
소프트웨어시스템분석및설계

최종 보고서

과목명 : 소프트웨어시스템분석및설계

학과명 : 소프트웨어공학과

팀원 : 강연범, 박지수, 백민재, 정세연



목 차

1. 프로젝트 개요	1
1.1 주제 선정 동기	1
1.1.1 브레인 스토밍(Brain Storming)을 통한 아이템 선정	1
1.2 프로젝트 주제 소개	2
1.2.1 주제 소개	2
1.2.2 출처	2
1.3 프로젝트 추진전략 및 방법	2
1.3.1 팀원 소개	2
1.3.2 WBS(Work Breakdown Structure)	3
1.3.3 LRC(Linear Responsibility Chart)	4
1.4 프로젝트 수행 일정	5
1.4.1 간트 차트를 활용한 프로젝트 수행 일정	5
1.4.2 회의록	6
2. 프로젝트 수행 내용 및 결과	25
2.1 Use case diagram	25
2.1.1 요구사항 분석	25
2.1.2 Use case 명세	26
2.1.3 Use case diagram 작성	34
2.2 Activity diagram	35
2.2.1 객체 선정 및 처리 순서 결정	35
2.2.2 Activity diagram 작성	35

2.3 Class diagram	37
2.3.1 Class 분석	37
2.3.2 Class 간 연관관계 도출	40
2.3.3 Class diagram 작성	44
2.4 Sequence diagram	44
2.4.1 객체 분류	44
2.4.2 Sequence diagram 작성	45
2.5 Communication diagram	52
2.5.1 객체 분류	52
2.5.2 Communication diagram 작성	52
2.6 State machine diagram	60
2.6.1 Event, Guard condition, Action	60
2.6.2 State machine diagram 작성	61
2.7 수정	62
2.7.1 Use case diagram	62
2.7.2 Activity diagram	64
2.7.3 Class diagram	64
2.7.4 Sequence diagram	67
2.7.5 Communication diagram	68
2.7.6 State machine diagram	70

3. 소스 코드	71
3.1 소스코드	71
3.2 실행결과 화면	125
4. 후기	132
4.1 어려운 점 및 개선 사항	132
4.2 프로젝트 후기	133

1. 프로젝트 개요

1.1 주제 선정 동기

UML을 활용한 응용 소프트웨어 분석 프로젝트의 주제는 참여 객체와 그 상호작용을 UML 다이어그램으로 명확하게 작성할 수 있어야 한다.

브레인스토밍의 결과로 주제는 여행 관리 시스템으로 선정했다. 여행 관리 시스템은 모바일 어플리케이션으로 구현되어 사용자와의 상호작용이 명확하다. 그러므로 boundary 객체 구분이 쉽고, 사용자의 입력 수신과 결과 출력에 대해 간결하게 나타낼 수 있다. 또한 시스템의 필요한 기능만을 구현하여 기능적 요구사항이 명확하고 객체의 주 흐름과 대체 흐름을 파악하기 쉽다는 장점이 있다.

1.1.1 브레인 스토밍(Brain Storming)을 통한 아이템 선정

1차 브레인스토밍 결과		
	아이디어 도출	선택
○ : 전원 찬성	1. 당근마켓	×
△ : 일부 찬성	2. 도서 관리 시스템	○
✖ : 전원 반대	3. 인스타그램	✖
	4. 마켓컬리	△
	5. 오늘의 집	✖
	6. 벨로그	✖
	7. 크림	✖
	8. 쇼핑몰 사이트	△
	9. 스타벅스	○
	10. 여행 관리 시스템	○
선정된 아이디어	도서 관리 시스템	

도서 관리 시스템, 스타벅스, 도서 관리 시스템이 전원 찬성으로 선정되었다. 그 중에서 도서 관리 시스템이 최종 선정되었다.

아이디어 평가		실용성	UML 적합성	난이도	합계
		4	3		
도서 관리 시스템		3	2	3	8
스타벅스		5	3	4	12
여행 관리 시스템					
선정된 최종 아이디어	여행 관리 시스템				

도서 관리 시스템으로 하려고 하였으나 클래스가 너무 방대하다고 생각되어 아이디어 평가를 통해 앞에서 선정된 3가지를 각각의 평가 기준에 따라 점수를 매겨보았다. 그 결과, 여행 관리 시스템이 최종적으로 선정되었다.

1.2 프로젝트 주제 소개

1.2.1 주제 소개

여행 관리 앱 ‘Travel Maker’

여행 시 일정 관리와 예산 관리는 필수적이다. 이 때 여행 관리 어플리케이션을 이용하면 일정과 예산을 쉽게 관리할 수 있다.

Travel Maker는 여행 일정 관리, 관광지 검색, 여행 지출 관리를 주 기능으로 가지고 있다. 여행 일정 관리 기능은 여행 전 일정 계획을 세울 때, 여행 중 일정을 확인할 때 유용하다. 그리고 여행 지출 관리 기능은 가계부를 쓸 때 사용할 수 있다. 또한 지역 별로 관광지를 검색할 수 있으며 검색 결과를 일정으로 추가할 수 있다.

1.2.2 출처

<https://github.com/rin-k645/travelmaker>

1.3 프로젝트 추진전략 및 방법

1.3.1 팀원 소개

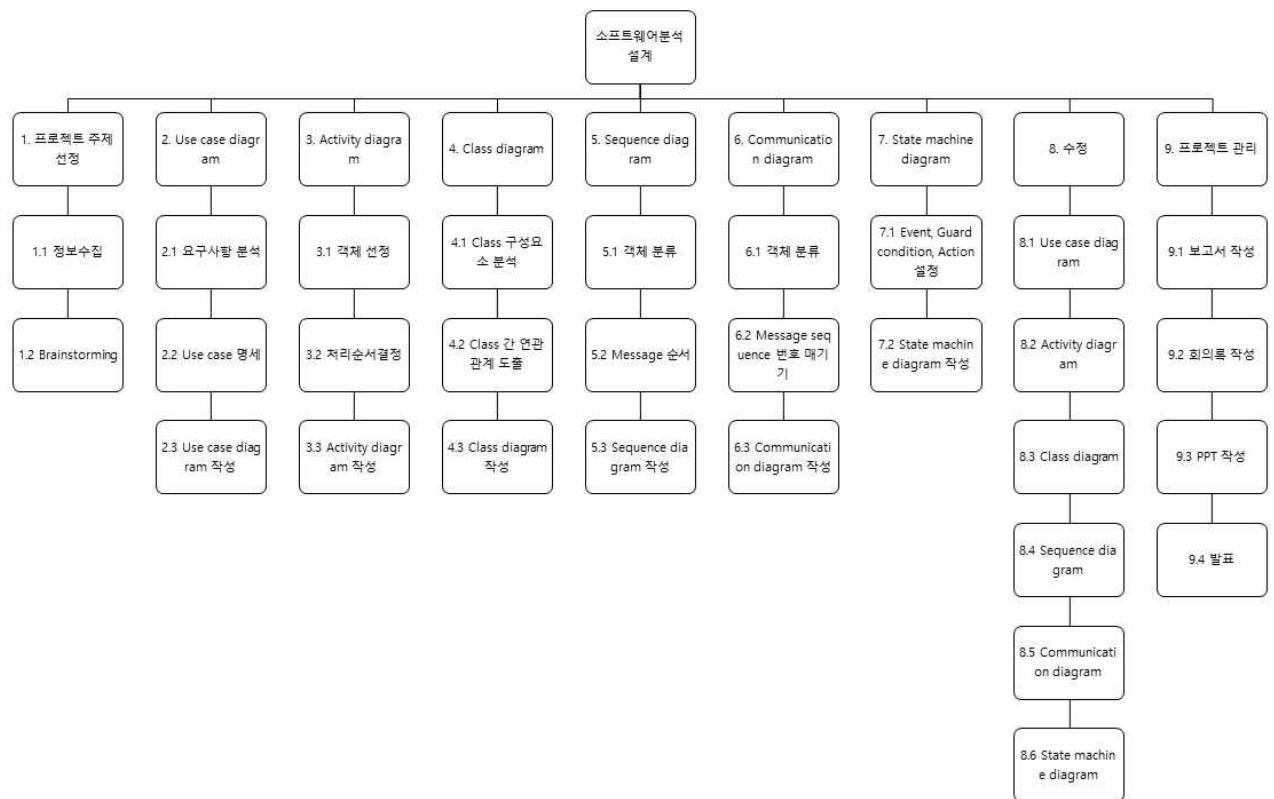
강연범 : 이번 프로젝트 진행함에 있어서 Use case 명세 도출과 State machine diagram 을 중점적으로 맡았고, 최종 발표 때 발표를 맡게 되었습니다.

박지수 : Communication diagram을 주도하여 그렸고, Sequence diagram, Class diagram, Use case diagram은 보조로 담당하여 그렸습니다. 최종 보고서와 회의록 또한 주로 담당하였습니다.

백민재 : Class 구성요소 분석, Sequence diagram 작성 및 수정, PPT 제작을 주로 책임져서 활동하였고, 그 외에 팀원들과 협업을 통해 유기적으로 협조하여 업무를 분담하였습니다.

정세연 : Use case diagram을 주도하여 그렸으며 이를 바탕으로 Activity diagram을 주도하였습니다. Class diagram에 많은 기여를 했으며 Sequence diagram, Communication diagram, State machine diagram은 조력자 역할을 하였습니다.

1.3.2 WBS(Work Breakdown Structure)



1.3.3 LRC(Linear Responsibility Chart)

	강연법	박지수	백민재	정세연
1. 프로젝트 주제선정	2	4	1	2
1.1 정보수집	3	4	1	3
1.2 Brainstorming	2	1	2	2
2. Use case diagram	2	4	3	1
2.1 요구사항 분석	2	1	2	4
2.2 Use case 명세	1	4	2	2
2.3 Use case diagram 작성	2	2	2	1
3. Activity diagram	2	2	2	1
3.1 객체선정	2	2	2	1
3.2 처리순서결정	2	2	2	1
3.3 Activity diagram 작성	2	2	2	1
4. Class diagram	2	2	3	1
4.1 Class 구성요소 분석	2	2	1	4
4.2 Class 간 연관관계 도출	2	4	2	1
4.3 Class diagram 작성	2	1	2	4
5. Sequence diagram	2	2	1	2
5.1 객체 분류	2	1	2	4
5.2 Message 순서	2	2	1	2
5.3 Sequence diagram 작성	2	2	1	2
6. Communication diagram	2	1	2	2
6.1 객체 분류	2	1	2	4
6.2 Message sequence 번호 매기기	2	1	2	2
6.3 Communication diagram 작성	2	1	2	2
7. State machine diagram	1	3	2	2
7.1 Event, Guard condition, Action 설정	1	3	2	2
7.2 State machine diagram 작성	1	3	2	2
8. 수정	2	2	2	1
8.1 Use case diagram	2	2	2	1
8.2 Activity diagram	2	2	2	1
8.3 Class diagram	2	2	2	1
8.4 Sequence diagram	2	2	1	2
8.5 Communication diagram	2	1	2	2
8.6 State machine diagram	1	2	2	2
9. 프로젝트 관리	2	1	2	4
9.1 보고서 작성	3	1	3	2
9.2 회의록작성	3	1	3	2
9.3 PPT 작성	4	3	1	3
9.4 발표	1	3	4	3
Key :				
1 = 주 책임자				
2 = 보조				
3 = 검토자				
4 = 최종승인				

1.4 프로젝트 수행 일정

1.4.1 간트 차트를 활용한 프로젝트 수행 일정

	10월		11월				12월		
	3	4	1	2	3	4	1	2	3
1. 프로젝트 주제선정									
1.1 정보수집									
1.2 Brainstorming									
2. Use case diagram									
2.1 요구사항 분석									
2.2 Use case 명세									
2.3 Use case diagram 작성									
3. Activity diagram									
3.1 객체 선정									
3.2 처리순서결정									
3.3 Activity diagram 작성									
4. Class diagram									
4.1 Class 구성요소 분석									
4.2 Class 간 연관관계 도출									
4.3 Class diagram 작성									
5. Sequence diagram									
5.1 객체 분류									
5.2 Message 순서									
5.3 Sequence diagram 작성									
6. Communication diagram									
6.1 객체 분류									
6.2 Message sequence 번호 매기기									
6.3 Communication diagram 작성									
7. State machine diagram									
7.1 Event, Guard condition, Action 설정									
7.2 State machine diagram 작성									
8. 수정									
8.1 Use case diagram									
8.2 Activity diagram									
8.3 Class diagram									
8.4 Sequence diagram									
8.5 Communication diagram									
8.6 State machine diagram									
9. 프로젝트 관리									
9.1 보고서 작성									
9.2 회의록 작성									
9.3 PPT 작성									
9.4 발표									

1.4.2 회의록

1주차

일 시	2022.10.27.목요일 / 15:00 ~ 16:00	장소	과방
작성자	박지수		
참석자	강연범, 박지수, 백민재, 정세연, 한상현		

회의 주제	1. 조장 선정 2. 주제 선정		
회의 내용	<p>1. 조장 선정 사다리 타기를 통하여 공정하게 조장을 선정하였다. 그 결과 박지수가 선정이 되었다.</p> <p>2. 주제 선정 브레인스토밍을 통하여 주제를 선정하였다.</p> <table border="1"><tr><td>○ : 전원 찬성 △ : 일부 찬성 × : 전원 반대</td><td>1. 당근마켓 × 2. 도서 관리 시스템 ○ 3. 인스타그램 × 4. 마켓컬리 △ 5. 오늘의 집 × 6. 블로그 × 7. 크림 × 8. 쇼핑몰 사이트 △ 9. 스타벅스 ○</td></tr></table> <p>도서 관리 시스템과 스타벅스가 전원 찬성으로 선정되었고, 그 중 도서 관리 시스템이 최종 선정되었다.</p>	○ : 전원 찬성 △ : 일부 찬성 × : 전원 반대	1. 당근마켓 × 2. 도서 관리 시스템 ○ 3. 인스타그램 × 4. 마켓컬리 △ 5. 오늘의 집 × 6. 블로그 × 7. 크림 × 8. 쇼핑몰 사이트 △ 9. 스타벅스 ○
○ : 전원 찬성 △ : 일부 찬성 × : 전원 반대	1. 당근마켓 × 2. 도서 관리 시스템 ○ 3. 인스타그램 × 4. 마켓컬리 △ 5. 오늘의 집 × 6. 블로그 × 7. 크림 × 8. 쇼핑몰 사이트 △ 9. 스타벅스 ○		



2주차

일 시	2022.11.3.목요일 / 17:00 ~ 18:30	장소	과방
작성자	박지수		
참석자	강연범, 박지수, 백민재, 정세연, 한상현		

회의 주제	1. 간트차트 작성 2. LRC 작성 3. WBS 작성																																	
회의 내용	<p>1. 간트차트를 엑셀파일을 사용하여 작성하였다.</p>  <table border="1" style="margin-top: 10px;"> <thead> <tr> <th>10월</th> <th>11월</th> <th>12월</th> </tr> <tr> <th>3주</th> <th>4주</th> <th>1주</th> <th>2주</th> <th>3주</th> <th>4주</th> <th>1주</th> <th>2주</th> <th>3주</th> </tr> </thead> <tbody> <tr> <td>1.1 정보 수집</td> <td>1.2 모티브</td> <td>2.1 use case 분석</td> <td>2.2 use case 명세</td> <td>2.3 use case diagram 작성</td> <td>2.4 diagram 수정</td> <td>3.1 족제 설정</td> <td>3.2 swim lane 정의</td> <td>3.3 족제 순서 결정</td> <td>3.4 activity diagram 작성</td> <td>3.5 diagram 수정</td> <td>4.1 class 구조요소 분석</td> <td>4.2 class 간 연관관계 도출</td> <td>4.3 class diagram 작성</td> <td>4.4 diagram 수정</td> <td>5.1 요구사항 분석을 통한 객체 파악</td> <td>5.2 객체와 참이 객체의 x축 나열</td> <td>5.3 객체 메세지 정의 메세지 호출을 시간 순서에 따라 표현</td> <td>5.4 sequence diagram 작성</td> <td>5.5 diagram 수정</td> <td>6.1 message, object, actor 선정</td> </tr> </tbody> </table>	10월	11월	12월	3주	4주	1주	2주	3주	4주	1주	2주	3주	1.1 정보 수집	1.2 모티브	2.1 use case 분석	2.2 use case 명세	2.3 use case diagram 작성	2.4 diagram 수정	3.1 족제 설정	3.2 swim lane 정의	3.3 족제 순서 결정	3.4 activity diagram 작성	3.5 diagram 수정	4.1 class 구조요소 분석	4.2 class 간 연관관계 도출	4.3 class diagram 작성	4.4 diagram 수정	5.1 요구사항 분석을 통한 객체 파악	5.2 객체와 참이 객체의 x축 나열	5.3 객체 메세지 정의 메세지 호출을 시간 순서에 따라 표현	5.4 sequence diagram 작성	5.5 diagram 수정	6.1 message, object, actor 선정
10월	11월	12월																																
3주	4주	1주	2주	3주	4주	1주	2주	3주																										
1.1 정보 수집	1.2 모티브	2.1 use case 분석	2.2 use case 명세	2.3 use case diagram 작성	2.4 diagram 수정	3.1 족제 설정	3.2 swim lane 정의	3.3 족제 순서 결정	3.4 activity diagram 작성	3.5 diagram 수정	4.1 class 구조요소 분석	4.2 class 간 연관관계 도출	4.3 class diagram 작성	4.4 diagram 수정	5.1 요구사항 분석을 통한 객체 파악	5.2 객체와 참이 객체의 x축 나열	5.3 객체 메세지 정의 메세지 호출을 시간 순서에 따라 표현	5.4 sequence diagram 작성	5.5 diagram 수정	6.1 message, object, actor 선정														

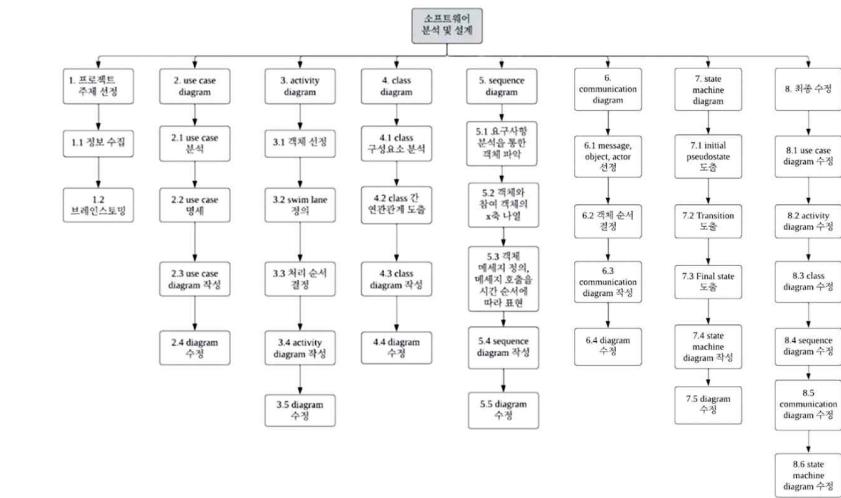
The diagram illustrates the relative duration of different UML diagram types. The timeline is divided into four colored segments:

- Blue segment (0-10): 6.2 객체 순서 결정, 6.3 communication diagram 작성, 6.4 diagram 수정
- Purple segment (10-20): 7 state machine diagram, 7.1 initial pseudostate 도출, 7.2 Transition 도출, 7.3 Final state 도출, 7.4 state machine diagram 작성, 7.5 diagram 수정
- White segment (20-30): 8 최종 수정
- Pink segment (30-40): 8.1 use case diagram 수정, 8.2 activity diagram 수정, 8.3 class diagram 수정, 8.4 sequence diagram 수정, 8.5 communication diagram 수정, 8.6 state machine diagram 수정

2. LRC를 작성하였고, 표의 숫자는 프로젝트를 진행하면서 채울 예정이다.

	박지수	한상현	정세연	강연범	백민재
1.프로젝트 주제선정					
1.1. 정보 수집					
1.2 브레인스토밍					
2. Use case Diagram					
2.1. Use case 분석					
2.2. Use case 명세					
2.3. Use case diagram 작성					
2.4. diagram 수정					
3.activity diagram					
3.1. 객체 선정					
3.2. swim lane 정의					
3.3. 처리 순서 결정					
3.4. activity diagram 작성					
3.5. diagram 수정					
4. Class diagram 작성					
4.1. class 구성요소 분석					
4.2. Class 간 연관관계 도출					
4.3. Class diagram 작성					
4.4. Diagram 수정					
5. Sequence diagram 작성					
5.1. 객체파악(요구사항 분석)					
5.2. 객체 및 참여객체 파악					
5.3. 객체 메세지 정의 시간 순서에 따른 표현					
5.4.Diagram 작성					
6. Communication diagram					
6.1. message, object, actor 설정					
6.2. 객체 순서 결정					
6.3. diagram 작성					
7. Sate machine diagram					
7.1. initial pseudo state 도출					
7.2. Transition 도출					
7.3. Final state 도출					
7.4. Diagram 작성					
8. 최종 수정					
8.1. Use case diagram 최종 수정					
8.2. Activity diagram 최종 수정					
8.3. Class diagram 최종 수정					
8.4. Sequence diagram 최종 수정					
8.5. Communication diagram 최종 수정					
8.6. State machine diagram 최종 수정					

3. WBS를 진행의 흐름을 토대로 작성하였다.

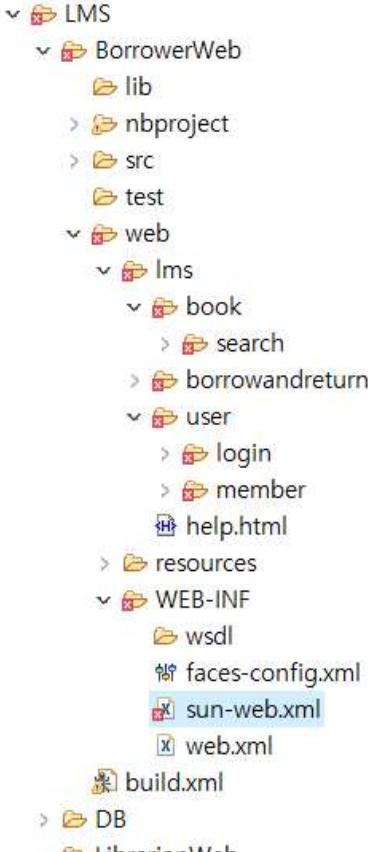


회의 사진



3주차

일 시	2022.11.8.화요일 / 13:00 ~ 14:00	장소	과방
작성자	박지수		
참석자	강연범, 박지수, 백민재, 정세연, 한상현		

회의 주제	1. 오류 코드 수정
	<p>1. 오류 코드 수정</p> <p>본격적인 다이어그램을 그리기에 앞서 실행이 되어야 하기 때문에 이클립스에서 코드를 열어보았지만 오류가 많았다.</p>  <pre> LMS BorrowerWeb lib nbproject src test web Ims book search borrowandreturn user login member help.html resources WEB-INF wsdl faces-config.xml sun-web.xml web.xml build.xml DB LibrarianWeb </pre> <p>The sun-web.xml file is highlighted in blue.</p> <p>sun-web.xml Error:</p> <pre> 1 The superclass "javax.servlet.http.HttpServlet" was not found on the Java Build Path 2<!-- 3 Document : UserRegistration 4 Generated by JBoss Seam 2.3.0.Final 5 --> </pre>
회의 내용	<p>위와 같은 오류가 발생하였는데, tomcat과 필요한 라이브러리들을 설치해</p>

주니 해결되었다.

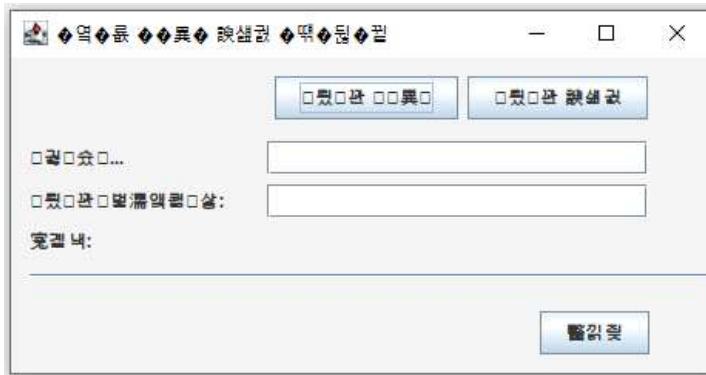
<해결된 모습>

```
UserRegistration.jsp
1 <?xml version="1.0" encoding="UTF-8"?>
2<!--
3     Document : UserRegistration
4     Created on : 2008. 6. 23, 오전 1:12:33-->
```

<설치한 라이브러리들>

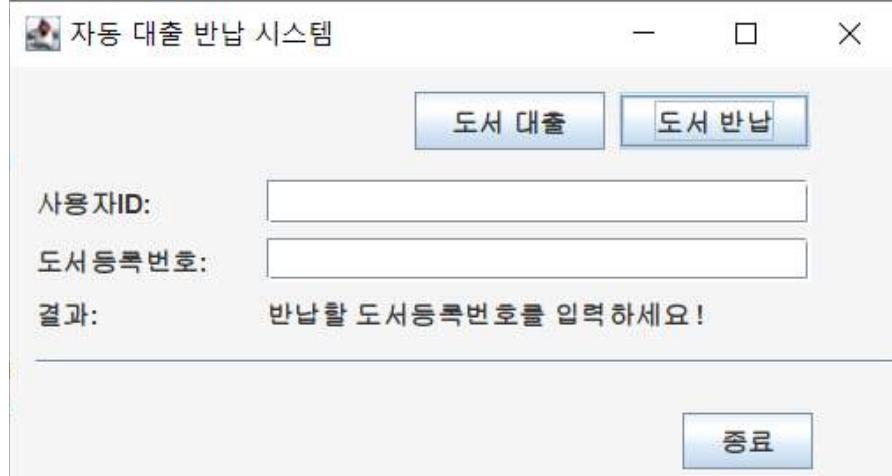
- Referenced Libraries
 - > dataprovider.jar - C:\Users\user\
 - > javax.annotation-api-1.2.jar - C:\U
 - > javax.ejb-api-3.2.jar - C:\Users\us
 - > javax.faces-api-2.0.jar - C:\Users\U
 - > javax.persistence-api-2.2.jar - C:\U
 - > LMS-CommonEntity.jar
 - > LMS-CommonType.jar
 - > rave-web-1.0.jar - C:\Users\user\
 - > webui-jsf-4.0.2.10.jar - C:\Users\U

실행을 시켜보니 글자가 깨지는 오류가 발생하였다.



인코딩 문제 때문에 글자가 깨지는 것이었다. 이클립스에서 인코딩타입을 변경하면 다음과 같이 문제가 해결되는 것을 볼 수 있다.

<해결된 모습>



1주차에서 모든 사람의 찬성을 받았던 도서관리 시스템 말고 스타벅스 또한 실행을 시켜보았다. 스타벅스는 안드로이드 스튜디오에서 실행을 시켰더니 다음과 같이 실행이 되었다. 하지만 GUI만 있고, 기능이 별로 없기 때문에 이 프로젝트와는 적합하지 않다고 생각되었다.

