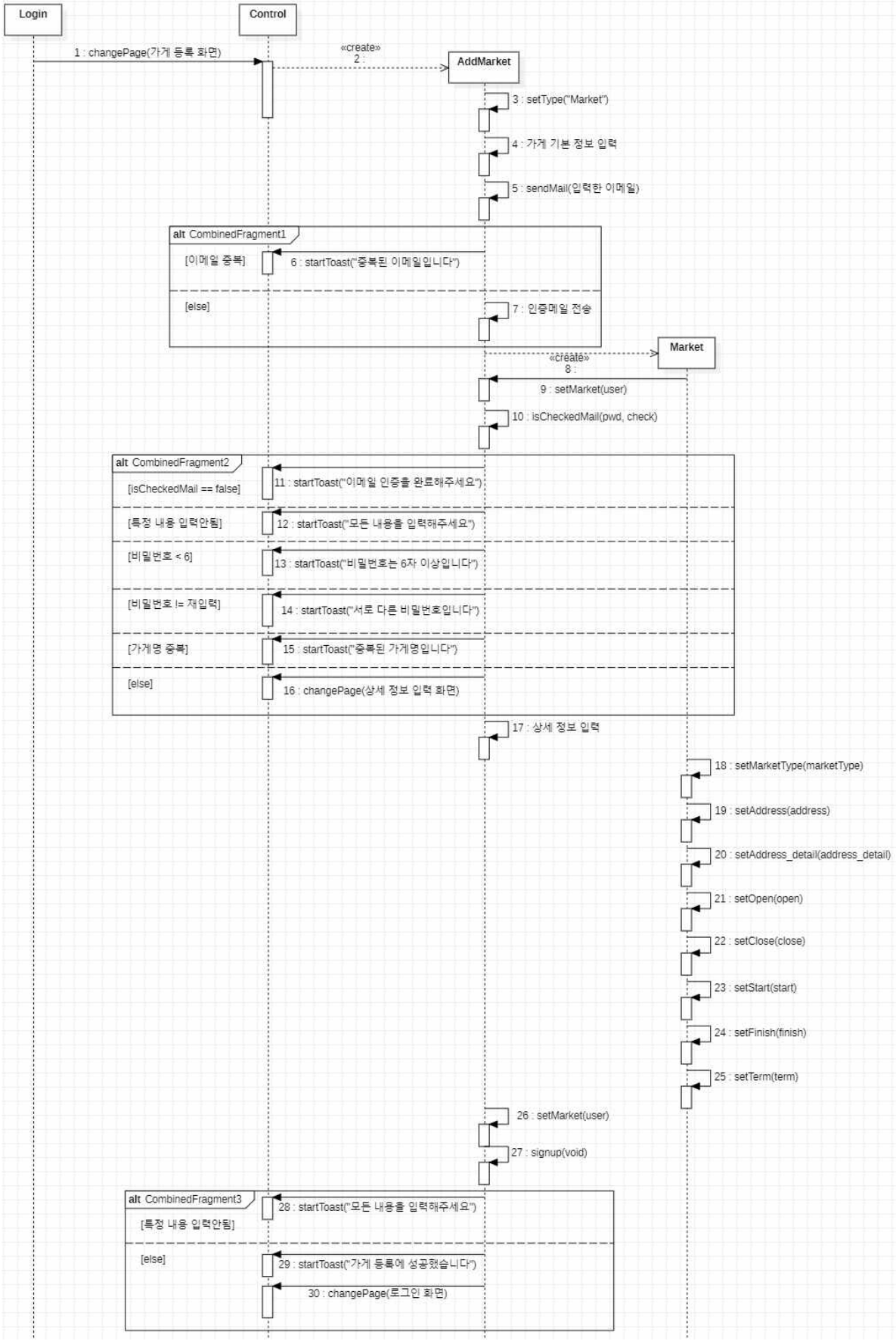
Login Use case 에서의 Sequence Diagram이다. Login 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 유저들이 입력한 이메일과 비밀번호로 login 메소드를 실행시키면 입력된 내용들의 적합성을 평가한 후 제대로 입력이 되었다면 실제 MNY에 존재하는 유저인지 확인을 한다. 최종 확인이 끝나고나면 setType 메소드를 통해 현재 로그인한 유저가 Customer인지 Market인지 저장하고 유저의 정보를 데이터베이스에서 가져와 getUser 메소드를 통해 loginUser에 저장하게 된다. 이때 만약 해당 유저가 정지상태라면 팝업창을 띄워주고 바로 MNY를 종료시킨다.

2) Join



Join Use case 에서의 Sequence Diagram이다. Join 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. MNY의 고객으로써 등록하기를 원할 때 시작되는 Use case로 로그인 화면에서 고객 등록 화면으로 넘어오게 된다. 고객 등록이기 때문에 setType 메소드로 Customer임을 저장해준다. 유저가 정보들을 입력하고나면 인증메일을 sendMail 메소드를 통해 보내게 된다. 이때, 중복된 이메일이라면 전송되지 않는다. 인증메일을 전송하고나면 setCustomer 메소드를 통해 입력된 메소드를 Customer 객체를 생성하여 저장해준다. 그리고 signup 메소드를 통해 고객등록 절차가 시작되는데 isCheckedMail 메소드로 인증이 되었는지 확인을 우선 한다. 확인이 되지않으면 우선 인증하도록 권하고 확인이 되었다면 입력한 정보들의 적합성을 검사한다. 모든 정보들이 통과하면 데이터베이스에 고객을 등록한다. 로그인 화면으로 돌아가며 끝이 난다.

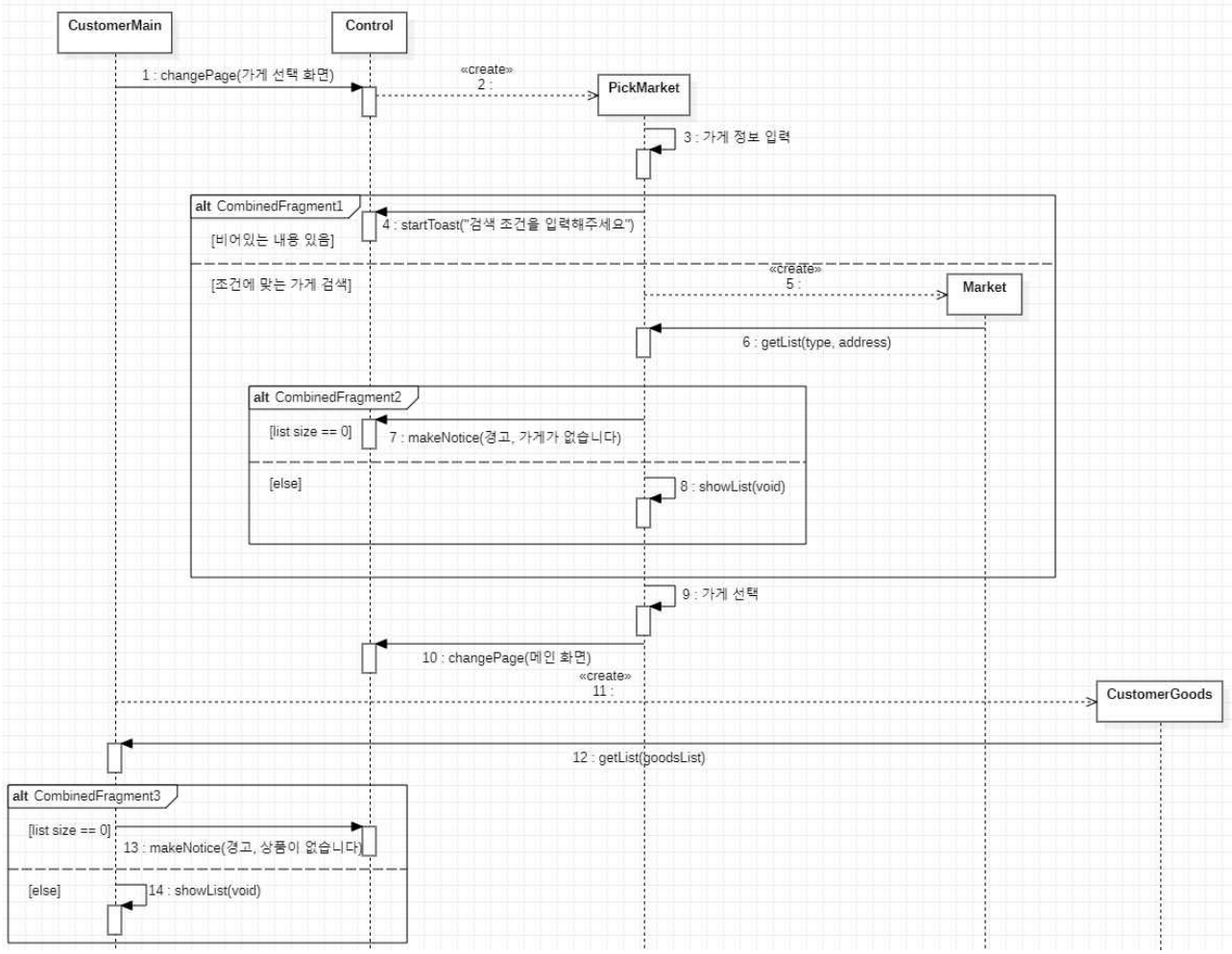
3) Add Market


Login
Control
AddMarket
Market
1 : changePage(가게 등록 화면)
«create»
2 :
3 : setType("Market")
4 : 가게 기본 정보 입력
5 : sendMail(입력한 이메일)
alt CombinedFragment1
[이메일 중복]
6 : startToast("중복된 이메일입니다")
[else]
7 : 인증메일 전송
«create»
8 :
9 : setMarket(user)
10 : isCheckedMail(pwd, check)
alt CombinedFragment2
[isCheckedMail == false]
11 : startToast("이메일 인증을 완료해주세요")
[특정 내용 입력안됨]
12 : startToast("모든 내용을 입력해주세요")
[비밀번호 < 6]
13 : startToast("비밀번호는 6자 이상입니다")
[비밀번호 != 재입력]
14 : startToast("서로 다른 비밀번호입니다")
[가게명 중복]
15 : startToast("중복된 가게명입니다")
[else]
16 : changePage(상세 정보 입력 화면)
17 : 상세 정보 입력
18 : setMarketType(marketType)
19 : setAddress(address)
20 : setAddress_detail(address_detail)
21 : setOpen(open)
22 : setClose(close)
23 : setStart(start)
24 : setFinish(finish)
25 : setTerm(term)
26 : setMarket(user)
27 : signup(void)
alt CombinedFragment3
[특정 내용 입력안됨]
28 : startToast("모든 내용을 입력해주세요")
[else]
29 : startToast("가게 등록에 성공했습니다")
30 : changePage(로그인 화면)

AddMarket Use Case 에서의 Sequence Diagram이다. AddMarket 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. MNY의 가게로써 등록하기를 원할 때 시작되는 Use case로 로그인 화면에서 가게 등록 화면으로 넘어오게 된다. 가게 등록이기 때문에 setType 메소드로 Market임을 저장해준다. 유저가 가게의 기본 정보들을 입력하고나면 인증메일을 sendMail 메소드를 통해 보내게 된다. 이때, 중복된 이메일이라면 전송되지 않는다. 인증메일을 전송하고나면 Market 객체를 생성하여 setMarket 메소드를 통해 입력된 정보들을 모두 저장한다. 그리고 isChekcedMail 메소드를 통해 인증이 되었는지 확인을 하는고 입려된 정보들의 적합성까지 확인을 한다. 확인이 통과되면 1차적으로 정보들이 저장되며 상세 정보 입력 화면으로 전환된다.

