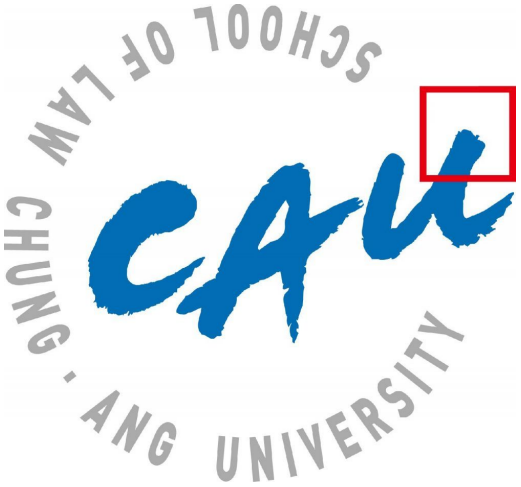
자료구조 설계(02)

310 NAVIGATION

Final Report



Subject	자료구조설계
Professor	정재은 교수님
Team Name	또구설계
Member (Student Number)	권도경(20154077) 김성재(20133659) 박미지(20150207) 이　용(20120596) 조원희(20153129)

Index

1. About Team

1.1 Team Name

또구설계 - 또또(Leader)와 자료구조 설계 팀원들을 의미합니다.

1.2 Team Member

권도경(20154077) : 안드로이드 구현 , Program 구조 설계

김성재(20133659) : 안드로이드 구현, 안드로이드 코드 총괄

박미지(20150207) : 안드로이드 구현, UI 및 시나리오 작성

이　용(20120596) : 알고리즘 설계, 알고리즘 구현

조원희(20153129) : 알고리즘 설계, 알고리즘 구현

1.3 Reference

GitHub - https://github.com/DDoGuSulGye

2. About Project

2.1 Title

- ❏ 310 Navigation

2.2 Goal

- ❏ 사용자는 출발지와, 목적지를 입력함으로서 해당 되는 경로를 알 수 있어야한다.
- ❏ 사용자는 경로와 함께 예상 시간을 알 수 있어야 한다.
- ❏ 사용자는 경유지를 포함하여 경로를 탐색할 수 있어야한다.
- ❏ 사용자는 계단을 이용할 때와 엘레베이터를 이용하는 총 2가지의 결과를 얻을 수 있어야한다.
- ❏ 사용자는 310관의 모든 강의실을 출발지, 목적지, 경유지로 설정할 수 있어야 한다.

2.3 Advantage of Project

- ❏ 해당 시간의 혼잡도를 수강 시간표를 통해 분석하여 더 정확한 값을 얻을 수 있다.
- ❏ 엘레베이터가 이용되는 여러 상황을 분석해 얻은 확률 값으로 시간의 정확도를 높일 수 있다.
- ❏ 계단을 통해 소요되는 시간은 직접 측정해서 얻은 결과로 시간의 정확도를 높일 수 있다.
- ❏ 사용자는 310관에 존재하는 모든 장소를 탐색할 수 있다.
- ❏ 설정한 구역은 단면도를 통해 획득한 구역으로 오차확률이 매우 낮다.
- ❏ 비상용 엘레베이터 까지 엘레베이터에 포함 시켜서 사용자는 더 다양한 엘레베이터를 이용할 수 있다.

3. Data Analysis

3.1 Data Analysis

3.1.1 강의실 별 수강인원(시간대별 혼잡도 파악)

❏ 강의실별로 월, 화, 수, 목, 금요일을 총 9교시로 나눠 수강인원을 조사하였습니다. 강의실 별로 배치되어 있는 시간표를 참고하여, 그 수업의 수강인원이 몇 명있는지 조사한후 데이터를 생성했습니다. 이 데이터는 추후에 시간대 별 혼잡도를 파악하는데 이용됩니다.

❏ 데이터 예시 (한 Column기준)

강의실이름	월요일 1-9교시의 각각 수강인원	화요일 1-9교시의 각각 수강인원	수요일 1-9교시의 각각 수강인원	목요일 1-9교시의 각각 수강인원	금요일 1-9교시의 각각 수강인원	근처 엘레베이터 구역	근처 계단 구역	층 수

❏ 데이터 실제 예

A	B	C	D	E	F	G	H	I	J	K	L	M	N
강의실이름	mon1	mon2	mon3	mon4	mon5	mon6	mon7	mon8	mon9	tue1	tue2	tue3	tue4
717		17	17		37	37				47	47		
721		97	97		99	99	78		78	87	90	90	25
722					80	80	84	84		82	82	82	
723	91	91		92			81	81					
726	50	50	75		59	59	45		45			80	80
727			24	24	81	29	38	38		70	70	35	
728			80	80	81	81	80					49	
729		47	47								68	68	
730		24	24				29	29					

3.1.2 노드 사이의 직선거리 (총 소요시간을 예측)

❏ 310관 각층의 단면도를 이용해서 구역을 나누고, 그 구역의 중심점과 각각의 강의실 사이의 직선거리를 구했습니다. 이 거리는 추후에 사용자가 몇 미터를 걸어야지 추측하는데 사용하며, 1분당 사람이 평균 걷는 속도를 이 공식에 더해서 실제 소요시간을 구현하는데 사용했습니다. 지도의 비율상 1cm 당 실제로 7m정도의 거리로 변환 가능하고, 사람은 1분에 70m를 걷기 때문에 총 소요시간을 예측하는데 중요하게 사용했습니다.

❏ 데이터 예시 (한 Column 기준)

장소(해당 구역의 중심)	인접장소	지도 상의 거리

❏ 데이터 실제 예

장소	인접장소	거리(cm)
D	920대형강의실	0.5
D	921중형강의실	0.5
D	917팀플룸	2
D	락카룸	0.5
D	954남자화장실	0.5
D	955여자화장실	0.5
D	912MBA세미나실	2

❑ 엘리베이터 대기 횟수를 위한 분석 차트 예시

3층 시간-구역별 혼잡도

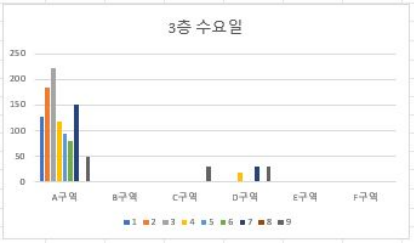
	1	2	3	4	5	6	7	8	9
A구역	112	58	193	214	78	94	157	104	53
B구역	0	0	0	0	0	0	0	0	0
C구역	0	0	41	0	0	0	0	0	11
D구역	0	0	0	17	0	0	0	0	0
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0