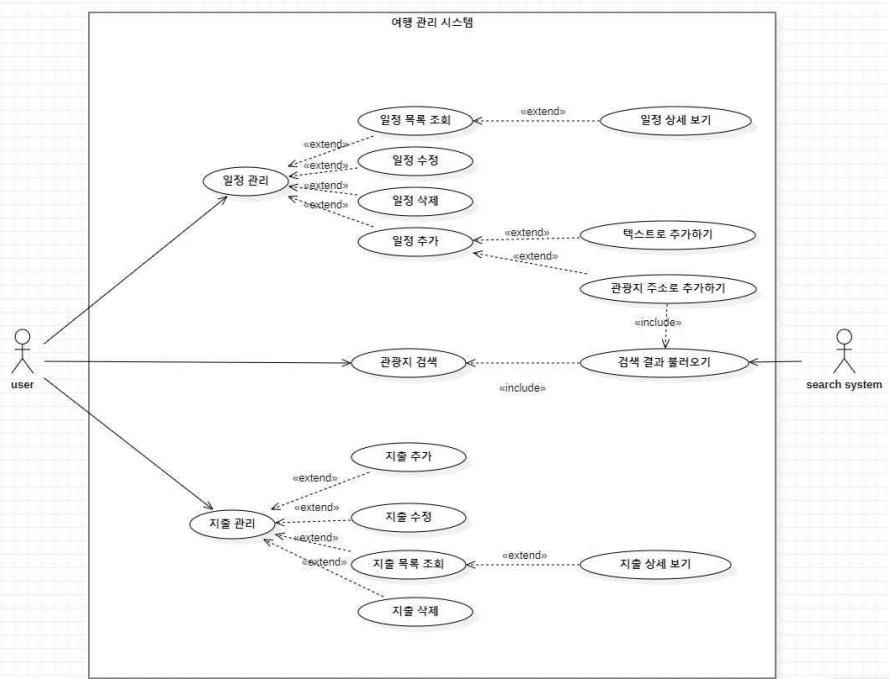
회의 사진	<p><스타벅스를 실행한 모습></p>  <p>3:40 7/25★</p> <p>18★ until Gold Level</p> <p>What's New Coupon</p> <p>4월 5일부터 일주일간! 사이렌 오더 위크에 참여하세요!</p> <p>2022.04.05 ~ 2022.04.11</p> <p>#가치위해같이해요 에코별 챌린지에 참여해 보세요!</p> <p>개인컵을 사용할수록 에코별이 늘어나요. 우리 함께 에코 라이프를 실천해 보아요!</p> <p>천일님을 위한 추천 메뉴</p> <p>자동 허니 블랙 티 아이스 아메리카노 스크립 라</p> <p>Home</p> <p>최종적으로는 도서관리 시스템이 선정되었고, 다음 주차부터는 본격적으로 다이어그램을 그릴 것이다.</p> 
-------	---

4주차

일 시	2022.11.15.화요일 / 17:00 ~ 20:00	장소	과방																																
작성자	박지수																																		
참석자	강연범, 박지수, 백민재(코로나로 인해 줌으로 참석), 정세연, 한상현																																		
회의 주제	1. 주제 변경 2. use case 명세 3. use case diagram 작성																																		
회의 내용	1. 주제 변경 도서 관리 시스템으로 하려고 했으나 클래스가 너무 방대하다고 생각되어 여행 관리 시스템으로 주제를 변경하였다. 2. use case 명세 use case를 총 16개로 도출해보았다. <table border="1"> <tr><td>UC-001</td><td>일정관리</td></tr> <tr><td>UC-002</td><td>일정목록조회</td></tr> <tr><td>UC-003</td><td>일정 상세 보기</td></tr> <tr><td>UC-004</td><td>일정 삭제</td></tr> <tr><td>UC-005</td><td>일정 수정</td></tr> <tr><td>UC-006</td><td>일정 추가</td></tr> <tr><td>UC-007</td><td>텍스트로 추가하기</td></tr> <tr><td>UC-008</td><td>관광지 주소로 추가하기</td></tr> <tr><td>UC-009</td><td>관광지 검색</td></tr> <tr><td>UC-010</td><td>지출관리</td></tr> <tr><td>UC-011</td><td>지출 목록 조회</td></tr> <tr><td>UC-012</td><td>지출 수정</td></tr> <tr><td>UC-013</td><td>지출 삭제</td></tr> <tr><td>UC-014</td><td>지출 추가</td></tr> <tr><td>UC-015</td><td>지출상세 보기</td></tr> <tr><td>UC-016</td><td>검색결과 불러오기</td></tr> </table> 그 중 하나인 UC-016을 첨부하였다.			UC-001	일정관리	UC-002	일정목록조회	UC-003	일정 상세 보기	UC-004	일정 삭제	UC-005	일정 수정	UC-006	일정 추가	UC-007	텍스트로 추가하기	UC-008	관광지 주소로 추가하기	UC-009	관광지 검색	UC-010	지출관리	UC-011	지출 목록 조회	UC-012	지출 수정	UC-013	지출 삭제	UC-014	지출 추가	UC-015	지출상세 보기	UC-016	검색결과 불러오기
UC-001	일정관리																																		
UC-002	일정목록조회																																		
UC-003	일정 상세 보기																																		
UC-004	일정 삭제																																		
UC-005	일정 수정																																		
UC-006	일정 추가																																		
UC-007	텍스트로 추가하기																																		
UC-008	관광지 주소로 추가하기																																		
UC-009	관광지 검색																																		
UC-010	지출관리																																		
UC-011	지출 목록 조회																																		
UC-012	지출 수정																																		
UC-013	지출 삭제																																		
UC-014	지출 추가																																		
UC-015	지출상세 보기																																		
UC-016	검색결과 불러오기																																		

유스케이스 이름	검색결과 불러오기		
유스케이스 고유번호	UC-016		
액터	주액터: 사용자 부액터: 검색 시스템		
개요	사용자가 관광지를 검색하면 검색 시스템이 그 결과를 불러온다.		
사전 조건	사용자가 관광지를 검색해야 한다.		
사후 조건	해당 없음		
	사용자	시스템	검색 시스템
기본 흐름	1. 관광지 검색		
	2. 관광지 검색 요청(->)		
		3. 요청한 관광지 결과 불러오기(->)	
	4. 사용자에게 검색된 관광지 리스트 출력		
대체 흐름	5. 검색 결과가 없으면 안내 메시지 출력		

3. use case diagram 작성

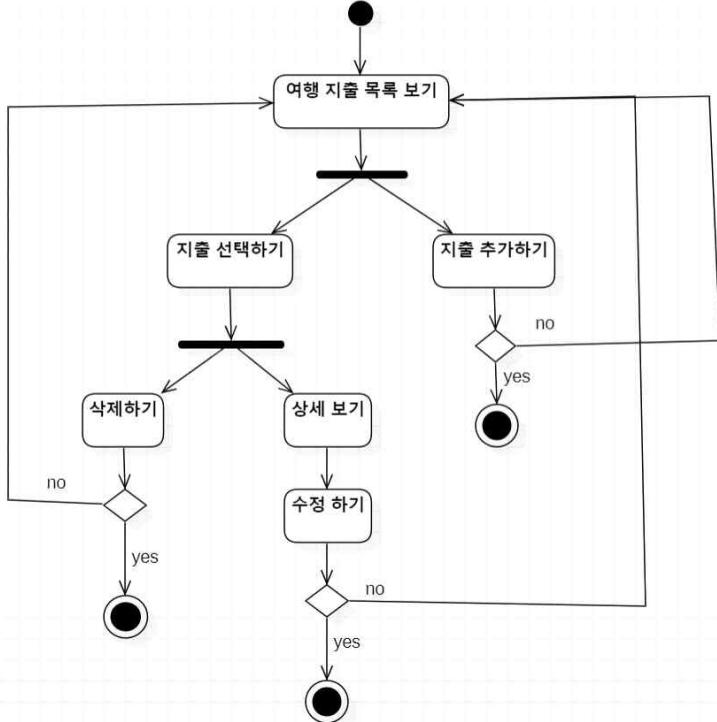




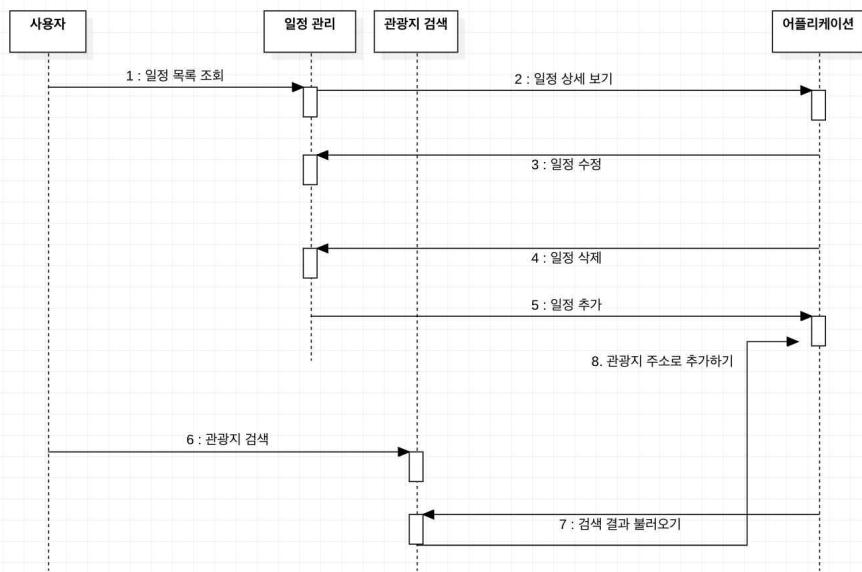
5주차

일 시	2022.11.22.화요일 / 17:00 ~ 18:30	장소	과방
작성자	박지수		
참석자	강연범, 박지수, 백민재, 정세연		

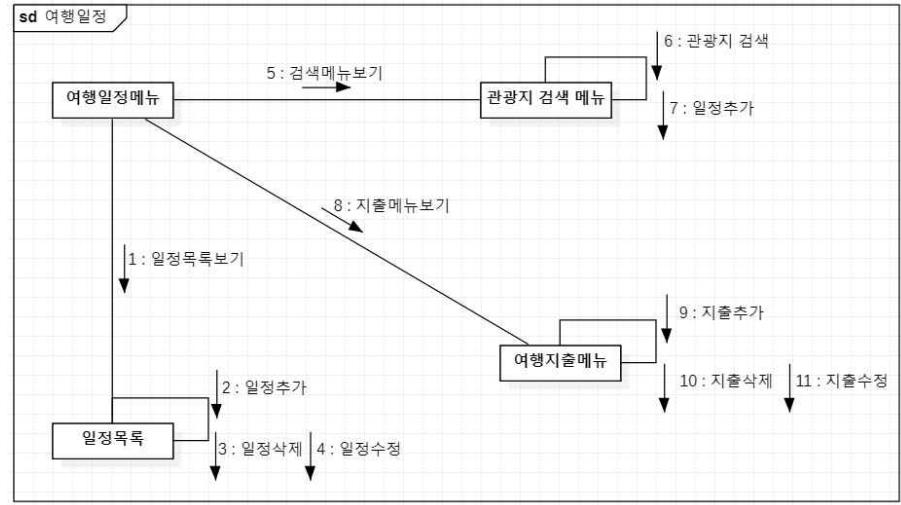
회의 주제	1. activity diagram 작성 2. sequence diagram 작성 3. communication diagram 작성 4. state machine diagram 작성
회의 내용	1. activity diagram 작성



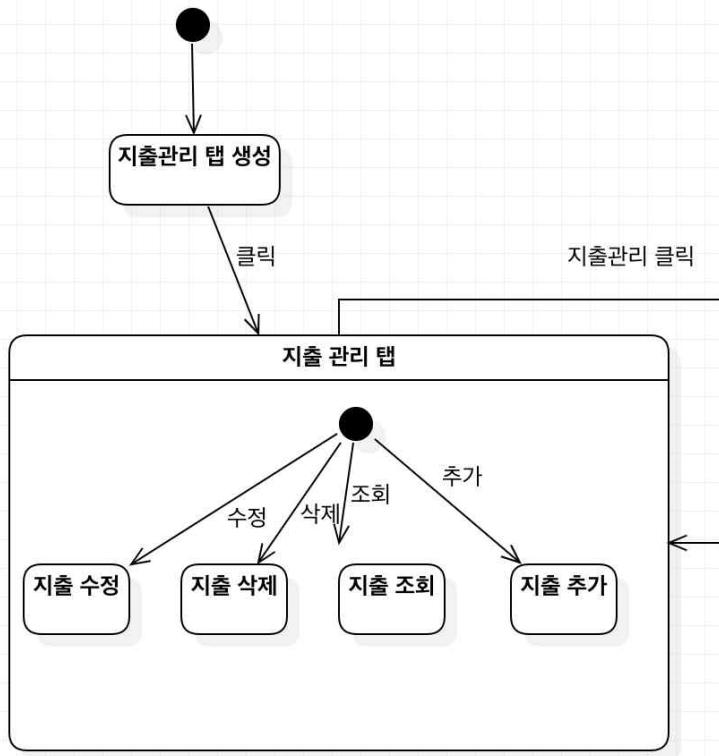
2. sequence diagram 작성



3. communication diagram 작성



4. state machine diagram 작성



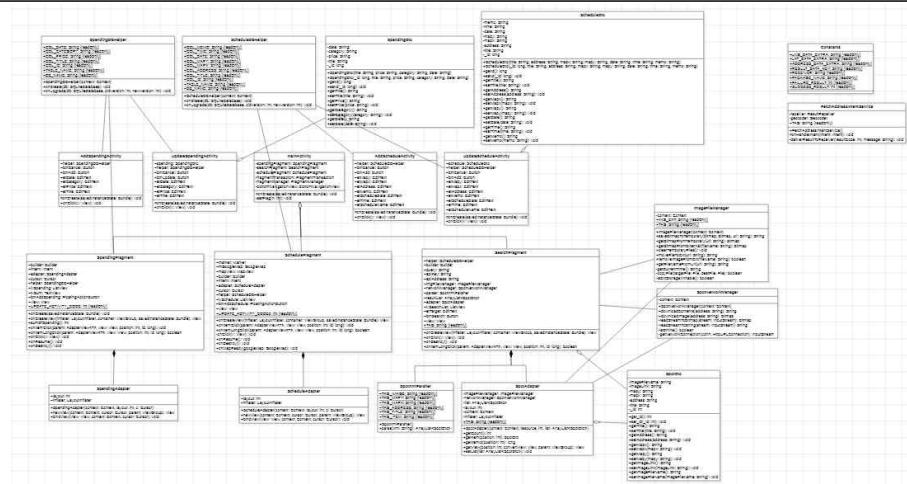
다음 주차부터는 class diagram을 그리고, 그렸던 나머지 diagram들을 수정할 예정이다.



6주차

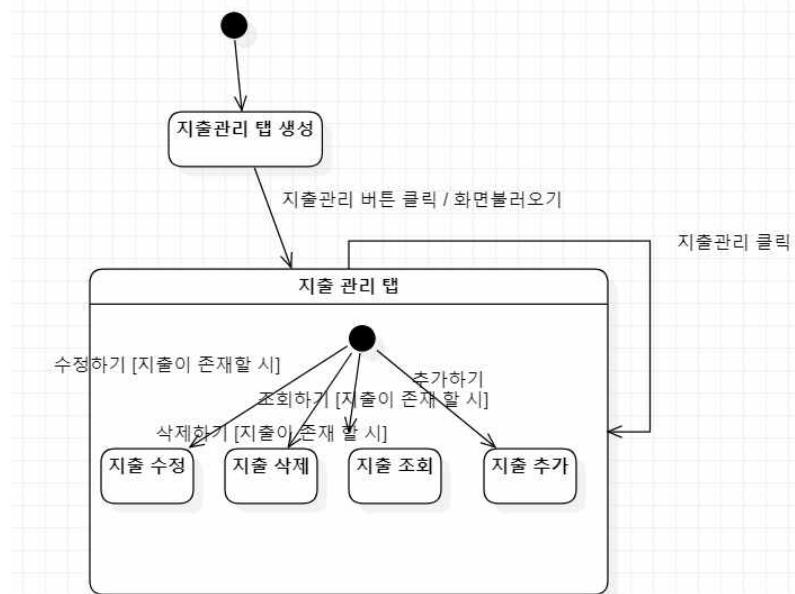
일 시	2022.11.29.화요일 / 17:00 ~ 18:30	장소	과방
작성자	박지수		
참석자	강연범, 박지수, 백민재, 정세연		

회의 주제	1. class diagram 작성 2. state machine diagram 수정 3. sequence diagram 수정
회의 내용	1. class diagram 작성 class 사이의 관계를 잘 생각하여 다음과 같이 나타내어 보았다. 제출 전 까지 다이어그램을 계속해서 살펴보면서 수정할 계획이다.



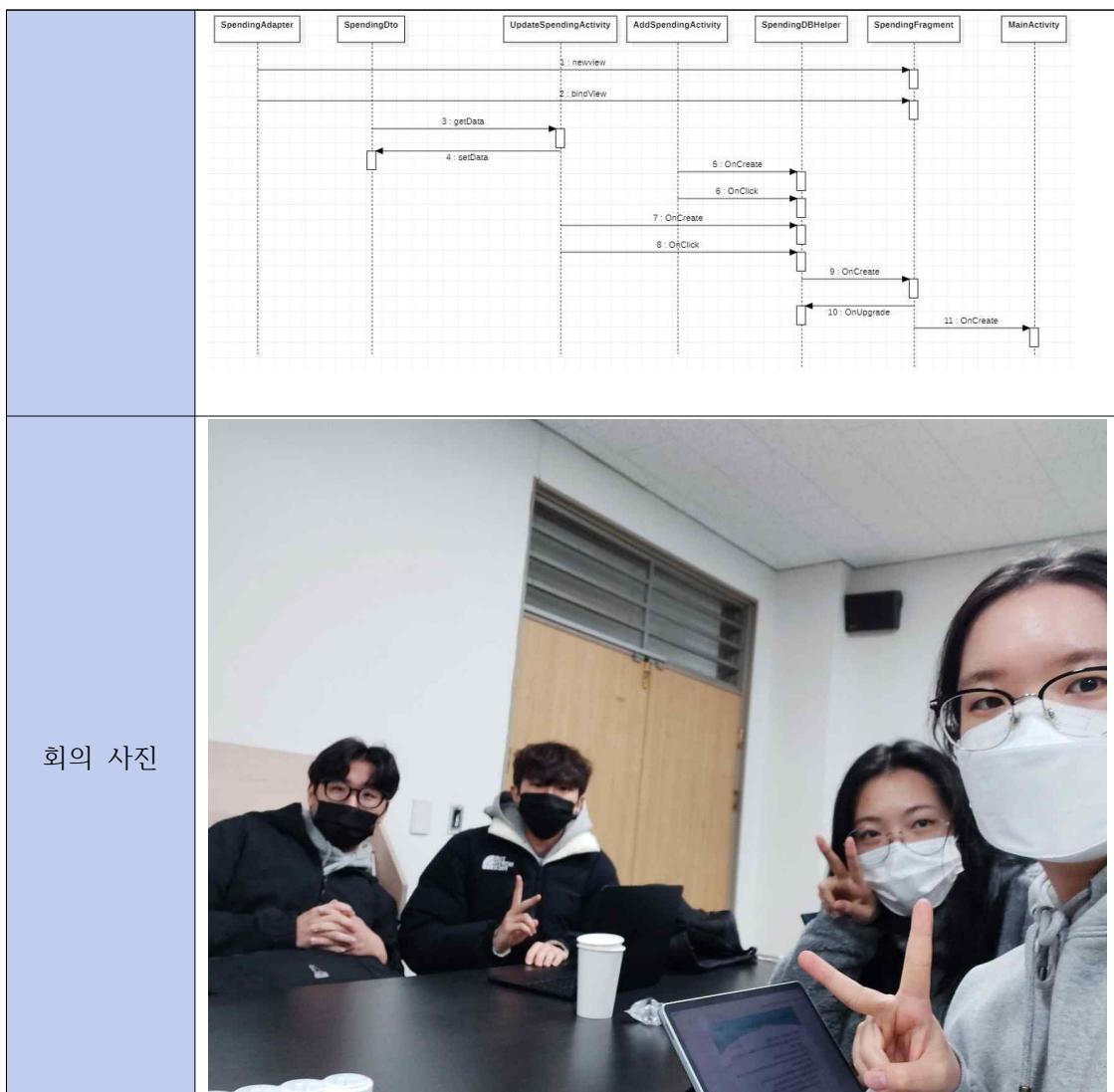
2. state machine diagram 설정

event, condition, action 등을 생각하여 다시 그려보았다. 그 중 일부를 다음과 같이 나타내어 보았다.



3. sequence diagram 설정

이전에는 한글을 이용하여 흐름을 이해해 보았다면 이번에는 코드를 사용하여 sequence diagram을 수정해 보았다. 그 중 일부를 다음과 같이 나타내어 보았다.

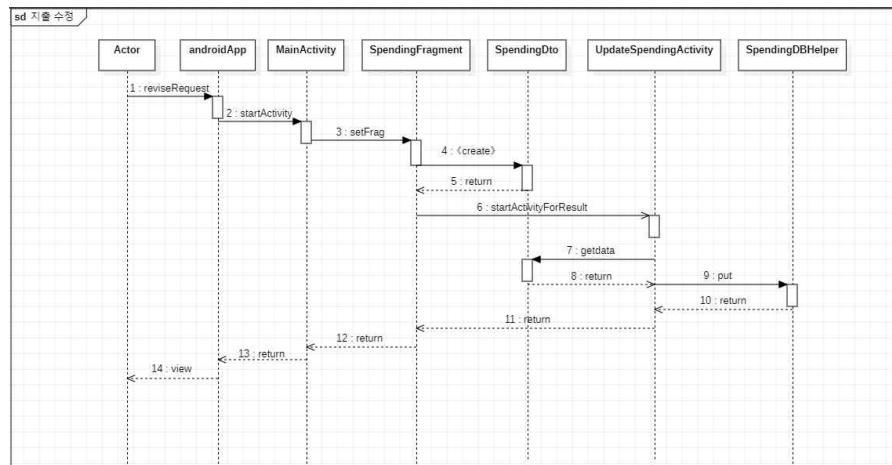


7주차

일 시	2022.12.6.화요일 / 14:00 ~ 16:00, 18:00 ~ 20:00	장소	과방
작성자	박지수		
참석자	강연범, 박지수, 백민재, 정세연		

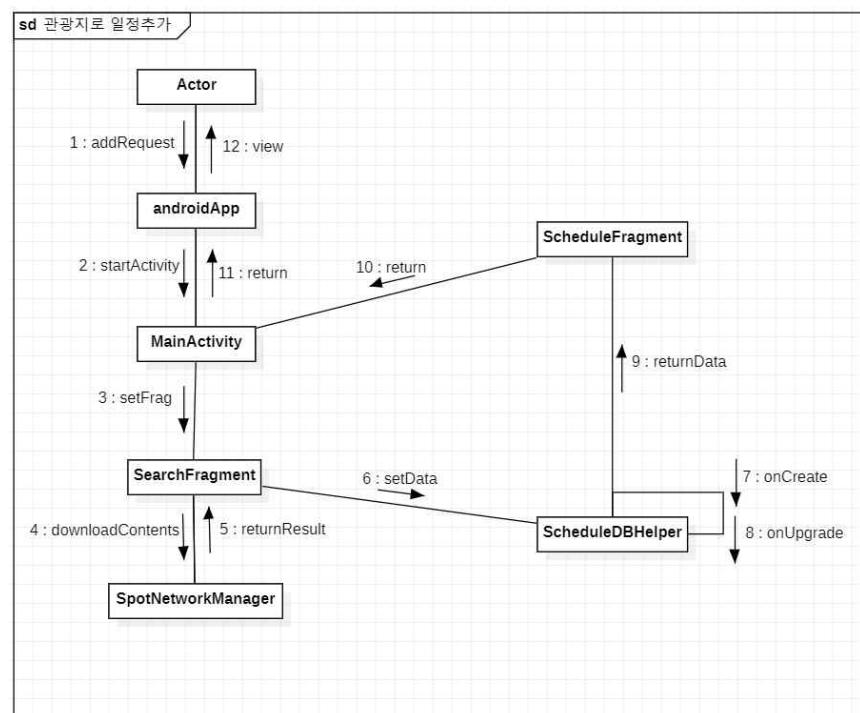
회의 주제	1. sequence diagram 수정 2. communication diagram 수정
회의 내용	generic form은 모든 가능한 interaction을 표현하는 것이고, instance form은 특정한 scenario를 표현한 것이다. 과제를 수행할 때는 instance form을 사용하라고 하셔서 각 use case 마다 sequence diagram을 그려

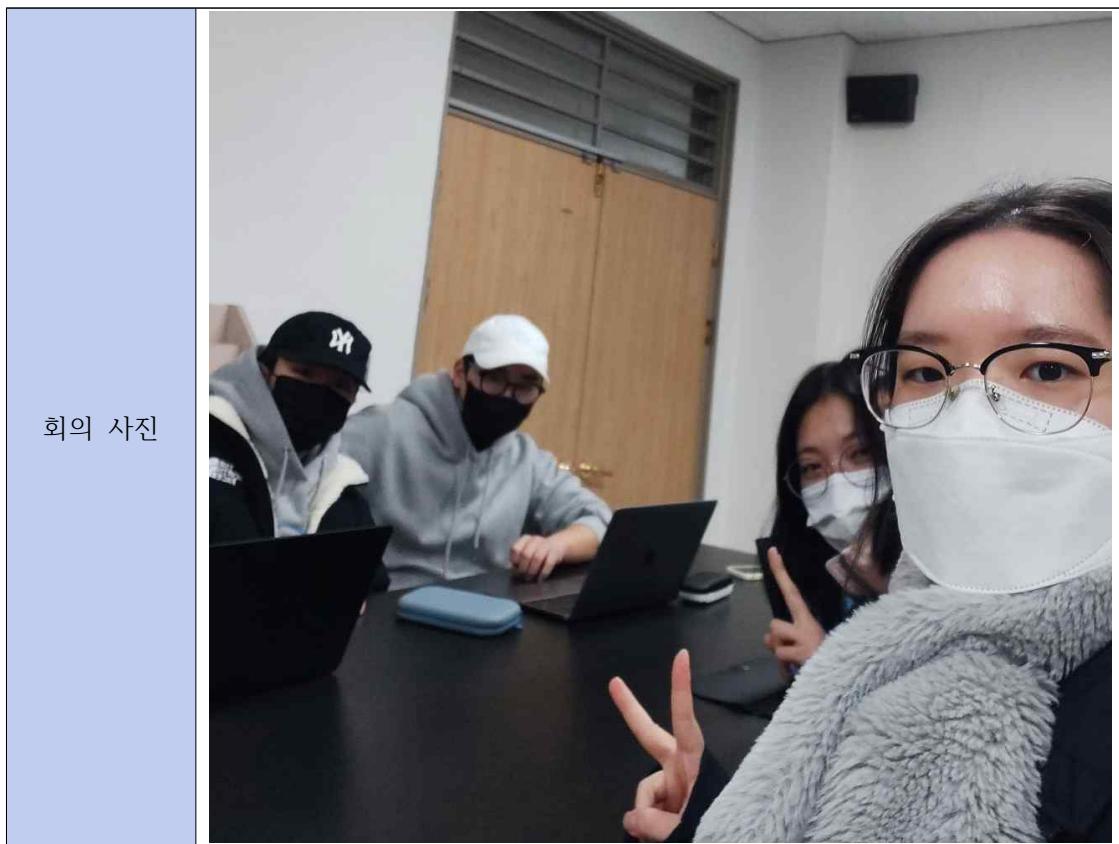
보았다. 그 중 하나인 지출 수정에 관한 내용을 첨부하였다.



2. communication diagram 설정

sequence diagram과 마찬가지로 instance form을 사용하여 diagram을 그려보았다. 그 중 하나인 관광지로 일정추가에 관한 내용을 첨부하였다.

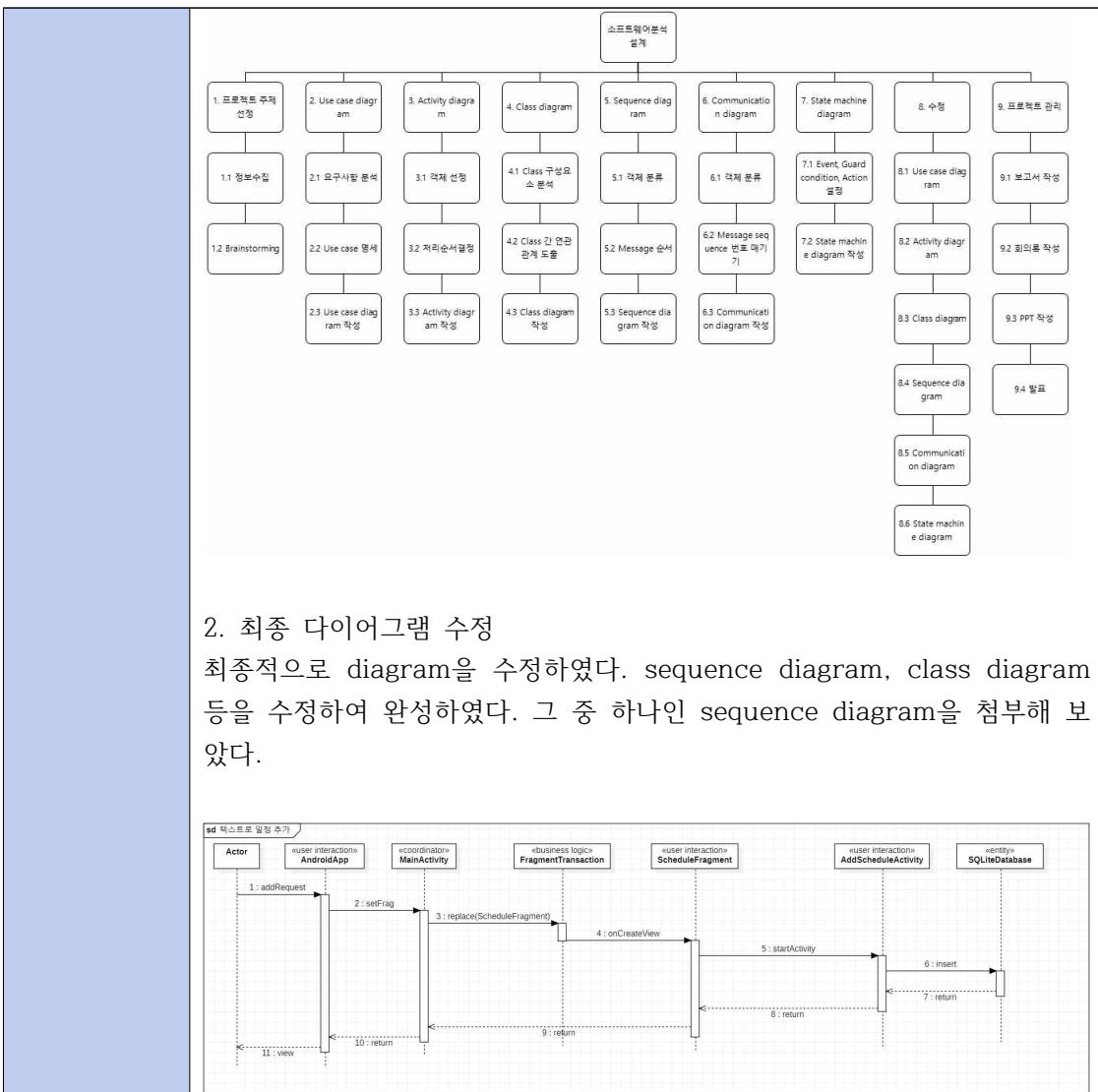


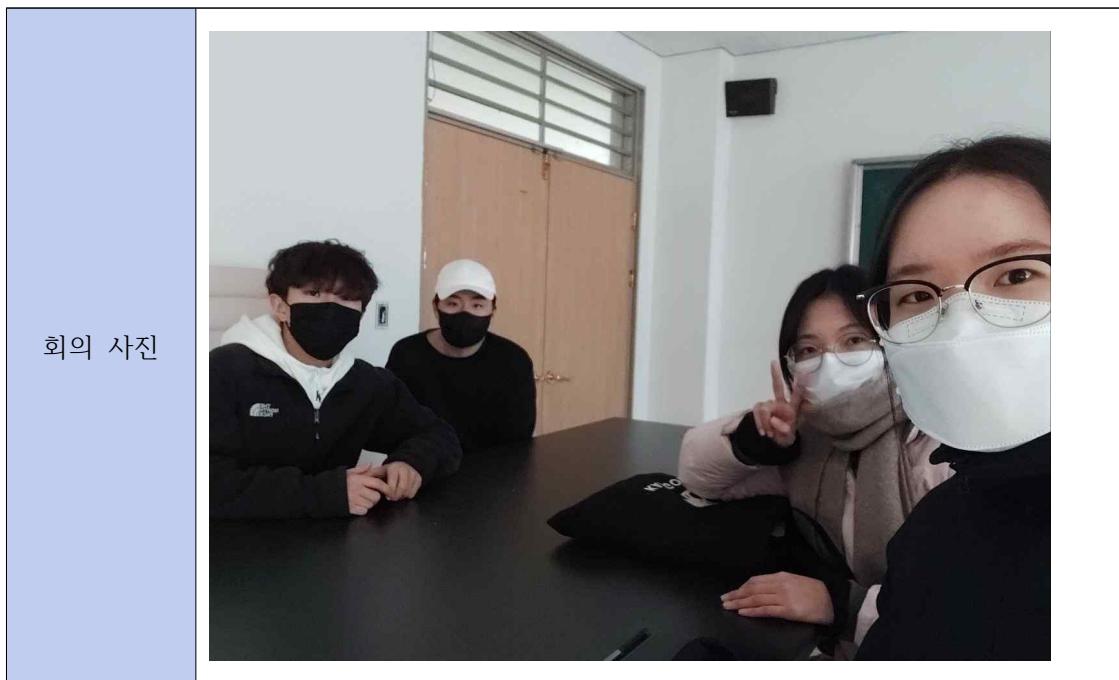


8주차

일 시	2022.12.13.화요일 / 17:00 ~ 18:30	장소	과방
작성자	박지수		
참석자	강연범, 박지수, 백민재, 정세연		

회의 주제	1. 최종 보고서 작성 2. 최종 diagram 수정
회의 내용	<p>1. 최종 보고서 작성</p> <p>과제 제출에 필요한 보고서를 작성하였다. 회의록, 간트 차트, WBS 등을 첨부하여 과제를 어떤식으로 진행했는지 파악할 수 있다. 그 중 하나인 WBS를 첨부해 보았다.</p>





2. 프로젝트 수행 내용 및 결과

2.1 Use case diagram

2.1.1 요구사항 분석

요구사항 번호	요구사항	Actor	Use case
1	User는 여행 일정 및 지출을 관리하기 위해서 여행 관리 시스템을 사용한다.	User	일정 관리, 일정 목록 조회, 일정 상세 보기, 지출 관리, 지출 목록 조회, 지출 상세 보기
2	User는 여행 일정을 추가, 수정, 삭제할 수 있다.	User	일정 추가, 일정 수정, 일정 삭제, 텍스트로 추가하기
3	User는 관광지 검색과 관광지 일정 추가를 할 수 있다.	User, search system	관광지 검색으로 추가하기, 관광지 검색
4	User는 여행 지출을 추가, 수정, 삭제할 수 있다.	User	지출 추가, 지출 수정, 지출 삭제

2.1.2 Use case 명세

UC-001	일정 관리
UC-002	일정 목록 조회
UC-003	일정 상세 보기
UC-004	일정 삭제
UC-005	일정 수정
UC-006	일정 추가
UC-007	텍스트로 추가하기
UC-008	관광지 검색으로 추가하기
UC-009	관광지 검색
UC-010	지출 관리
UC-011	지출 목록 조회
UC-012	지출 수정
UC-013	지출 삭제
UC-014	지출 추가
UC-015	지출 상세 보기

다음과 같이 정의할 수 있다.