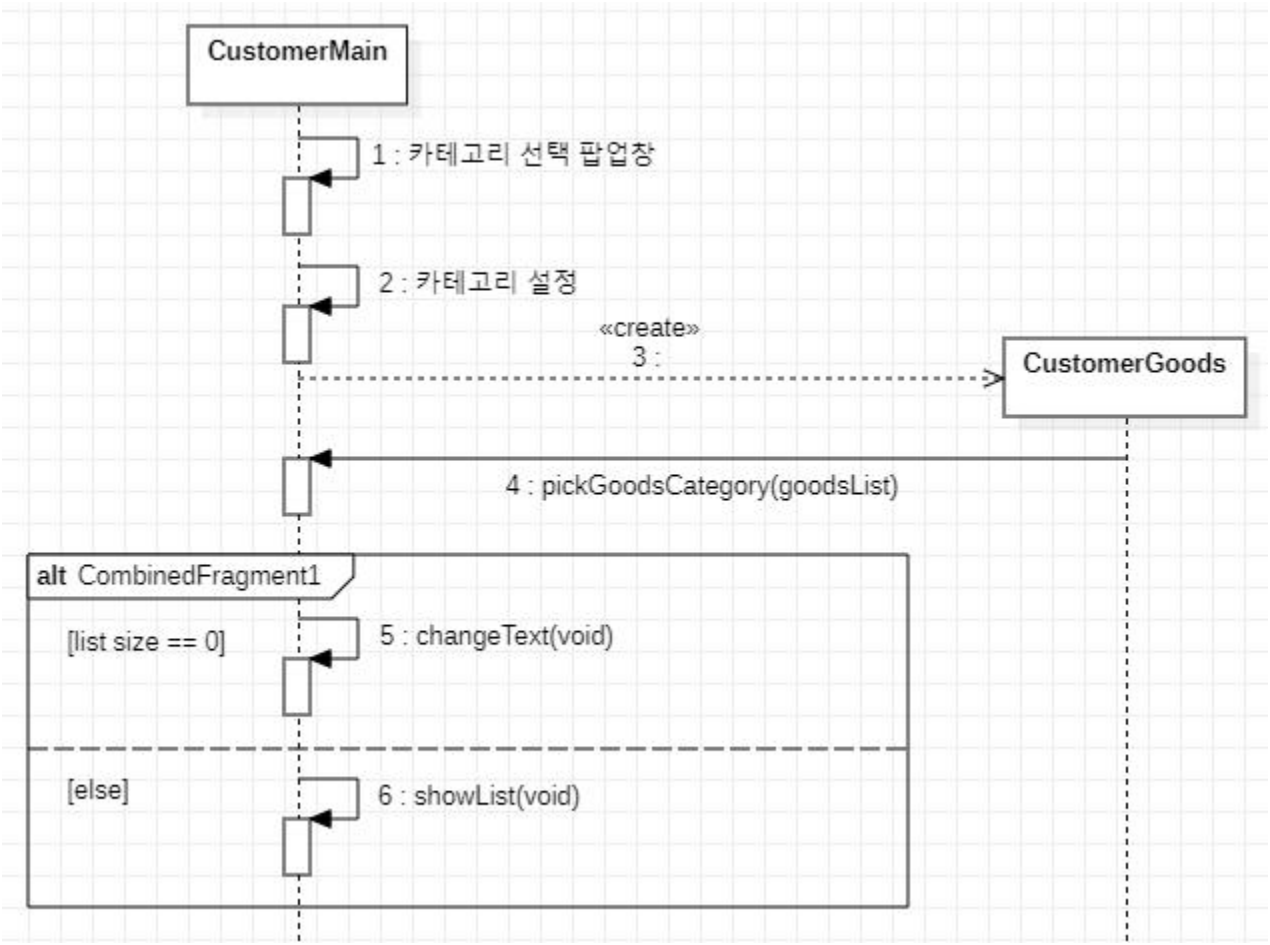
여기서 유저들은 가게의 상세한 정보들을 입력하게 된다. 입력이 모두 끝나면 Market 객체의 각종 set 메소드들을 통해 위에서 입력한 정보들 외의 것들도 모두 저장해준다. 그리고 setMarket 메소드를 통해 그렇게 최종저장된 market으로 업데이트해준다. 마지막으로, signup 메소드가 실행되는데 우선 입력한 정보들의 적합성 확인이 필요하다. 모두 통과되고나면 최종적으로 MNY의 회원으로 등록되며 로그인 화면으로 넘어가게 되고 절차는 끝이 난다.

4) Pick Market



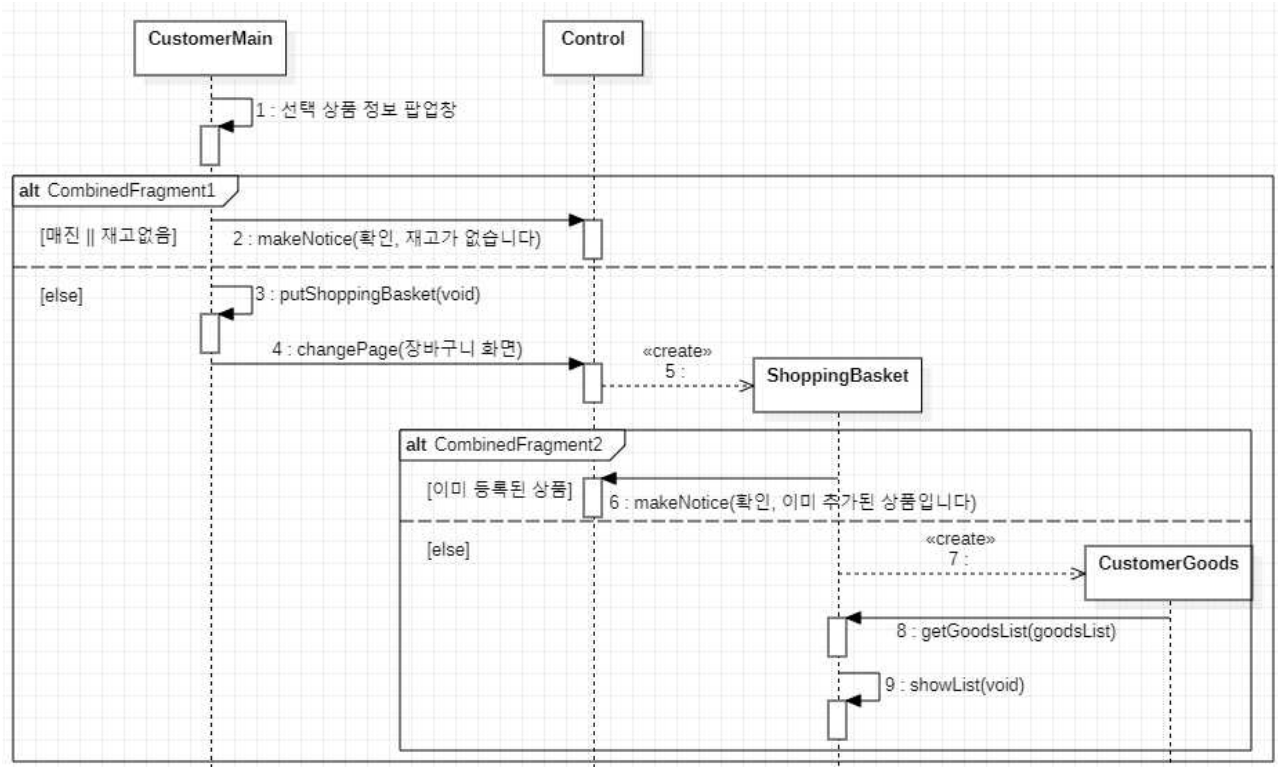
Pick Market Use case 에서의 Sequence Diagram이다. PickMarket, CustomerMain 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 고객들은 살펴보고 싶은 가게를 직접 선택해야한다. changePage 메소드로 가게 선택 화면으로 이동하면 PickMarket이 본격적으로 시작된다. 우선, 찾고싶은 가게의 정보를 입력해준다. 그 후, 입력된 정보의 적합성 확인이 우선적으로 이루어진다. 정보들이 통과되고나면 Market 객체를 생성하여 getList 메소드를 통해 데이터베이스에서 조건에 맞는 가게들을 각 Market객체에 저장하여 리스트에 저장한 후 가져온다. 이때, 리스트의 길이를 통해 조건에 맞는 가게가 있는지 없는지 확인을 한다. 조건의 가게가 존재하면 showList 메소드를 통해 화면에 표시하게 된다. 유저가 가게를 선택하고나면 changePage메소드로 메인화면으로 이동하게 된다. 메인화면에서는 선택된 가게에 등록된 상품들을 CustomerGoods 객체를 생성하고 getList 메소드를 통해 데이터베이스에서 가져온다. 이때도 리스트의 길이를 통해 상품이 있는지 없는지 확인을 하게 된다. 상품이 존재하지 않는다면 다시 가게를 선택하도록 가게선택 화면으로 이동하게 되며, 존재한다면 showList 메소드를 통해 화면에 상품들이 보여지게 된다.