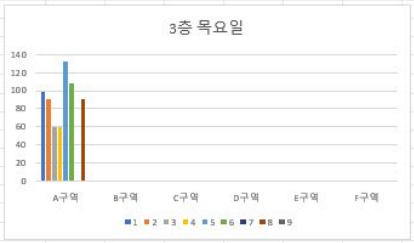
	1	2	3	4	5	6	7	8	9
A구역	262	163	136	136	0	108	169	0	186
B구역	0	0	42	0	0	0	36	0	36
C구역	0	0	0	0	0	0	0	0	0
D구역	0	0	0	0	0	0	0	0	0
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0

	1	2	3	4	5	6	7	8	9
A구역	126	184	222	117	94	80	151	0	47
B구역	0	0	0	0	0	0	0	0	0
C구역	0	0	0	0	0	0	0	0	30
D구역	0	0	0	18	0	0	29	0	29
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0

	1	2	3	4	5	6	7	8	9
A구역	99	91	60	60	132	108	0	90	0
B구역	0	0	0	0	0	0	0	0	0
C구역	0	0	0	0	0	0	0	0	0
D구역	0	0	0	0	0	0	0	0	0
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0

	1	2	3	4	5	6	7	8	9
A구역	0	0	0	104	0	104	89	0	89
B구역	0	0	0	0	0	0	0	0	0
C구역	0	0	0	0	0	0	28	0	28
D구역	0	0	0	97	0	80	98	0	98
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0





5층 시간-구역별 혼잡도

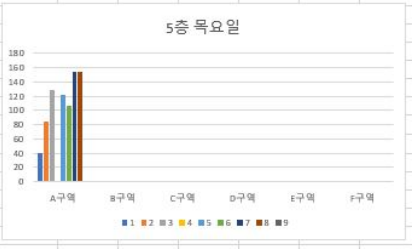
	1	2	3	4	5	6	7	8	9
A구역	0	0	0	0	0	0	0	0	0
B구역	0	0	0	0	0	0	0	0	0
C구역	0	0	0	0	0	0	0	0	0
D구역	0	108	38	70	41	41	88	39	49
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0

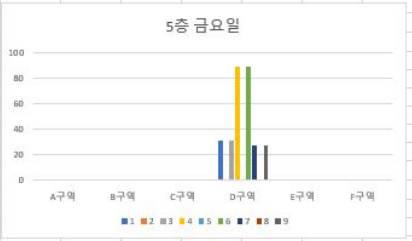
	1	2	3	4	5	6	7	8	9
A구역	99	99	129	176	119	119	116	155	59
B구역	0	171	216	122	0	77	17	0	0
C구역	75	75	121	132	81	86	150	150	0
D구역	0	78	78	0	93	93	97	97	0
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0

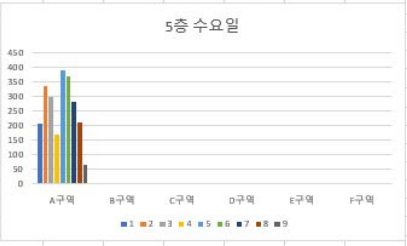
	1	2	3	4	5	6	7	8	9
A구역	204	337	299	168	388	370	280	209	65
B구역	0	0	0	0	0	0	0	0	0
C구역	0	0	0	0	0	0	0	0	0
D구역	0	0	0	0	0	0	0	0	0
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0

	1	2	3	4	5	6	7	8	9
A구역	41	85	129	0	122	107	154	154	0
B구역	0	0	0	0	0	0	0	0	0
C구역	0	0	0	0	0	0	0	0	0
D구역	0	0	0	0	0	0	0	0	0
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0

	1	2	3	4	5	6	7	8	9
A구역	0	0	0	0	0	0	0	0	0
B구역	0	0	0	0	0	0	0	0	0
C구역	0	0	0	0	0	0	0	0	0
D구역	31	0	31	89	0	89	27	0	27
E구역	0	0	0	0	0	0	0	0	0
F구역	0	0	0	0	0	0	0	0	0



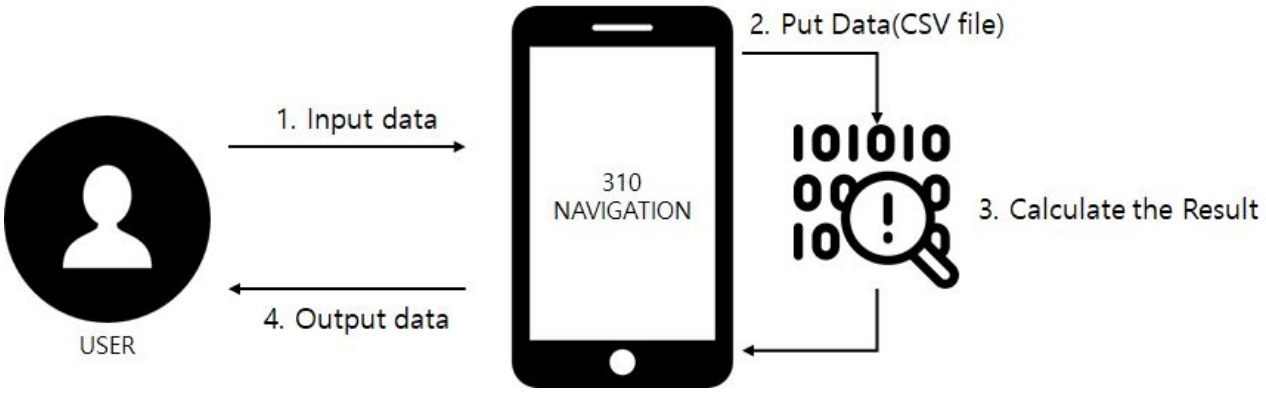




위 사진과 같이 각 층별 혼잡도의 최대, 최소값과 평균값을 분석하여 정원 24명의 Elevator에 탑승한다고 하였을 때, 혼잡한 시간(Boom Time)에 최대 몇번(1~3번) 엘리베이터를 대기 해야 하는지에 대한 정보를 얻었습니다.

4. Implementation

4.1 Entire Architecture of Project



<Entire Architecture Picture>

Project는 크게 4 파트로 분류될 수 있습니다. 각 파트별로 설명을 하면 아래와 같습니다.

1) Input Data

Input Data 파트는 어플리케이션을 사용하는 사용자가 탐색하고자 하는 경로를 입력하는 파트입니다.

크게 2가지 타입의 Input이 들어올 수 있으며, “경유지가 존재”하는 input이거나, “경유지가 존재 하지 않는” input이 올 수 있습니다.

2) Using Data(Put Data)

Put Data part는 본 Project를 위해 제작한 Algorithm을 돌리기 위해 필요한 Data들을 가져오는 작업을 의미합니다. 현재 본 Project에서는 각 층별 강의실에 대한 정보와 이동에 필요한 경로 Data set을 이용하고 있으며 이 Data set은 Github에서 확인할 수 있고, 타입은 CSV File입니다.