유스케이스 이름	일정 관리	
유스케이스 고유번호	UC-001	
액터	주액터: 사용자	
개요	사용자는 여행 일정 목록에 등록된 일정을 관리한다.	
사전 조건	해당 없음	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 여행 일정 메뉴 진입(->)	
대체 흐름	해당 없음	

유스케이스 이름	일정 목록 조회	
유스케이스 고유번호	UC-002	
액터	주액터: 사용자	
개요	사용자는 여행 일정 목록에 등록된 일정을 조회한다.	
사전 조건	해당 없음	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 여행 일정 메뉴 진입(->)	
	2. 사용자에게 추가/수정한 일정 목록을 출력	
대체 흐름	해당 없음	

유스케이스 이름	일정 상세 보기	
유스케이스 고유번호	UC-003	
액터	주액터: 사용자	
개요	사용자는 여행 일정 목록의 일정을 상세 확인한다.	
사전 조건	여행 일정 목록에 해당 일정이 있어야 한다.	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 여행 일정 메뉴 진입(->)	
	2. 상세 일정 정보 요청(->)	
	3. 일정 상세 정보 확인	
	4. 일정의 세부 사항(시간, 날짜, 메모, 위치)을 출력	
대체 흐름	해당 없음	

유스케이스 이름	일정 삭제	
유스케이스 고유번호	UC-004	
액터	주액터: 사용자	
개요	사용자는 여행 일정 목록에 등록된 일정을 삭제한다.	
사전 조건	여행 일정 목록에 해당 일정이 있어야 한다.	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 여행 일정 메뉴 진입(->)	
	2. 삭제 할 여행 일정 길게 눌러 선택(->)	
	3. 사용자에게 삭제 여부 화면을 보여줌	
	4. 확인 버튼을 누름(->)	
	5. 해당 여행 일정을 삭제	
대체 흐름	6. 취소 버튼을 누르면 삭제가 취소되고 여행 일정 목록 화면으로 돌아감	

유스케이스 이름	일정 수정	
유스케이스 고유번호	UC-005	
액터	주액터: 사용자	
개요	사용자는 여행 일정 목록에 등록된 일정을 수정한다.	
사전 조건	여행 일정 목록에 해당 일정이 있어야 한다.	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 여행 일정 메뉴 진입(->)	
	2. 수정 할 여행 일정 선택(->)	
	3. 사용자에게 수정 화면을 보여줌	
	4. 수정 할 내역을 작성(->)	
	5. 수정 버튼을 누름(->)	
	6. 해당 여행 일정이 수정	
대체 흐름	7. 필수 정보 미 입력 시 필수 사항을 입력하라는 메시지 출력 8. 취소 버튼을 누르면 수정이 취소되고 여행 일정 목록 화면으로 돌아감	

유스케이스 이름	일정 추가	
유스케이스 고유번호	UC-006	
액터	주액터: 사용자	
개요	사용자는 여행 일정 목록에 일정을 추가한다.	
사전 조건	해당 없음	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 여행 일정 메뉴 진입(->)	
	2. 플러스(+) 추가 버튼 누름(->)	
	3. 사용자에게 추가 화면을 보여줌	
대체 흐름	해당 없음	

유스케이스 이름	텍스트로 추가하기	
유스케이스 고유번호	UC-007	
액터	주액터: 사용자	
개요	사용자가 텍스트로 여행 일정을 추가한다.	
사전 조건	해당 없음	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 여행 일정 메뉴 진입(->)	
	2. 텍스트로 일정 추가 요청 (->)	
	4. 사용자에게 추가한 일정을 보여줌	
대체 흐름	5. 필수 정보 미 입력 시 필수 사항을 입력하라는 메시지 출력 6. 취소 버튼을 누르면 추가가 취소되고 여행 일정 목록 화면으로 돌아감	

유스케이스 이름	관광지 검색으로 추가하기	
유스케이스 고유번호	UC-008	
액터	주액터: 사용자	
개요	사용자가 관광지 검색으로 여행 일정을 추가한다.	
사전 조건	관광지를 검색한다.	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 관광지 검색 요청(->)	
	2. 검색 결과 출력(->)	
	3. 추가할 관광지 선택 후 추가	
대체 흐름	4. 취소 버튼을 누르면 추가가 취소되고 관광지 검색 결과 화면으로 돌아감	

유스케이스 이름	관광지 검색		
유스케이스 고유번호	UC-009		
액터	주액터: 사용자 부액터: 검색 시스템		
개요	사용자가 관광지를 검색하면 검색 시스템이 그 결과를 불러온다.		
사전 조건	해당 없음		
사후 조건	해당 없음		
기본 흐름	사용자	시스템	검색 시스템
	1. 관광지 검색		
	2. 관광지 검색 요청(->)		
		3. 요청한 관광지 결과 불러오기(->)	
대체 흐름	4. 사용자에게 검색된 관광지 리스트 출력		
	5. 검색 결과가 없으면 안내 메시지 출력		

유스케이스 이름	지출 관리	
유스케이스 고유번호	UC-010	
액터	주액터: 사용자	
개요	사용자의 지출 내역들을 관리할 수 있다.	
사전 조건	해당 없음	
사후 조건	해당 없음	
	사용자	시스템
기본 흐름	1. 여행 지출 메뉴 진입(->)	
대체 흐름	해당 없음	

유스케이스 이름	지출 목록 조회	
유스케이스 고유번호	UC-011	
액터	주액터: 사용자	
개요	사용자의 추가/수정한 지출 목록을 조회할 수 있다.	
사전 조건	해당 없음	
사후 조건	해당 없음	
	사용자	시스템
기본 흐름	1. 여행 지출 메뉴 진입(->)	
	2. 사용자에게 추가/수정한 여행 지출 목록을 보여줌	
대체 흐름	해당 없음	

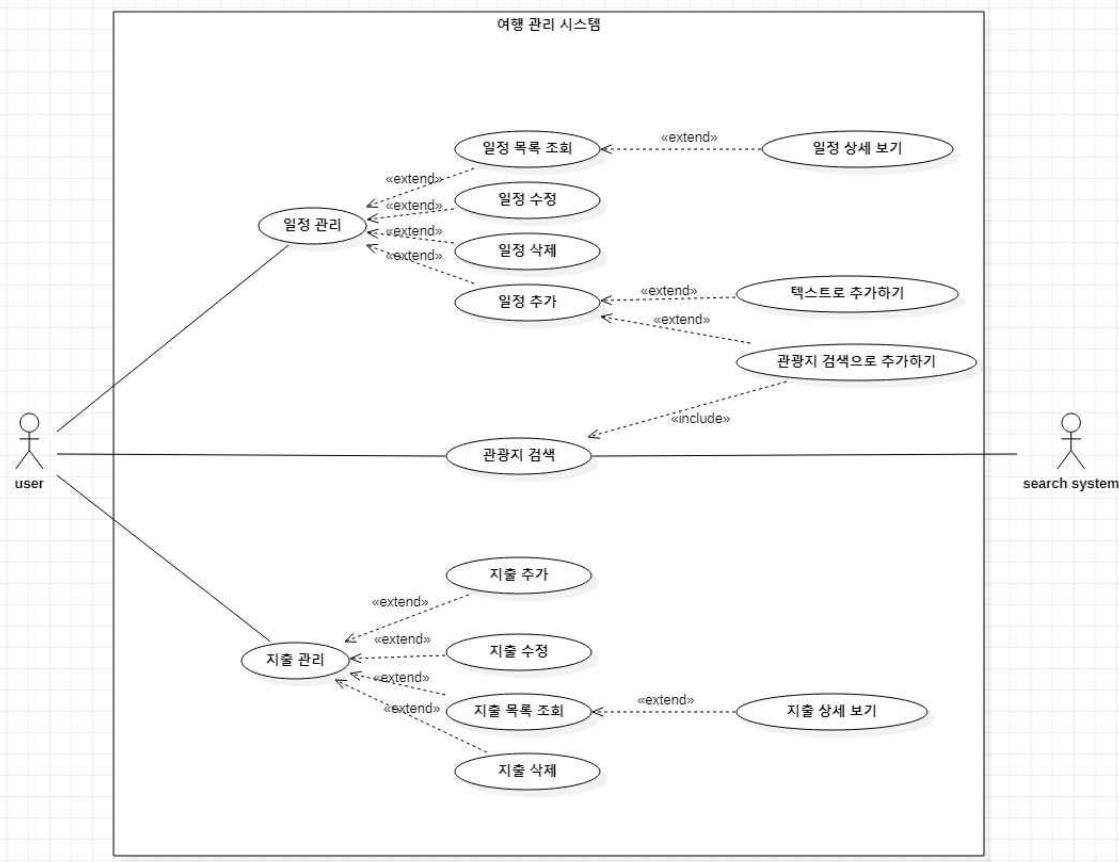
유스케이스 이름	지출 수정	
유스케이스 고유번호	UC-012	
액터	주액터: 사용자	
개요	사용자의 추가한 지출 내역을 수정한다.	
사전 조건	지출 목록에 해당 지출이 있어야 한다.	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 여행 지출 메뉴 진입(->)	
	2. 해당되는 지출 내역 클릭(->)	
	3. 사용자가 선택한 특정 지출 내역을 수정(->)	
	4. 해당 지출 내역 수정	
대체 흐름	5. 필수 정보 미 입력 시 필수 사항을 입력하라는 메시지 출력	
	6. 취소 버튼을 누르면 수정이 취소되고 여행 지출 목록 화면으로 돌아감	

유스케이스 이름	지출 삭제	
유스케이스 고유번호	UC-013	
액터	주액터: 사용자	
개요	사용자는 지출 목록에 등록된 지출을 삭제한다.	
사전 조건	지출 목록에 해당 지출이 있어야 한다.	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 지출 목록 진입(->)	
	2. 삭제 할 지출 내역 선택(->)	
	3. 사용자에게 삭제 확인 화면을 보여줌	
	4. 확인 버튼을 누름 (->)	5. 해당 지출 내역을 삭제
대체 흐름	6. 취소 버튼을 누르면 삭제가 취소되고 여행 지출 목록 화면으로 돌아감	

유스케이스 이름	지출 추가	
유스케이스 고유번호	UC-014	
액터	주액터: 사용자	
개요	사용자는 지출 목록에 지출 내역을 추가한다.	
사전 조건	해당 없음	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 지출 목록 진입(->)	
	2. 지출 추가 요청(->)	
	3. 사용자에게 지출 추가 화면을 보여줌	
	4. 추가 할 내역을 작성(->)	
	5. 확인 버튼을 누름(->)	
	6. 해당 지출 내역을 추가	
대체 흐름	7. 필수 정보 미 입력 시 필수 사항을 입력하라는 메시지를 보여줌 8. 취소 버튼을 누르면 추가가 취소되고 여행 지출 목록 화면으로 돌아감	

유스케이스 이름	지출 상세 보기	
유스케이스 고유번호	UC-015	
액터	주액터: 사용자	
개요	사용자는 지출 목록에 지출 내역을 상세 확인한다.	
사전 조건	지출 내역이 있어야 한다.	
사후 조건	해당 없음	
기본 흐름	사용자	시스템
	1. 지출 목록 진입(->)	
	2. 상세 지출 정보 요청(->)	
	3. 지출 상세 정보 확인	
대체 흐름	4. 사용자에게 지출 상세 정보 화면을 보여줌	

2.1.3 Use case diagram 작성



Use case diagram은 기능적 요구사항을 바탕으로 한 다이어그램이다. 여행 관리 시스템의 use case는 크게 일정관리, 관광지 검색, 지출 관리로 나누어지며 이들은 서로 다른 use case들로 확장된다. 일정 관리는 목록 조회, 추가, 수정, 삭제 기능으로 확장되며 지출 관리도 이와 같다.

Actor는 use case와 의사소통하는 외부 시스템, 사용자 등이 될 수 있다. 여행 관리 시스템에서 actor는 검색 결과를 불러오는 검색 시스템과 사용자이다. 여기서 검색 시스템은 한국관광공사 open API이다.

extend 관계

확장 관계는 선택적인 상호작용을 명시할 때 또는 예외적인 경우를 다룰 때 사용한다. ‘목록조회’와 ‘상세 보기’의 경우 상세 보기가 목록 조회에서 일어날 수 있는 예외적이고 선택적인 기능이므로 확장 관계이다. ‘일정 추가’의 확장 사례로는 ‘텍스트로 추가하기’와 ‘관광지 검색으로 추가하기’가 있다. 이 때 확장은 extend use case에서 base use case 방향이다.

include 관계

포함 관계는 여러 use case가 공통적인 부분이 있을 경우 또는 하나의 use case가 다른 use case의 실행을 전제로 할 때 사용된다. 여행 관리 시스템에서는 후자의 이유로 사용하였다.

다. 관광지 검색으로 일정을 추가하기 위해서는 관광지 주소를 불러와야하고, 관광지 주소를 불러오기 위해서는 관광지 검색 기능을 사용해야 한다. 이 때 화살표는 base use case에서 include use case 방향이다.

2.2 Activity diagram

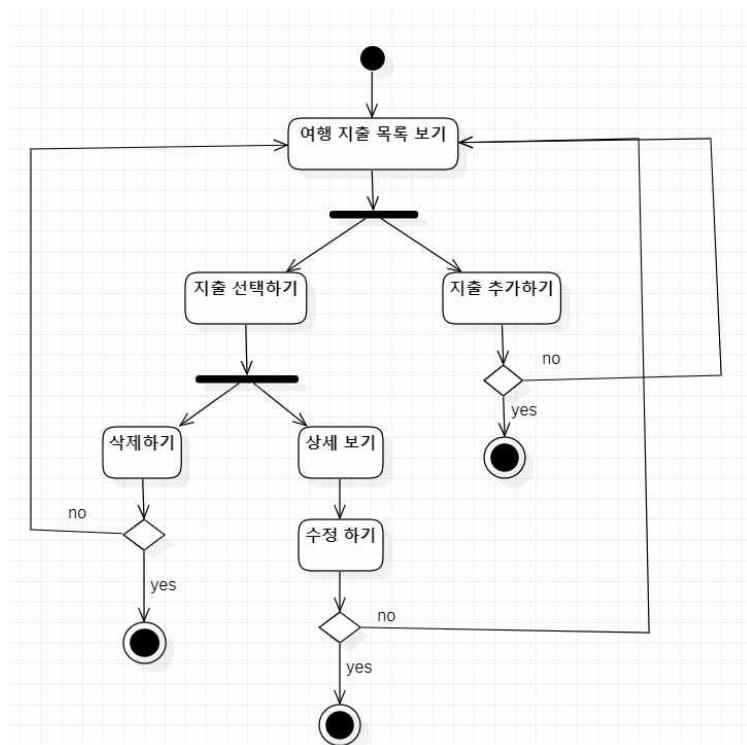
2.2.1 객체 선정 및 처리 순서 결정

Activity diagram은 Use case diagram의 주 흐름과 대체 흐름을 모두 포함하여 기능의 흐름을 나타내는 diagram이다. 따라서 Use case diagram에서 사용한 Use case를 객체로 선정한다.

Use case diagram은 처리 순서가 중요하지 않았지만, Activity diagram은 필수적으로 처리 순서를 결정해야 한다. 따라서 분석 대상인 여행 관리 앱의 실제 흐름에 따라 순서를 정의하였다. 흐름이 분리되어 수행되는 것을 포크 노드로, 사용자의 선택에 따라 흐름이 달라지는 것을 결점 노드로 나타냈다.

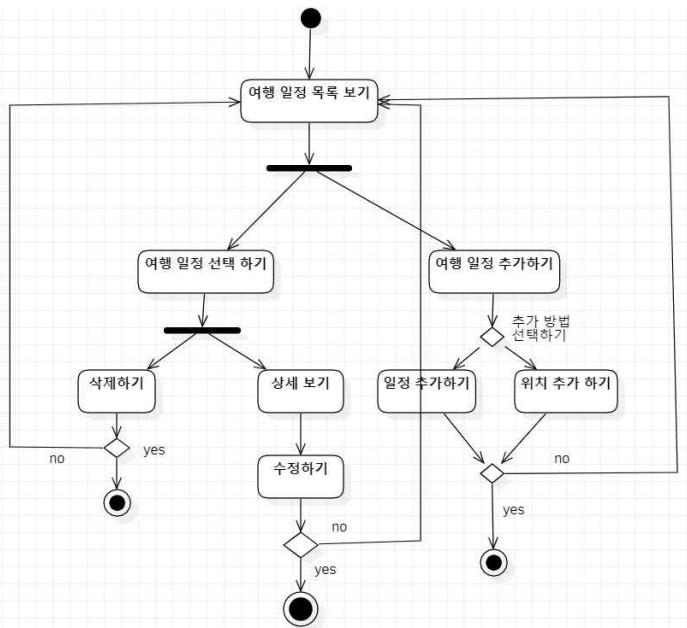
2.2.2 Activity diagram 작성

지출관리 Activity diagram



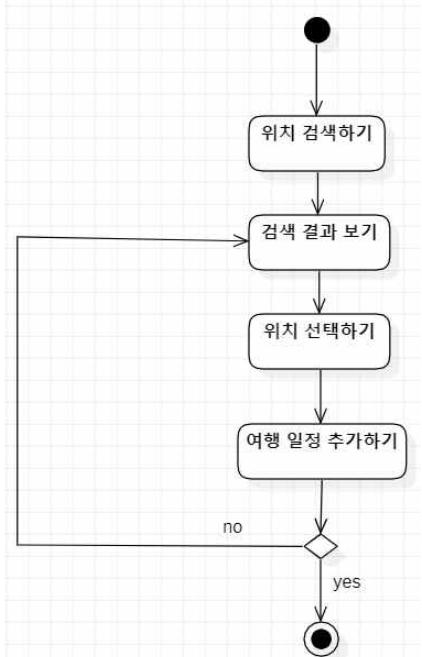
여행 지출 목록 보기에 들어가면 지출 선택하기와 지출 추가하기를 병렬적으로 처리할 수 있다. 이 때 지출 선택하기는 삭제, 상세보기로 흐름이 나눠질 수 있다. 삭제하기, 수정하기, 지출 추가하기는 각각 취소와 확인으로 제어 가능하다.

여행일정 관리하기 Activity diagram



여행 일정 목록 보기에 들어가면 일정 선택하기와 일정 추가하기를 병렬적으로 처리할 수 있다. 이 때 일정 선택하기는 삭제, 상세보기로 흐름이 나눠질 수 있다. 삭제하기, 수정하기는 각각 취소와 확인으로 제어 가능하다. 여행 일정 추가하기는 일정 추가하기와 위치 추가하기로 추가 방법을 선택할 수 있으며 선택 후에 종료한다.

검색 Activity diagram



검색하기 템에서 위치를 검색하면 검색 결과를 볼 수 있다. 이후 관광지 위치 선택 후에 여행 일정을 추가한 뒤 종료한다.

2.3 Class diagram

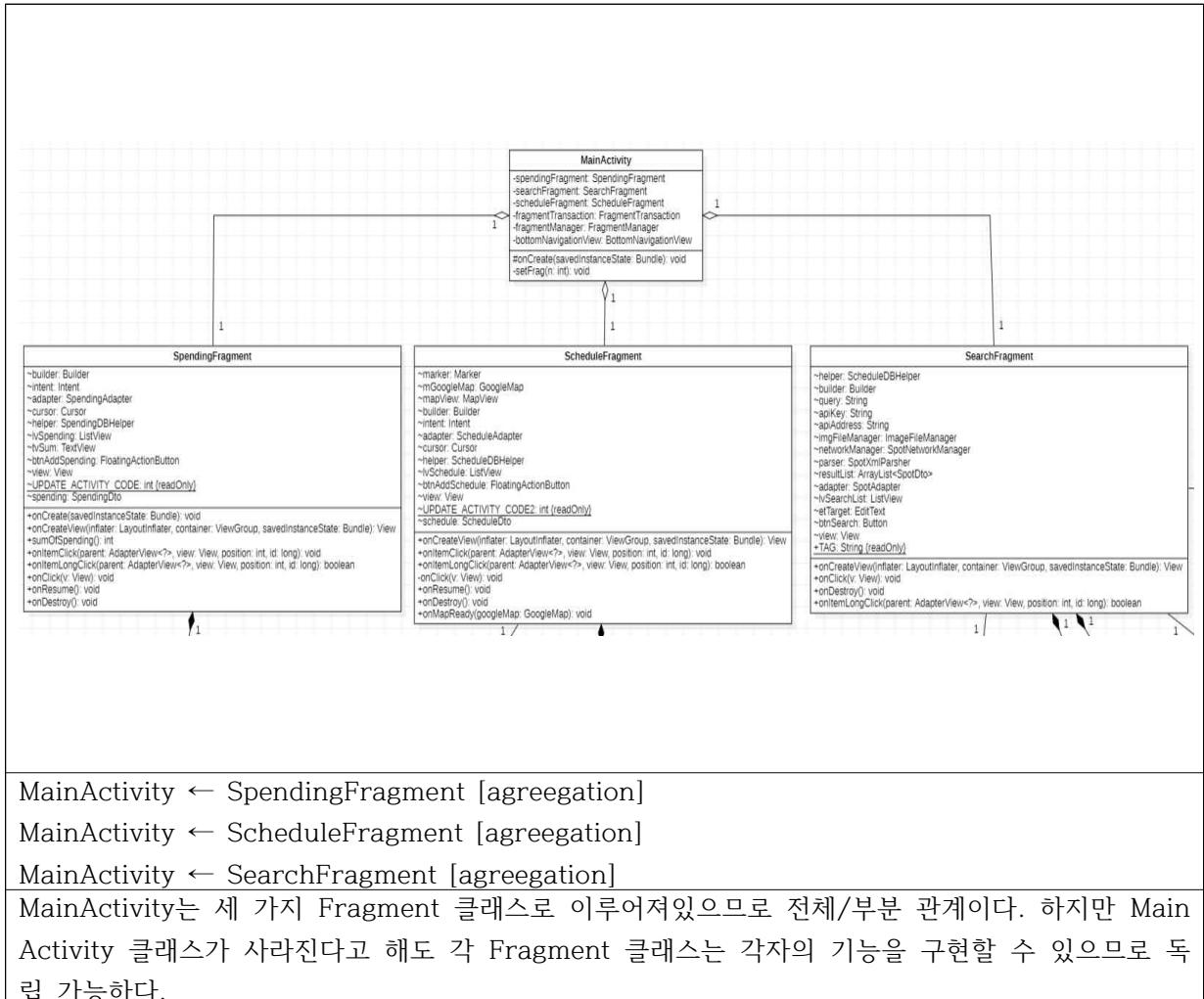
2.3.1 Class 분석

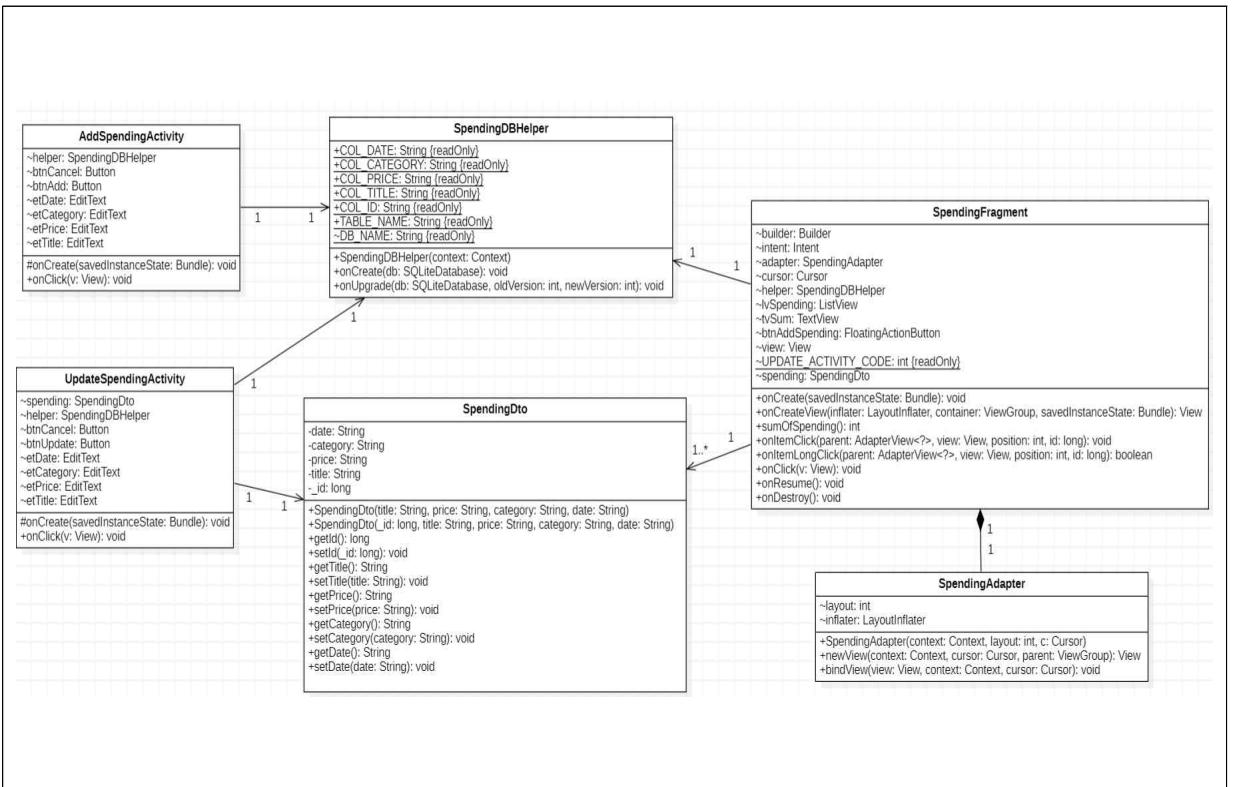
AddScheduleActivity	일정을 추가할 수 있는 클래스이다. title, time, date, memo, address, mapx, mapy를 입력 받아 일정을 추가할 수 있다. 이때, 필수항목인 일정 명을 입력하지 않고 추가 하게 되면 '필수 항목을 입력하지 않았습니다.'라는 메시지가 뜨면서 일정 추가를 할 수 없다. 필수항목을 입력하게 되면 추가가 가능하다.
AddSpendingActivity	지출을 추가할 수 있는 클래스이다. title, price, category, date를 입력 받아 지출을 추가할 수 있다. 이때, 필수항목인 지출 명, 가격, 카테고리를 입력하지 않고 추가 하게 되면 '필수 항목을 입력하지 않았습니다.'라는 메시지가 뜨면서 지출 추가를 할 수 없다. 필수항목을 입력하게 되면 추가가 가능하다.
ImageFileManager	관광지에 알맞은 이미지를 추출하는 클래스이다. Bitmap과 Bitmap 다운로드에 사용한 URL을 전달받아 내부저장소에 JPG 파일로 저장한 후 파일 이름을 반환한다. 파일 저장 실패 시 null을 반환한다. 파일 저장 성공 시 내부 저장소에 파일을

	생성하여 파일용 output 스트림을 생성하고 비트맵 이미지를 파일에 기록한다. 이때, Bitmap의 크기가 클 경우 조정을 한다.
MainActivity	메인 클래스이다. scheduleFragment, searchFragment, spendingFragment로 구성되며, 처음에는 setFrag(0)으로 첫 fragment 화면을 지정한다. fragment를 교체하여 원하는 메뉴로 들어갈 수 있다.
ScheduleAdapter	일정화면과 일정에 넣을 데이터를 연결하는 클래스이다. 일정의 원본을 받아 관리하고, Adapter view가 출력할 수 있는 형태로 데이터를 제공하는 중간 객체 역할을 한다.
ScheduleDBhelper	ScheduleDBhelper 클래스를 생성하여 다양한 DB 명령어를 수행하는 메서드를 추가해준다. 해당 명령어를 호출하여 로컬 DB에 일정 데이터 테이블을 생성(onCreate()), 저장(onUpgrade()) 할 수 있다.
ScheduleDto	ScheduleDto 클래스는 데이터의 전송을 담당하는 클래스이다. 즉 저장된 데이터베이스 각각의 일정 데이터들을 반환하는 역할을 하게 된다.
ScheduleFragment	ScheduleFragment 클래스는 일정관리 fragment를 위한 클래스로, ScheduleDBhelper 클래스의 객체를 가지게 된다. ScheduleDBhelper 클래스에게 데이터베이스를 요청하게 되면 ScheduleDto 클래스를 이용해서 dto를 생성하여 일정 관련 데이터베이스를 가져오게 된다. 해당 클래스에는 선택하는 onClick(), 계속 진행하는 onResume(), 삭제하는 onDestroy() 등이 있다.
SearchFragment	ScheduleFragment 클래스는 검색 기능 fragment를 위한 클래스이다. 검색 내용을 보기위한 onCreateView 함수, onClick 함수를 이용해서 openApi주소와 query 조합 후에 서버에서 데이터를 가져오고 가져온 데이터는 어댑터에 설정하게 된다.
SpendingAdapter	지출 화면과 지출에 넣을 데이터를 연결하는 클래스이다. 지출의 원본을 받아 관리하고, Adapter view가 출력할 수 있는 형태로 데이터를 제공하는 중간 객체 역할을 한다.
SpendingDBhelper	SpendingDBhelper 클래스를 생성하여 다양한 DB 명령어를 수행하는 메서드를 추가해준다 해당 명령어를 호출하여 로컬 DB에 지출 데이터를 생성(onCreate()), 저장(onUpgrade()) 할 수 있다.
SpendingDto	SpendingDto 클래스는 데이터의 전송을 담당하는 클래스이다. 즉 저장된 데이터베이스 각각의 일정 데이터들을 반환하는 역할을 하게 된다.
SpendingFragment	SpendingFragment 클래스는 지출관리 fragment를 위한 클래스로, SpendingDBhelper 클래스의 객체를 가지게 된다. SpendingDBhelper 클래스에게 데이터베이스를 요청하게 되면 SpendingDto 클래스를 이용해서 dto를 생성하여 지출관련

	데이터베이스를 가져오게 된다. 해당 클래스에는 클릭하는 onClick(), 계속 진행하는 onResume(), 삭제하는 onDestroy(), 버튼을 길게 눌렸을 때 삭제 유무를 물어보는 onItemLongClick()이 존재한다.
SpotAdapter	관광지 화면과 관광지에 넣을 데이터를 연결하는 클래스이다. 관광지의 원본을 받아 관리하고, Adapter view가 출력할 수 있는 형태로 데이터를 제공하는 중간 객체 역할을 한다.
SpotDto	SpotDto 클래스는 데이터의 전송을 담당하는 클래스이다. 즉 저장된 데이터베이스 각각의 지도정보를 반환하는 역할을 하게된다.
SpotNetworkManager	map 정보를 추출하는 클래스이다. 설정된 주소에 접속하여 문자열 데이터를 수신한 후 반환하는 downloadContents(), 주소를 전달받아 bitmap을 다운로드 후 반환하는 downloadImage(), InputStream을 전달받아 비트맵으로 변환 후 반환하는 readStreamToBitmap(), InputStream을 전달받아 문자열로 변환 후 반환하는 readStreamToString(), 네트워크 환경 조사하는 isOnline(), URLConnection을 전달받아 연결정보 설정 후 연결, 연결 후 수신한 InputStream을 반환하는 getNetworkConnection() 클래스들이 있다.
SpotXmlParsher	xmlparser는 xml 문서의 plain text 데이터를 읽어들여, 그것을 xml dom 객체로 parsing 한다. 또한 xml 문서가 적합한 형식을 갖추고 있는지와 문법상의 오류는 없는지를 검사하는 클래스이다.
UpdateScheduleActivity	dto에서 정보를 받아 값을 수정하는 클래스이다. 일정관련 정보를 다룬다.
UpdateSpendingActivity	dto에서 정보를 받아 값을 수정하는 클래스이다. 지출관련 정보를 다룬다.