5) Pick Goods



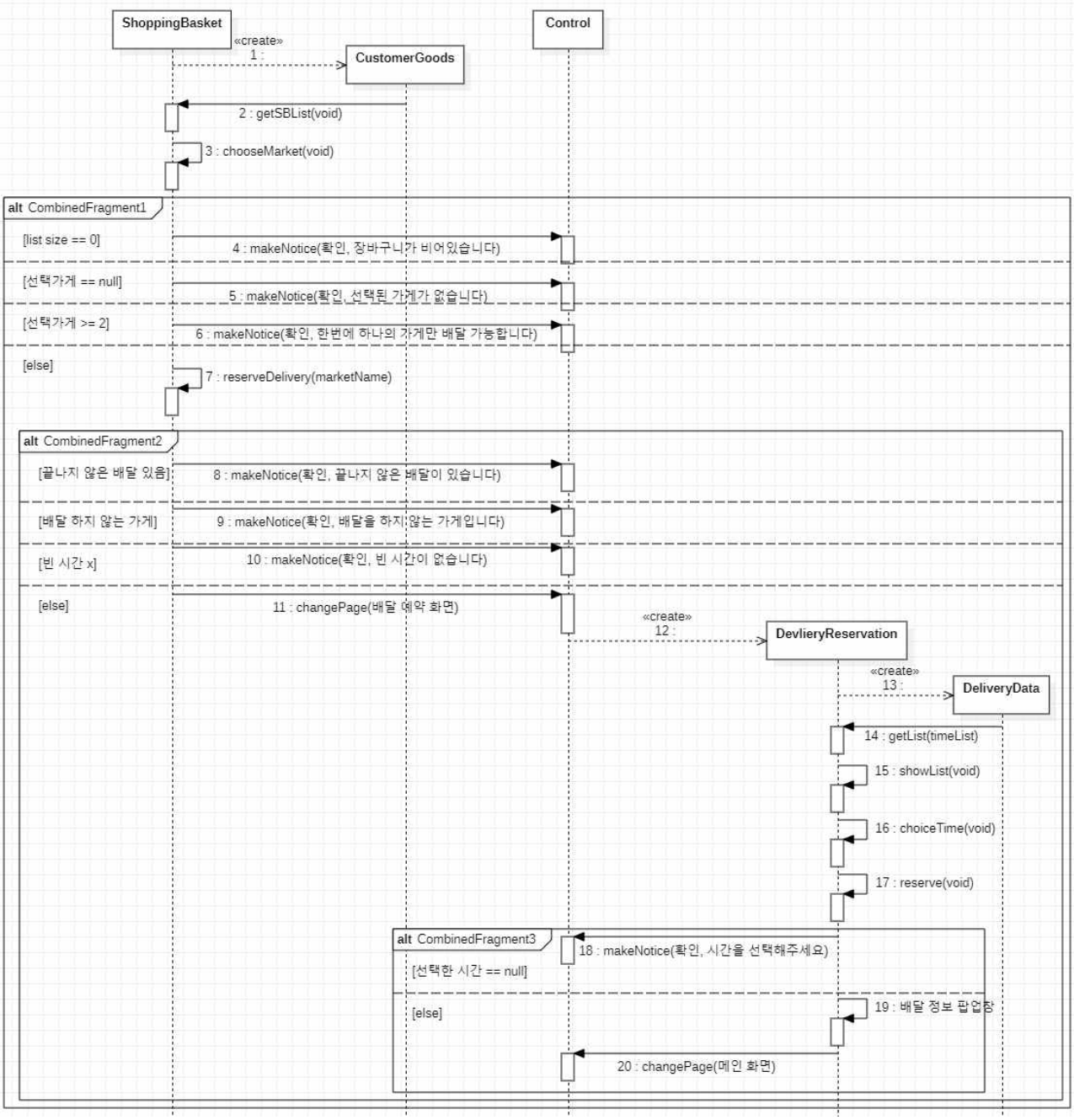
Pick Goods Use case 에서의 Sequence Diagram이다. CustomerMain 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 고객들은 직접 원하는 카테고리의 상품들을 검색할 수 있다. 메인화면에서 카테고리 선택 버튼을 누르면 카테고리를 선택할 수 있는 팝업창이 뜨게 된다. 원하는 카테고리를 설정하면 CustomerGoods 객체를 생성하게 되고, pickGoodsCategory 메소드를 통해 해당 가게의 데이터베이스에서 카테고리에 맞는 상품들을 가져오게 된다. 이때, 리스트의 사이즈를 통해 상품이 존재하지 않는다면 화면 중앙에 텍스트로 없음을 보여주고, 있다면 해당 상품들을 화면에 보여주게 된다.

6) Add To Shopping Basket



Add To Shopping Basket Use case 에서의 Sequence Diagram이다. CustomerMain, ShoppingBasket 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 고객들은 상품들을 둘러보다 마음에 드는 상품들을 장바구니에 담을 수 있다. 나열된 상품들 중, 원하는 상품에서 장바구니 담기를 누르면 상품의 정보들을 간단하게 표시한 팝업창이 띄워진다. 이때, 재고의 상황을 체크하여 알려주게 된다. 만약 재고가 있다면 putShoppingBasket 메소드를 통해 장바구니에 저장하게 된다. 저장된 결과를 확인하기 위해 changePage 메소드를 통해 장바구니 화면으로 이동하게 된다. 이때 ShoppingBasket 객체를 생성하는데 그전에 이미 추가된 상품인지 확인 과정이 먼저 수행된다. 이 과정이 통과하고나면 CustomerGoods 객체도 바로 생성하여 getGoodsList 메소드를 통해 장바구니에 저장된 상품들을 데이터베이스에서 가져온다. showList 메소드로 해당 상품들을 화면에 보여주며 끝이 난다.

7) Reserve Delivery

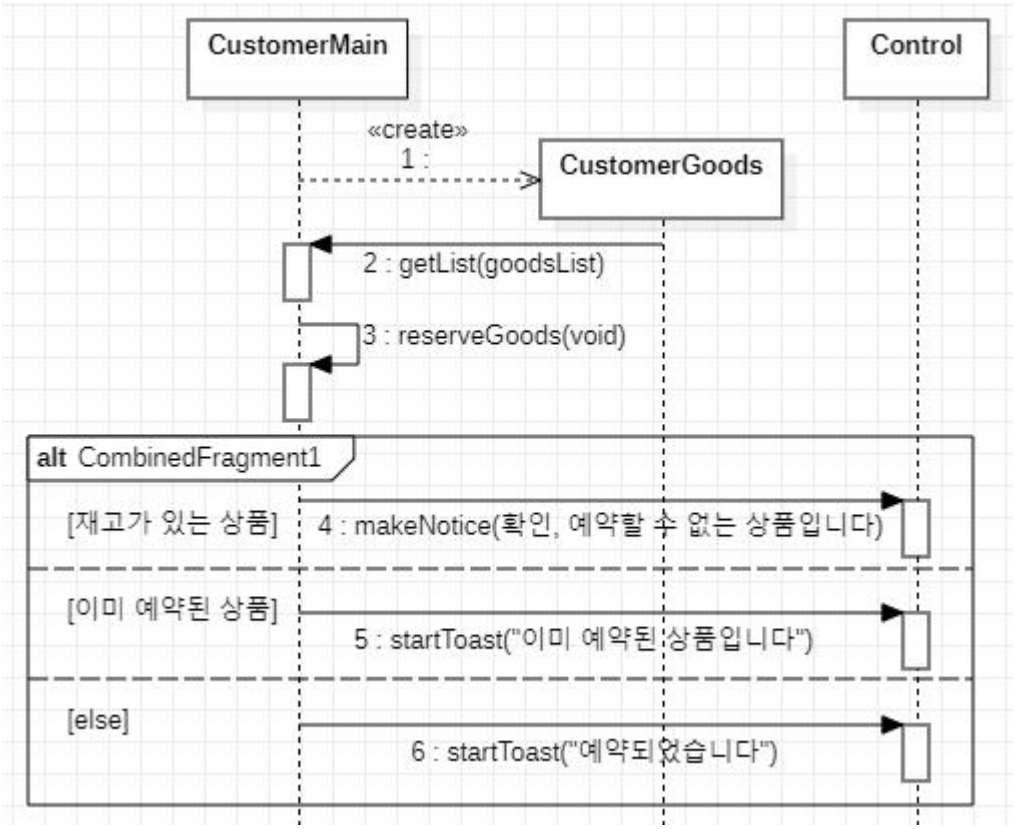


Reserve Delivery Use case 에서의 Sequence Diagram이다. ShoppingBasket, DeliveryReservation 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 고객들은 장바구니 담아놓은 상품들을 후에 배달을 예약하여 받아볼 수 있게 된다. 우선, 장바구니 화면에 들어가면 CustomerGoods 객체를 생성하여 데이터베이스에서 getList 메소드로 상품목록을 받아온다. 해당 장바구니에는 여러 가게의 여러 상품들이 존재할 것이다. 그러나, 배달은 한 번에 하나의 가게만 에약할 수 있도록 하였다. 그래서 chooseMarket 메소드를 통해 가게를 선택해야한다. 그 후, reserveDelivery 메소드를 통해 예약 절차가 시작되는데, 그전에 필수적으로 거쳐야하는 확인과정을 거치게 된다. 모두

통과가 되고나면 메소드가 실행되는데, 이때, 해당 가게가 배달을 하지 않는다면 예약이 아예 불가능하기에 우선적으로 확인을 해준다. 그후, 예약 가능 시간이 있는지 확인을 하고 예약이 간으하다면 changePage 메소들르 통해 배달 예약 화면으로 이동하게 된다.