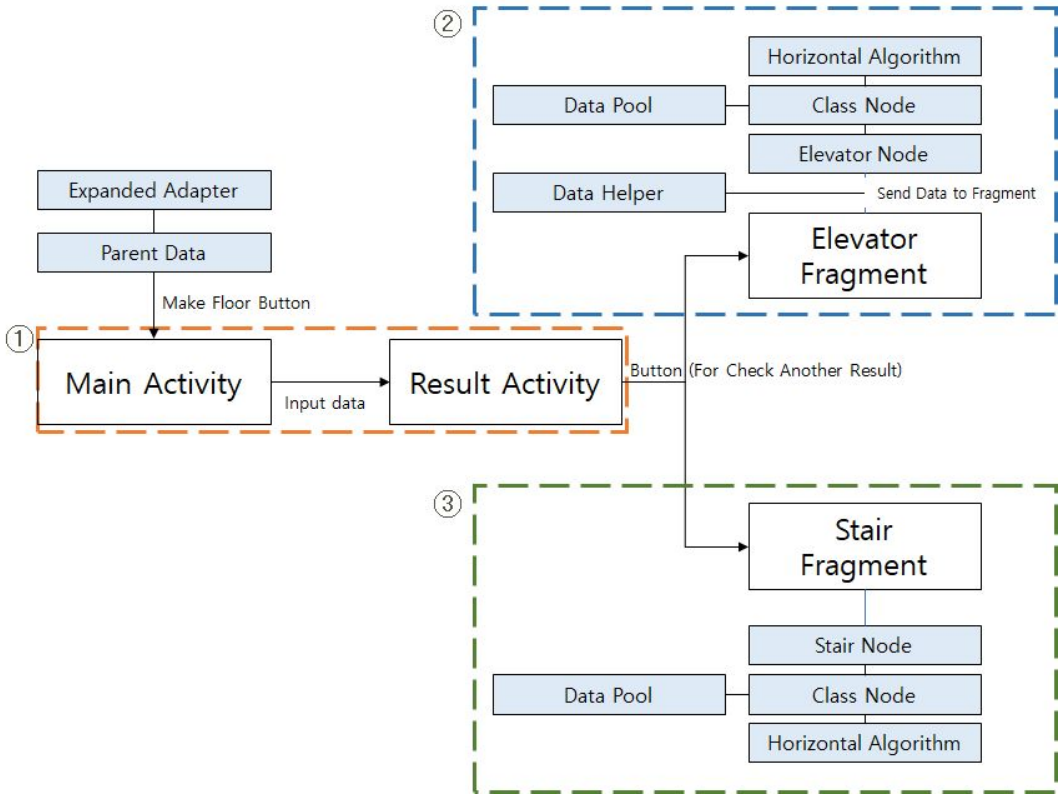
3) Calculate the Result

Calculate the Result파트는 4.3절에서 언급하는 Algorithm을 이용하여 사용자가 입력한 데이터에 대한 최적의 경로를 찾고, 예상 소요시간을 계산하는 파트를 의미합니다. 최종 결과로 ArrayList<ArrayList String>의 형태의 자료구조를 리턴하게 됩니다.

4) Output Data

(3)번 파트에서 얻은 결과를 실제 User가 볼 수 있는 결과 화면에 띄워주는 역할을 합니다. 안드로이드의 “Fragment”를 이용하여 구현하였으며, 버튼을 이용해 계단을 이용하는 경로와 엘리베이터를 이용하는 경로 두가지 Output을 확인해 볼 수 있습니다. 전체적 결과는 6절의 테스트 결과에서 확인하실 수 있습니다.

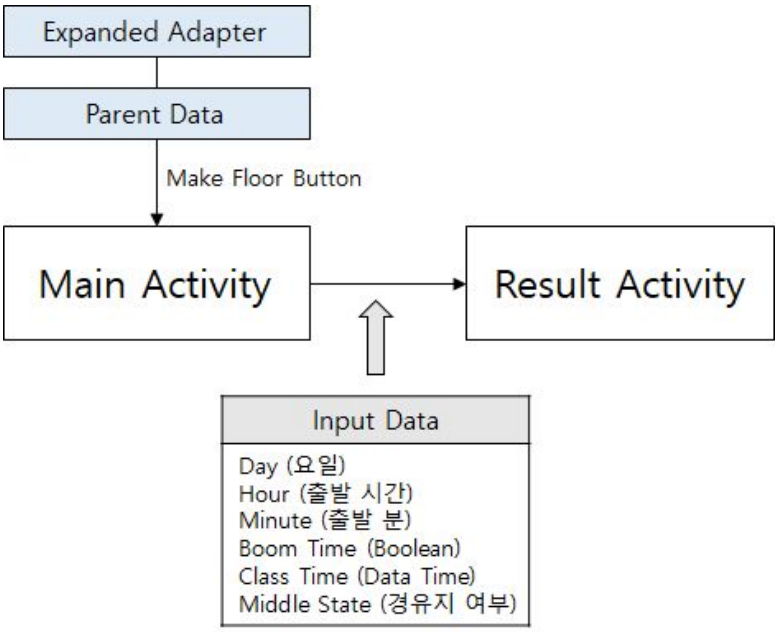
4.2 Class Diagram with Data Structure



<전체 Class Layout>

본 프로젝트는 상단의 <전체 Class Layout> 사진과 같이 총 11개의 클래스로 이루어져 있습니다. 각 파트별 상세한 클래스 설명은 하단과 같습니다.