2.3.2 Class 간 연관관계 도출





SpendingFragment ← SpendingAdapter [Composition]

SpendingFragment → SpendingDto [Directed Association] 1...*

SpendingFragment → SpendingDBHelper [Directed Association]

AddSpendingActivity → SpendingDBHelper [Directed Association]

UpdateSpendingActivity → SpendingDBHelper [Directed Association]

UpdateSpendingActivity → SpendingDto [Directed Association]

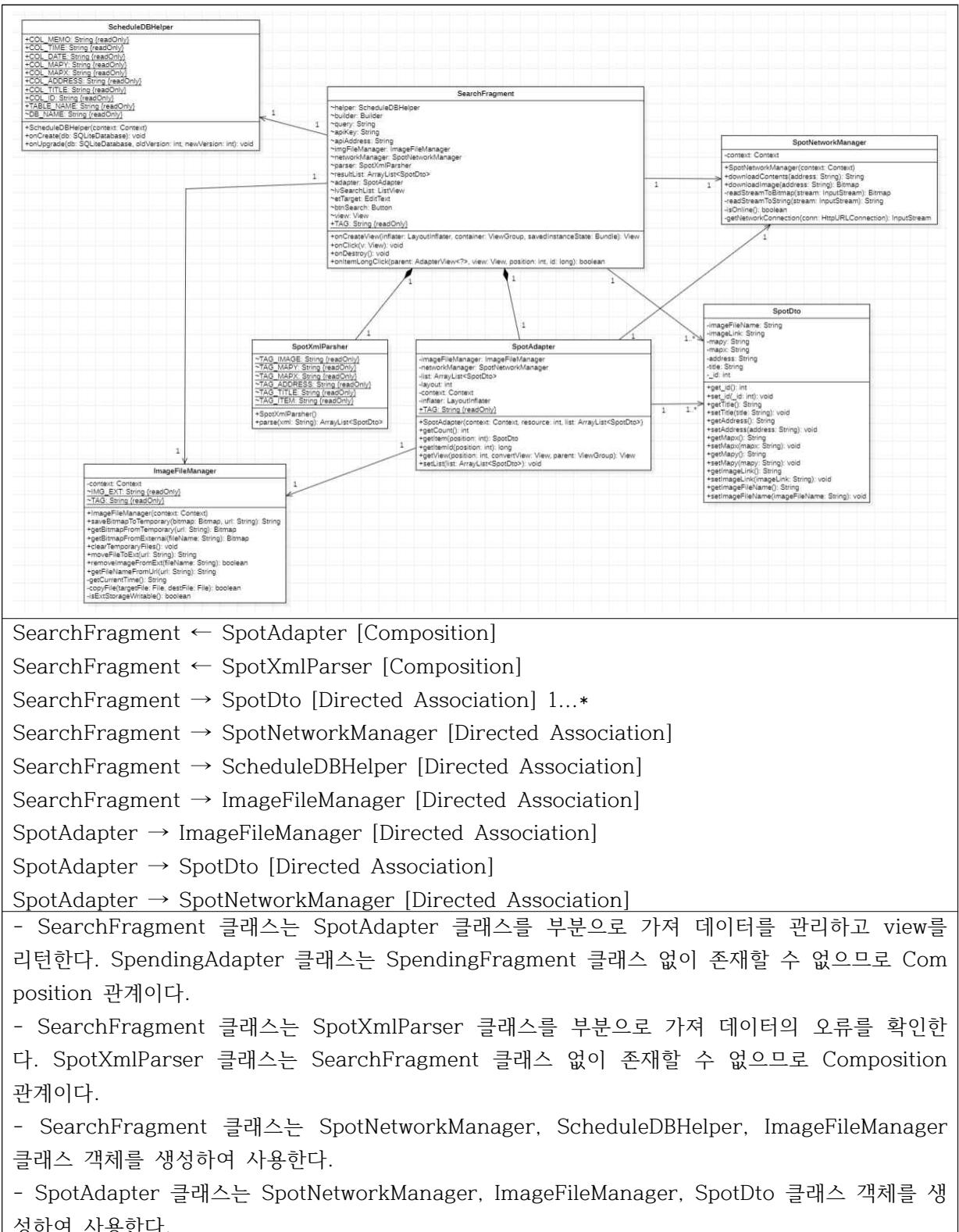
- SpendingFragment 클래스는 SpendingAdapter 클래스를 부분으로 가져 데이터를 관리하고 view를 리턴한다. SpendingAdapter 클래스는 SpendingFragment 클래스 없이 존재할 수 없으므로 Composition 관계이다.

- SpendingFragment 클래스는 SpendingDto, SpendingDBHelper 객체를 생성하여 사용한다.

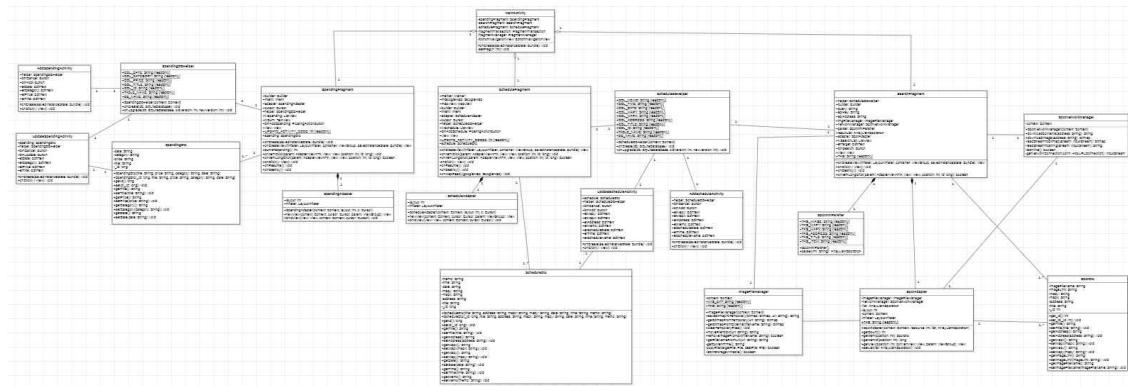
- AddSpendingActivity 클래스는 SpendingDBHelper 객체를 생성하여 사용한다.

- UpdateSpendingActivity 클래스는 SpendingDBHelper, SpendingDto 클래스 객체를 생성하여 사용한다.





2.3.3 Class diagram 작성



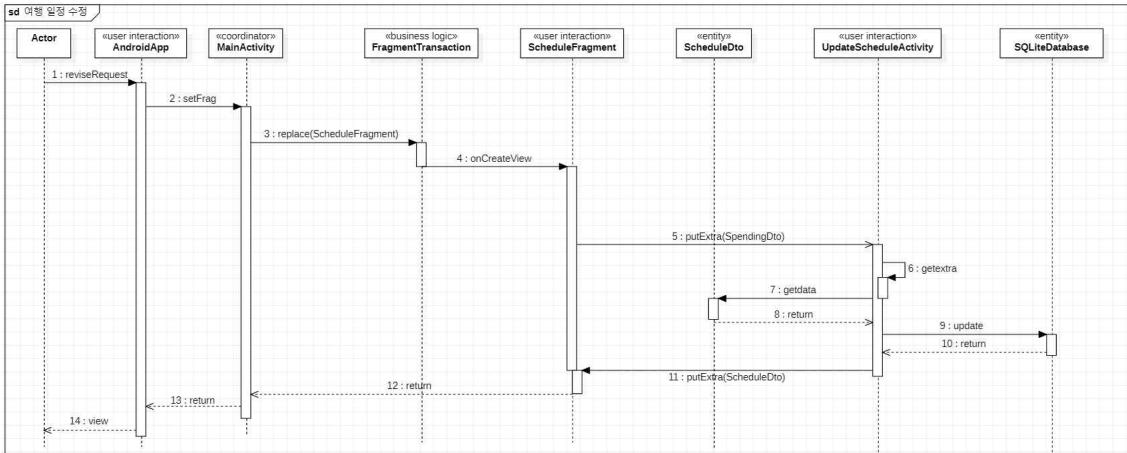
2.4 Sequence diagram

2.4.1 객체 분류

user interaction	AndroidApp
	SpendingFragment
	ScheduleFragment
	SearchFragment
	AddSpendingActivity
	AddScheduleActivity
	UpdateSpendingActivity
	UpdateScheduleActivity
	SpendingAdapter
	ScheduleAdapter
proxy	SpotNetworkManager
	SpotXmlParsher
entity	SpendingDto
	SQLiteDatabase
	ScheduleDto
coordinator	MainActivity
business logic	FragmentTransaction

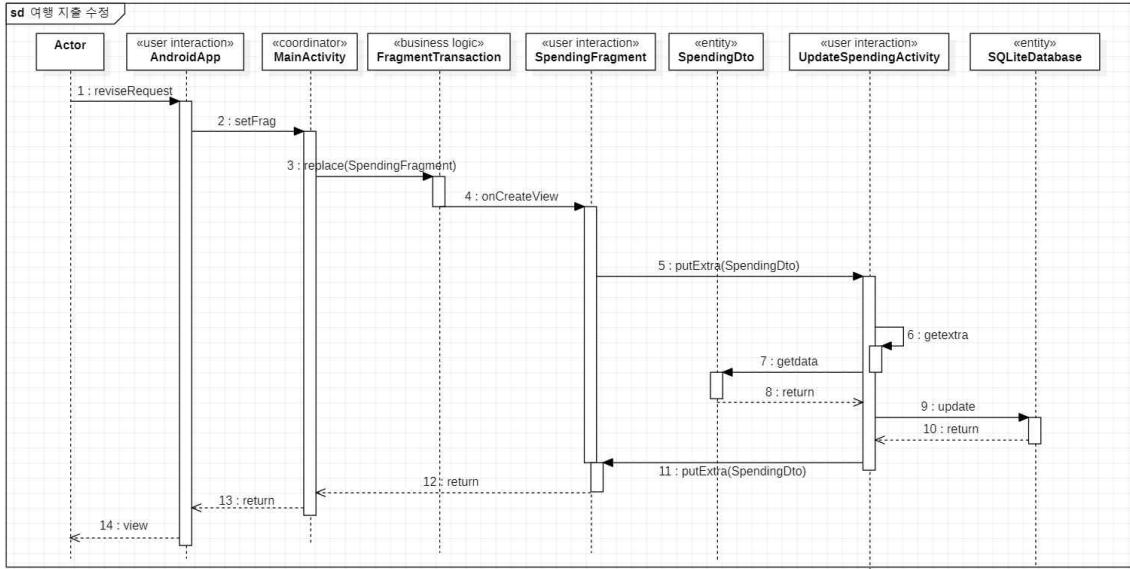
2.4.2 Sequence diagram 작성

여행 일정 수정



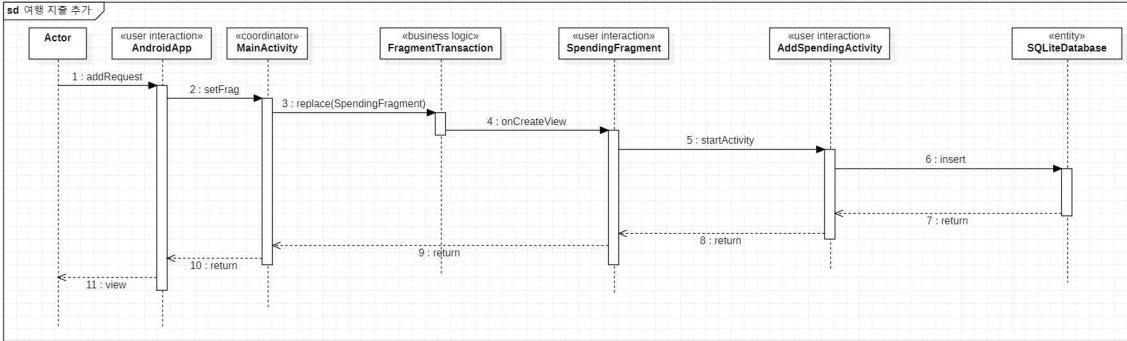
1. reviseRequest : 사용자는 안드로이드 앱을 통해 수정 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(ScheduleFragment) : MainActivity는 FragmentTransaction에게 ScheduleFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 ScheduleFragment를 실행시킨다.
5. putExtra(SpendingDto) : ScheduleFragment는 intent 객체를 통해 UpdateScheduleActivity에게 Dto 객체를 전달한다.
6. getextra : UpdateScheduleActivity는 intent 객체를 통해 Dto 객체를 전달 받는다.
7. getdata : Dto 객체를 통해 각 요소들의 수정 사항을 입력 받는다.
8. return : Dto가 수정 사항을 반환한다.
9. update: SQLiteDatabase에게 수정 사항을 업데이트 시킨다.
10. return
11. putExtra(ScheduleDto) : 수정된 Dto 객체를 intent 객체에 저장하여 ScheduleFragment에게 보낸다.
- 12 ~ 13 : return
14. view : 안드로이드 앱이 사용자에게 수정 사항을 보여준다.

여행 지출 수정



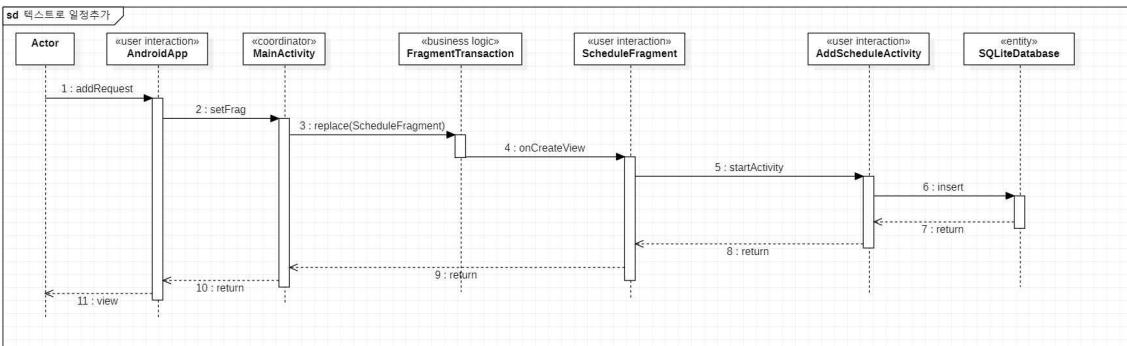
1. reviseRequest : 사용자는 안드로이드 앱을 통해 수정 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SpendingFragment) : MainActivity는 FragmentTransaction에게 SpendingFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 SpendingFragment를 실행시킨다.
5. putExtra(SpendingDto) : SpendingFragment는 intent 객체를 통해 UpdateSpendingActivity에게 Dto 객체를 전달한다.
6. getextra : UpdateSpendingActivity는 intent 객체를 통해 Dto 객체를 전달 받는다.
7. getdata : Dto 객체를 통해 각 요소들의 수정 사항을 입력 받는다.
8. return : Dto가 수정 사항을 반환한다.
9. update : SQLiteDatabase에게 수정 사항을 업데이트 시킨다.
10. return
11. putExtra(SpendingDto) : 수정된 Dto 객체를 intent 객체에 저장하여 SpendingFragment에게 보낸다.
- 12 ~ 13 : return
14. view : 안드로이드 앱이 사용자에게 수정 사항을 보여준다.

여행 지출 추가



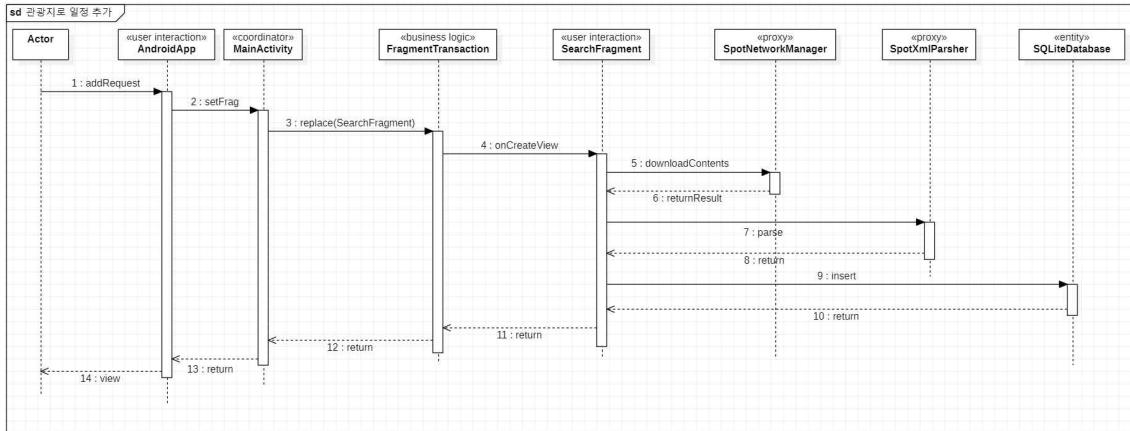
1. addRequest : 사용자는 안드로이드 앱을 통해 추가 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SpendingFragment) : MainActivity는 FragmentTransaction에게 SpendingFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 SpendingFragment를 실행시킨다.
5. startActivityForResult : SpendingFragment는 AddSpendingActivity를 실행시킨다.
6. insert : AddSpendingActivity는 SQLiteDatabase에게 데이터를 삽입한다.
- 7 ~ 10 : return
11. view : 안드로이드 앱이 사용자에게 추가 사항을 보여준다.

텍스트로 일정 추가

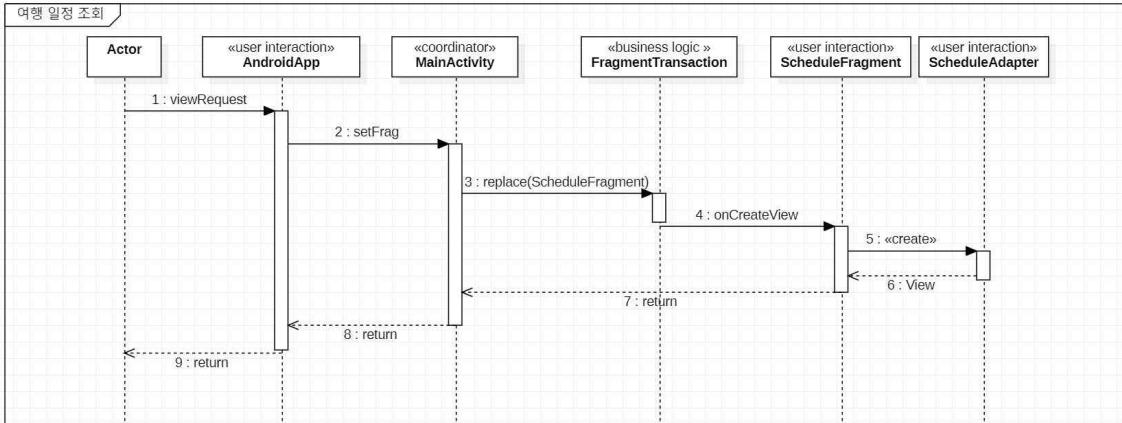


1. addRequest : 사용자는 안드로이드 앱을 통해 추가 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(ScheduleFragment) : MainActivity는 FragmentTransaction에게 ScheduleFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 ScheduleFragment를 실행시킨다.
5. startActivityForResult : ScheduleFragment는 AddScheduleActivity를 실행시킨다.
6. insert : AddScheduleActivity는 SQLiteDatabase에게 데이터를 삽입한다.
- 7 ~ 10 : return
11. view : 안드로이드 앱이 사용자에게 추가 사항을 보여준다.

관광지로 일정 추가

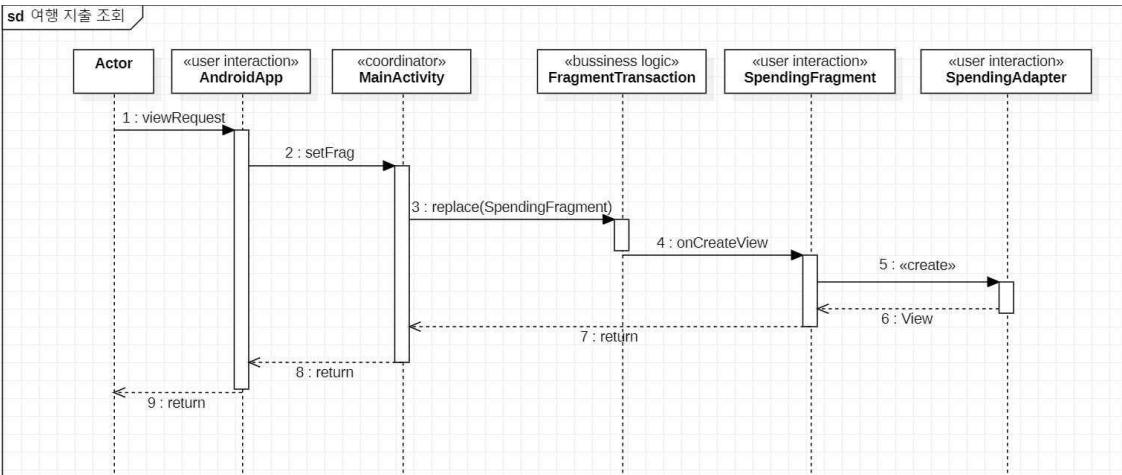


여행 일정 조회



1. ViewRequest : 사용자는 안드로이드 앱을 통해 조회 버튼을 눌러 조회를 요청한다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(ScheduleFragment) : MainActivity는 FragmentTransaction에게 ScheduleFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 ScheduleFragment를 실행시킨다.
5. «create» : ScheduleFragment는 ScheduleAdapter를 생성하여 조회할 정보를 저장한다.
6. view : ScheduleAdapter가 ScheduleFragment에게 조회할 정보를 반환 한다.
- 7 ~ 8 : return
9. return : 사용자에게 원하는 항목을 보여준다.

여행 지출 조회

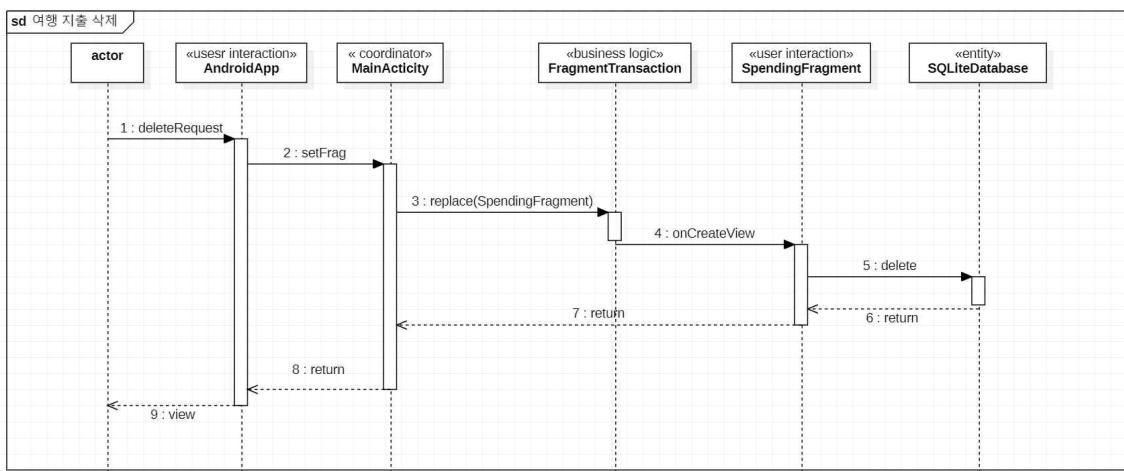


1. viewRequest : 사용자는 안드로이드 앱을 통해 조회 버튼을 눌러 조회를 요청한다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SpendingFragment) : MainActivity는 FragmentTransaction에게 SpendingFr

gment를 호출시킨다.

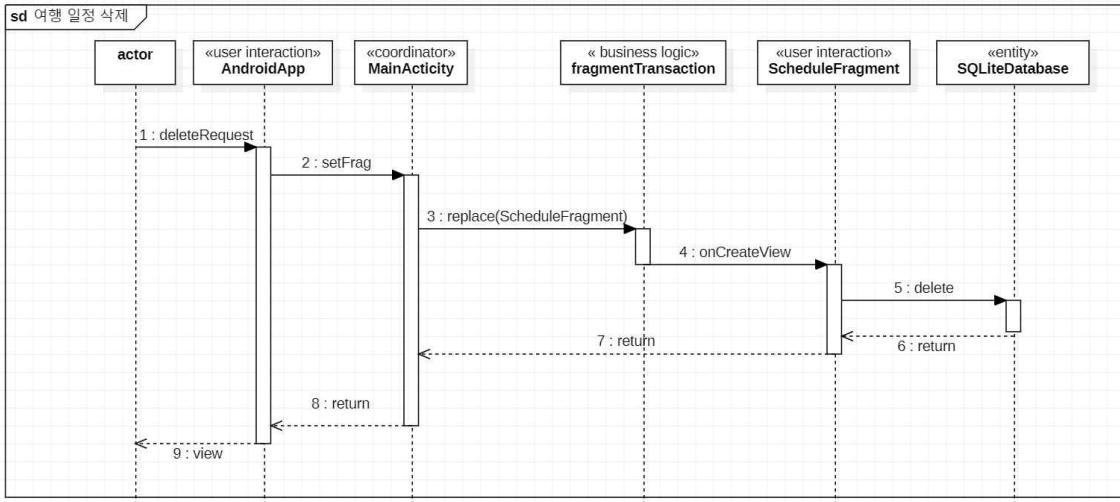
4. onCreateView : FragmentTransaction은 SpendingFragment를 실행시킨다.
5. <<create>> : SpendingFragment는 SpendingAdapter를 생성하여 조회할 정보를 저장한다.
6. view : SpendingAdapter가 SpendingFragment에게 조회할 정보를 반환 한다.
- 7 ~ 8 : Return
9. return : 사용자에게 원하는 항목을 보여준다.

여행 지출 삭제



1. deleteRequest : 사용자는 안드로이드 앱을 통해 삭제 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SpendingFragment) : MainActivity는 FragmentTransaction에게 SpendingFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 SpendingFragment를 실행시킨다.
5. delete : SQLiteDatabase에게 해당 정보 삭제를 요청한다.
- 6 ~ 8 : return
9. view : 안드로이드 앱이 사용자에게 수정된 사항을 보여준다.

여행 일정 삭제



1. deleteRequest : 사용자는 안드로이드 앱을 통해 삭제 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(ScheduleFragment) : MainActivity는 FragmentTransaction에게 ScheduleFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 ScheduleFragment를 실행시킨다.
5. delete : SQLiteDatabase에게 해당 정보 삭제를 요청한다.
- 6 ~ 8 : return
9. view : 안드로이드 앱이 사용자에게 수정된 사항을 보여준다.

2.5 Communication diagram

2.5.1 객체 분류

user interaction	AndroidApp
	SpendingFragment
	ScheduleFragment
	SearchFragment
	AddSpendingActivity
	AddScheduleActivity
	UpdateSpendingActivity
	UpdateScheduleActivity
	SpendingAdapter
	ScheduleAdapter
proxy	SpotNetworkManager
	SpotXmlParsher
entity	SpendingDto
	SQLiteDatabase
	ScheduleDto
coordinator	MainActivity
business logic	FragmentTransaction

2.5.2 Communication diagram 작성

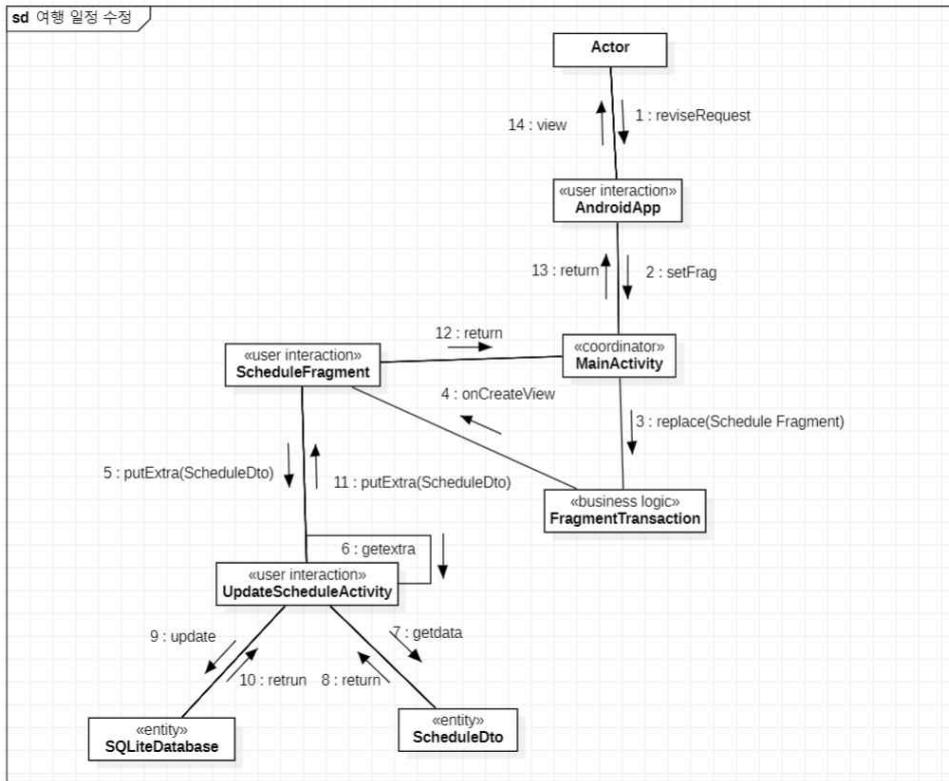
객체 선정

Communication diagram은 Use case diagram에서 이미 정의된 시스템과 액터 사이에 메시지를 표현한다. 여행 관리 앱의 경우 시스템의 종류가 많지 않아 각 기능에 참여하는 클래스로 시스템을 대체하였다.