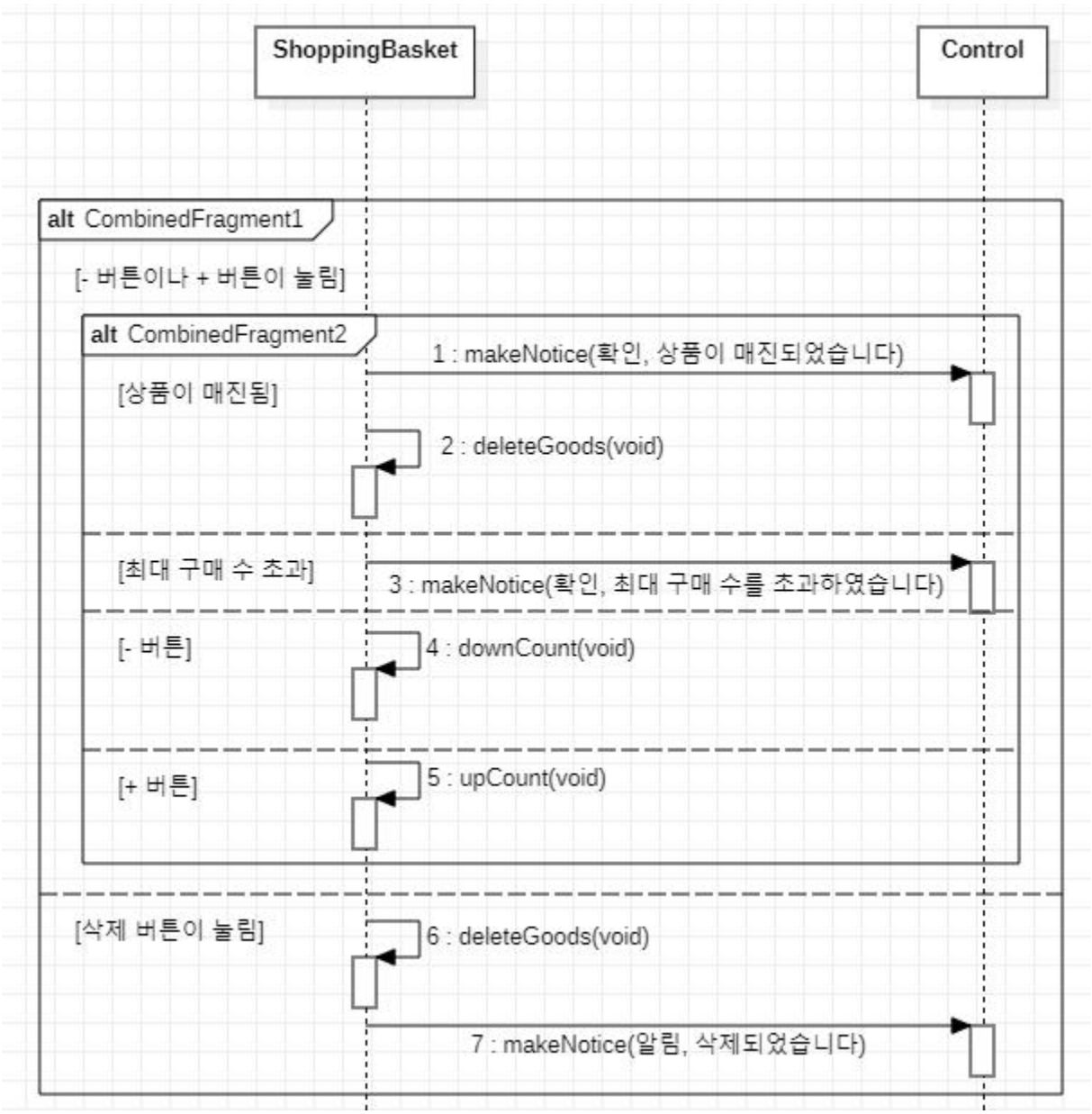
이때, DeliveryReservation 객체가 생성되며 DeliveryData 객체를 생성하여 데이터베이스에서 가게가 등록한 배달 일정들의 현황을 getList 메소드로 가져오게 된다. showList 메소드로 배달 일정 현황 목록을 화면에 보여주고 고객들은 choiceTime 메소드로 예약이 가능한 시간대를 선택하게 된다. 선택이 끝나고나면 reserve 메소드로 마지막단계로 들어가는데 이때 선택한 시간대가 있는지 우선 확인하고 있다면 정보를 팝업창으로 띄워주고 예약 정보가 데이터베이스에 저장된다. changePage 메소드로 메인화면으로 돌아가며 끝이 난다.

8) Reserve Goods



Reserve Goods Use case 에서의 Sequence Diagram이다. CustomerMain 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 고객들은 현재 입고가 안되었거나, 매진된 상품들에 대해 예약을 할 수 있다. 우선 메인화면이 시작되면 CustomerGoods 객체를 생성하여 getList 메소드로 데이터베이스에서 가게의 상품들을 가져오게 된다. 그 후, reserveGoods 메소드를 통해 상품 예약이 시작된다. 재고가 있는 상품은 예약이 불가능하기에 확인 절차가 필요하다. 또한, 이미 예약된 상품의 경우에는 확인을 시켜주기 위해 메시지로 알려준다. 최종적으로 예약이 끝나면 메시지로 예약되었음을 알려주며 끝이 난다.

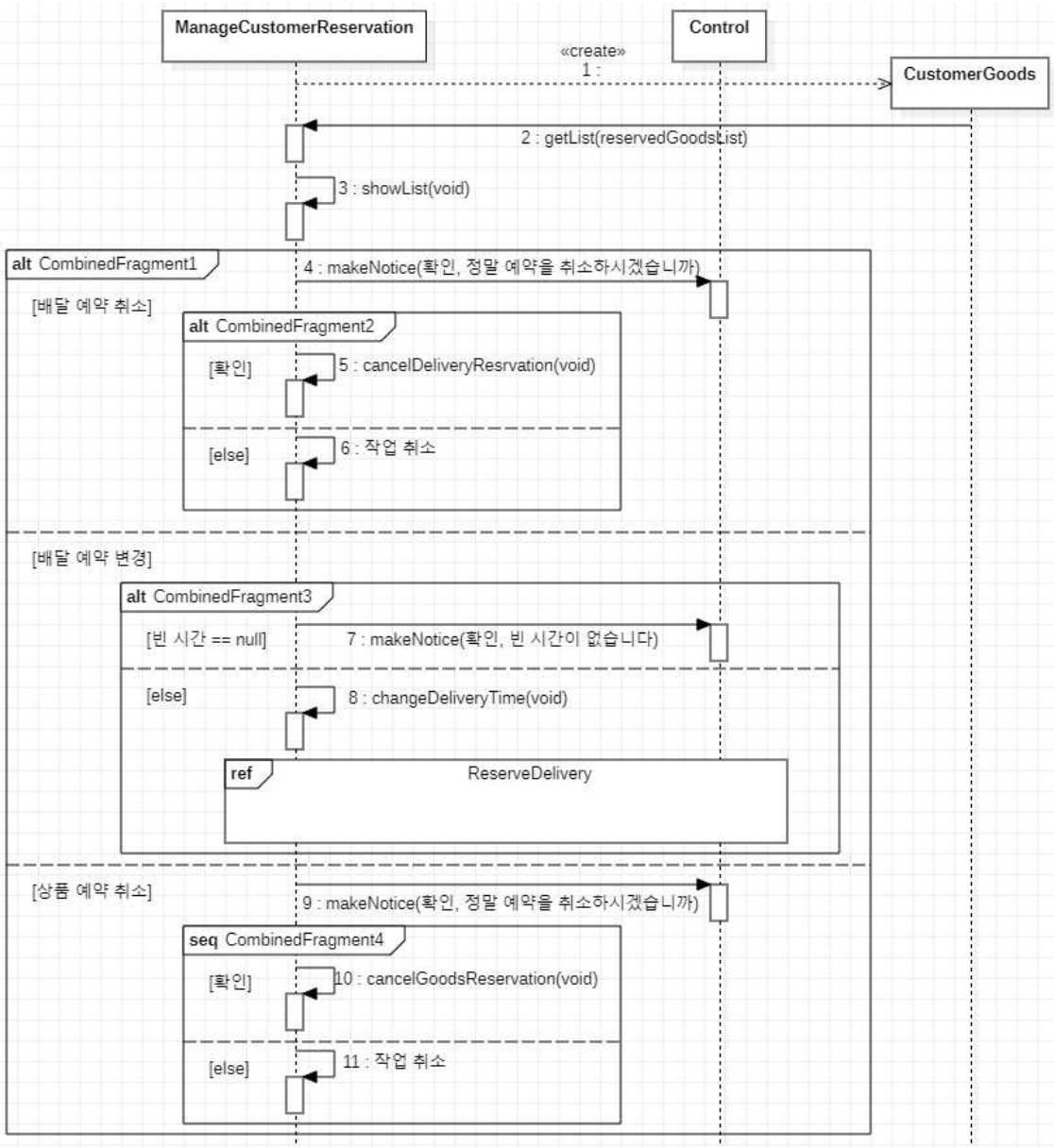
9) Manage Shopping Basket



Manage Shopping Basket Use case 에서의 Sequence Diagram이다. ShoppingBasket 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 고객들은 장바구니에 저장해둔 상품들에 대해 관리할 필요가 있다. 이 관리에는 개수 조정, 삭제 두가지의 경우가 존재한다. 우선 개수 조정의 경우이다. 만약 - 버튼이나 + 버튼이면 그에 맞는 downCount 메소드나 upCount 메소드가 호출되어 개수가 -1 +1 된다. 물론 그전에 해당 상품이 현재 매진되었다면 장바구니에 있을 필요가 없기 때문에 deleteGoods 메소드를 통해 삭제를 하게 되고, 정해진 최대 구매 수를 초과하게 되면 메시지로 알려주어야 한다. 두번째로는 삭제의 경우이다. 삭제의 경우는 단순하게 deleteGoods 메소드를 통해 삭제하게 된다. 그후, 변경된 정보를 저장하기 위해 fixInfo 메소드가 실행되면 변경점들이 데이터베이