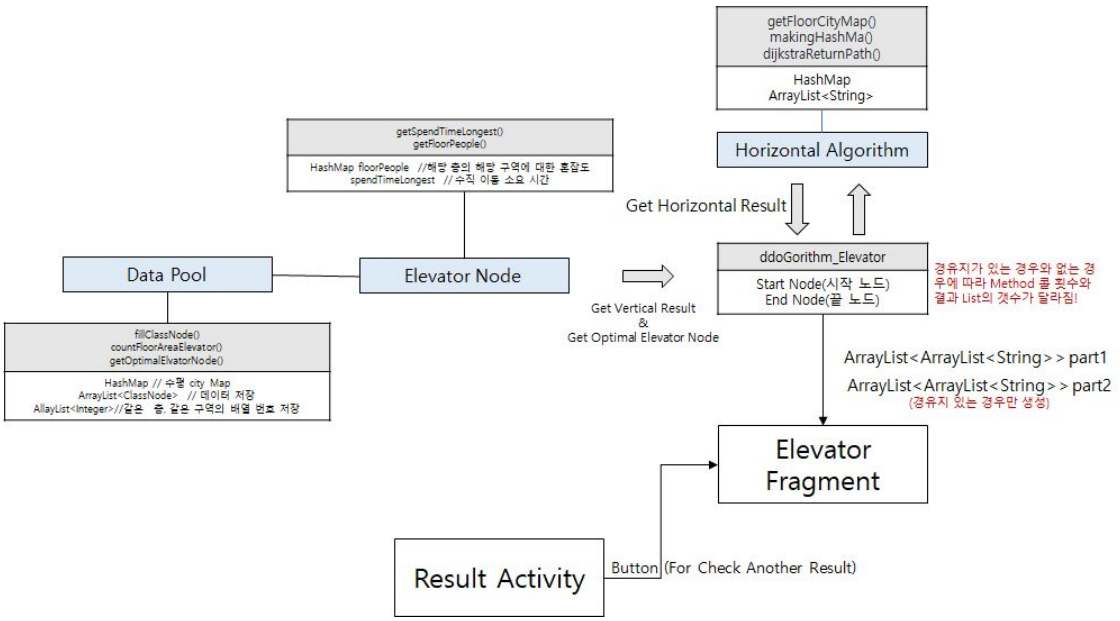
1)



<Main Activity>

Main Activity Class는 User로부터 Input을 받아오는 역할을 합니다. Input Data로는 상단 에 있는 <Main Activity>사진의 Input Data Table의 내용을 받아옵니다.

2)

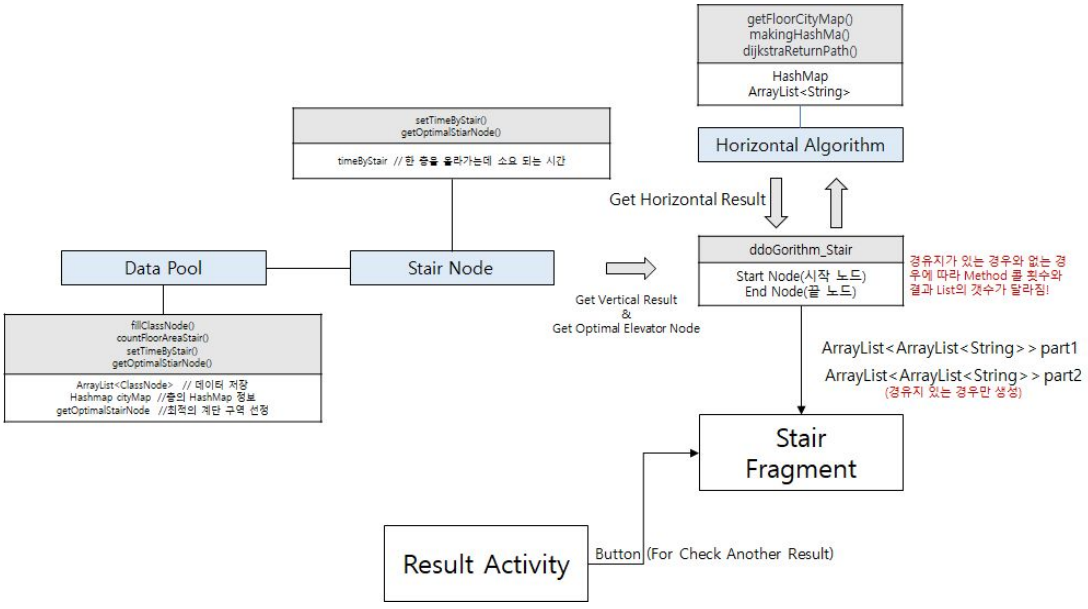


<Result Activity_Elevator Fragment>

Result Activity는 알고리즘 계산 결과를 받아오는 곳으로 크게 두 파트인 Elevator와 Stair파트로 이루어져 있습니다. Elevator는 순수 Elevator를 이용했을 때에 예상되는 이동 시간을 나타내며 Stair는 순수 계단만을 이용했을 때 예상되는 이동 시간을 나타냅니다. 그 중 첫 파트인 Elevator Fragment class를 채우기 위해서는 “Data Pool” class와 “Eelevator Node”, “Horizontal Algorithm” class들이 필요합니다.

각 클래스들의 세부적인 get, set함수의 설명은 생략하였으며, 주로 알고리즘을 계산하거나, 결과값을 리턴하는 함수들만 <Result Activity_Elevator Fragment> 사진에 표기를 해두었습니다.

3)



<Result Activity_Stair Fragment>

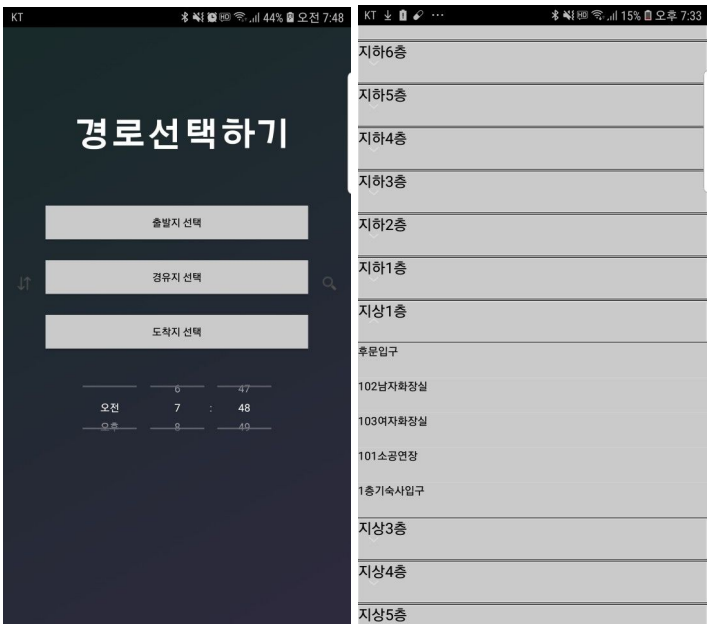
두 번째 파트인 Stair Fragment class를 채우기 위해서는 “Data Pool” class와 “Stair Node”, “Horizontal Algorithm” class들이 필요합니다.

이 부분 역시 각 클래스들의 세부적인 get, set함수의 설명은 생략하였으며, 주로 알고리즘을 계산하거나, 결과값을 리턴하는 함수들만 <Result Activity_Elevator Fragment> 사진에 표기를 해두었습니다.

4.3 UI Diagram

4.2절에서 언급한 각 클래스들의 실제 UI와 사용 설명에 대한 파트입니다.