객체 순서 결정

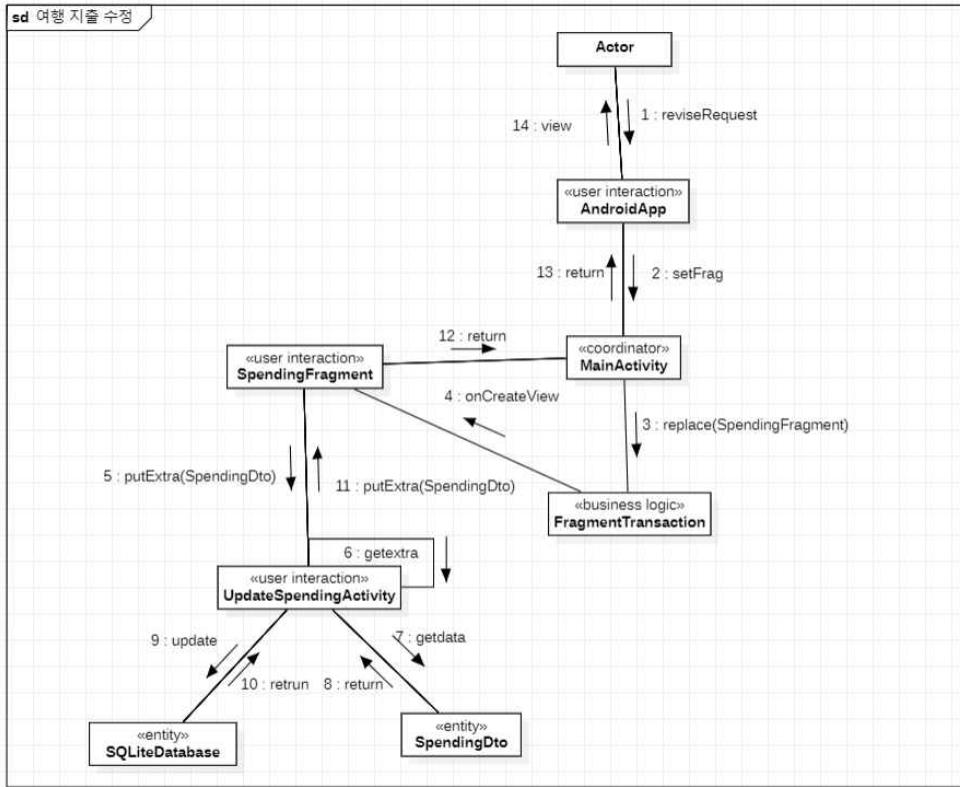
Communication diagram은 Sequence diagram을 바탕으로 그렸으며 각 메시지의 순서는 Sequence diagram과 같다. 객체는 메시지 순서대로 나열하였으며 각 객체의 상호작용이 잘 보이도록 자리를 설정하였다.

여행 일정 수정



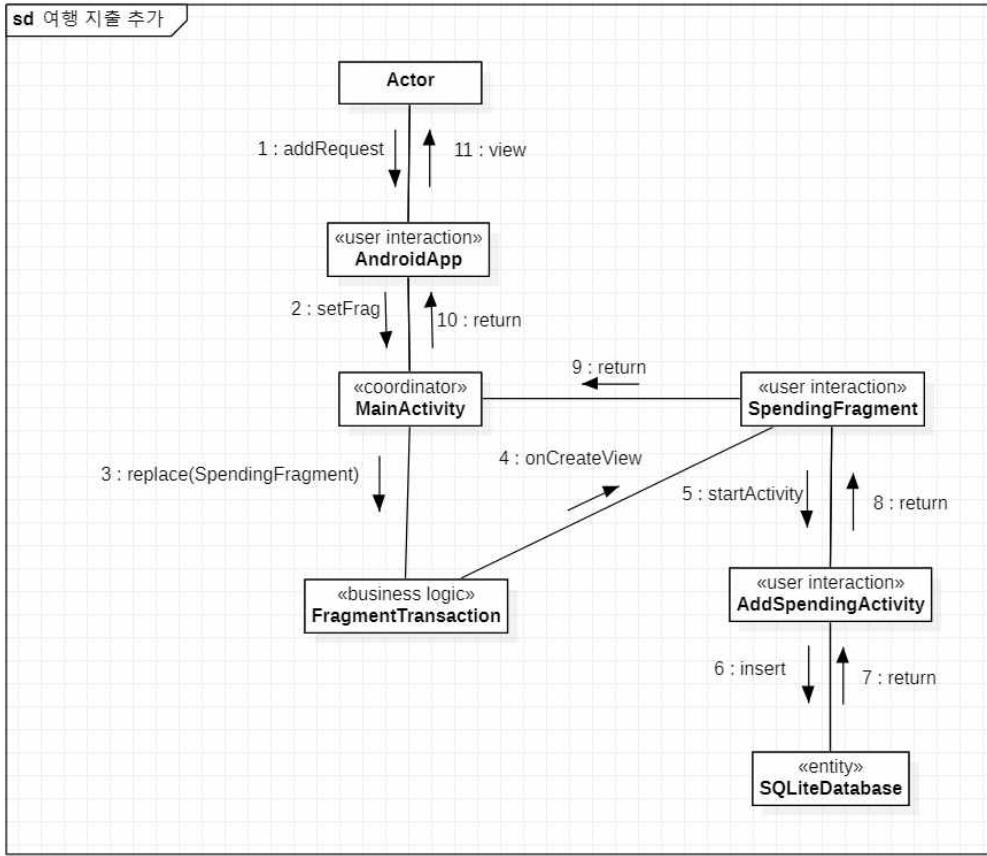
1. reviseRequest : 사용자는 안드로이드 앱을 통해 수정 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(ScheduleFragment) : MainActivity는 FragmentTransaction에게 ScheduleFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 ScheduleFragment를 실행시킨다.
5. putExtra(SpendingDto) : ScheduleFragment는 intent 객체를 통해 UpdateScheduleActivity에게 Dto 객체를 전달한다.
6. getextra : UpdateScheduleActivity는 intent 객체를 통해 Dto 객체를 전달 받는다.
7. getdata : Dto 객체를 통해 각 요소들의 수정 사항을 입력 받는다.
8. return : Dto가 수정 사항을 반환한다.
9. update: SQLiteDatabase에게 수정 사항을 업데이트 시킨다.
10. return
11. putExtra(ScheduleDto) : 수정된 Dto 객체를 intent 객체에 저장하여 ScheduleFragment에게 보낸다.
- 12 ~ 13 : return
14. view : 안드로이드 앱이 사용자에게 수정 사항을 보여준다.

여행 지출 수정



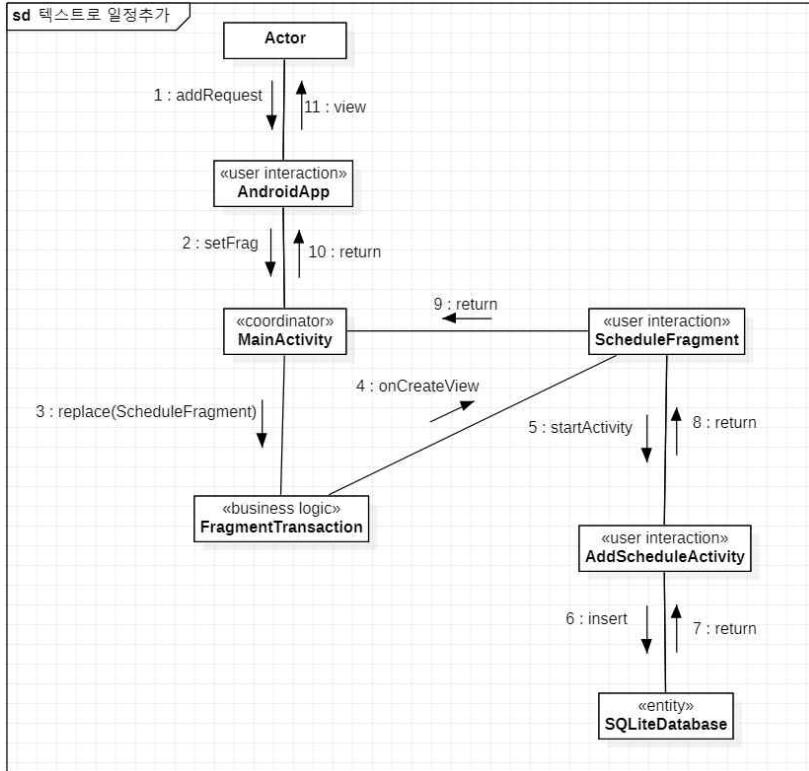
1. reviseRequest : 사용자는 안드로이드 앱을 통해 수정 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SpendingFragment) : MainActivity는 FragmentTransaction에게 SpendingFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 SpendingFragment를 실행시킨다.
5. putExtra(SpendingDto) : SpendingFragment는 intent 객체를 통해 UpdateSpendingActivity에게 Dto 객체를 전달한다.
6. getextra : UpdateSpendingActivity는 intent 객체를 통해 Dto 객체를 전달 받는다.
7. getdata : Dto 객체를 통해 각 요소들의 수정 사항을 입력 받는다.
8. return : Dto가 수정 사항을 반환한다.
9. update : SQLiteDatabase에게 수정 사항을 업데이트 시킨다.
10. return
11. putExtra(SpendingDto) : 수정된 Dto 객체를 intent 객체에 저장하여 SpendingFragment에게 보낸다.
- 12 ~ 13 : return
14. view : 안드로이드 앱이 사용자에게 수정 사항을 보여준다.

여행 지출 추가



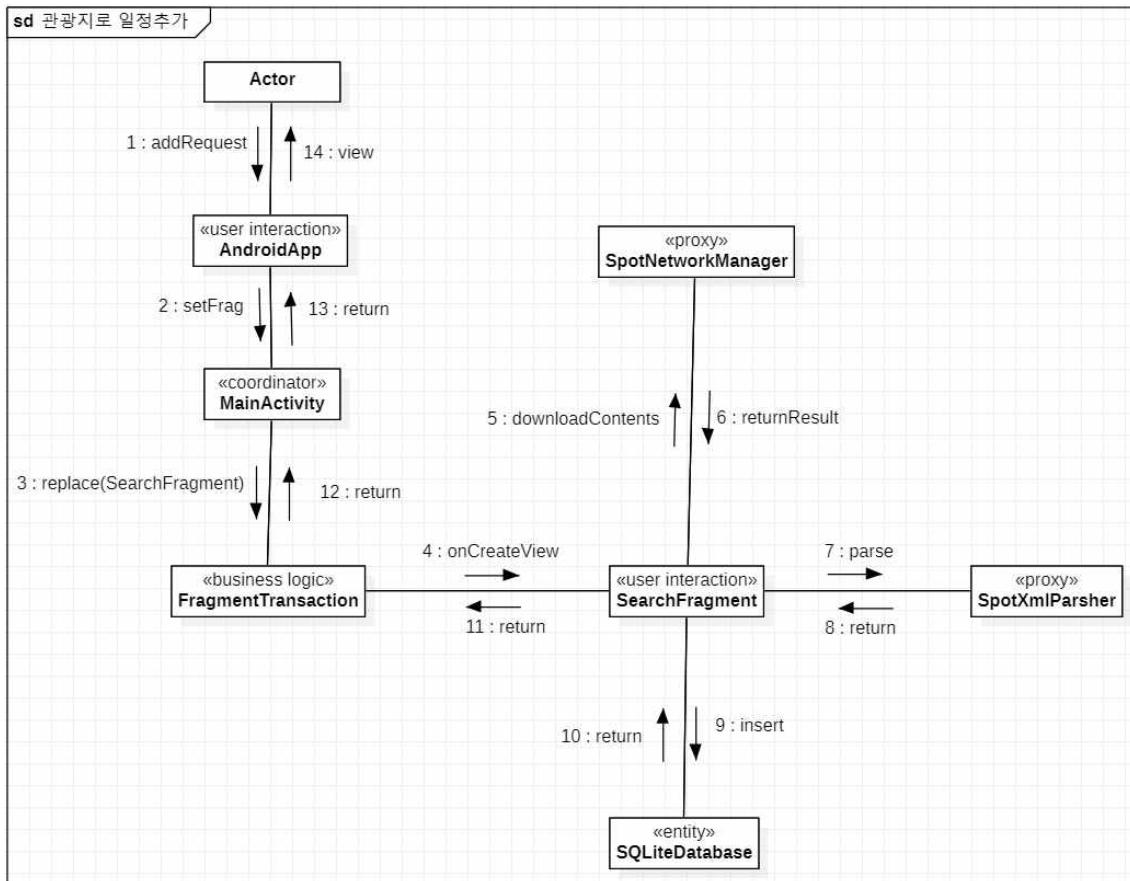
1. addRequest : 사용자는 안드로이드 앱을 통해 추가 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SpendingFragment) : MainActivity는 FragmentTransaction에게 SpendingFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 SpendingFragment를 실행시킨다.
5. startActivityForResult : SpendingFragment는 AddSpendingActivity를 실행시킨다.
6. insert : AddSpendingActivity는 SQLiteDatabase에게 데이터를 삽입한다.
- 7 ~ 10 : return
11. view : 안드로이드 앱이 사용자에게 추가 사항을 보여준다.

텍스트로 일정 추가



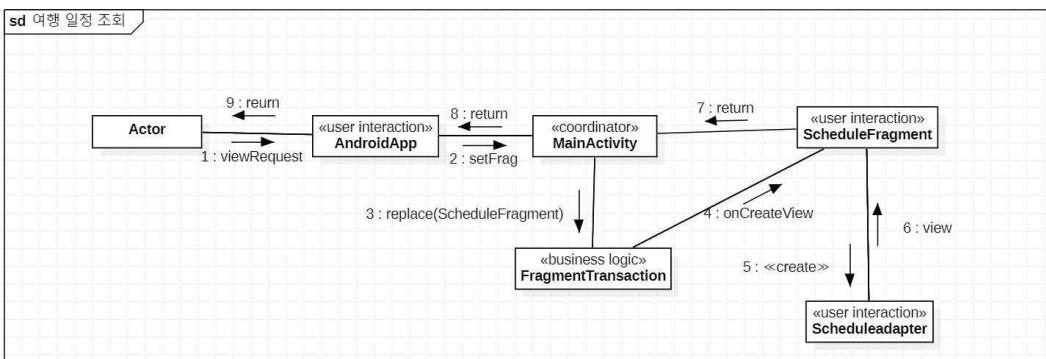
1. addRequest : 사용자는 안드로이드 앱을 통해 추가 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(ScheduleFragment) : MainActivity는 FragmentTransaction에게 ScheduleFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 ScheduleFragment를 실행시킨다.
5. startActivityForResult : ScheduleFragment는 AddScheduleActivity를 실행시킨다.
6. insert : AddScheduleActivity는 SQLiteDatabase에게 데이터를 삽입한다.
- 7 ~ 10 : return
11. view : 안드로이드 앱이 사용자에게 추가 사항을 보여준다.

관광지로 일정 추가



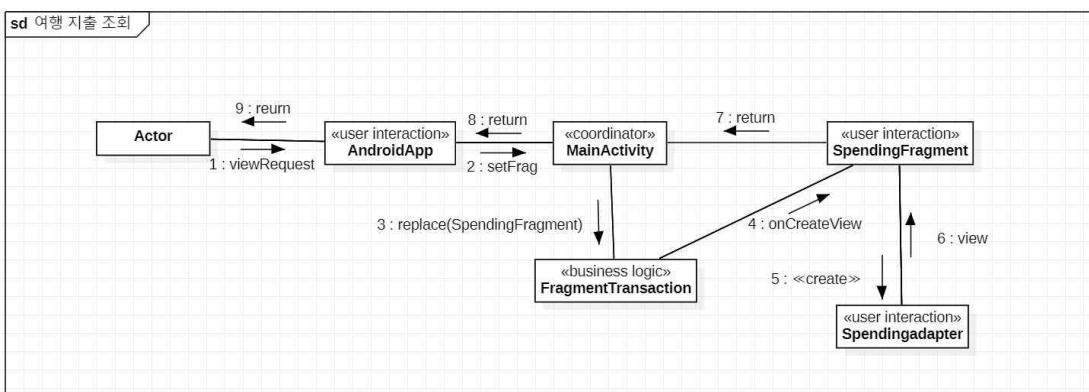
1. addRequest : 사용자는 안드로이드 앱을 통해 추가 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SearchFragment) : MainActivity는 FragmentTransaction에게 SearchFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 SearchFragment를 실행시킨다.
5. downloadContents : SearchFragment는 SpotNetworkManager로 downloadContents 를 요청한다. 그 결과, 관광지에 맞는 사진을 다운로드하게 된다.
6. returnResult : 관광지에 맞는 사진을 다운로드하면 SearchFragment로 사진을 반환한다.
7. parse : SpotXmlParsher는 SpotNetworkManager로부터 반환 받은 사진의 태입을 바꿔 준다.
8. return : SpotXmlParsher에서 태입을 바꾼 사진을 다시 받는다.
9. insert : SearchFragment는 SpotXmlParsher로부터 받은 데이터를 SQLiteDatabase에 게 삽입한다.
- 10 ~ 13 : return
14. view : 안드로이드 앱이 사용자에게 추가 사항을 보여준다.

여행 일정 조회



1. ViewRequest : 사용자는 안드로이드 앱을 통해 조회 버튼을 눌러 조회를 요청한다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(ScheduleFragment) : MainActivity는 FragmentTransaction에게 ScheduleFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 ScheduleFragment를 실행시킨다.
5. <<create>> : ScheduleFragment는 ScheduleAdapter를 생성하여 조회할 정보를 저장한다.
6. view : ScheduleAdapter가 ScheduleFragment에게 조회할 정보를 반환 한다.
- 7 ~ 8 : return
9. return : 사용자에게 원하는 항목을 보여준다.

여행 지출 조회

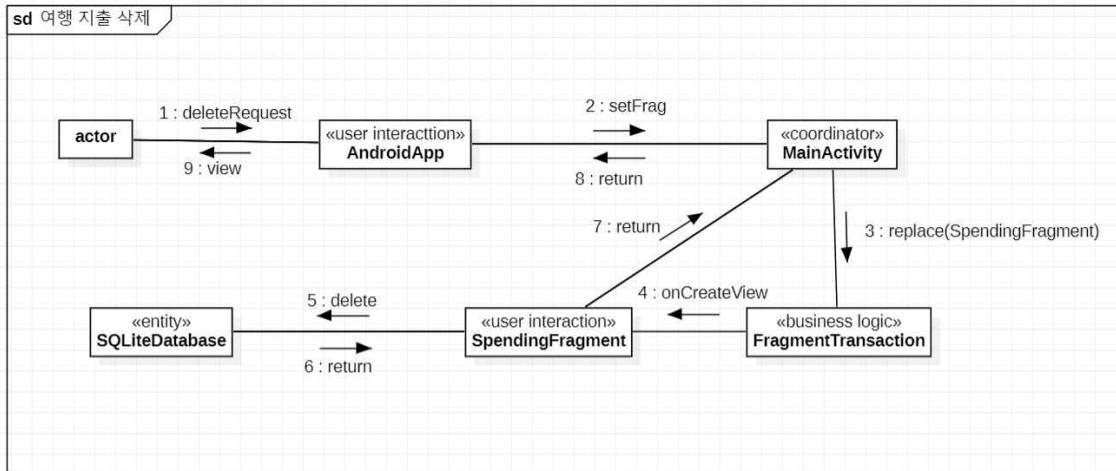


1. viewRequest : 사용자는 안드로이드 앱을 통해 조회 버튼을 눌러 조회를 요청한다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SpendingFragment) : MainActivity는 FragmentTransaction에게 SpendingFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 SpendingFragment를 실행시킨다.
5. <<create>> : SpendingFragment는 SpendingAdapter를 생성하여 조회할 정보를 저장한다.
6. view : SpendingAdapter가 SpendingFragment에게 조회할 정보를 반환 한다.
- 7 : return
- 8 : return
- 9 : return

다.

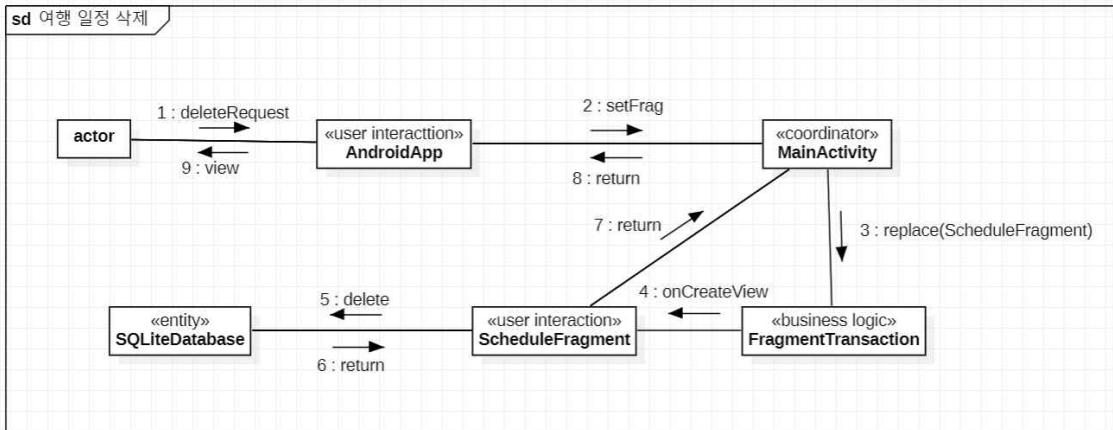
6. view : SpendingAdapter가 SpendingFragment에게 조회할 정보를 반환 한다.
- 7 ~ 8 : Return
9. return : 사용자에게 원하는 항목을 보여준다.

여행 지출 삭제



1. deleteRequest : 사용자는 안드로이드 앱을 통해 삭제 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(SpendingFragment) : MainActivity는 FragmentTransaction에게 SpendingFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 SpendingFragment를 실행시킨다.
5. delete : SQLiteDatabase에게 해당 정보 삭제를 요청한다.
- 6 ~ 8 : return
9. view : 안드로이드 앱이 사용자에게 수정된 사항을 보여준다.

여행 일정 삭제



1. deleteRequest : 사용자는 안드로이드 앱을 통해 삭제 버튼을 누른다.
2. setFrag : 안드로이드 앱은 MainActivity부터 화면 전환을 명령한다.
3. replace(ScheduleFragment) : MainActivity는 FragmentTransaction에게 ScheduleFragment를 호출시킨다.
4. onCreateView : FragmentTransaction은 ScheduleFragment를 실행시킨다.
5. delete : SQLiteDatabase에게 해당 정보 삭제를 요청한다.
- 6 ~ 8 : return
9. view : 안드로이드 앱이 사용자에게 수정된 사항을 보여준다.

2.6 State machine diagram

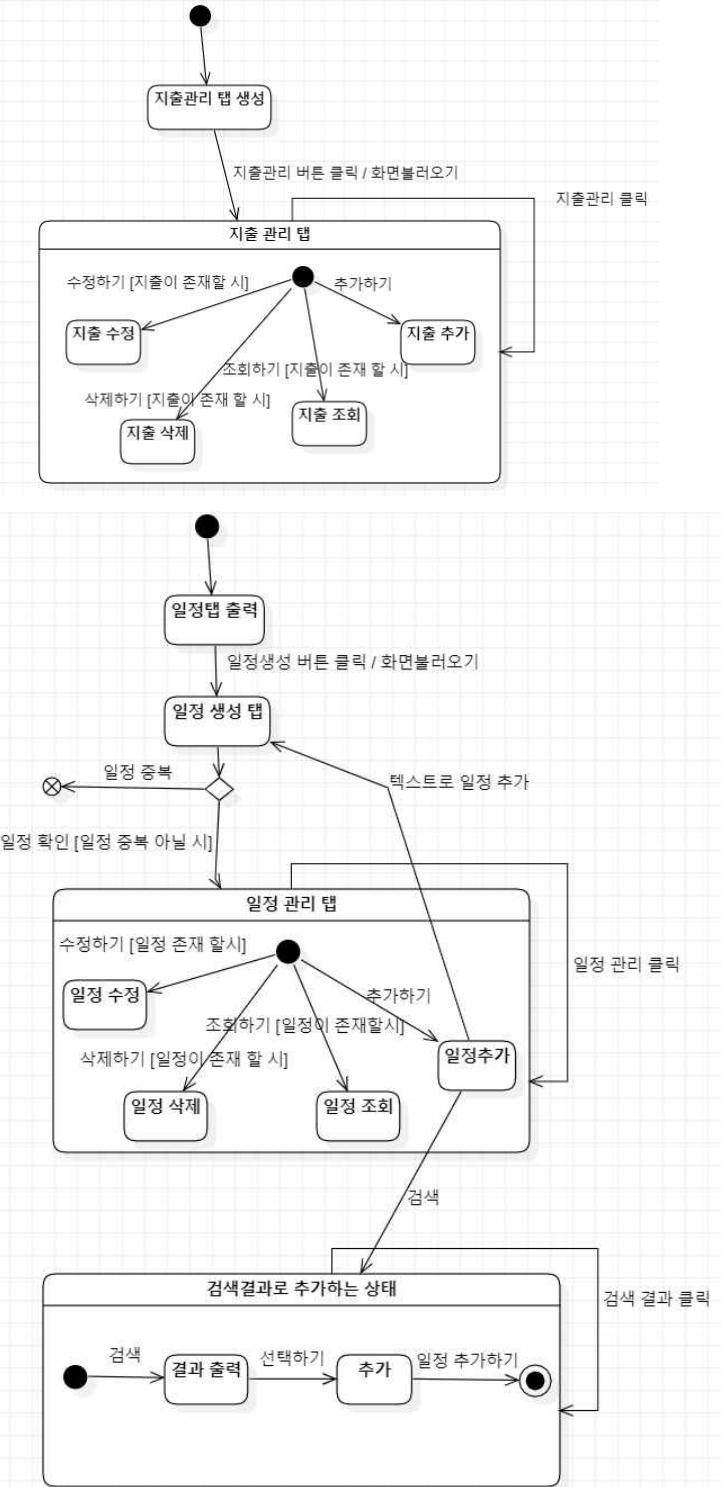
2.6.1 Event, Guard condition, Action

Event는 특정한 순간에 한 번에 발생하게 된다. 버튼 클릭, 텍스트로 일정 추가, 일정 관리 탭 클릭 등, 상태 전이를 유발하는 행동들을 event로 설정하여 디어그램에 나타내었다.

Guard condition은 조건부의 상태를 나타내고 Event[Condition]으로 나타낸다. 다음의 디어그램에서는 일정이 중복되거나 지출이 존재하는 조건을 Guard로 나타내었다. 즉 일정 중복이 아닐 시에만 해당 event인 일정확인을 수행하는 것이다.

Action은 상태 전이의 결과로 실행되어야 하는 행위로 /Action으로 나타낸다. 다음의 디어그램에서는 일정 생성 버튼을 클릭하는 event가 실행되면 화면 불러오기 같은 action이 실행되는 모습을 볼 수 있다.