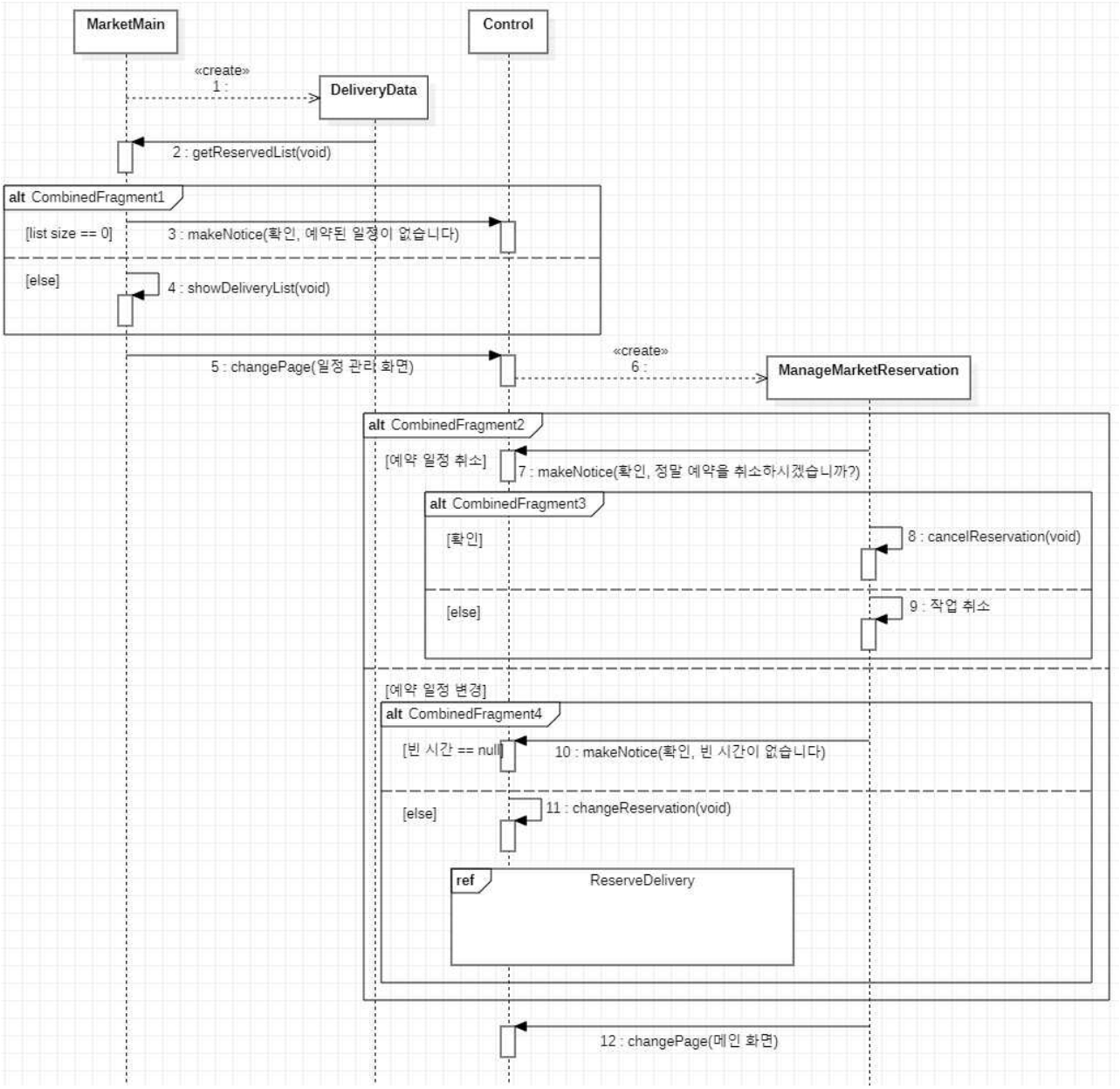
스에도 최종적으로 변경된다. 이때, 변경점이 있어야만 fixInfo 메소드가 정상적으로 실행이 가능하다. 정상적으로 실행이 되었다면 CustomerGoods 객체를 생성하여 변경된 상품들의 정보들을 getList 메소드로 가져와서 showList를 통해 장바구니 화면에서 볼 수 있게 되며 끝이 난다.

10) Manage Reservation



Manage Reservation Use case 에서의 Sequence Diagram이다. ManageCustomerReservation 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 고객들은 본인의 예약정보들을 관리할 필요가 있다. 고개들이 할 수 있는 예약은 배달, 상품 2가지이다. 첫 번째 배달예약 관리이다. 배달예약에는 취소와 변경이라는 두가지 선택지가 존재한다. 우선 취소하는 경우라면, 정말 취소할지 재확인 후, cancelReservation 메소드를 통해 해당 예약을 취소하게 된다. 변경하는 경우라면 우선적으로 빈 시간이 있는지 확인하여 있다면, 7) 의 Reserve Delivery 에서와 같은 방식으로 예약 일정을 변경하게 된다. 두 번째로 상품예약 관리이다. 상품예약에는 취소의 경우만 존재한다. 취소하길 원하면 cancelGoodsReservation 메소드를 통해 상품이 예약 목록에서 제거되며 예약관리는 끝이 난다.

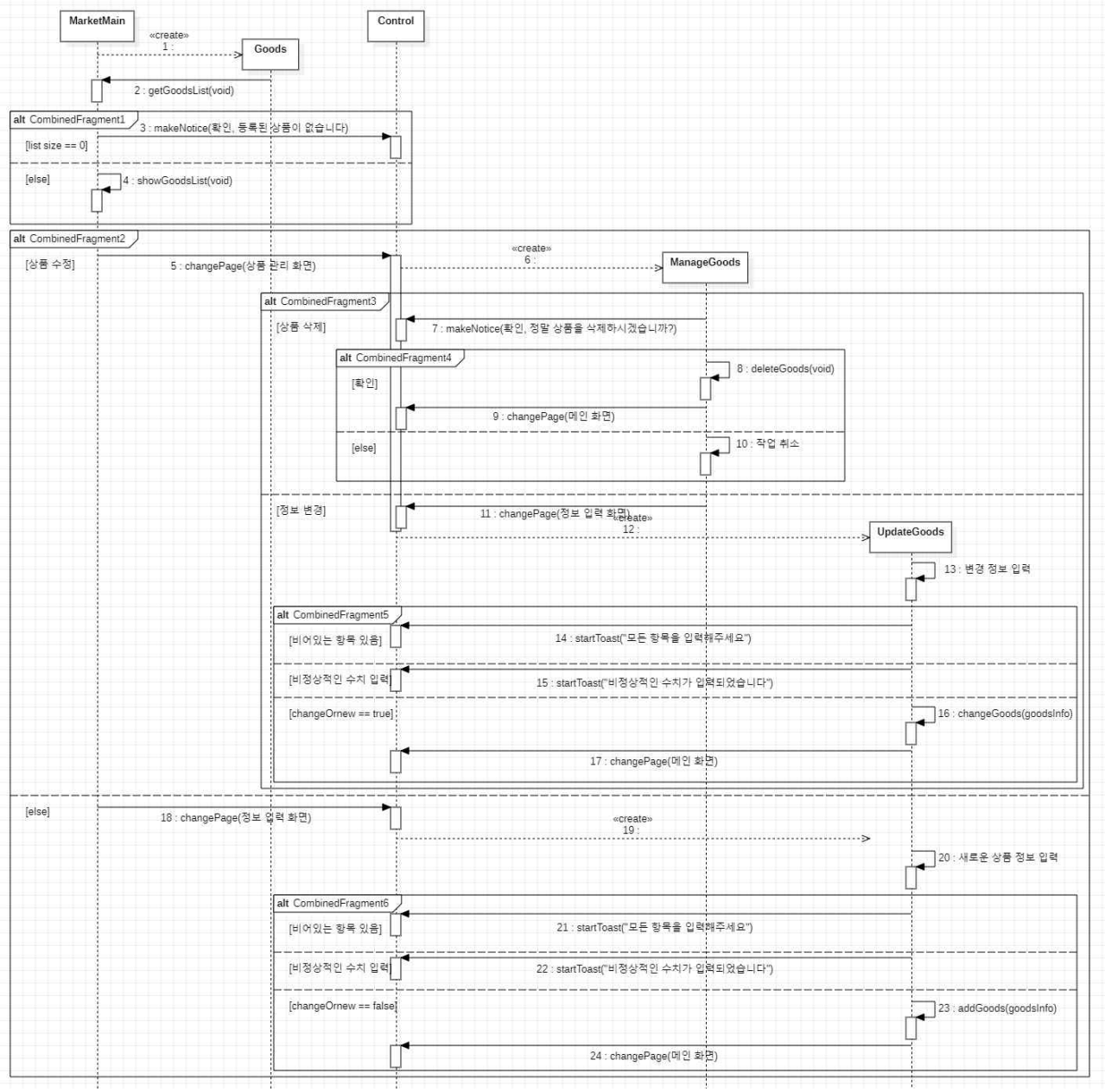
11) Manage Delivery Reservation



Manage Delivery Reservation Use case 에서의 Sequence Diagram이다. MarketMain, ManageMarketReservation 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 가게 관리자들은 자신의 가게의 배달 일정들을 관리할 수 있다. 그래서 일정들을 보기위해 DeliveryData 객체를 생성하여 getDeliveryList 메소드를 통해 데이터베이스에서 일정 정보들을 가져오게 된다. 이때, 리스트의 사이즈를 통해 정보가 존재한다면 showDeliveryList 메소드를 통해 화면에 보여주게 된다. 그후, 관리하고 싶은 특정 일정을 선택하면 일정 관리 화면으로 이동하게 되며, ManageMarketReservation 클래스가 시작된다. 배달 일정 관리에는 2가지가 존재한다. 첫 번째는 배달취소이다. 이 경우, 재확인이 필요하며 cancelReservation 메소드를 통해 해당 배달 일정은 취소된다. 두 번째는

배달변경이다. 이 경우, 빈시간이 있는지 확인이 필요하며 존재한다면 changeReservation 메소들르 실행하여 7) ReserveDelivery 와 같은 방식으로 배달 일정을 변경하게 된다. 최종 관리가 끝나면 changePage 메소드로 메인 화면으로 다시 이동하여 변경된 점들을 확인할 수 있다.