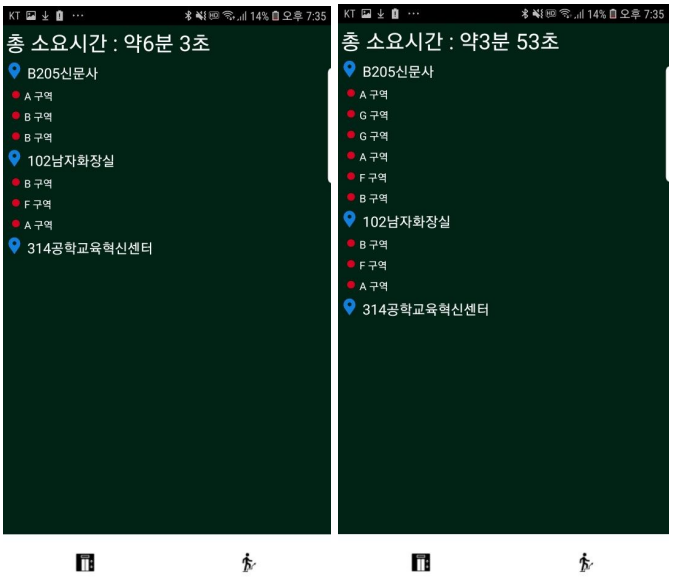
1) Main Activity



<Main Activity>

사용자가 어플리케이션을 시작했을 때 가장 먼저 접하는 화면입니다. 사용자는 “출발지 선택”, “경유지 선택”, “도착지 선택”을 누를 시 <Main Activity>의 오른쪽 사진과 같이 이동하고자 하는 장소 정보를 선택할 수 있습니다. 모든 정보를 선택한 후에, <Main Activity>의 시간 정보를 입력 하고, 검색 버튼을 누르게 되면 Result Activity로 넘어가고 결과를 확인할 수 있습니다.

2) Result Activity



<Result Activity>

사용자가 검색 버튼을 누른 후 본 프로젝트의 Algorithm을 통해 얻은 결과는 <Result Activity>와 같습니다. 위 사진에서 왼쪽은 엘리베이터에 대한 예상 이동 시간, 오른쪽은 계단에 대한 예상 이동 시간 결과입니다. 화면 하단의 엘리베이터와 계단 사진 버튼을 누르면 각각의 결과를 확인할 수 있습니다.

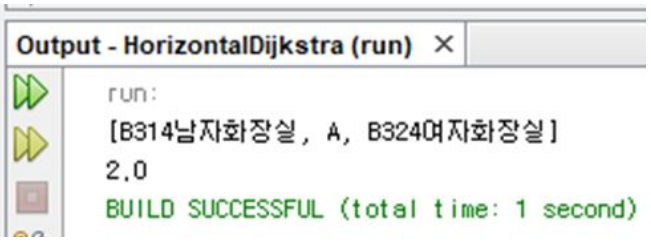
4.4 Algorithm

프로그램에서 사용되는 주요 알고리즘을 설명 드리자면 수평 그리고 수직 이동 알고리즘이 있습니다.

수평 알고리즘은 같은 층 내부에서 A라는 장소에서 B라는 장소까지의 최소 소요 시간과 경로를 리턴해줍니다.

만약 사용자가 1층 A장소를 출발지 그리고 7층 B장소를 목적지로 입력을 하면 수평 알고리즘은 1층 A장소에서 층간 이동이 가능한 장소까지의 최소 소요 시간과 경로를 계산하고 이어 7층에서 층간이동이 가능한 장소에서 B장소까지의 최소 소요 시간과 경로를 계산하게 됩니다. 즉 수평 이동 알고리즘은 사용자 입력에 따른 최소 소요 시간과 경로 계산 과정에서 총 두번 사용됩니다.

수평 이동 알고리즘의 input parameter는 다음과 같습니다. 사는 계단) 그리고 해당 층의 그래프 모델이 저장된 자료구조입니다. 층간 이동 가능 장소가 엘리베이터인 경우의 예시를 들어 보겠습니다. 만약 사용자가 7층 726호를 출발지 그리고 1층의 어느 장소를 목적지로 입력을 하면 수평 알고리즘은 7층 726호에서 탑승 가능한 모든 엘리베이터까지의 거리를 계산합니다. 7층 726호에서 탑승 가능한 엘리베이터란 사실상 7층의 모든 엘리베이터입니다. 7층에는 A, B, C구역 각각 2~5대의 엘리베이용자 출발 장소, 같은 층내의 층간 이동 가능 장소(엘리베이터 또터가 존재합니다. 수평 알고리즘은 사용자가 입력한 출발지와 A, B, C구역 각각의 엘리베이터간의 최소 소요 시간과 경로를 해당 층의 그래프 모델이 저장된 자료구조를 바탕으로 다익스트라(dijkstra) 알고리즘을 이용하여 ArrayList 형식으로 리턴합니다. 이 값은 추후에 최종 경로를 판별하기 위해 필요합니다.



(같은 층 내 두 장소의 최단 경로 및 최소 소요 시간 예시)

수직 알고리즘은 말 그대로 층간 이동을 하는데 걸리는 소요 시간(엘리베이터의 경우 대기 시간 포함)을 계산합니다. 수평 알고리즘과 달리 다익스트라 알고리즘이 사용되지 않으며 팀원들끼리 상의하여 만든 엘리베이터 이동 시간과 대기 시간을 표현한 수학적 공식을 이용합니다. 해당 공식은 앞서 말한 Boom Time에 영향을 받아 Boom Time일 경우와 아닌 경우로 이벤트가 나누어지며 또 해당 이벤트에서 엘리베이터를 총 몇번 보내야(사람이 너무 많아 탈 기회를 놓치는 경우) 하는지에 대한 두 가지 이벤트로 나누어 집니다.

공식은 아래와 같습니다.