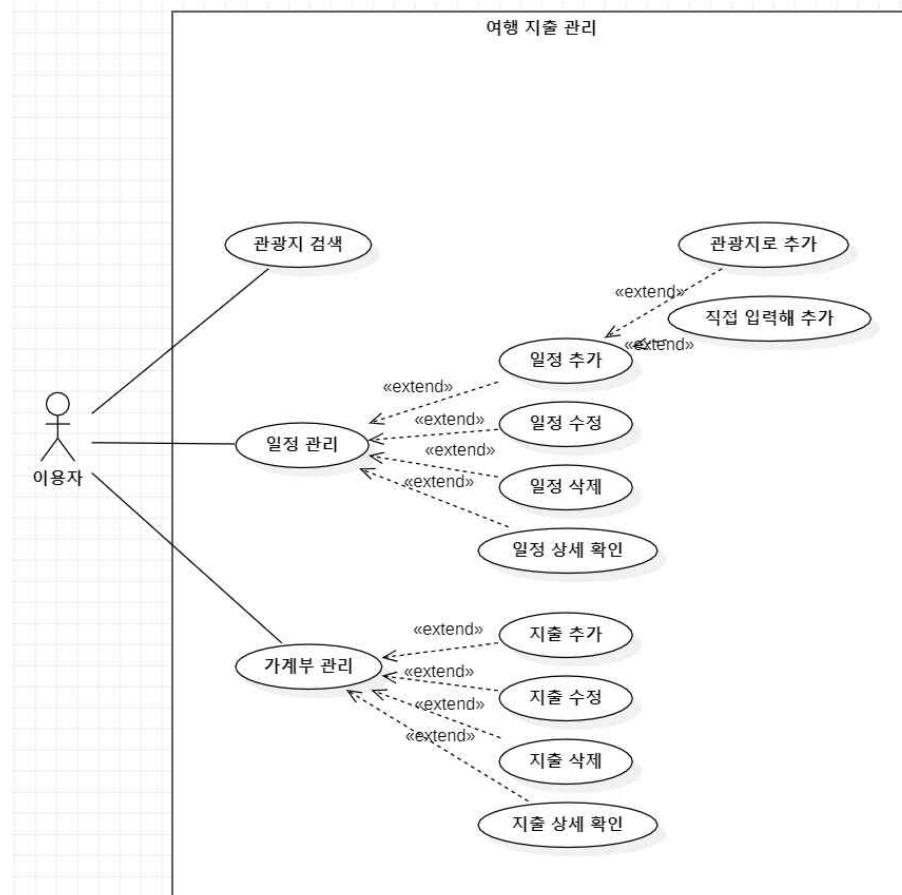
2.6.2 State machine diagram 작성



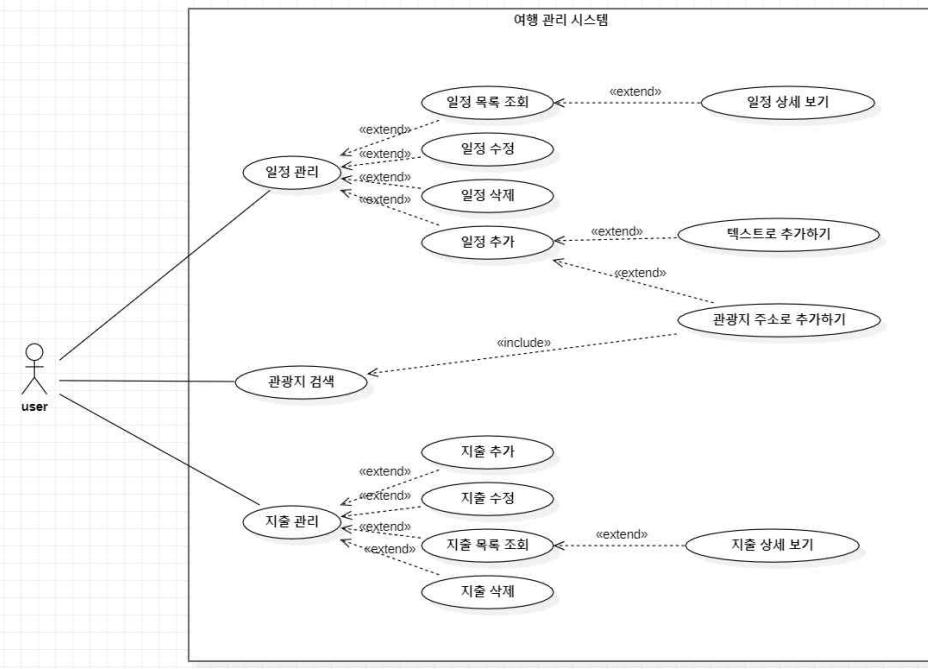
State machine diagram은 시스템 또는 객체의 전체적인 제어와 흐름을 모델링하기 위해 사용되는 다이어그램이다. 시스템의 동적인 측면을 그려내는 다이어그램이다. 상태는 크게 일정 관리 탭, 검색결과로 추가하는 상태, 지출관리 탭으로 보았다.

2.7 수정

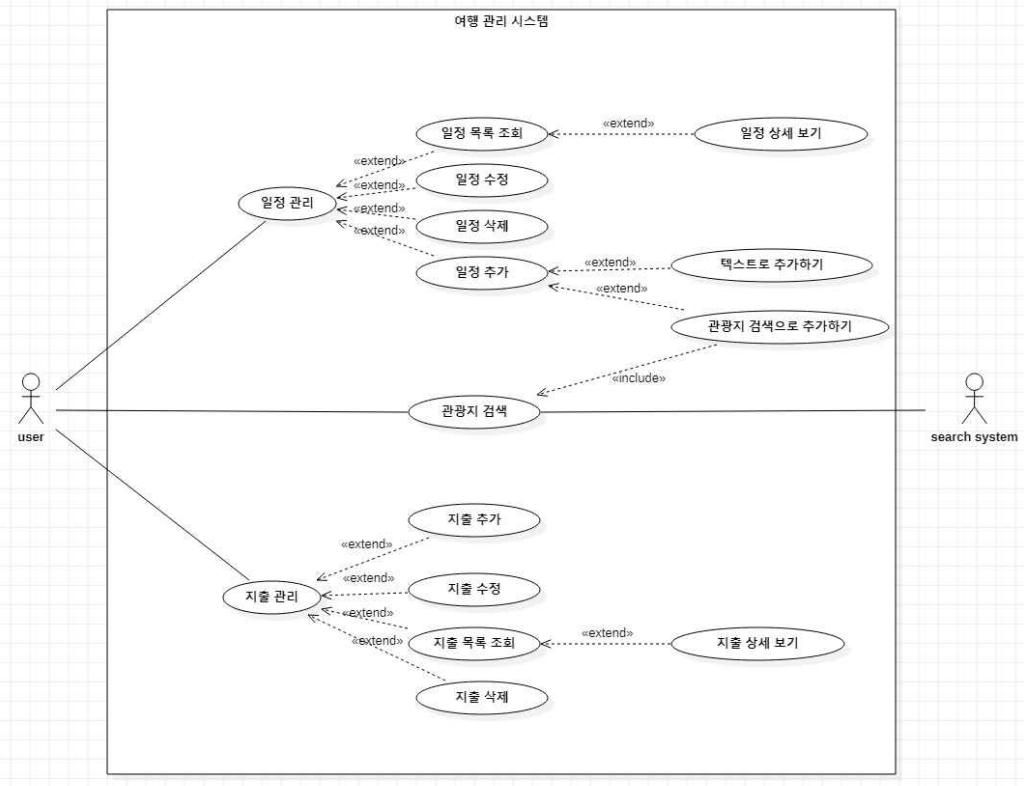
2.7.1 Use case diagram



처음으로 그린 Use case diagram이다. 깔끔하게 잘 정리가 되어 있지도 않고, 번잡스러운 느낌이 든다.



그 다음으로 그린 Use case diagram이다. 전보다는 깔끔해지고 정리가 잘 된 모습으로 보인다. 하지만 sub system이 부족하기 때문에 다시 그리기로 하였다.

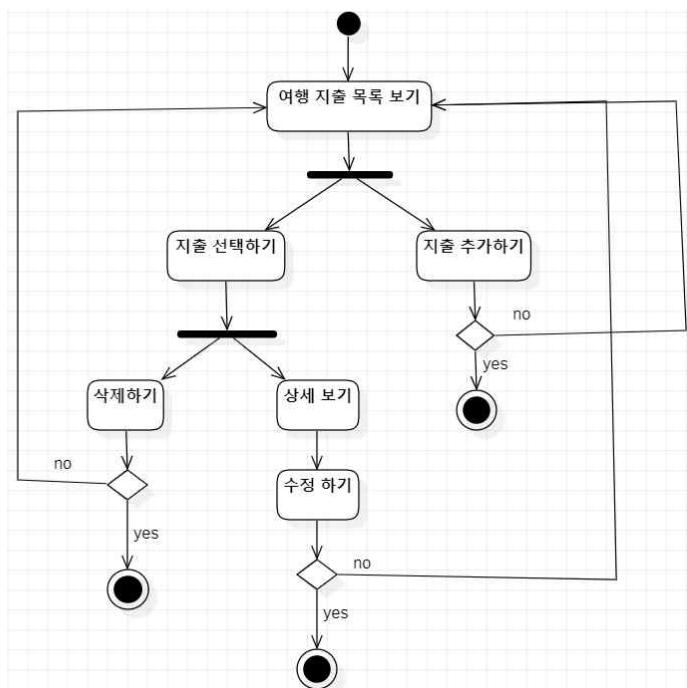


최종 Use case diagram이다. extend와 include를 다시 고민해보고, search system도 추

가하여 보다 완성도 높은 Use case diagram이 되었다.

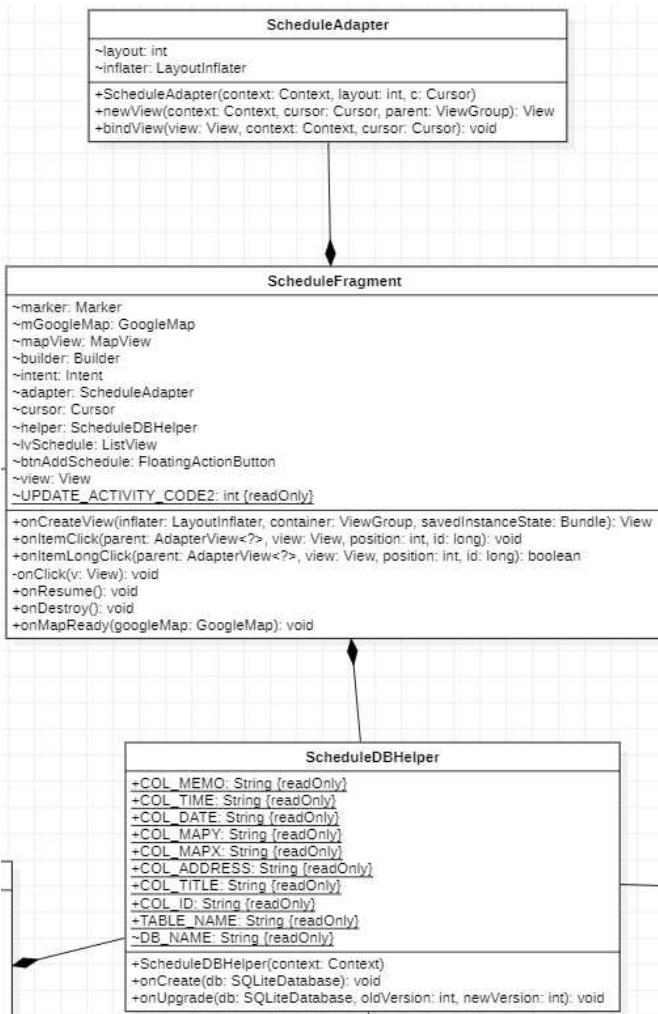
2.7.2 Activity diagram

Use case diagram을 토대로 Activity diagram을 그려 보았다. 그 중 하나인 지출 관리 Activity diagram이다. Activity diagram에 대한 팀원들의 의견은 변경할 부분이 없다고 판단하여 완성했다.

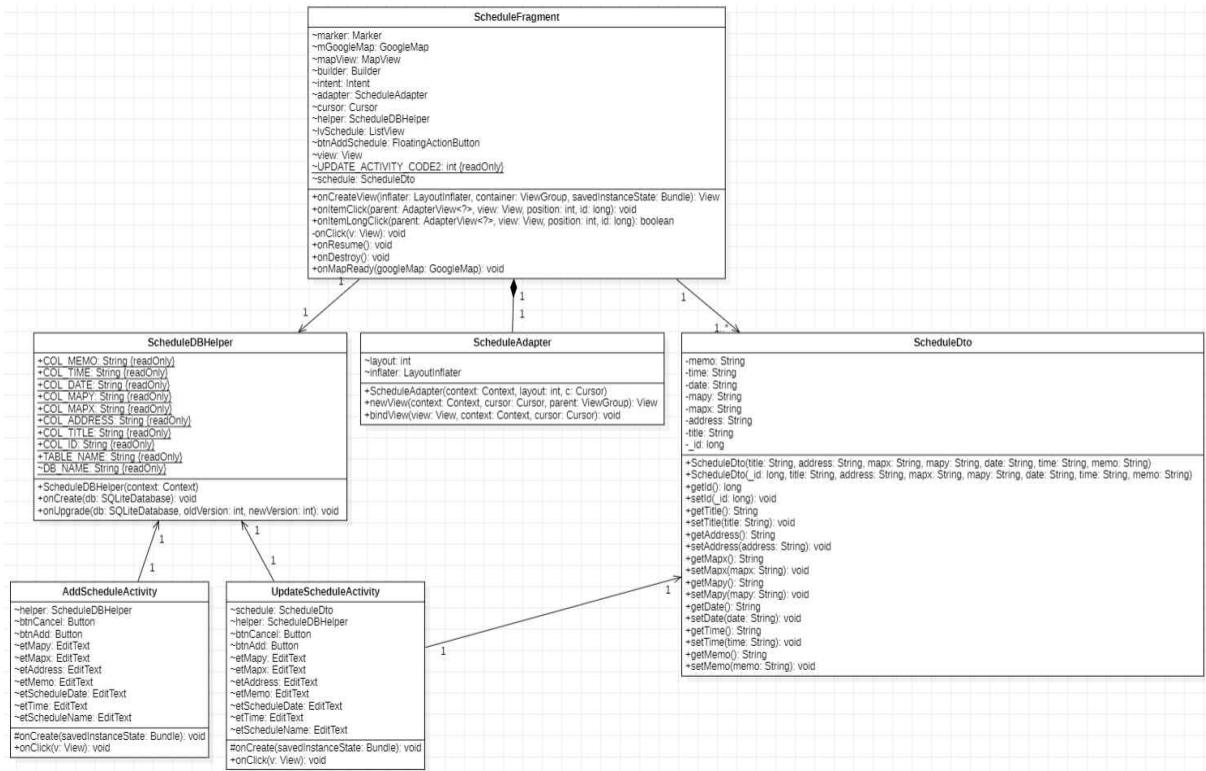


2.7.3 Class diagram

전체적인 Class diagram을 보여주기는 힘들기 때문에, 특정 부분에 대해서 발전 과정을 서술해 보았다.



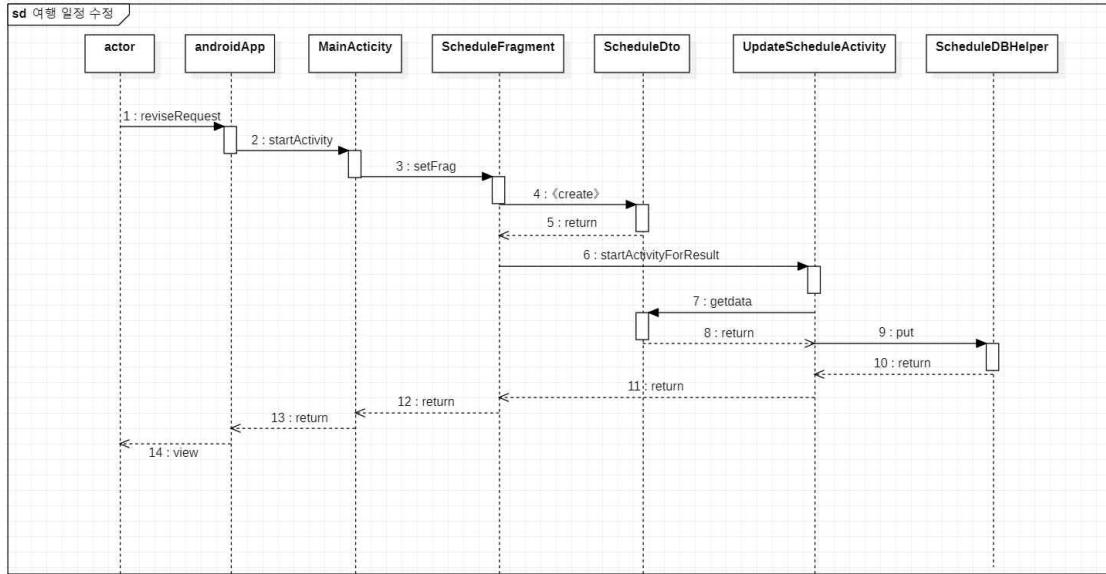
다중도 표시도 되어 있지 않고, 클래스 간의 관계가 모두 composition으로 표현되어 있는 모습을 확인할 수 있다. 그래서 이 클래스는 적절치 않다고 판단되어 다시 그려보았다.



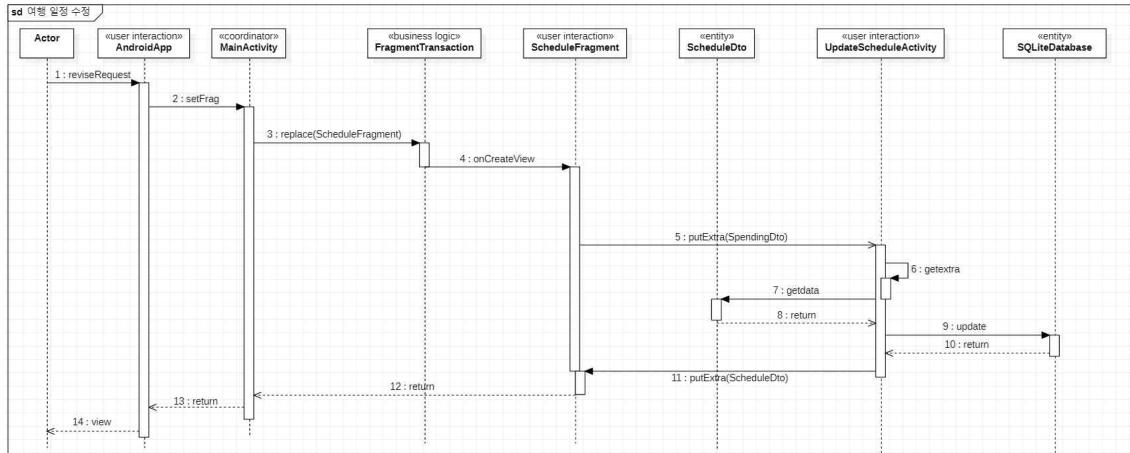
최종적으로 완성된 Class diagram이다. Fragment와 Adapter, DBHelper간의 관계 뿐 아니라 다중도 또한 잘 나타내고 있는 것을 확인할 수 있다.

2.7.4 Sequence diagram

Sequence diagram을 다 보여주기는 어렵기 때문에 일정 수정에 대한 변화 과정을 서술해보았다. instance form을 사용하였다.



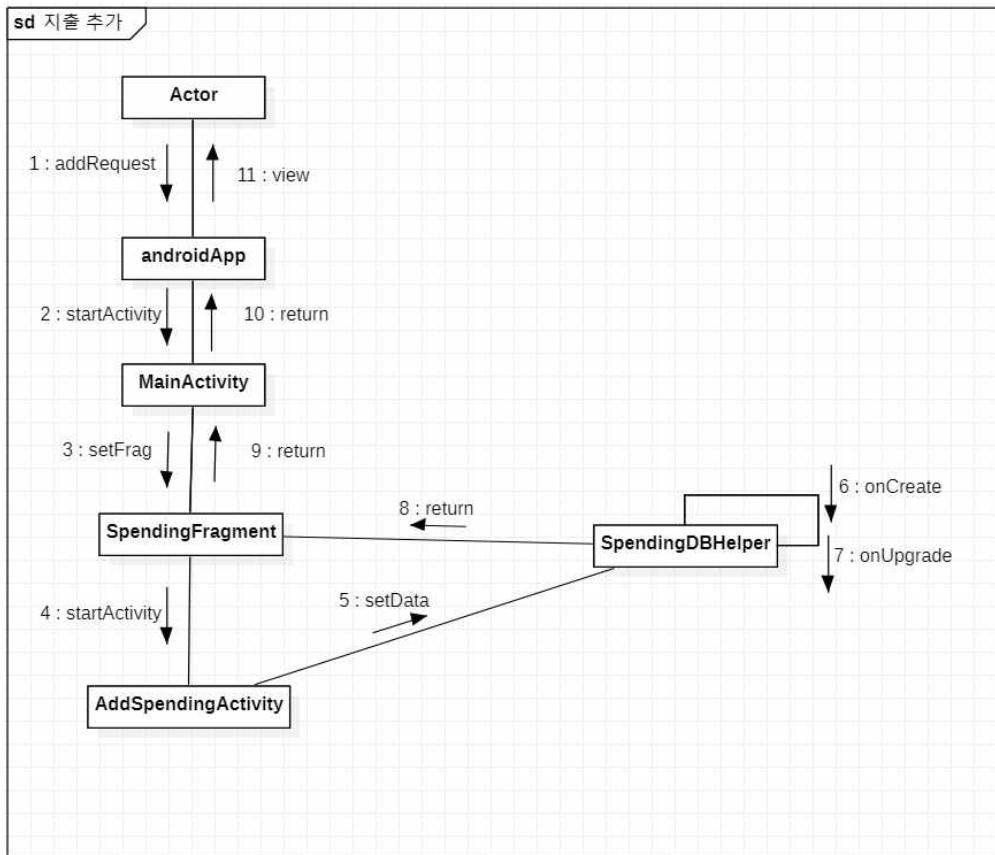
처음으로 그려 본 Sequence diagram이다. 객체 분류 한 것을 적용하지도 않고, 사용한 클래스도 맞지 않다고 판단되어 다시 그려보았다.



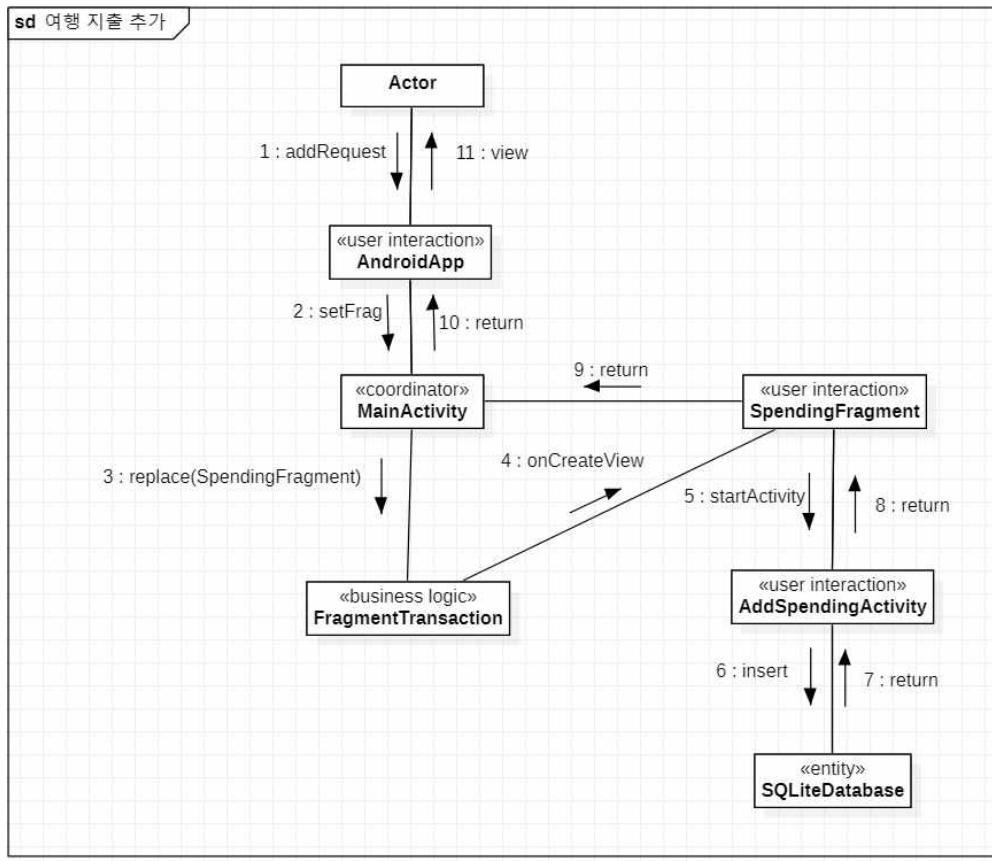
최종적으로 완성된 Sequence diagram이다. 객체 분류도 하고, 시간 순서에 따라 다시 배열하여 완성하였다.

2.7.5 Communication diagram

Communication diagram을 다 보여주기는 어렵기 때문에 지출 추가에 대한 변화 과정을 서술해보았다.

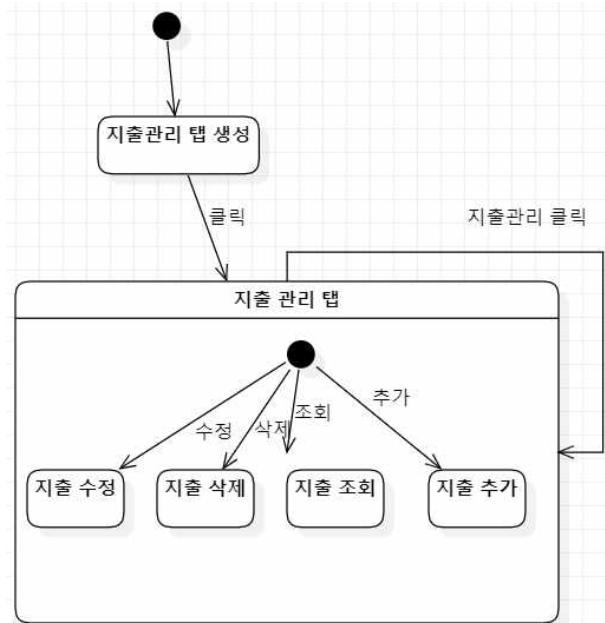


처음으로 그려 본 Communication diagram이다. 객체 분류 한 것을 적용하지도 않고, 사용한 클래스도 맞지 않다고 판단되어 다시 그려 보았다.

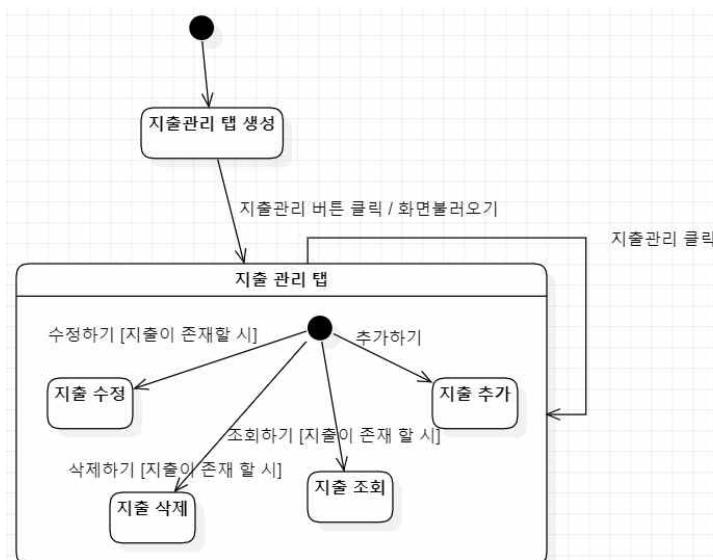


최종적으로 완성된 Communication diagram이다. 객체 분류를 하였고, 객체 사이의 상호 작용 또한 잘 확인할 수 있다.

2.7.6 State machine diagram



처음으로 그린 State machine diagram이다. Event만 썼기 때문에 Guard condition, Action에 대한 부분이 부족하다고 느꼈다.



최종적으로 완성된 State machine diagram이다. Event, Guard condition, Action을 적절하게 추가하여 상태 전이가 잘 보이도록 완성했다.

3. 소스 코드

3.1 소스코드

```
AddScheduleActivity  
package ddwucom.mobile.finalproject.ma01_20170922;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.ContentValues;  
import android.content.Intent;  
import android.database.sqlite.SQLiteDatabase;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
  
public class AddScheduleActivity extends AppCompatActivity {  
  
    EditText etScheduleName;  
    EditText etTime;  
    EditText etScheduleDate;  
    EditText etMemo;  
    EditText etAddress;  
    EditText etMapx;  
    EditText etMapy;  
  
    Button btnAdd;  
    Button btnCancel;  
  
    ScheduleDBHelper helper;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_add_schedule);  
    }  
}
```

```

etScheduleName = findViewById(R.id.etScheduleName);
etTime = findViewById(R.id.etTime);
etScheduleDate = findViewById(R.id.etScheduleDate);
etMemo = findViewById(R.id.etMemo);
etAddress = findViewById(R.id.etAddress);
etMapx = findViewById(R.id.etMapx);
etMapy = findViewById(R.id.etMapy);

btnAdd = findViewById(R.id.btnAdd2);
btnCancel = findViewById(R.id.btnCancel2);

helper = new ScheduleDBHelper(this);
}

public void onClick(View v) {
switch (v.getId()) {
case R.id.btnAdd2:
String title = etScheduleName.getText().toString();
String time = etTime.getText().toString();
String date = etScheduleDate.getText().toString();
String memo = etMemo.getText().toString();
String address = etAddress.getText().toString();
String mapx = etMapx.getText().toString();
String mapy = etMapy.getText().toString();

//필수 항목 입력하지 않고 추가 클릭 시
if (title.matches(""))
Toast.makeText(this, "필수 항목을 입력하지 않았습니다.", Toast.LENGTH_SHORT).show();
else {
SQLiteDatabase db = helper.getWritableDatabase();
ContentValues value = new ContentValues();

value.put(helper.COL_TITLE, title);
value.put(helper.COL_TIME, time);
value.put(helper.COL_DATE, date);
value.put(helper.COL_MEMO, memo);
value.put(helper.COL_ADDRESS, address);
value.put(helper.COL_MAPX, mapx);
value.put(helper.COL_MAPY, mapy);
}
}
}

```

```

db.insert(helper.TABLE_NAME, null, value);

Intent resultIntent = new Intent();
setResult(RESULT_OK, resultIntent);
helper.close();
finish();
}

break;
case R.id.btnCancel2:
setResult(RESULT_CANCELED);
finish();
break;
}
}
}
}

```

```

AddSpendingActivity
package ddwucom.mobile.finalproject.ma01_20170922;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AddSpendingActivity extends AppCompatActivity {
EditText etTitle;
EditText etPrice;
EditText etCategory;
EditText etDate;

Button btnAdd;
Button btnCancel;

```

```

SpendingDBHelper helper;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_add_spending);

etTitle = findViewById(R.id.etTitle);
etPrice = findViewById(R.id.etPrice);
etCategory = findViewById(R.id.etCategory);
etDate = findViewById(R.id.etDate);

btnAdd = findViewById(R.id.btnAdd);
btnCancel = findViewById(R.id.btnCancel);

helper = new SpendingDBHelper(this);
}

public void onClick(View v) {
switch (v.getId()) {
case R.id.btnAdd:
String title = etTitle.getText().toString();
String price = etPrice.getText().toString();
String category = etCategory.getText().toString();
String date = etDate.getText().toString();

//필수 항목 입력하지 않고 추가 클릭 시
if (title.matches("") || price.matches("") || category.matches(""))
Toast.makeText(this, "필수 항목을 입력하지 않았습니다.",
Toast.LENGTH_SHORT).show();
else {
SQLiteDatabase db = helper.getWritableDatabase();
ContentValues value = new ContentValues();

value.put(helper.COL_TITLE, title);
value.put(helper.COL_PRICE, price);
value.put(helper.COL_CATEGORY, category);
value.put(helper.COL_DATE, date);

db.insert(helper.TABLE_NAME, null, value);
}
}
}

```

```

Intent resultIntent = new Intent();
setResult(RESULT_OK, resultIntent);
helper.close();
finish();
}
break;
case R.id.btnCancel:
setResult(RESULT_CANCELED);
finish();
break;
}
}
}
}

```

Constants

```

package ddwucom.mobile.finalproject.ma01_20170922;

/* 앱에 사용할 상수 값을 정의한 클래스*/
public class Constants {
public static final int SUCCESS_RESULT = 0;
public static final int FAILURE_RESULT = 1;
public static final String PACKAGE_NAME = "mobile.example.lbs.geocodingtest";
public static final String RECEIVER = PACKAGE_NAME + ".RECEIVER";
public static final String RESULT_DATA_KEY = PACKAGE_NAME +
".RESULT_DATA_KEY";
public static final String ADDRESS_DATA_EXTRA = PACKAGE_NAME +
".ADDRESS_DATA_EXTRA";
public static final String LAT_DATA_EXTRA = PACKAGE_NAME + ".LATITUDE";
public static final String LNG_DATA_EXTRA = PACKAGE_NAME + ".LONGITUDE";
}