12) Manage Goods



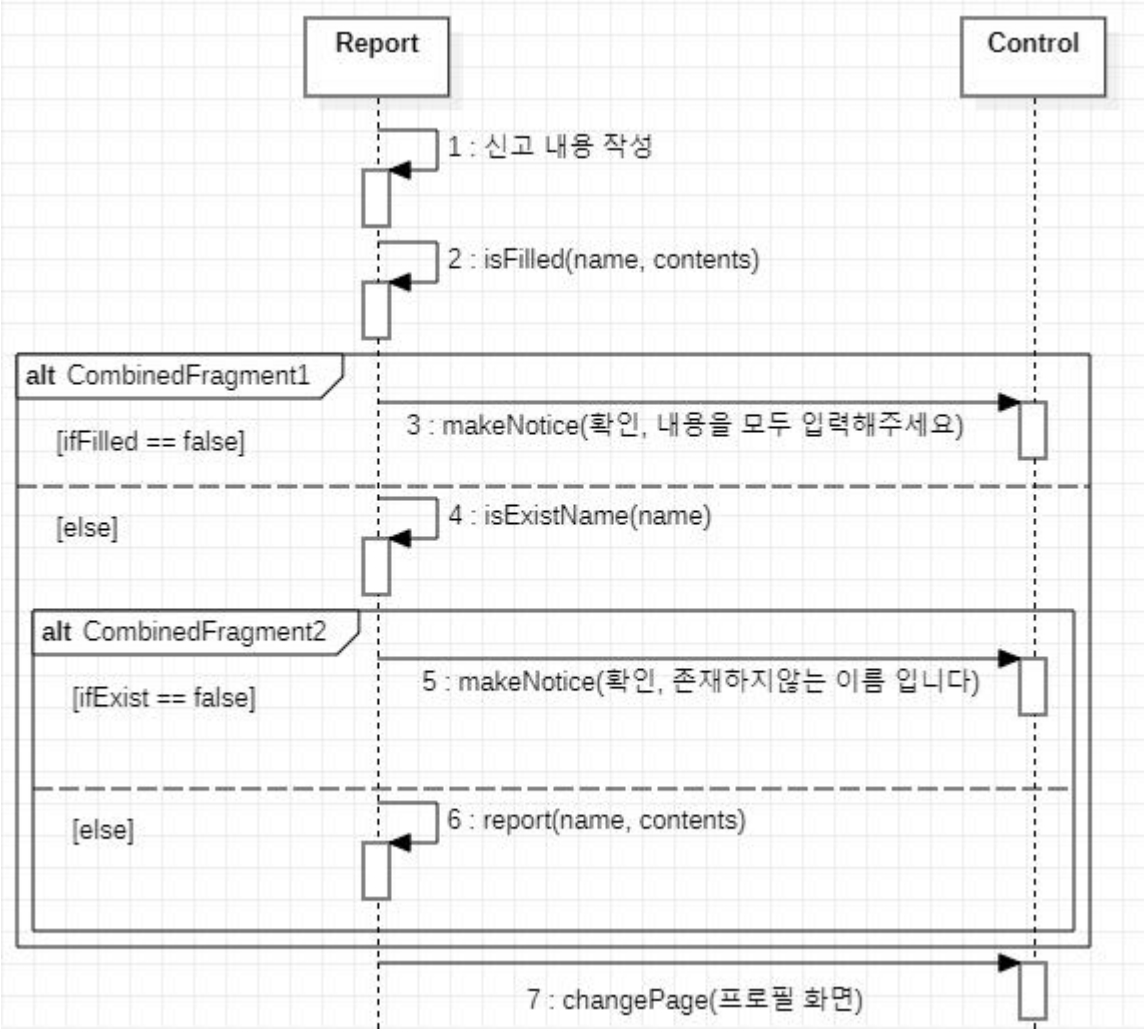
Manage Goods Use case 에서의 Sequence Diagram이다. MarketMain, ManageGoods, UpdateGoods 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 가게 관리자들은 가게의 등록된 상품들에 대해서 관리할 수 있다. 우선 등록된 상품들을 보기위해 Goods 객체를 생성하여 getGoodsList 메소드를 통해 받아와서 상품이 존재한다면 showGoodsList 메소드를 통해 화면에 보여지게 된다. 이제 관리를 위해 상품을 선택하게 되는데 2가지의 경우로 나뉘게 된다.

첫 번째는 기존 상품의 수정이다. 우선, 수정하고자 하는 상품이 선택되면 checkPage 메소드를 통해 상품 관리 화면으로 이동하며 ManageGoods 객체가 시작된다. 수정에는 역시

2가지로 나뉜다. 삭제와 정보 변경이다. 상품 삭제의 경우 재확인을 걸쳐 삭제가 결정되면 deleteGoods 메소드를 통해 해당 상품을 삭제하고 changePage 메소드로 메인화면으로 이동하여 변경된 점을 확인할 수 있다. 정보 변경의 경우 changePage 메소드를 통해 정보 입력 화면으로 이동하며 UpdateGoods 메소드가 시작된다. 가게 관리자들은 해당 상품의 변경 정보들을 입력한다. 입력된 정보들의 적합성을 확인한 후 통과가 되면 changeGoods 메소드를 통해 해당 상품의 정보가 변경된다. 여기서 changeOrnew 변수의 값을 조건에 걸리게 하는데 현재 입력 정보가 수정된 것인지 새로운 것인지 확인하기 위함이다. 최종적으로 changePage 메소드로 메인화면으로 이동하여 변경된 점을 확인할 수 있다.

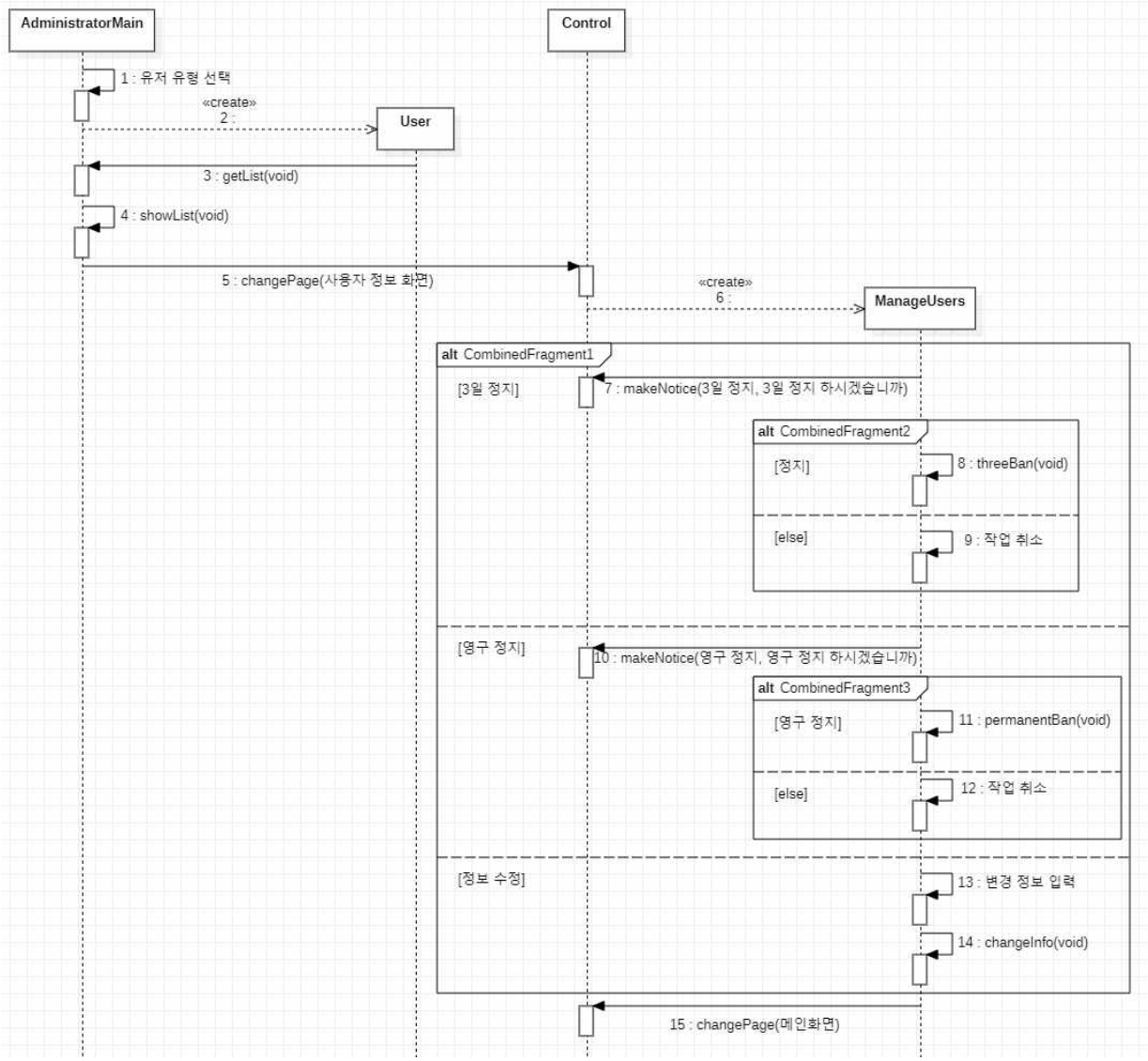
두 번째는 새로운 상품의 추가이다. 이 경우, 바로 changePage 메소드로 정보 입력 화면으로 이동하게 된다. 이때, UpdateGoods 객체가 생성되며 가게 관리자들은 새로운 상품들의 정보를 입력하게 된다. 역시 입력된 정보들의 적합성을 확인한 후 통과가 되면 addGoods 메소드를 통해 해당 상품의 정보가 등록된다. 여기서 역시 changeOrnew 변수의 값을 조건에 걸리게 하여 위의 경우와 다르게 추가의 경우에 필요한 addGoods 메소드를 호출하도록 한다. 최종적으로 changePage 메소드로 메인화면으로 이동하여 등록된 것을 확인할 수 있다.

13) Report



Report Use case 에서의 Sequence Diagram이다. Report 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 고객 또는 가게 관리자들은 특정 고객, 가게를 신고할 수 있다. 신고하기 화면으로 들어오게 되면 Report 객체가 시작된다. 우선 유저들은 신고 내용을 작성하게 된다. 이 내용에는 타겟 유저의 이름과, 신고 내용으로 구성된다. 최종 report 메소드가 수행되기 위해서는 두가지의 절차가 필요하다. 먼저 isFilled 메소드를 통해 비어있는 입력이 있는지 확인한다. 모두 입력이 되었다면 다음은 isExist 메소드를 통해 존재하는 고객 또는 가게인지 확인한다. 최종적으로 모두 확인이 되면, report 메소드가 수행되며 신고 내용이 시스템 관리자의 데이터베이스에 기록된다. changePage 메소드로 프로필 화면으로 이동하며 끝이 난다.