Boom Time?	Event	Spend Time 평균 EV 대기 시간 / EV 탑승 후 목적지까지 이동 시간
Yes	EV 기다리고 오면 바로 타는 경우	$\frac{10*(all-1)+1.5*all}{2} + 10*n+1.5*(n-1)$
	EV 기다리고 오면 못 타고 n번 다시 기다리고 타는 경우	$\frac{10*(all-1)+1.5*all}{2} * n + 10*n+1.5*(n+1)$ EV 대기 횟수(최대 3으로 제한)
No	EV 기다리고 오면 바로 타는 경우	$\frac{10*\left(\frac{all}{2}-1\right)+1.5*\frac{all}{2}}{2} + 10*\frac{n}{2}+1.5*(\frac{n}{2}-1)$
	EV 기다리고 오면 못 타고 n번 다시 기다리고 타는 경우	$\frac{10*\left(\frac{all}{2}-1\right)+1.5*\frac{all}{2}}{2} * n + 10*\frac{n}{2}+1.5*(\frac{n}{2}-1)$

(Boom Time 여부와 세부 이벤트로 분류된 엘리베이터 층간 이동 공식)

수직 알고리즘의 input parameter는 다음과 같습니다. 수평 알고리즘에서 계산한 여러 경로들의 층간 이동 가능 장소(엘리베이터 또는 계단), 각 엘리베이터의 정차 가능한 층이 저장된 자료구조, 엘리베이터 대기 시간 (엘리베이터를 보내고 기다려야 하는 횟수) 그리고 Boom Time 여부입니다. 위의 값들이 모두 입력되면 수학적 공식을 이용하여 각 층간 이동 장소에서 층간 이동 소요 시간(엘리베이터는 대기 시간 포함)이 계산됩니다. 층간 이동의 경우 일직선으로 수직 상승 또는 하강을 하므로 경로는 리턴되지 않고 순수 소요 시간만 리턴이 됩니다.

이렇듯, 수평 그리고 수직 알고리즘을 이용하여 출발지까지의 여러개의 경로와 소요 시간이 계산이 됩니다. 최종적으로 여러개의 경로와 소요 시간 중 가장 적은 소요 시간을 가진 경로가 최종 결과로 선택이 됩니다.

간단히 예를 들면 사용자가 7층 726호를 출발지 그리고 1층 기숙사 입구를 목적지로 입력하면:

1) 726호에서 A구역 엘리베이터를 타고 층간 이동을 하고 1층의 A구역 엘리베이터에서 내려서 기숙사 입구까지 걸린 소요 시간 = 10분(가정)

2) 726호에서 B구역 엘리베이터를 타고 층간 이동을 하고 1층의 B구역 엘리베이터에서 내려서 기숙사 입구까지 걸린 소요 시간 = 15분(가정)

3) 726호에서 C구역 엘리베이터를 타고 층간 이동을 하고 1층의 C구역 엘리베이터에서 내려서 기숙사 입구까지 걸린 소요 시간 = 20분(가정)

수평 그리고 수직 이동 알고리즘을 이용하여 위와 같은 세 개의 경로와 소요 시간이 계산이 되며 최종적으로 가장 소요 시간이 적은 1번 경로가 선택이 되어 알고리즘은 해당 경로와 소요 시간을 리턴하며 종료합니다.

5. Data Structure

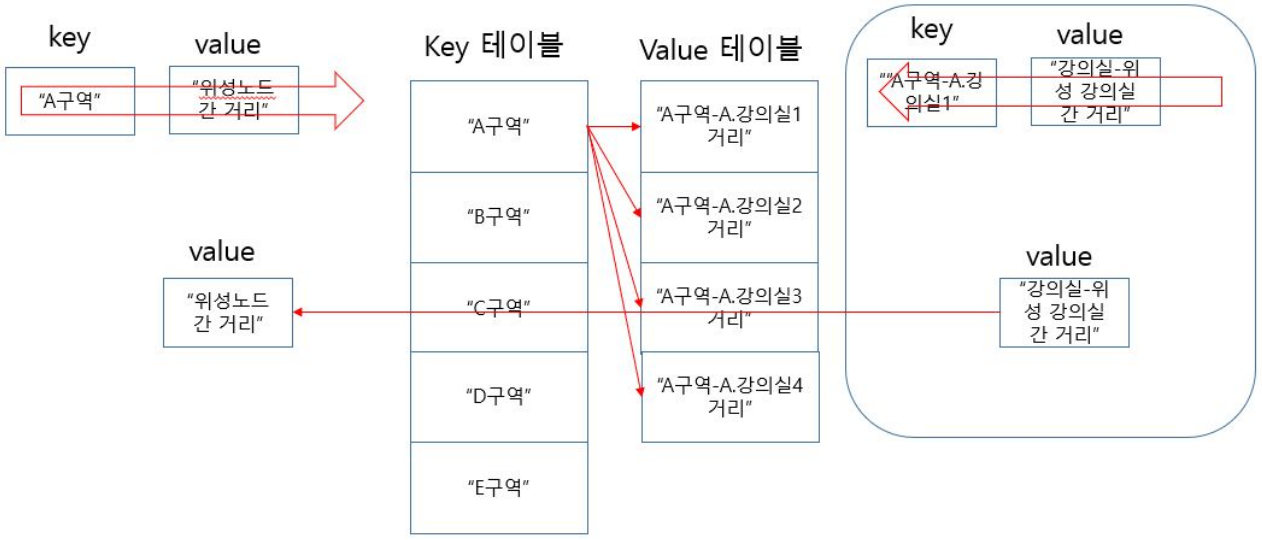
5.1 층간 데이터 저장을 위한 자료구조

수평 알고리즘을 구현하기 위해서는 각 구역을 나누는 A,B,C,D,E,F,G(Big node)들과 그 구역들에 연결되어있는 Small node 들로 나누어져 있습니다.

	A	B	C
1	장소	인접장소	거리(cm)
2	A	B507여자화장실	5
3	A	B506남자화장실	5
4	A	B502대형강의실-3	5
5	A	B501대형강의실-1	7
6	A	B5도서관입구	9
7	A	B507여자화장실	5
8	A	B506남자화장실	5
9	A	B502대형강의실-3	5
10	A	B501대형강의실-1	7
11	A	B5도서관입구	9
12	A	G	4
13	G	A	4
14	B507여자화장실	A	5
15	B506남자화장실	A	5
16	B502대형강의실-3	A	5
17	B501대형강의실-1	A	7
18	B5도서관입구	A	9
19	B507여자화장실	A	5

<Horizontal algorithm 에서 사용되는 엑셀 데이터>

데이터를 활용하기 위해 엑셀에는 층별에 따라 구역을 나누는 A,B,C,D,E,F,G 구역의 Big node들과 이들의 구역별에 연결된 Small node들이 있습니다. 이러한 데이터들을 저희는 엑셀에 저장하여 활용할 수 있도록 하였습니다.



<한 개의 키값에 두개의 Value를 넣기 위해 사용된 이중 해쉬맵 구조1>

```
public Map<String,Map<String,Double>> makingHashMap(int i){
```

<한 개의 키값에 두개의 Value를 넣기 위해 사용된 이중 해쉬맵 구조2>

이러한 데이터들에서 노드간의 최단거리를 구하기 위하여 저희는 이중 해쉬맵 구조를 활용하여 구현하였습니다. 해쉬맵은 Key값과 Value값으로 구현되며 처음 key 값에는 해당 층에 포함된 모든 노드들을 입력해주고 Value값에는 Key와 연결된 노드를 새로운 해쉬맵의 Key값, 그 사이의 거리값을 새로운 해쉬맵의 Value로 넣어 줍니다.