```

FetchAddressIntentService

```

package ddwucom.mobile.finalproject.ma01_20170922;

import android.app.IntentService;
import android.content.Intent;
import android.location.Address;
import android.location.Geocoder;
```

```

import android.os.Bundle;
import android.os.ResultReceiver;
import android.text.TextUtils;
import android.util.Log;

import androidx.annotation.Nullable;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

public class FetchAddressIntentService extends IntentService {

    final static String TAG = "FetchAddress";

    private Geocoder geocoder;
    private ResultReceiver receiver;

    public FetchAddressIntentService() {
        super("FetchLocationIntentService");
    }

    @Override
    protected void onHandleIntent(@Nullable Intent intent) {
        geocoder = new Geocoder(getApplicationContext(), Locale.getDefault());

        // MainActivity 가 전달한Intent 로부터 위도/경도와Receiver 객체 설정
        if (intent == null) return;
        double latitude = intent.getDoubleExtra(Constants.LAT_DATA_EXTRA, 0);
        double longitude = intent.getDoubleExtra(Constants.LNG_DATA_EXTRA, 0);
        receiver = intent.getParcelableExtra(Constants.RECEIVER);

        List<Address> addresses = null;

        // 위도/경도에 해당하는 주소 정보를Geocoder 에게 요청
        try {
            addresses = geocoder.getFromLocation(latitude, longitude, 1);
        } catch (IOException e) {
    }
}

```

```

e.printStackTrace();
} catch (IllegalArgumentException e) {
e.printStackTrace();
}

// 결과로부터 주소 추출
if (addresses == null || addresses.size() == 0) {
Log.e(TAG, getString(R.string.no_address_found));
deliverResultToReceiver(Constants.FAILURE_RESULT, null);
} else {
Address addressList = addresses.get(0);
ArrayList<String> addressFragments = new ArrayList<String>();

for(int i = 0; i <= addressList.getMaxAddressLineIndex(); i++) {
addressFragments.add(addressList.getAddressLine(i));
}
Log.i(TAG, getString(R.string.address_found));
deliverResultToReceiver(Constants.SUCCESS_RESULT,
TextUtils.join(System.getProperty("line.separator"),
addressFragments));
}
}

// ResultReceiver에게 결과를 Bundle 형태로 전달
private void deliverResultToReceiver(int resultCode, String message) {
Bundle bundle = new Bundle();
bundle.putString(Constants.RESULT_DATA_KEY, message);
receiver.send(resultCode, bundle);
}

}

```

ImageFileManager

```

package ddwucom.mobile.finalproject.ma01_20170922;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;

```

```

import android.net.Uri;
import android.os.Environment;
import android.util.Log;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class ImageFileManager {

    final static String TAG = "ImageFileManager";
    final static String IMG_EXT = ".jpg";

    private Context context;

    public ImageFileManager(Context context) {
        this.context = context;
    }

    /* Bitmap 과 Bitmap 다운로드에 사용한 URL 을 전달받아 내부저장소에 JPG 파일로 저장 후
    파일 이름을 반환, 파일 저장 실패 시 null 반환*/
    public String saveBitmapToTemporary(Bitmap bitmap, String url) {
        String fileName = null;
        try {
            fileName = getFileNameFromUrl(url);

            // 내부 저장소에 파일 생성
            File saveFile = new File(context.getFilesDir(), fileName);
            // FileOutputStream 스트림 생성
            FileOutputStream fos = new FileOutputStream(saveFile);
            // 비트맵 이미지를 파일에 기록, Bitmap의 크기가 클 경우 이 부분에서 조정
            bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fos);

            fos.flush();
            fos.close();
        } catch (FileNotFoundException e) {

```

```

e.printStackTrace();
fileName = null;
} catch (IOException e) {
e.printStackTrace();
fileName = null;
}
return fileName;
}

/*
이미지 다운로드에 사용하는 url 을 전달받아
url 에 해당하는 내부 저장소에 이미지 파일이 있는지 확인 후 있으면 Bitmap 반환, 없을
경우null 반환*/
public Bitmap getBitmapFromTemporary(String url) {
String fileName = getFileNameFromUrl(url);
String path = context.getFilesDir().getPath() + "/" + fileName;

// 비트맵일 경우의 읽기
Bitmap bitmap = null;
bitmap = BitmapFactory.decodeFile(path);
Log.i(TAG, path);

return bitmap;
}

/*
이미지 파일명을 전달받아 외부저장소에 이미지 파일이 있는지 확인 후 있으면 Bitmap
반환, 없을 경우null 반환*/
public Bitmap getBitmapFromExternal(String fileName) {
if (!isExtStorageWritable()) return null;
String path = context.getExternalFilesDir(Environment.DIRECTORY_PICTURES) + "/" +
fileName;
Log.i(TAG, path);
Bitmap bitmap = BitmapFactory.decodeFile(path);
return bitmap;
}

/*
내부저장소에 저장한 모든 임시 파일을 제거*/
public void clearTemporaryFiles() {
File internalStoragefiles = context.getFilesDir();

```

```

File[] files = internalStoragefiles.listFiles();
for (File target : files) {
    target.delete();
}

}

/* Bitmap 과Bitmap 다운로드에 사용한URL 을 전달받아
내부저장소에 저장한 임시파일을 외부저장소로 옮긴 후 현재시간으로 지정한 파일명을
반환*/
public String moveFileToExt(String url) {
    if (!isExtStorageWritable()) return null;

    // 내부 저장소에 저장하고 있는 원본 파일 준비
    if(url != null) {
        String fileName = getFileNameFromUrl(url);
        File internalFile = new File(context.getFilesDir(), fileName);

        // 외부 저장소에 저장할 복사본 파일 준비
        String newfileName = getCurrentTime() + IMG_EXT; // 현재시간.jpg 로 파일이름
        지정
        File externalFile = new
        File(context.getExternalFilesDir(Environment.DIRECTORY_PICTURES), newfileName);
        // 복사 수행
        boolean result = copyFile(internalFile, externalFile);

        // 정상적으로 복사 완료 시 내부저장소의 파일 삭제 및 외부저장소에 저장한 파일명
        반환
        if (result) {
            internalFile.delete();
            return newfileName;
        }
    }

    return null;
}

/* 외부저장소에 보관 중인 이미지 파일을 제거*/
public boolean removeImageFromExt(String fileName) {

```

```
if (!isExtStorageWritable()) return false;
File targetFile = new
File(context.getExternalFilesDir(Environment.DIRECTORY_PICTURES), fileName);
return targetFile.delete();
}
```

```
/* URL에서 파일명에 해당하는 부분을 추출(예):
http://www.dongduk.ac.kr/main/main.jpg → main.jpg
사용하는URL에 따라 달라질 수 있으므로 사용 시 확인 필요*/
public String getFileNameFromUrl(String url) {
String fileName = Uri.parse(url).getLastPathSegment();
return fileName.replace("\n", "");
}
```

```
/* 현재 시간(초단위)을 문자열로 단위*/
private String getCurrentTime() {
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMdd_HHmmss");
//SSS가 밀리세컨드 표시
return dateFormat.format(new Date());
}
```

```
/* target을dest로 복사, 실패할 경우false반환*/
private boolean copyFile(File targetFile, File destFile) {
FileInputStream fis = null;
FileOutputStream fos = null;
byte[] buffer = null;
int length = 0;
try {
fis = new FileInputStream(targetFile);
fos = new FileOutputStream(destFile) ;
buffer = new byte[1024];
while((length=fis.read(buffer)) != -1){
fos.write(buffer, 0, length);
}
} catch (Exception e) {
e.printStackTrace();
return false;
} finally{
```

```

try {
    fis.close();
    fos.close();
} catch (IOException e) {
    e.printStackTrace();
}
}

return true;
}

/* 외부 저장소가 읽기/쓰기가 가능한지 확인 */
private boolean isExtStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

}

```

MainActivity

```

package ddwucom.mobile.finalproject.ma01_20170922;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import android.os.Bundle;
import android.view.MenuItem;

import com.google.android.material.bottomnavigation.BottomNavigationView;

public class MainActivity extends AppCompatActivity {
    private BottomNavigationView bottomNavigationView; //바텀 네비게이션 뷰
    private FragmentManager fragmentManager;

```

```

private FragmentTransaction fragmentTransaction;
private ScheduleFragment scheduleFragment;
private SearchFragment searchFragment;
private SpendingFragment spendingFragment;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

////바텀 네비게이션 뷰
bottomNavigationView = findViewById(R.id.bottomNavi);
bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
switch (menuItem.getItemId()) {
case R.id.action_schedule:
setFrag(0);
break;
case R.id.action_search:
setFrag(1);
break;
case R.id.action_spending:
setFrag(2);
break;
}
return true;
}
});

scheduleFragment = new ScheduleFragment();
searchFragment = new SearchFragment();
spendingFragment = new SpendingFragment();
setFrag(0); //첫 프래그먼트 화면 설정
}

//프래그먼트 교체
private void setFrag(int n)

```

```

{
fragmentManager = getSupportFragmentManager();
fragmentTransaction= fragmentManager.beginTransaction();
switch (n)
{
case 0:
fragmentTransaction.replace(R.id.Main_Frame, scheduleFragment);
fragmentTransaction.commit();
break;

case 1:
fragmentTransaction.replace(R.id.Main_Frame, searchFragment);
fragmentTransaction.commit();
break;

case 2:
fragmentTransaction.replace(R.id.Main_Frame, spendingFragment);
fragmentTransaction.commit();
break;
}
}
}

}

```

ScheduleAdapter

```

package ddwucom.mobile.finalproject.ma01_20170922;

import android.content.Context;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;

public class ScheduleAdapter extends CursorAdapter {
LayoutInflater inflater;
int layout;

```

```

public ScheduleAdapter(Context context, int layout, Cursor c) {
    super(context, c, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    this.inflater = (LayoutInflater)
        context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    this.layout = layout;
}

@Override
public View newView(Context context, Cursor cursor, ViewGroup parent) {
    View view = inflater.inflate(layout, parent, false);
    ScheduleAdapter.ViewHolder holder = new ScheduleAdapter.ViewHolder();
    view.setTag(holder);

    return view;
}

@Override
public void bindView(View view, Context context, Cursor cursor) {
    ScheduleAdapter.ViewHolder holder = (ScheduleAdapter.ViewHolder)view.getTag();

    if(holder.tvTitle == null) {
        holder.tvTitle = view.findViewById(R.id.tvScheduleName);
        holder.tvScheduleAddress = view.findViewById(R.id.tvScheduleAddress);
    }

    holder.tvTitle.setText(cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_TITLE)));
    holder.tvScheduleAddress.setText(cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_ADDRESS)));
}

static class ViewHolder {
    public ViewHolder() {
        tvTitle = null;
        tvScheduleAddress = null;

    }

    TextView tvTitle;
    TextView tvScheduleAddress;
}

```

```

}

}

ScheduleDBHelper
package ddwucom.mobile.finalproject.ma01_20170922;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class ScheduleDBHelper extends SQLiteOpenHelper {
final static String DB_NAME = "schedule.db";
public final static String TABLE_NAME = "schedule_table";
public final static String COL_ID = "_id";
public final static String COL_TITLE = "title";
public final static String COL_ADDRESS = "address";
public final static String COL_MAPX = "mapx";
public final static String COL_MAPY = "mapy";
public final static String COL_DATE = "date";
public final static String COL_TIME = "time";
public final static String COL_MEMO = "memo";

public ScheduleDBHelper(Context context) {
super(context, DB_NAME, null, 1);
}

@Override
public void onCreate(SQLiteDatabase db) {
String sql = "CREATE TABLE " + TABLE_NAME
+ "(" + COL_ID + "integer primary key autoincrement, "
+ COL_TITLE + "TEXT, " + COL_ADDRESS + "TEXT, " + COL_MAPX + "TEXT, " +
COL_MAPY + "TEXT, " + COL_DATE + "TEXT, " + COL_TIME + "TEXT, " +
COL_MEMO + "TEXT )";
db.execSQL(sql);

//샘플 데이터
db.execSQL("INSERT INTO " + ScheduleDBHelper.TABLE_NAME + "VALUES (NULL,
'오죽현', '강원도 강릉시 율곡로3139번길24', '128.8776814918', '37.7787651808',

```

```

'2020.12.23', '14:00', '택시타고가기');");
db.execSQL("INSERT INTO " + ScheduleDBHelper.TABLE_NAME + "VALUES (NULL,
'초당맷돌순두부', '강원도 강릉시 초당순두부길75', '128.9154527370', '37.7901229368',
'2020.12.23', '18:00', '해물순두부맛집');");
db.execSQL("INSERT INTO " + ScheduleDBHelper.TABLE_NAME + "VALUES (NULL,
'경포대', '강원도 강릉시 경포로365', '128.8965126086', '37.7955691591', '2020.12.23',
'18:00', '근처에경포해변');");
db.execSQL("INSERT INTO " + ScheduleDBHelper.TABLE_NAME + "VALUES (NULL,
'하슬라아트월드', '강원도 강릉시 강동면 율곡로1441', '129.0102329067', '37.7065316769',
'2020.12.24', '18:00', '강릉시내에서버스타고가기');");
db.execSQL("INSERT INTO " + ScheduleDBHelper.TABLE_NAME + "VALUES (NULL,
'안목해변', '강원도 강릉시 창해로14번길20-1', '128.9473504054', '37.7726505813',
'2020.12.25', '10:00', '커피거리에서커피마시기');");
}

```

```

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE " + TABLE_NAME);
    onCreate(db);
}
}

```

ScheduleDto

```
package ddwucom.mobile.finalproject.ma01_20170922;
```

```

import java.io.Serializable;

public class ScheduleDto implements Serializable {
    private long _id;
    private String title;
    private String address;
    private String mapx;
    private String mapy;
    private String date;
    private String time;
    private String memo;

    public ScheduleDto(String title, String address, String mapx, String mapy, String
date, String time, String memo) {

```

```

this.title = title;
this.address = address;
this.mapx = mapx;
this.mapy = mapy;
this.date = date;
this.time = time;
this.memo = memo;
}

public ScheduleDto(long _id, String title, String address, String mapx, String mapy,
String date, String time, String memo) {
this._id = _id;
this.title = title;
this.address = address;
this.mapx = mapx;
this.mapy = mapy;
this.date = date;
this.time = time;
this.memo = memo;
}

public long getId() { return _id; }

public void setId(long _id) { this._id = _id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getAddress() { return address; }

public void setAddress(String address) { this.address = address; }

public String getMapx() { return mapx; }

public void setMapx(String mapx) { this.mapx = mapx; }

public String getMapy() { return mapy; }

public void setMapy(String mapy) { this.mapy = mapy; }

```

```
public String getDate() { return date; }

public void setDate(String date) { this.date = date; }

public String getTime() { return time; }

public void setTime(String time) { this.time = time; }

public String getMemo() { return memo; }

public void setMemo(String memo) { this.memo = memo; }

}
```

```
ScheduleFragment
package ddwucom.mobile.finalproject.ma01_20170922;

import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapView;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.LatLng;
```

```

import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

public class ScheduleFragment extends Fragment implements
AdapterView.OnItemClickListener, AdapterView.OnItemLongClickListener,
OnMapReadyCallback {
final static int UPDATE_ACTIVITY_CODE2 = 300;

View view;
FloatingActionButton btnAddSchedule;

ListView lvSchedule = null;
ScheduleDBHelper helper;
Cursor cursor;
ScheduleAdapter adapter;

Intent intent;
AlertDialog.Builder builder;

MapView mapView;
GoogleMap mGoogleMap;
Marker marker;

@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
view = inflater.inflate(R.layout.fragment_schedule, container, false);

btnAddSchedule = view.findViewById(R.id.btnAddSchedule);
btnAddSchedule.setOnClickListener(this::onClick);

//리스트뷰 처리
helper = new ScheduleDBHelper(getActivity());

lvSchedule = (ListView) view.findViewById(R.id.lvSchedule);
adapter = new ScheduleAdapter(getActivity(), R.layout.listview_schedule, null);
lvSchedule.setAdapter(adapter);

//리스트뷰 클릭리스너
}

```

```

lvSchedule.setOnItemClickListener(ScheduleFragment.this);
lvSchedule.setOnItemLongClickListener(ScheduleFragment.this);

//맵뷰
MapView = (MapView)view.findViewById(R.id.map);
MapView.onCreate(savedInstanceState);
MapView.onResume();
MapView.getMapAsync(this);

//뷰리턴
return view;
}

//리스트뷰 클릭 처리
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
//Toast.makeText(getApplicationContext(), id + "선택", Toast.LENGTH_SHORT).show();

SQLiteDatabase db = helper.getReadableDatabase();

//id로 데이터 선택하기
cursor = db.rawQuery("SELECT * FROM " + ScheduleDBHelper.TABLE_NAME +
" WHERE " + ScheduleDBHelper.COL_ID + "=" + id + "", null);

String title = null;
String address = null;
String mapx = null;
String mapy = null;
String date = null;
String time = null;
String memo = null;

while(cursor.moveToNext()) {
title = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_TITLE));
address =
cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_ADDRESS));
mapx = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_MAPX));
mapy = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_MAPY));
date = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_DATE));
time = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_TIME));
memo = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_MEMO));
}

```

```

}

//dto 생성
ScheduleDto schedule = new ScheduleDto(id, title, address, mapx, mapy, date, time,
memo);

intent = new Intent(getActivity(), UpdateScheduleActivity.class);
intent.putExtra("schedule", schedule);
startActivityForResult(intent, UPDATE_ACTIVITY_CODE2);
}

@Override
public boolean onItemLongClick(AdapterView<?> parent, View view, int position,
long id) {
cursor = (Cursor) adapter.getItem(position);
String title = cursor.getString(cursor.getColumnIndex("title"));

builder = new AlertDialog.Builder(getActivity());

builder.setTitle("일정 삭제")
.setMessage(title + "을(를) 삭제하시겠습니까?")
.setPositiveButton("삭제", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int which) {
//삭제 수행
Toast.makeText(getActivity(), "삭제되었습니다", Toast.LENGTH_SHORT).show();
SQLiteDatabase sqLiteDatabase = helper.getWritableDatabase();

String whereClause = helper.COL_ID + "=?";
String[] whereArgs = new String[] { String.valueOf(id) };
sqLiteDatabase.delete(helper.TABLE_NAME, whereClause, whereArgs);

cursor = sqLiteDatabase.rawQuery("select * from " +
ScheduleDBHelper.TABLE_NAME, null);
adapter.changeCursor(cursor);

helper.close();
}
})
.setNegativeButton("취소", null)
.setCancelable(false)

```

```

.show();

return true;
}

private void onClick(View v) {
switch (v.getId()) {
case R.id.btnAddSchedule: //플로팅버튼
intent = new Intent(getActivity(), AddScheduleActivity.class);
startActivity(intent);
break;
}
}

@Override
public void onResume() {
super.onResume();
SQLiteDatabase db = helper.getReadableDatabase();
cursor = db.rawQuery("select * from " + ScheduleDBHelper.TABLE_NAME, null);

adapter.changeCursor(cursor);
helper.close();

//맵뷰
mapView.onResume();
}

@Override
public void onDestroy() {
super.onDestroy();

if (cursor != null) cursor.close();
}

@Override
public void onMapReady(GoogleMap googleMap) {
mGoogleMap = googleMap;

//db에서 일정 정보 읽어와서 지도에 마커로 표시
SQLiteDatabase db = helper.getReadableDatabase();

```

```

cursor = db.rawQuery("select * from " + ScheduleDBHelper.TABLE_NAME, null);

long id = 0;
String title = null;
String mapx = null;
String mapy = null;

while(cursor.moveToNext()) {
    id = cursor.getLong(cursor.getColumnIndex(ScheduleDBHelper.COL_ID));
    title = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_TITLE));
    mapx = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_MAPX));
    mapy = cursor.getString(cursor.getColumnIndex(ScheduleDBHelper.COL_MAPY));

    if(id == 1) {
        LatLng currentLoc = new LatLng(Double.parseDouble(mapy),
                                         Double.parseDouble(mapx));
        mGoogleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(currentLoc, 12));

        mGoogleMap.addMarker(new MarkerOptions()
            .position(currentLoc)
            .title(title));
    } else{
        if(mapy != null && mapx !=null) {
            mGoogleMap.addMarker(new MarkerOptions()
                .position(new LatLng(Double.parseDouble(mapy), Double.parseDouble(mapx)))
                .title(title));
        }
    }
}
}

```

SearchFragment

```

package ddwucom.mobile.finalproject.ma01_20170922;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.ContentValues;

```

```

import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.SearchView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;

import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.ArrayList;

```

public class SearchFragment **extends** Fragment **implements**
AdapterView.OnItemLongClickListener {
public static final String **TAG** = "SearchActivity";

View **view**:

```

Button btnSearch;
EditText etTarget;
ListView lvSearchList;

SpotAdapter adapter;
ArrayList<SpotDto> resultList;
SpotXmlParsher parser;
SpotNetworkManager networkManager;

```

```

ImageFileManager imgFileManager;

String apiAddress;
String apiKey;
String query;

AlertDialog.Builder builder;

ScheduleDBHelper helper;

@Nullable
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    view = inflater.inflate(R.layout.fragment_search, container, false);

    btnSearch = view.findViewById(R.id.btnSearch);
    btnSearch.setOnClickListener(this::onClick);
    etTarget = view.findViewById(R.id.etTarget);
    lvSearchList = view.findViewById(R.id.lvSearchList);

    resultList = new ArrayList();
    adapter = new SpotAdapter(getActivity(), R.layout.listview_spot, resultList);
    lvSearchList.setAdapter(adapter);

    apiAddress = getResources().getString(R.string.api_url);
    apiKey = getResources().getString(R.string.api_key);
    parser = new SpotXmlParsher();
    networkManager = new SpotNetworkManager(getActivity());
    imgFileManager = new ImageFileManager(getActivity());

    helper = new ScheduleDBHelper(getActivity());

    //리스트뷰 클릭 리스너
    lvSearchList.setOnItemLongClickListener(SearchFragment.this);

    //뷰리턴
    return view;
}

public void onClick(View v) {

```

```

switch(v.getId()) {
    case R.id.btnSearch:
        query = etTarget.getText().toString(); // UTF-8 인코딩 필요
        // OpenAPI 주소와 query 조합 후 서버에서 데이터를 가져옴
        try {
            String result;
            result = apiAddress + apiKey + "&keyword=" + URLEncoder.encode(query, "UTF-8")
            +
            "&areaCode=&sigunguCode=&cat1=&cat2=&cat3=&listYN=Y&MobileOS=ETC&MobileApp=
            TourAPI3.0_Guide";
            new NetworkAsyncTask().execute(result);
            Log.d(TAG, result);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        // 가져온 데이터는 파싱 수행 후 어댑터에 설정
    }

    break;
}

@Override
public void onDestroy() {
    super.onDestroy();
    // 임시 파일 삭제
    imgFileManager.clearTemporaryFiles();
}

@Override
public boolean onItemLongClick(AdapterView<?> parent, View view, int position,
long id) {

    builder = new AlertDialog.Builder(getActivity());

    builder.setTitle("일정 추가")
        .setMessage(resultList.get(position).getTitle() + "을(를) 추가하시겠습니까?")
        .setPositiveButton("추가", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // 추가 수행
                Toast.makeText(getApplicationContext(), "추가되었습니다", Toast.LENGTH_SHORT).show();
            }
        });
}

```

```

SQLiteDatabase db = helper.getWritableDatabase();
ContentValues value = new ContentValues();

value.put(helper.COL_TITLE, resultList.get(position).getTitle());
value.put(helper.COL_TIME, "");
value.put(helper.COL_DATE, "");
value.put(helper.COL_MEMO, "");
value.put(helper.COL_ADDRESS, resultList.get(position).getAddress());
value.put(helper.COL_MAPX, resultList.get(position).getMapx());
value.put(helper.COL_MAPY, resultList.get(position).getMapy());

db.insert(helper.TABLE_NAME, null, value);

Intent resultIntent = new Intent();
getActivity(). setResult(Activity.RESULT_OK, resultIntent);
helper.close();

}

.setNegativeButton("취소", null)
.setCancelable(false)
.show();

return true;
}

class NetworkAsyncTask extends AsyncTask<String, Integer, String> {
ProgressDialog progressDlg;

@Override
protected void onPreExecute() {
super.onPreExecute();
progressDlg = ProgressDialog.show(getActivity(), "Wait", "Downloading...");
}

@Override
protected String doInBackground(String... strings) {
String address = strings[0];
String result = null;
// networking

```

```

result = networkManager.downloadContents(address);
if (result == null) return "Error!";

Log.d(TAG, result);

// parsing
resultList = parser.parse(result); // -> 반환타입 바꿔줘야한다.

return result;
}

```

```

@Override
protected void onPostExecute(String result) {

    adapter.setList(resultList);    // Adapter 의 결과List 를 설정 후 notify
    progressDlg.dismiss();
}

}
}

```

```

SpendingAdapter
package ddwucom.mobile.finalproject.ma01_20170922;

import android.content.Context;
import android.database.Cursor;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CursorAdapter;
import android.widget.TextView;

public class SpendingAdapter extends CursorAdapter {
    LayoutInflater inflater;
    int layout;

    public SpendingAdapter(Context context, int layout, Cursor c) {
        super(context, c, CursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER);
    }
}

```

```

this.inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
this.layout = layout;
}

@Override
public View newView(Context context, Cursor cursor, ViewGroup parent) {
View view = inflater.inflate(layout, parent, false);
ViewHolder holder = new ViewHolder();
view.setTag(holder);

return view;
}

@Override
public void bindView(View view, Context context, Cursor cursor) {
ViewHolder holder = (ViewHolder)view.getTag();

if(holder.tvTitle == null) {
holder.tvTitle = view.findViewById(R.id.tvTitle);
holder.tvPrice = view.findViewById(R.id.tvPrice);
holder.tvCategory = view.findViewById(R.id.tvCategory);
}

holder.tvTitle.setText(cursor.getString(cursor.getColumnIndex(SpendingDBHelper.COL_TITLE)));
holder.tvPrice.setText(cursor.getString(cursor.getColumnIndex(SpendingDBHelper.COL_PRICE)));
holder.tvCategory.setText(cursor.getString(cursor.getColumnIndex(SpendingDBHelper.COL_CATEGORY)));
}

static class ViewHolder {

public ViewHolder() {
tvTitle = null;
tvPrice = null;
tvCategory = null;
}
}

```

TextView tvTitle;

```

    TextView tvPrice;
    TextView tvCategory;
}
}

SpendingDBHelper
package ddwucom.mobile.finalproject.ma01_20170922;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class SpendingDBHelper extends SQLiteOpenHelper {
final static String DB_NAME = "spending.db";
public final static String TABLE_NAME = "spending_table";
public final static String COL_ID = "_id";
public final static String COL_TITLE = "title";
public final static String COL_PRICE = "price";
public final static String COL_CATEGORY = "category";
public final static String COL_DATE = "date";

public SpendingDBHelper(Context context) {
super(context, DB_NAME, null, 1);
}

@Override
public void onCreate(SQLiteDatabase db) {
String sql = "CREATE TABLE " + TABLE_NAME
+ "(" + COL_ID + "integer primary key autoincrement, "
+ COL_TITLE + "TEXT, " + COL_PRICE + "TEXT, " + COL_CATEGORY + "TEXT, "
+ COL_DATE + "TEXT )";
db.execSQL(sql);

//샘플 데이터
db.execSQL("INSERT INTO " + SpendingDBHelper.TABLE_NAME + "VALUES (NULL, "
'KTX왕복', '51000', '교통비', '2020.12.23');");
db.execSQL("INSERT INTO " + SpendingDBHelper.TABLE_NAME + "VALUES (NULL, "
'숙소', '76000', '숙박비', '2020.12.23');");
db.execSQL("INSERT INTO " + SpendingDBHelper.TABLE_NAME + "VALUES (NULL, "

```

```

'오죽헌', '3000', '입장료', '2020.12.23');");
db.execSQL("INSERT INTO " + SpendingDBHelper.TABLE_NAME + "VALUES (NULL,
'해물순두부', '8000', '식비', '2020.12.23');");
db.execSQL("INSERT INTO " + SpendingDBHelper.TABLE_NAME + "VALUES (NULL,
'하슬라아트월드', '12000', '입장료', '2020.12.23');");
db.execSQL("INSERT INTO " + SpendingDBHelper.TABLE_NAME + "VALUES (NULL,
'엽서', '1000', '쇼핑', '2020.12.23');");

}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE " + TABLE_NAME);
    onCreate(db);
}
}

```

```

SpendingDto
package ddwucom.mobile.finalproject.ma01_20170922;

import java.io.Serializable;

public class SpendingDto implements Serializable {
    private long _id;
    private String title;
    private String price;
    private String category;
    private String date;

    public SpendingDto(String title, String price, String category, String date) {
        this.title = title;
        this.price = price;
        this.category = category;
        this.date = date;
    }

    public SpendingDto(long _id, String title, String price, String category, String date)
    {
        this._id = _id;
    }
}

```

```
this.title = title;
this.price = price;
this.category = category;
this.date = date;
}

public long getId() { return _id; }

public void setId(long _id) { this._id = _id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getPrice() { return price; }

public void setPrice(String price) { this.price = price; }

public String getCategory() { return category; }

public void setCategory(String category) { this.category = category; }

public String getDate() { return date; }

public void setDate(String date) { this.date = date; }
}
```

SpendingFragment

```
package ddwucom.mobile.finalproject.ma01_20170922;
```

```
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

public class SpendingFragment extends Fragment implements
        AdapterView.OnItemClickListener, AdapterView.OnItemLongClickListener {
    final static int UPDATE_ACTIVITY_CODE = 200;

    View view;
    FloatingActionButton btnAddSpending;
    TextView tvSum;

    ListView lvSpending = null;
    SpendingDBHelper helper;
    Cursor cursor;
    SpendingAdapter adapter;

    Intent intent;
    AlertDialog.Builder builder;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup
            container, @Nullable Bundle savedInstanceState) {
        view = inflater.inflate(R.layout.fragment_spending, container, false);

        //setHasOptionsMenu(true);
    }
}
```

```

btnAddSpending = view.findViewById(R.id.btnAddSpending);
btnAddSpending.setOnClickListener(this::onClick);

//리스트뷰 처리
helper = new SpendingDBHelper(getActivity());

lvSpending = (ListView) view.findViewById(R.id.lvSpending);
adapter = new SpendingAdapter(getActivity(), R.layout.listview_spending, null);
lvSpending.setAdapter(adapter);

//리스트뷰 클릭리스너
lvSpending.setOnItemClickListener(SpendingFragment.this);
lvSpending.setOnItemLongClickListener(SpendingFragment.this);

//지출 합계
tvSum = view.findViewById(R.id.tvSum);
tvSum.setText(Integer.toString(sumOfSpending()));


//뷰리턴
return view;
}

public int sumOfSpending() {
int sum = 0;

SQLiteDatabase db = helper.getReadableDatabase();

cursor = db.rawQuery("select * from " + SpendingDBHelper.TABLE_NAME, null);

String price = null;

while(cursor.moveToNext()) {
price = cursor.getString(cursor.getColumnIndex(SpendingDBHelper.COL_PRICE));

sum += Integer.parseInt(price);
}

return sum;
}