14) Manage Users

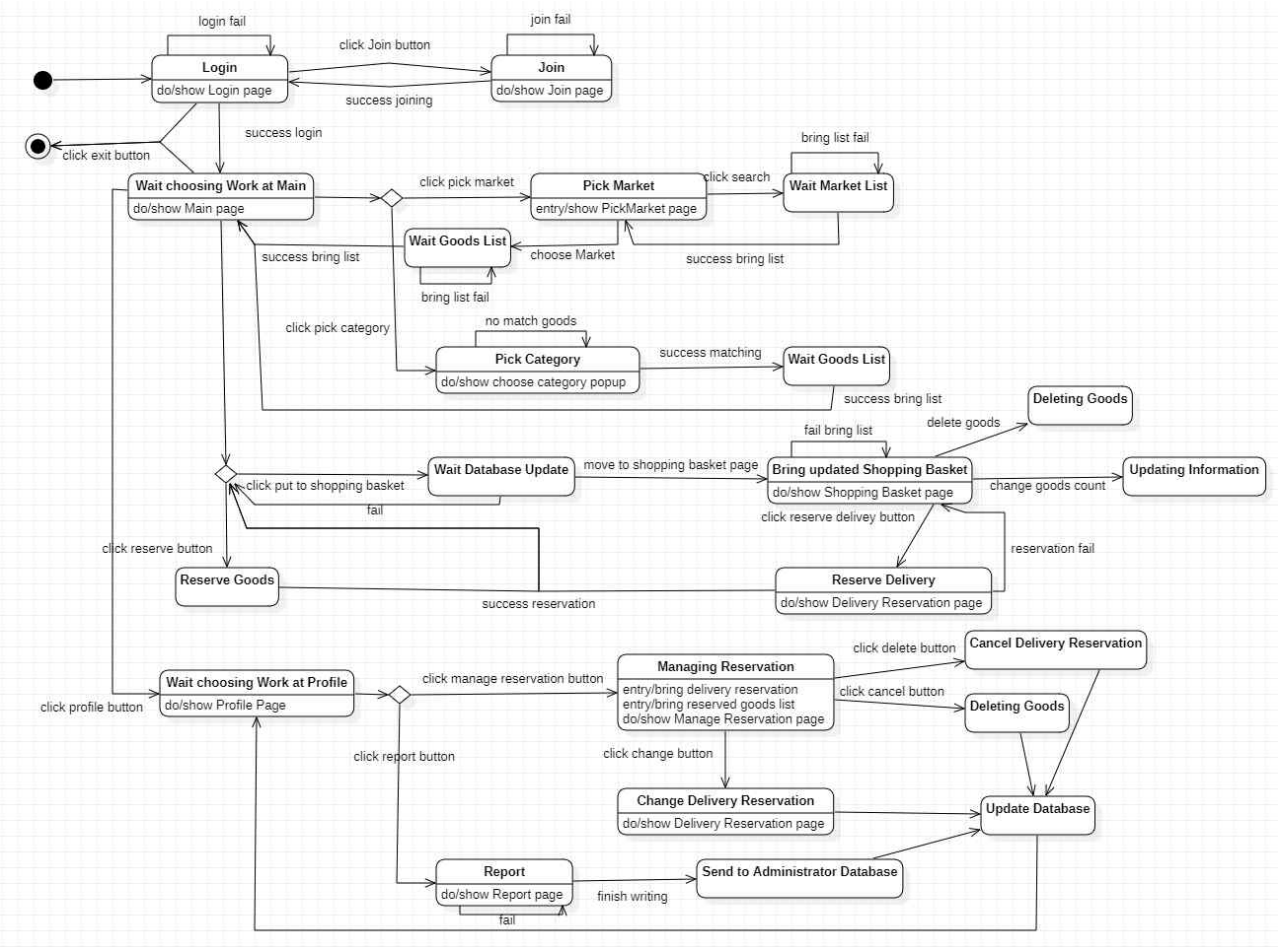


Manage Users Use case 에서의 Sequence Diagram이다. AdministratorMain, ManageUsers 클래스가 핵심 기능들을 하게 될 Controller 클래스이다. 시스템 관리자들은 신고 또는 여러 가지 사유로 특정 유저들에 대해 정지를 주거나 정보를 변경할 경우가 생긴다. 우선 유저 유형을 선택할 필요가 있다. 이 과정에서 User 객체가 생성되며 getList 메소드를 통해 유형에 맞는 유저들의 리스트를 가져온다. 그 후, showList 메소드를 통해 화면에 보여지고 chooseUser 메소드를 통해 특정 유저를 선택하게 된다. changePage 메소드를 통해 사용자 정보 화면으로 이동하며 ManageUsers 객체가 생성된다. 유저들의 관리에는 3가지가 존재한다. 첫 번째 3일정지이다. 우선 재확인이 필요하며 정지하기로 결정이 되면, threeBan 메소드로 해당 유저를 3일동안 정지시킨다. 두 번째는 영구정지이다. 역시 재확인이 필요하며 영구정지가 결정되면 permanentBan 메소드를 통해 해당 유저는 영구정지된다. 마지막으로 정보 수정이다. 변경된 정보를 입력하고 changeInfo 메소드를 통해 변경된 정보가 데이터베이스에 반영되며 끝이 난다.

4. State machine diagram

총 3가지의 경우로 나누어서 설명을 하겠다.

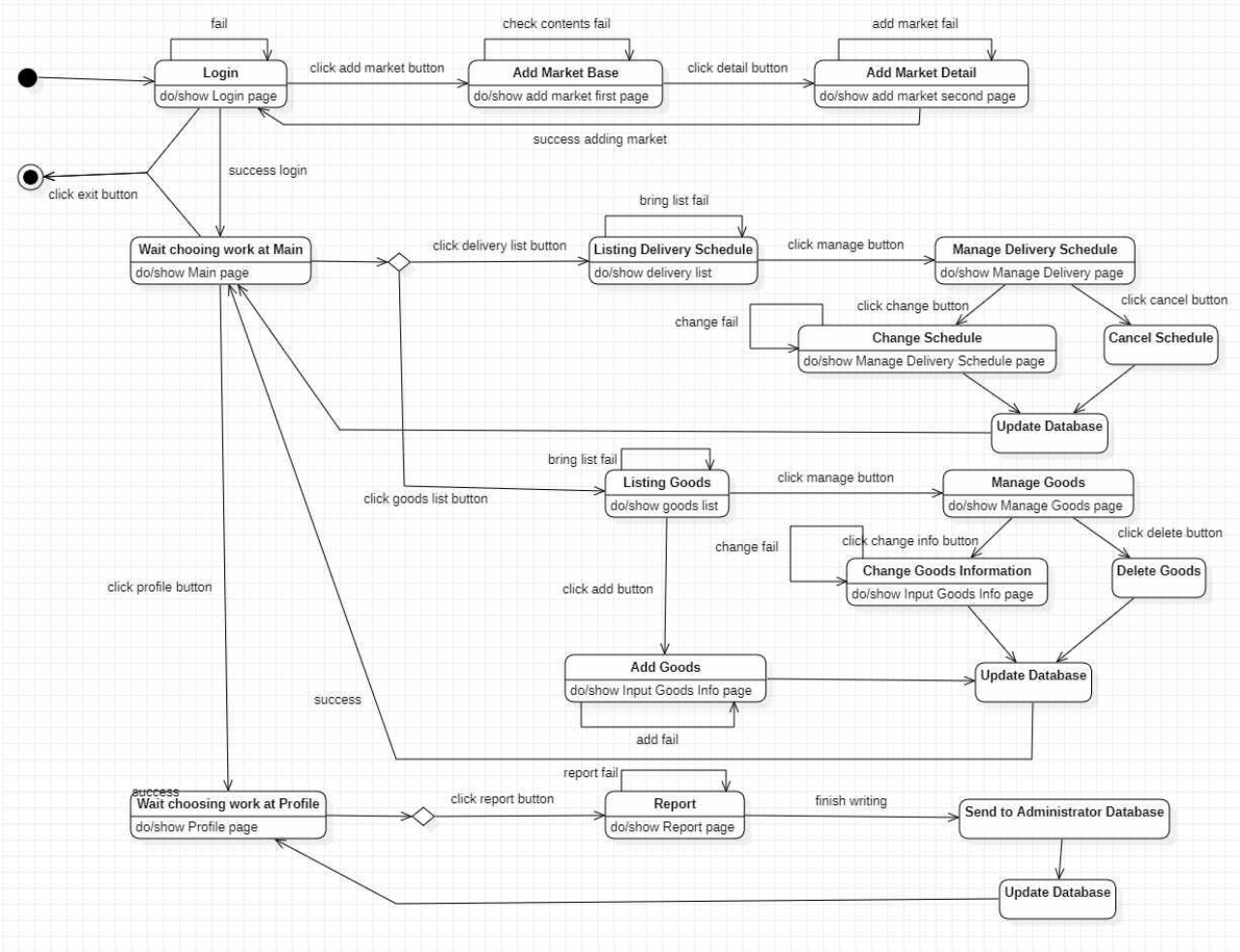
1) Customer



Customer 가 MNY 서비스를 이용하며 겪게 되는 시스템의 흐름을 나타낸다. 우선 MNY 의 서비스들을 이용하기 위해서는 Login 상태를 거쳐야 한다. 이때, 아직 MNY에 고객으로 등록되기 전이라면 Join 상태로 진입하여 등록 절차를 진행해야한다. 성공적으로 로그인을 하면 선택권이 주어진다. 우선 MNY의 주 목적인 가게와 상품의 선택이다. Pick Market 상태로 진입하게 될 경우, 원하는 조건을 입력하면 Wait Market List 상태로 진입하며 데이터베이스에서 조건의 가게 리스트를 가져오게 된다. 성공적으로 가져오면 Pick Market 상태로 되돌아오게되는데, 여기서 한 곳의 가게를 선택하면 해당 가게의 상품들을 가져오기 위해 Wait Goods List 상태로 진입하며 성공적으로 가져오고 나면 Wait chooising work at Main 상태로 재진입하게 된다. 이 다음은 위에서 가져온 상품들을 이용한 기능들의 선택이다. 상품들 중 특정 상품의 장바구니 담기 버튼이 눌리게 되면 Wait Database Update 상태로 진입하여 데이터베이스에 해당 상품이 장바구니로 담기게 되고 Bring updated Shopping Basket 상태에서 업데이트된 장바구니 데이터들을 보여주게 된다. 여기서 고객