5.2 수직간의 데이터들을 저장하기 위한 자료구조

305								104		104	89		89	B	E	3
310								17			98		98	A	D	3
311								80		80				A	D	3
312			24											A	D	3
315	49		66											A	D	3
316	59													A	D	3
321		33		33							28		28	A	D	3
410											30		30	A	D	4

<Elevator algorithm 에서 사용되는 엑셀 데이터>

층간 이동을 하는 수직 데이터들에 대한 데이터에는 엘리베이터를 효율적으로 타기 위해 해당 층에 대한 시간별 인원, 그리고 해당 위치와 가장 가까운 엘리베이터, 계단 그리고 층에 대한 정보가 담겨져 있습니다.

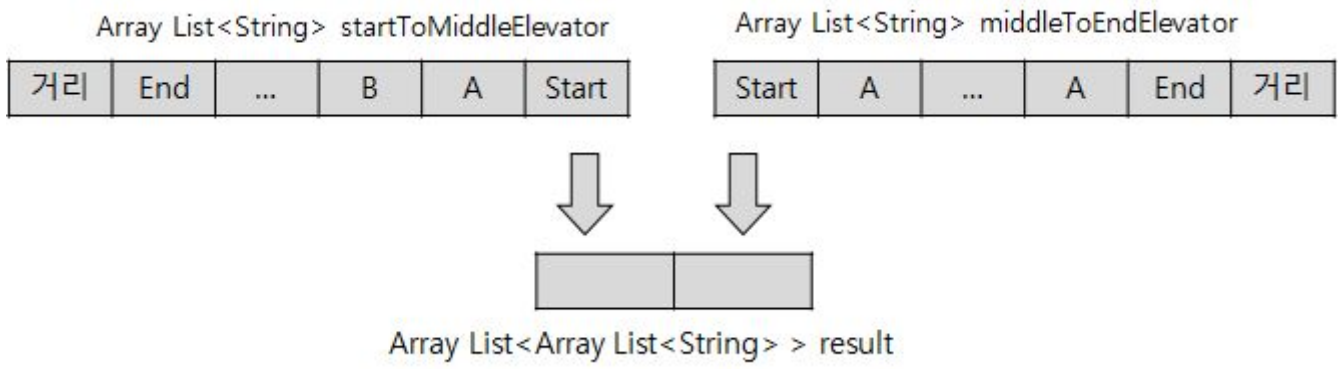
```
String ClassName;
ArrayList<Integer> monday = new ArrayList<>();
ArrayList<Integer> tuesday  = new ArrayList<>();
ArrayList<Integer> wednesday  = new ArrayList<>();
ArrayList<Integer> thursday  = new ArrayList<>();
ArrayList<Integer> friday  = new ArrayList<>();
String nearbyElevator;
String nearbyStair;
String floor;
```

<Horizontal algorithm 에서 사용되는 Class>

이러한 데이터들을 효율적으로 저장하기 위하여, 각각의 요일에 대한 강의정보와 수강인원이 들어간 Arraylist를 ClassNode로 정의하여 구현하였습니다. 이를 통하여 ClassNode에서는 각 층에 대한 정보들을 저장하여, 층간이동을 하는 경우에 엘리베이터의 복잡도를 계산하여 효율적인 엘리베이터 탑승을 돕습니다.

5.3 전체 이동경로를 저장하기 위한 데이터 구조

전체 이동된 데이터들의 경로를 효과적으로 저장하기 위하여 저희는 ArrayList 안에 ArrayList 형태로 수평이동 - 수직이동 - 수평이동을 각각 나누어서 저장하였습니다.



<전체 이동경로를 저장하기 위한 이중 ArrayList>

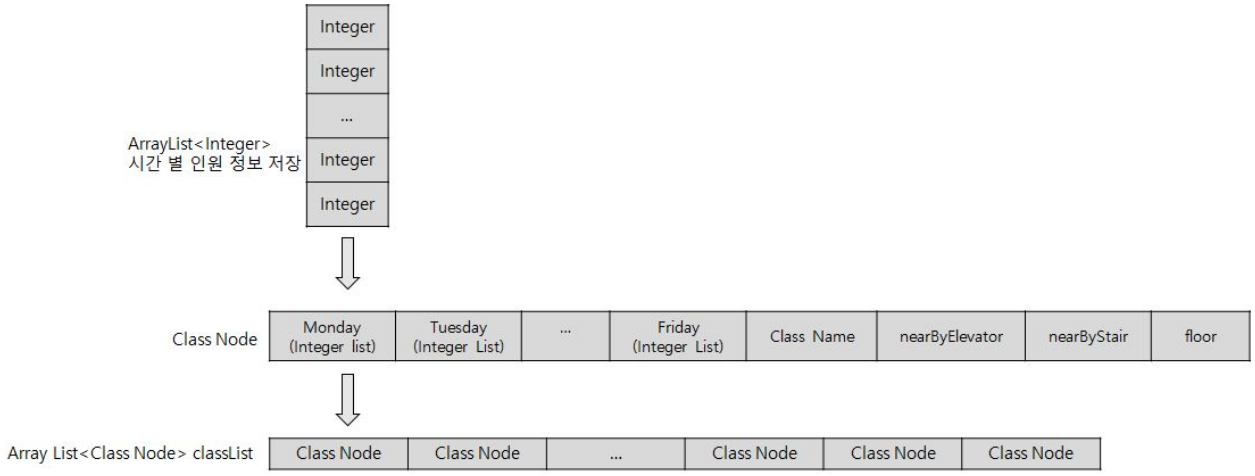
이를 통하여 최초의 ArrayList에는 입구에서 부터 수직으로 이동하기 이전까지의 경로가 저장되어있고, 두번째 ArrayList에는 엘리베이터의 혼잡도를 통한 층간의 수직이동이 저장되어있습니다. 마지막 수평이동 ArrayList에는 엘리베이터/ 혹은 계단의 구역에서 나와 원하는 최종 목적지까지의 경로가 ArrayList로 저장되어있습니다.

```
ArrayList<String> startPath = horizontalAlgorithm.dijkstraReturnPath(start, "G", cityMapStart);

for(Iterator iterator1 = cityMapMiddle.keySet().iterator() ; iterator1.hasNext();){
    String keyName = (String)iterator1.next();
    for(Iterator iterator2 = cityMapEnd.keySet().iterator() ; iterator2.hasNext();){
        if(keyName.equals((String)iterator2.next())){
            if(keyName.equals("D"))
                sameStair.put(0, keyName);
            if(keyName.equals("E"))
                sameStair.put(1, keyName);
            if(keyName.equals("F"))
                sameStair.put(2, keyName);
            if(keyName.equals("G"))
                sameStair.put(3, keyName);
        }
    }
}

StairNode optimalNode = dataPool.getOptimalStairNode(D, E, F, G, "G", sameStair, horizontalAlgorithm, cityMapMiddle);

ArrayList<String> middlePath = horizontalAlgorithm.dijkstraReturnPath("G", optimalNode.getArea(), cityMapMiddle);
ArrayList<String> endPath = horizontalAlgorithm.dijkstraReturnPath(optimalNode.getArea(), end, cityMapEnd);
returnArray.add(startPath);
returnArray.add(middlePath);
returnArray.add(endPath);
```