```

```

//리스트뷰 클릭 처리
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    SQLiteDatabase db = helper.getReadableDatabase();

    //id로 데이터 선택하기
    cursor = db.rawQuery("SELECT * FROM " + SpendingDBHelper.TABLE_NAME +
        " WHERE " + SpendingDBHelper.COL_ID + "=" + id + "", null);

    String title = null;
    String price = null;
    String category = null;
    String date = null;

    while(cursor.moveToNext()) {
        title = cursor.getString(cursor.getColumnIndex(SpendingDBHelper.COL_TITLE));
        price = cursor.getString(cursor.getColumnIndex(SpendingDBHelper.COL_PRICE));
        category =
            cursor.getString(cursor.getColumnIndex(SpendingDBHelper.COL_CATEGORY));
        date = cursor.getString(cursor.getColumnIndex(SpendingDBHelper.COL_DATE));
    }

    //dto 생성
    SpendingDto spending = new SpendingDto(id, title, price, category, date);

    intent = new Intent(getActivity(), UpdateSpendingActivity.class);
    intent.putExtra("spending", spending);
    startActivityForResult(intent, UPDATE_ACTIVITY_CODE);
}

@Override
public boolean onItemLongClick(AdapterView<?> parent, View view, int position,
    long id) {
    cursor = (Cursor) adapter.getItem(position);
    String title = cursor.getString(cursor.getColumnIndex("title"));

    builder = new AlertDialog.Builder(getActivity());

    builder.setTitle("소비 삭제")
        .setMessage(title + "을(를) 삭제하시겠습니까?")
        .setPositiveButton("삭제", new DialogInterface.OnClickListener() {

```

```

@Override
public void onClick(DialogInterface dialog, int which) {
    //삭제 실행
    Toast.makeText(getActivity(), "삭제되었습니다", Toast.LENGTH_SHORT).show();
    SQLiteDatabase sqLiteDatabase = helper.getWritableDatabase();

    String whereClause = helper.COL_ID + "=?";
    String[] whereArgs = new String[] { String.valueOf(id) };
    sqLiteDatabase.delete(helper.TABLE_NAME, whereClause, whereArgs);

    cursor = sqLiteDatabase.rawQuery("select * from " +
        SpendingDBHelper.TABLE_NAME, null);
    adapter.changeCursor(cursor);

    tvSum.setText(Integer.toString(sumOfSpending()));

    helper.close();
}

.setNegativeButton("취소", null)
.setCancelable(false)
.show();

return true;
}

public void onClick(View v) {
switch (v.getId()) {
case R.id.btnAddSpending: //플로팅버튼
    intent = new Intent(getActivity(), AddSpendingActivity.class);
    startActivity(intent);
    break;
}
}

```

```

@Override
public void onResume() {
super.onResume();
SQLiteDatabase db = helper.getReadableDatabase();
cursor = db.rawQuery("select * from " + SpendingDBHelper.TABLE_NAME, null);
}

```

```
adapter.changeCursor(cursor);

tvSum.setText(Integer.toString(sumOfSpending()));

helper.close();
}

@Override
public void onDestroy() {
super.onDestroy();

if (cursor != null) cursor.close();
}

}
```

```
SpotAdapter
package ddwucom.mobile.finalproject.ma01_20170922;

import android.content.Context;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.ArrayList;

public class SpotAdapter extends BaseAdapter {

public static final String TAG = "SpotAdapter";

private LayoutInflater inflater;
private Context context;
private int layout;
```

```

private ArrayList<SpotDto> list;
private SpotNetworkManager networkManager = null;
private ImageFileManager imageFileManager = null;

public SpotAdapter(Context context, int resource, ArrayList<SpotDto> list) {
    this.context = context;
    this.layout = resource;
    this.list = list;
    imageFileManager = new ImageFileManager(context);
    networkManager = new SpotNetworkManager(context);
    inflater =
        (LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
}

@Override
public int getCount() {
    return list.size();
}

@Override
public SpotDto getItem(int position) {
    return list.get(position);
}

@Override
public long getItemId(int position) {
    return list.get(position).get_id();
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
    Log.d(TAG, "getView with position : " + position);
    View view = convertView;
    ViewHolder viewHolder = null;
}

```

```

if (view == null) {
    view = inflater.inflate(layout, parent, false);
    viewHolder = new ViewHolder();
    viewHolder.tvTitle = view.findViewById(R.id.tvTitle);
    viewHolder.tvAddress = view.findViewById(R.id.tvAddress);
    viewHolder.ivImage = view.findViewById(R.id.ivImage);
    view.setTag(viewHolder);
} else {
    viewHolder = (ViewHolder)view.getTag();
}

SpotDto dto = list.get(position);

viewHolder.tvTitle.setText(dto.getTitle());
viewHolder.tvAddress.setText(dto.getAddress());

/*작성할 부분*/
// dto의 이미지 주소에 해당하는 이미지 파일이 내부저장소에 있는지 확인
// ImageFileManager 의 내부저장소에서 이미지 파일 읽어오기 사용
// 이미지 파일이 있을 경우bitmap, 없을 경우null 을 반환하므로bitmap 이 있으면
// 이미지뷰에 지정
// 없을 경우GetImageAsyncTask 를 사용하여 이미지 파일 다운로드 수행
if(dto.getImageLink() == null) {
    viewHolder.ivImage.setImageResource(R.mipmap.ic_launcher);
    return view;
}

// 파일에 있는지 확인
// dto 의 이미지 주소 정보로 이미지 파일 읽기
Bitmap savedBitmap =
imageFileManager.getBitmapFromTemporary(dto.getImageLink()); //파일이름

if(savedBitmap != null) {
    viewHolder.ivImage.setImageBitmap(savedBitmap);
    Log.d(TAG, "Image loading from file");
} else {
    viewHolder.ivImage.setImageResource(R.mipmap.ic_launcher);
    new GetImageAsyncTask(viewHolder).execute(dto.getImageLink());
    Log.d(TAG, "Image loading from network");
}

```

```

return view;
}

public void setList(ArrayList<SpotDto> list) {
    this.list = list;
    notifyDataSetChanged();
}

//    ※findViewById() 호출 감소를 위해 필수로 사용할 것
static class ViewHolder {
    public TextView tvTitle = null;
    public TextView tvAddress = null;
    public ImageView ivImage = null;
}

/* 책 이미지를 다운로드 후 내부저장소에 파일로 저장하고 이미지뷰에 표시하는 AsyncTask
*/
// 1. 네트워크에서 이미지 다운
// 2. 뷰홀더에 이미지 설정
// 3. 이미지 파일 저장
class GetImageAsyncTask extends AsyncTask<String, Void, Bitmap> {

    ViewHolder viewHolder;
    String imageAddress;

    public GetImageAsyncTask(ViewHolder holder) {
        viewHolder = holder;
    }

    @Override
    protected Bitmap doInBackground(String... params) {
        imageAddress = params[0];
        Bitmap result = null;
        result = networkManager.downloadImage(imageAddress);
        return result;
    }

    @Override
}

```

```

protected void onPostExecute(Bitmap bitmap) {
    /*작성할 부분*/
    /*네트워크에서 다운 받은 이미지 파일을ImageFileManager 를 사용하여 내부저장소에 저장
     * 다운받은bitmap 을 이미지뷰에 지정*/
    if(bitmap != null) {
        viewHolder.ivImage.setImageBitmap(bitmap);
        imageFileManager.saveBitmapToTemporary(bitmap, imageAddress);
    }

}

}

```

```

SpotDto
package ddwucom.mobile.finalproject.ma01_20170922;

import android.text.Html;
import android.text.Spanned;

import java.io.Serializable;

public class SpotDto implements Serializable {
    private int _id;
    private String title;
    private String address;
    private String mapx;
    private String mapy;
    private String imageLink;
    private String imageFileName;

    public int get_id() { return _id; }

    public void set_id(int _id) { this._id = _id; }

    public String getTitle() {

```

```
Spanned spanned = Html.fromHtml(title);
return spanned.toString();
}

public void setTitle(String title) { this.title = title; }

public String getAddress() { return address; }

public void setAddress(String address) { this.address = address; }

public String getMapx() { return mapx; }

public void setMapx(String mapx) { this.mapx = mapx; }

public String getMapy() { return mapy; }

public void setMapy(String mapy) { this.mapy = mapy; }

public String getImageLink() { return imageLink; }

public void setImageLink(String imageLink) { this.imageLink = imageLink; }

public String getImageFileName() { return imageFileName; }

public void setImageFileName(String imageName) { this.imageFileName =
imageFileName; }
}
```

```
SpotNetworkManager
package ddwucom.mobile.finalproject.ma01_20170922;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

import java.io.BufferedReader;
import java.io.IOException;
```

```

import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import javax.net.ssl.HttpsURLConnection;

public class SpotNetworkManager {

    private Context context;

    public SpotNetworkManager(Context context) {
        this.context = context;
    }

    /* 주소(address)에 접속하여 문자열 데이터를 수신한 후 반환*/
    public String downloadContents(String address) {
        HttpURLConnection conn = null;
        InputStream stream = null;
        String result = null;

        if (!isOnline()) return null;

        try {
            URL url = new URL(address);
            conn = (HttpURLConnection)url.openConnection();
            stream = getNetworkConnection(conn);
            result = readStreamToString(stream);
            if (stream != null) stream.close();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (conn != null) conn.disconnect();
        }

        return result;
    }

    /* 주소를 전달받아 bitmap 다운로드 후 반환*/
}

```

```

public Bitmap downloadImage(String address) {
    HttpURLConnection conn = null;
    InputStream stream = null;
    Bitmap result = null;

    try {
        URL url = new URL(address);
        conn = (HttpURLConnection)url.openConnection();
        stream = getNetworkConnection(conn);
        result = readStreamToBitmap(stream);
        if (stream != null) stream.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (conn != null) conn.disconnect();
    }

    return result;
}

// downloadContents() 외 downloadImage()를 하나의 메소드로 구현할 경우
// 사용 시] 매개변수로 String or Image 를 구분, 반환 값은 타입캐스팅 필요
// public Object download(String address, boolean isImage) {
//     HttpURLConnection conn = null;
//     InputStream stream = null;
//     Object result = null;

//     try {
//         URL url = new URL(address);
//         conn = (HttpURLConnection)url.openConnection();
//         stream = getNetworkConnection(conn);
//         if (isImage) {
//             result = readStreamToBitmap(stream);
//         } else {
//             result = readStreamToString(stream);
//         }
//         if (stream != null) stream.close();
//     } catch (Exception e) {
//         e.printStackTrace();
//     } finally {

```

```

//           if (conn != null) conn.disconnect();
//       }
//
//       return result;
//   }

/* InputStream을 전달받아 비트맵으로 변환 후 반환*/
private Bitmap readStreamToBitmap(InputStream stream) {
    return BitmapFactory.decodeStream(stream);
}

/* InputStream을 전달받아 문자열로 변환 후 반환*/
private String readStreamToString(InputStream stream){
    StringBuilder result = new StringBuilder();

    try {
        InputStreamReader inputStreamReader = new InputStreamReader(stream);
        BufferedReader bufferedReader = new BufferedReader(inputStreamReader);

        String readLine = bufferedReader.readLine();

        while (readLine != null) {
            result.append(readLine + "\n");
            readLine = bufferedReader.readLine();
        }

        bufferedReader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }

    return result.toString();
}

/* 네트워크 환경 조사*/
private boolean isOnline() {
    ConnectivityManager connMgr = (ConnectivityManager)

```

```

context.getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
return (networkInfo != null && networkInfo.isConnected());
}

/*
 * URLConnection 을 전달받아 연결정보 설정 후 연결. 연결 후 수신한InputStream
 * 반환*/
private InputStream getURLConnection(HttpURLConnection conn) throws
Exception {

// 클라이언트 아이디 및 시크릿 그리고 요청URL 선언
conn.setReadTimeout(3000);
conn.setConnectTimeout(3000);
conn.setRequestMethod("GET");
conn.setDoInput(true);

if (conn.getResponseCode() != HttpsURLConnection.HTTP_OK) {
throw new IOException("HTTP error code: " + conn.getResponseCode());
}

return conn.getInputStream();
}
}

```

```

SpotXmlParsher
package ddwucom.mobile.finalproject.ma01_20170922;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;

import java.io.StringReader;
import java.util.ArrayList;

public class SpotXmlParsher {
public enum TagType { NONE, TITLE, ADDRRESS, MAPX, MAPY, IMAGE };

```

```

final static String TAG_ITEM = "item";
final static String TAG_TITLE = "title";
final static String TAG_ADDRESS = "addr1";
final static String TAG_MAPX = "mapx";
final static String TAG_MAPY = "mapy";
final static String TAG_IMAGE = "firstimage";

public SpotXmlParsher() {
}

public ArrayList<SpotDto> parse(String xml) {

    ArrayList<SpotDto> resultList = new ArrayList();
    SpotDto dto = null;

    TagType tagType = TagType.NONE;

    try {
        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
        XmlPullParser parser = factory.newPullParser();
        parser.setInput(new StringReader(xml));

        int eventType = parser.getEventType();
        while (eventType != XmlPullParser.END_DOCUMENT) {
            switch (eventType) {
                case XmlPullParser.START_DOCUMENT:
                    break;
                case XmlPullParser.END_DOCUMENT:
                    break;
                case XmlPullParser.START_TAG:
                    if (parser.getName().equals(TAG_ITEM)) {
                        dto = new SpotDto();
                    } else if (parser.getName().equals(TAG_TITLE)) {
                        if (dto != null) tagType = TagType.TITLE;
                    } else if (parser.getName().equals(TAG_ADDRESS)) {
                        if (dto != null) tagType = TagType.ADDRRESS;
                    } else if (parser.getName().equals(TAG_MAPX)) {
                        if (dto != null) tagType = TagType.MAPX;
                    } else if (parser.getName().equals(TAG_MAPY)) {
                        if (dto != null) tagType = TagType.MAPY;
                    } else if (parser.getName().equals(TAG_IMAGE)) {

```

```

if (dto != null) tagType = TagType.IMAGE;
}
break;
case XmlPullParser.END_TAG:
if (parser.getName().equals(TAG_ITEM)) {
resultList.add(dto);
dto = null;
}
break;
case XmlPullParser.TEXT:
switch(tagType) {
case TITLE:
dto.setTitle(parser.getText());
break;
case ADDRESS:
dto.setAddress(parser.getText());
break;
case MAPX:
dto.setMapx(parser.getText());
break;
case MAPY:
dto.setMapy(parser.getText());
break;
case IMAGE:
dto.setImageLink(parser.getText());
break;
}
tagType = TagType.NONE;
break;
}
eventType = parser.next();
}
} catch (Exception e) {
e.printStackTrace();
}

return resultList;
}
}

```

```
UpdateScheduleActivity
package ddwucom.mobile.finalproject.ma01_20170922;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class UpdateScheduleActivity extends AppCompatActivity {

    EditText etScheduleName;
    EditText etTime;
    EditText etScheduleDate;
    EditText etMemo;
    EditText etAddress;
    EditText etMapx;
    EditText etMapy;

    Button btnAdd;
    Button btnCancel;

    ScheduleDBHelper helper;
    ScheduleDto schedule;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_schedule);

        schedule = (ScheduleDto) getIntent().getSerializableExtra("schedule");

        etScheduleName = findViewById(R.id.etScheduleName);
        etTime = findViewById(R.id.etTime);
        etScheduleDate = findViewById(R.id.etScheduleDate);
```

```

etMemo = findViewById(R.id.etMemo);
etAddress = findViewById(R.id.etAddress);
etMapx = findViewById(R.id.etMapx);
etMapy = findViewById(R.id.etMapy);

btnAdd = findViewById(R.id.btnAdd2);
btnCancel = findViewById(R.id.btnCancel2);

helper = new ScheduleDBHelper(this);

//기존 정보 보여주기
etScheduleName.setText(schedule.getTitle());
etTime.setText(schedule.getTime());
etScheduleDate.setText(schedule.getDate());
etMemo.setText(schedule.getMemo());
etAddress.setText(schedule.getAddress());
etMapx.setText(schedule.getMapx());
etMapy.setText(schedule.getMapy());
}

public void onClick(View v) {
switch(v.getId()) {
case R.id.btnUpdate2:
String title = etScheduleName.getText().toString();
String time = etTime.getText().toString();
String date = etScheduleDate.getText().toString();
String memo = etMemo.getText().toString();
String address = etAddress.getText().toString();
String mapx = etMapx.getText().toString();
String mapy = etMapy.getText().toString();

if(title.matches(""))
Toast.makeText(this, "필수 항목을 입력하지 않았습니다.", Toast.LENGTH_SHORT).show();
else {
schedule.setTitle(title);
schedule.setTime(time);
schedule.setDate(date);
schedule.setMemo(memo);
schedule.setAddress(address);
schedule.setMapx(mapx);
}
}
}

```

```
schedule.setMapy(mapy);

//dto 받아서 값 설정
SQLiteDatabase sqLiteDatabase = helper.getWritableDatabase();
ContentValues row = new ContentValues();

row.put(helper.COL_TITLE, schedule.getTitle());
row.put(helper.COL_TIME, schedule.getTime());
row.put(helper.COL_DATE, schedule.getDate());
row.put(helper.COL_MEMO, schedule.getMemo());
row.put(helper.COL_ADDRESS, schedule.getAddress());
row.put(helper.COL_MAPX, schedule.getMapx());
row.put(helper.COL_MAPY, schedule.getMapy());

String whereClause = helper.COL_ID + "=?";
String[] whereArgs = new String[] { String.valueOf(schedule.getId()) };
sqLiteDatabase.update(helper.TABLE_NAME, row, whereClause, whereArgs);

Intent resultIntent = new Intent();
resultIntent.putExtra("schedule", schedule);
setResult(RESULT_OK, resultIntent);

helper.close();
finish();
}

break;
case R.id.btnCancel3:
setResult(RESULT_CANCELED);
finish();
break;
}
}
}
```

```
UpdateSpendingActivity  
package ddwucom.mobile.finalproject.ma01_20170922;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.ContentValues;
```

```
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class UpdateSpendingActivity extends AppCompatActivity {

    EditText etTitle;
    EditText etPrice;
    EditText etCategory;
    EditText etDate;

    Button btnUpdate;
    Button btnCancel;

    SpendingDBHelper helper;
    SpendingDto spending;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_spending);

        spending = (SpendingDto) getIntent().getSerializableExtra("spending");

        etTitle = findViewById(R.id.etTitle);
        etPrice = findViewById(R.id.etPrice);
        etCategory = findViewById(R.id.etCategory);
        etDate = findViewById(R.id.etDate);
        btnUpdate = findViewById(R.id.btnUpdate);
        btnCancel = findViewById(R.id.btnCancel);

        helper = new SpendingDBHelper(this);

        //기존 정보 보여주기
        etTitle.setText(spending.getTitle());
        etPrice.setText(spending.getPrice());
        etCategory.setText(spending.getCategory());
    }
}
```

```

etDate.setText(spending.getDate());
}

public void onClick(View v) {
switch(v.getId()) {
case R.id.btnUpdate:
String title = etTitle.getText().toString();
String price = etPrice.getText().toString();
String category = etCategory.getText().toString();
String date = etDate.getText().toString();

if(title.matches("") || price.matches("") || category.matches(""))
Toast.makeText(this, "필수 항목을 입력하지 않았습니다.",
Toast.LENGTH_SHORT).show();
else {
spending.setTitle(title);
spending.setPrice(price);
spending.setCategory(category);
spending.setDate(date);

//dto 받아서 값 설정
SQLiteDatabase sqLiteDatabase = helper.getWritableDatabase();
ContentValues row = new ContentValues();

row.put(helper.COL_TITLE, spending.getTitle());
row.put(helper.COL_PRICE, spending.getPrice());
row.put(helper.COL_CATEGORY, spending.getCategory());
row.put(helper.COL_DATE, spending.getDate());

String whereClause = helper.COL_ID + "=?";
String[] whereArgs = new String[] { String.valueOf(spending.getId()) };
sqLiteDatabase.update(helper.TABLE_NAME, row, whereClause, whereArgs);

Intent resultIntent = new Intent();
resultIntent.putExtra("spending", spending);
setResult(RESULT_OK, resultIntent);

helper.close();
finish();
}
break;
}

```

```

case R.id.btnCancel:
    setResult(RESULT_CANCELED);
    finish();
break;
}
}
}

```

3.2 실행결과 화면

여행 일정 클릭 시



추가한 여행 일정 목록을 확인할 수 있다. 여기서 우측 하단의 플러스 버튼을 눌러 텍스트로 여행 일정을 추가할 수 있다. 일정명은 필수로 입력해야 한다. 필수 정보를 입력했다면 추가 클릭 시 일정 DB에 추가된다. 필수 정보를 입력하지 않았을 경우에는 토스트로 알려준다. 추가한 일정을 삭제하려면 길게 누르면 항목 삭제 상자가 나온다. 이때, 삭제 클릭 시 일정 DB에서 삭제된다.

관광지 검색 클릭 시



관광지를 검색할 수 있다.

여행 지출 클릭 시



여행 지출을 확인할 수 있다. 길게 누르면 삭제가 가능하다. 길게 누르면 항목 삭제 대화 상자가 나오고 삭제 클릭시 지출 DB에서 삭제된다. 그리고 합계와 리스트뷰가 갱신된다.

키워드 ‘seoul’ 검색 결과

6:18 ☀️ 🔍 🌐

관광지 검색

seoul

롯데월드 SEOUL SKY
서울특별시 송파구 올림픽로300

서울100K (SEOUL100K)
서울 중구 태평로 2가 17-3

서울국제작가축제(Seoul International...)
서울특별시 마포구 양화로 72

신림빌(SS-Seoulstay)
서울특별시 관악구 조원로31길 17

제7회 서울커피페스티벌 The 7th Seoul...
서울특별시 강남구 영동대로 (513, 코엑스)

조선 팰리스 서울 강남, 럭셔리 컬렉션 호...
서울특별시 강남구 테헤란로 231



여행 일정



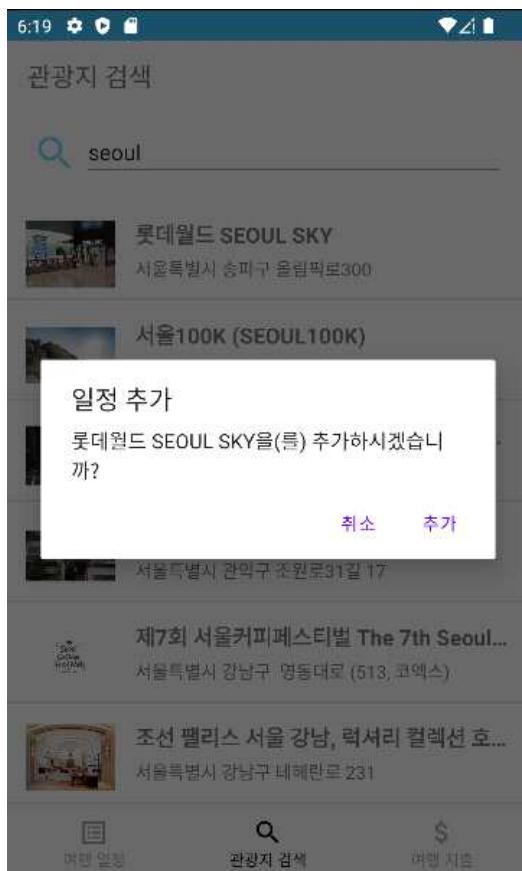
관광지 검색



여행 지출

검색 클릭 시 ‘한국 관광 공사 OPEN API’중 ‘키워드 검색’을 이용하여 해당 키워드가 들어간 관광 정보(관광지 이름, 주소, 위도, 경도, 사진)를 가져온다. 화면에는 리스트뷰로 관광지 이름, 주소, 사진만 출력한다.

길게 눌러서 일정 추가



길게 누르면 항목 추가 대화 상자가 나오고, 추가 클릭 시 여행 일정 DB에 추가된다. 이때 관광지 이름, 주소, 위도, 경도가 저장된다.

클릭 시 일정 수정



일정 수정

일정명	롯데월드 SEOUL SKY
시간	시간을 입력하세요
날짜	날짜를 입력하세요
메모	메모를 입력하세요

위치

주소	서울특별시 송파구 올림픽로300
위도	127.1040305171
경도	37.5142459111

취소

수정

기존 정보를 화면에 표시한다. 필수 정보를 입력했다면 수정 클릭 시 일정 DB가 수정이 되고, 여행 일정 화면으로 복귀하고 리스트뷰가 갱신된다. 필수 정보를 입력하지 않았을 경우에 토스트로 알려준다.

지출 추가



지출 추가

지출명

가격

카테고리

날짜

취소

추가

지출명, 가격, 카테고리를 필수로 입력한다. 필수 정보를 모두 입력했다면 추가 클릭 시 지출 DB에 추가된다. 필수 정보를 모두 입력하지 않았을 경우에는 토스트로 알려준다. 추가 완료시 지출화면으로 복귀해 합계와 리스트뷰가 갱신된다.

4. 후기

4.1 어려운 점 및 개선 사항

강연범 : 가장 어려웠던 점은 주제선정 이였습니다. diagram을 그리기에 적합하고 너무 간단하지도 너무 복잡하지도 않는 최적의 프로그램을 찾는데 어려움이 있었지만 팀원 모두 발벗고 각자 여러 개의 프로그램을 찾아와서 취합한 후 장단점을 판단하여 좋은 주제를 선정했다고 생각합니다. 추가적으로 저희 팀이 처음구성원과 약간의 차이가 있는데 이 부분에 있어서는 서로의 소통이 부족하여 모두가 만족하는 결과를 도출해지 못하였다고 생각합니다. 추후에는 더 많은 대화와 소통으로 모든 구성원들이 만족하는 방향으로 프로젝트를 진행하고자 합니다.

박지수 : 다들 diagram에 대해 정확히 알지 못하고, diagram에 대해서 계속 공부하면서 그려야 했기 때문에 나중에 새로운 내용에 대해 배우게 되었을 때, diagram을 계속해서 수정하는 부분이 어려웠던 것 같습니다. 또한, diagram을 그리면서도 이렇게 그리는 것이 맞나 하는 생각이 자주 들었습니다. 그만큼 사람마다 그리는 방법이 달랐기 때문입니다. diagram을 그리는 데에 있어서 서로 의견이 충돌했던 부분도 있었지만 그 부분에 대해서는 토의를 통해 해결하였습니다. diagram에 대한 각자의 의견을 말하고 가장 적절하다고 생각하는 사람의 의견을 따랐습니다. 그럼에도 불구하고 아쉬운 점은 주어진 시간이 조금 부족하다고 생각해서 나중에 기회가 된다면 diagram을 조금 더 세세하게 파악해서 꼼꼼히 그려보고 싶습니다.

백민재 : 업무 효율성이 낮았던 것 같습니다. 의견이 갈릴 때 수행해야 될 과제량이 더 많아지는 느낌이었던 것 같습니다. 여러모로 소통이 중요하다는 점을 깨달았던 것 같습니다.

정세연 : 안드로이드 앱을 처음으로 접해봐서 코드 분석이 어려웠습니다. 모바일 앱 수업을 듣고 진행했다면 훨씬 빠르게 진행했을 것 같습니다. 그리고 주제를 다시 정하고, diagram을 처음부터 다시 그려야 하는 상황이 왔을 때 막막했지만 조원들과 함께 상황을 이겨냈습니다. 개선할 점은 여전히 안드로이드 앱에 대해 자세히 알지 못해서 사용자에게 화면이 어떤 단계로 나타나는지 알 수 없다는 점입니다. 만약 시간이 좀 더 주어진다면 그 부분에 대해 개선하고 싶습니다.

4.2 프로젝트 후기

강연범 : 이전 학기에 배웠던 소프트웨어공학개론, 소프트웨어 도메인 모델링 여러 방면에서 배웠었던 UML에 있어서 단순히 개념만 공부하는 것이 아니라 직접 분석하고 더 심도 높은 강의를 통해 더 많은 방면으로 UML을 완성해나가면서 비록 완벽하지는 않은 프로젝트였을 수는 있지만 노력의 결과가 배신하지 않는 프로젝트였다고 생각하고 분명히 배운 내용이지만 소홀히 공부해서 넘어갔던 여러 부분들을 이 수업을 통해서 깊이 있게 배웠다고 생각합니다.

박지수 : 이번 프로젝트를 진행하면서 짧다면 짧고 길다면 긴 8주였습니다. 8주 동안 팀원들과 매주 만나서 diagram에 대해 토의하고 같이 그리면서 많은 걸 배웠던 것 같습니다. 하지만 처음에 정한 주제를 하지 않고, 새로운 주제로 다시 그리게 되었을 때는 정말 막막했습니다. 왜냐하면 시간이 별로 없었기 때문입니다. 하지만 그럴 때일수록 마음을 다잡고 팀원들과 시간을 많이 보내려고 노력했던 것 같습니다. 직접 만나기가 어렵고 바쁠 때는 카카오톡을 활용하여서 최대한 많은 의견을 공유하려 노력하였습니다. 그 결과 무사히 diagram을 그릴 수 있었다고 생각합니다. 이번 diagram을 그릴 때는 StarUML을 사용하여 그렸는데, 다음에 기회가 된다면 다른 diagram tool을 사용해 보고 싶습니다.

백민재 : 프로젝트를 통해 시스템을 분석하고 이해하는 방법을 더 잘 알 수 있었던 것 같습니다. 클래스 혹은 객체 간 상관관계에 대해서 서로 의견을 주고받거나 피드백 하다 보니 더욱 구조에 대해 생각하게 되었고 관련 지식 향상이 잇따랐던 것 같습니다.

정세연 : 여러 강의를 통해 diagram에 대해 정확히 알고 있다고 생각했지만 실제 프로젝트를 진행해보니 아무것도 모르고 있었다고 느꼈습니다. 확실하게 정의할 수 있는 것들이 아니면 조원들과 의견을 맞춰야 하는 부분도 생겼고 토론 과정에서 diagram에 대해 더 자세히 알게 되었습니다. 문제가 생겼을 때 여러 diagram 레퍼런스를 참고하려고 시도하였으나 의견이 모두 달라 참고하기에 어려움이 있었습니다. 그래서인지 diagram에 대한 저의 의견이 새롭게 생긴 것 같아 뿌듯함이 남습니다.