들은 삭제와 정보 변경에 따른 각각의 상태로 진입할 수 있다. 그리고 예약 버튼을 누르게 되면 Reserve Delivery 상태로 진입하게 된다. 고객들은 가게가 등록한 배달 일정들 중에서 원하는 일정에 예약을 하고 나면 다시 Main 상태로 진입하게 된다. 다음은 Main에서 진입할 수 있는 마지막 상태인 Reserve Goods 상태이다. 메인화면에 떠있는 상품들 중에서 원하는 상품을 예약하고자할 때, 예약 버튼을 누르게 되면 진입할 수 있는 상태이다. 이 역시 예약이 성공적으로 진행되면 Main 상태로 재진입하게 된다. 마지막으로 Wait choosing work at Profile 상태이다. 이는 Main 상태에서 프로필 버튼을 누르게 되면 진입하게 된다. Profile 에서는 2가지의 경우로 나뉘게 된다. 첫 번째는 Managing Reservation 상태이다. 여기서 고객들은 배달 예약정보를 삭제하거나 변경할 수 있으며, 예약 상품들을 삭제할 수 있다. 그리고 그 선택에 따라 각각에 맞는 상태로 진입하게 된다. 두 번째는 Report 상태이다. 이름 그대로 특정 고객, 가게를 신고하고 싶을 때 진입하게 된다. 위의 2가지 경우 모두 성공적으로 진행이 되고나면 Update Database상태로 진입하게 되며, 업데이트까지 성공적으로 끝나게 되면 Profile 상태로 재진입하며 Customer가 MNY에서 이용하게 될 모든 상태들이 끝이 난다.

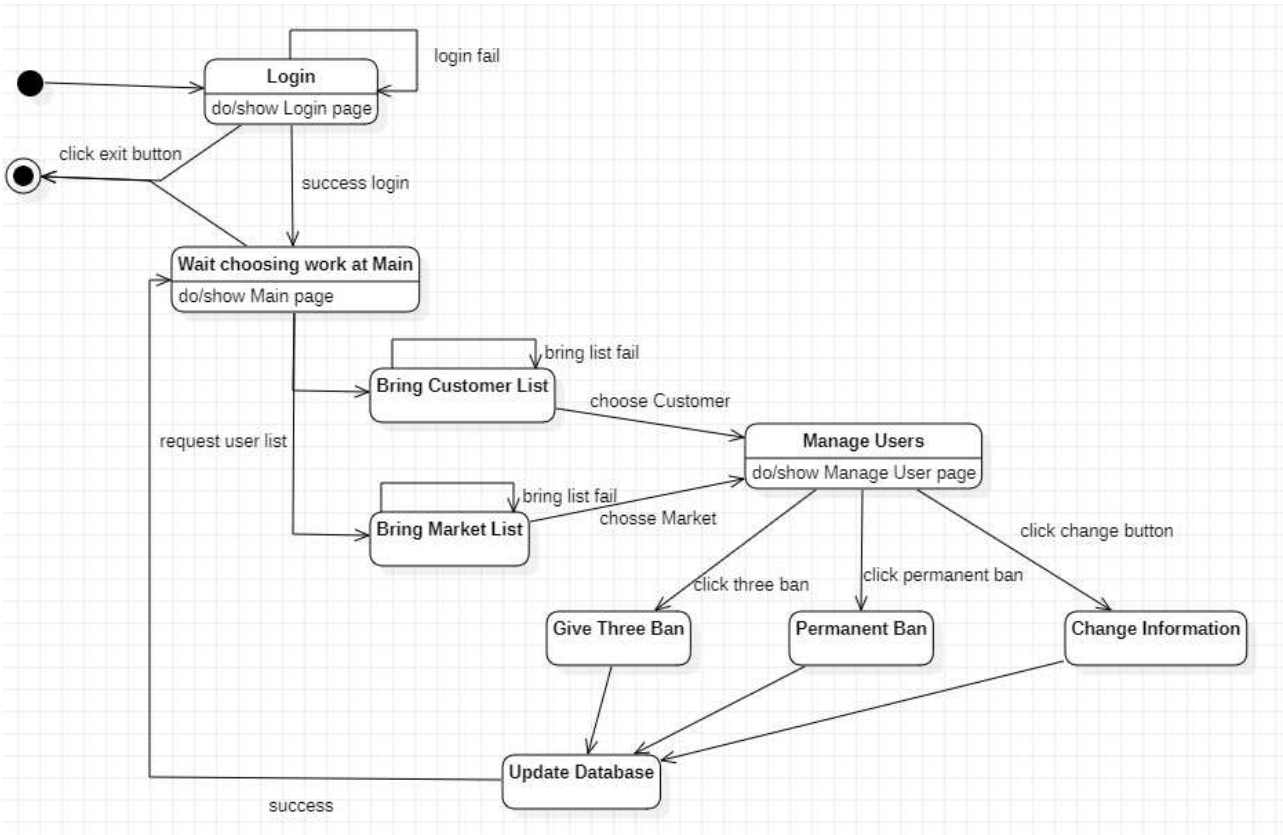
2) Market



Market 이 MNY 서비스를 이용하며 겪게 되는 시스템의 흐름을 나타낸다. 우선 MNY의 서비스들을 이용하기 위해서 Login 상태를 거쳐야 한다. 이때, 아직 MNY에 가게로 등록되기 전이라면 Add Marekt 상태로 진입하여 등록 절차를 진행해야한다. Add Market은 2단계의 절차로 나뉘어져있다. 첫 번째 Add Market Base에서 가게의 기본정보들이 입력되며 두 번째 Add Market Detail에서 세부적인 정보들이 모두 입력되며 Add Market 상태는 끝이 난다. 성공적으로 로그인을 하게되면 Wait choosing work at Main 상태로 진입하게 되는데 여기서 2가지의 선택권이 있다. 첫 번째는 Listing Delivery Schedule 상태이다. 이 상태에 진입하게 되면 자신이 등록한 배달 일정들에 대해 예약 현황들을 살펴볼 수 있다. 그 일들 중에서 하나의 일정을 선택하면 Manage Delivery Schedule 상태로 진입하며 배달 일정 관리 화면으로 넘어가게 된다. 배달 일정 관리에는 삭제, 일정 변경 2가지의 경우가 있으며 각 경우마다 진입하게 되는 상태가 존재한다. 두 경우의 상태가 끝이나면 Update Database 상태에 진입하며 삭제되거나 변경된 정보가 실제 데이터베이스에 적용된다. 그 후, Main 상태로 재진입하며 첫 번째의 경우는 끝이 난다. 두 번째로는 Listing Goods 상태이다. 이 상태에 진입하게 되면 가게에 등록된 상품들에 대해 살펴볼 수 있다.

이 중 특정 상품을 선택하면 Manage Goods 상태로 진입하게 되며 상품들에 대한 삭제 또는 정보 변경의 관리가 가능해진다. 역시 각각 상태가 존재한다. Listing Goods 상태에서는 Add Goods 상태로도 진입이 가능한데 이 경우는 새로운 상품의 추가일 때에만 진입이 가능하다. 역시 각 상태에서의 작업이 끝나면 Update Database 상태로 진입하여 데이터베이스에 적용을 한 후 Main 상태로 재진입하게 된다. Customer 때와 마찬가지로 Main 상태에서 Wait choosing work at Profile 상태로 진입이 가능하며 Report 상태가 존재하는데, 그 흐름은 Customer와 똑같이 진행된다.

3) Administrator



마지막으로 Administrator의 입장에서 나타낸 시스템의 흐름이다. 앞의 Customer, Market과 마찬가지로 Login 상태를 거치게 된다. 이때 Administrator는 시스템의 관리자로써 특별히 회원등록 과정은 거치지 않는다. 곧바로 로그인하여 Wait chossing work at Main 상태로 진입할 수 있다. Administrator는 MNY에 등록된 유저들에 대해 관리가 가능하다. 우선, 고객 또는 가게의 유형을 지정하여 리스트를 받아오게 된다. 그 각각의 상태가 Bring Customer List, Bring Market List가 된다. 리스트를 받아오고 Manage Users 상태로 진입하면 유저 관리 화면으로 이동하게 된다. 유저들의 관리에는 3일 정지, 영구 정지, 정보 변경 3가지로 구분되는데 역시 각각에 상태가 존재하게 된다. 각 상태가 끝나고나면 Update Database를 통해 모든 변경점들이 데이터베이스에 적용되며 Main 상태로 재진입하게 하고 Administrator의 MNY에서의 흐름은 끝이 난다.

5. Implementation requirements

S/W Requirements
(1) Android 5.0+ (Lollipop+)
(2) SDK 21+

H/W Requirements
(1) Android 기반의 스마트 기기
(2) 갤럭시 s6 (or 동급) 이상

6. Glossary

C	
Controller	MVC 디자인 패턴에서 3가지 중 C에 해당하며, 핵심 기능들을 갖는 클래스를 뜻한다.
M	
Method	클래스에 구현된 함수들을 Method라 칭한다.
Model	MVC 디자인 패턴에서 3가지 중 M에 해당하며, View에서 Controller를 통해 기능 수행에 대상이 되는 클래스들을 뜻한다. ex) 학생, 유저...

7. References

[홍드로이드 안드로이드 앱 만들기]
https://www.youtube.com/watch?v=UNKlX9J6m-A&list=PLC51MBz7PMyyyR2I4gGBMFMMUfYmBkZxm
[Android Developers Document]
https://developer.android.com/docs?hl=ko
[안드로이드의 MVC패턴]
https://jroomstudio.tistory.com/21
수업 중 사용된 강의자료 6~9