<전체 이동경로를 저장하기 위한 이중 ArrayList 구현 코드>

코드상에서 startPath, middlePath 그리고 endPath에서는 각각의 구역간의 이동경로를 기존에 만들어낸 CITYMAP과 함께 다익스트라를 돌려 최단거리의 경로를 각각의 ArrayList에 저장한 뒤, 이를 'returnArray'라는 ArrayList에 저장해 주었습니다. 이를 통하여 최종적으로 User가 원하는 이동경로의 최단거리의 총 경로를 탐색할 수 있도록 구현하였습니다.

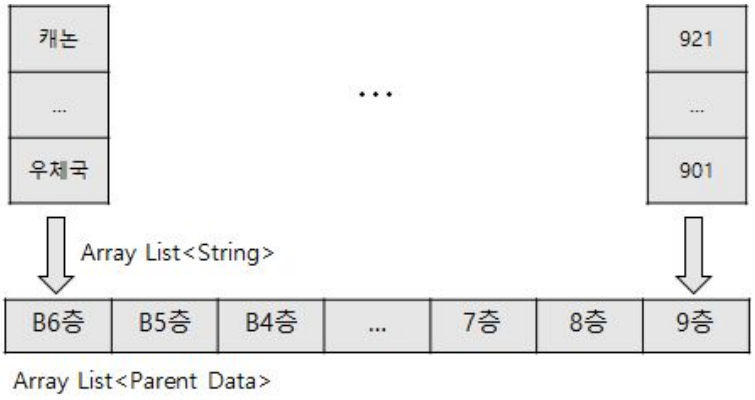
5.4 수직 이동을 위한 Class Node List 데이터 구조



<수직 이동 용 Class Node List 데이터 구조>

수직 이동 정보 저장과 Elevator Node 생성 등 강의실 별 인원 정보를 활용하는 곳에 이용되는 class node list의 자료구조 형태입니다. 전체 강의실에 대한 정보들은 classList라는 class Node Array List에 저장되며, class node Object 하나는 한 강의실에 대한 수업 정보와 수강 인원 정보, 그리고 해당하는 층(Floor), 가장 인접한 계단과 엘리베이터 정보를 담고 있습니다.

5.5 사용자의 UI 버튼 용 자료 저장 데이터 구조



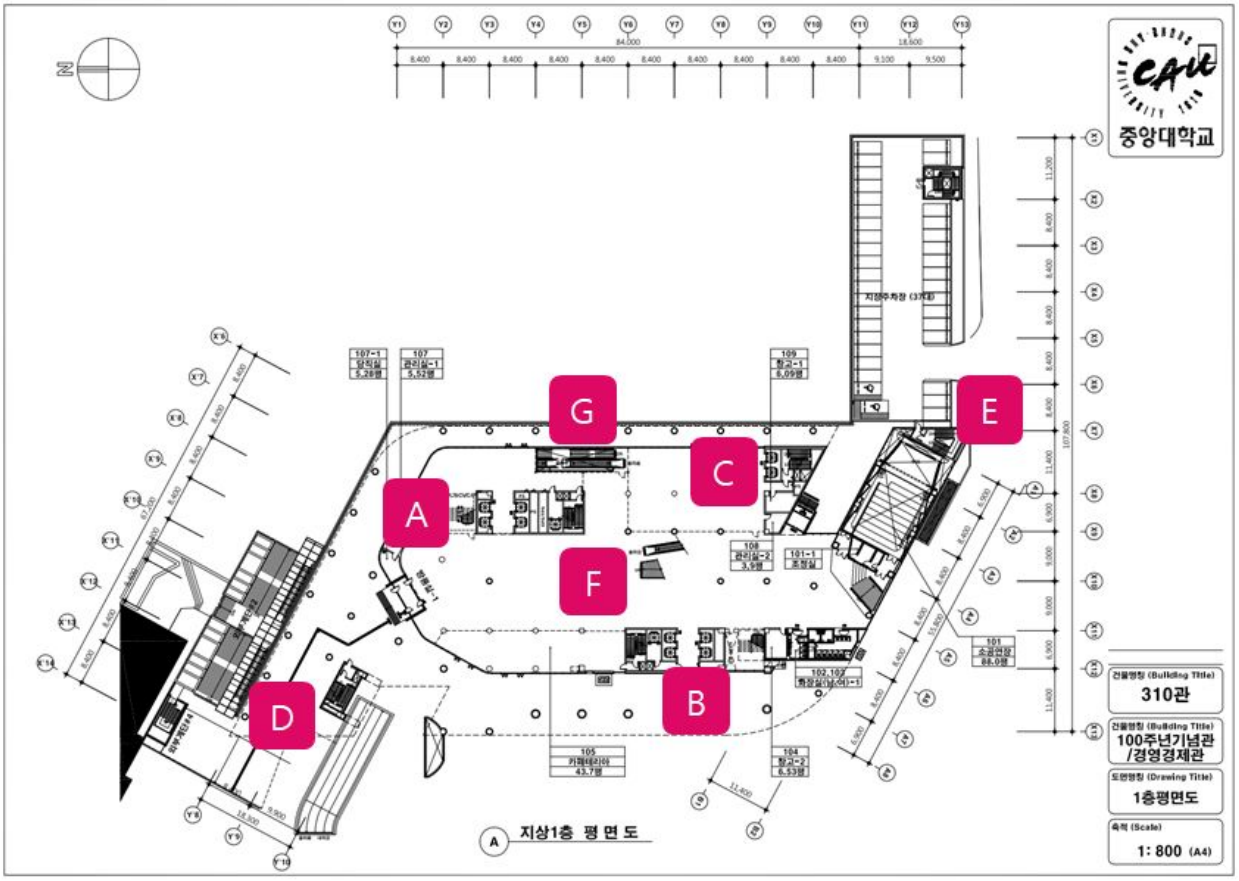
<버튼 생성을 위한 Object-String ArrayList 자료구조>

사용자는 출발지, 경유지, 도착지에 대한 정보를 입력할 때 <버튼 생성을 위한 Object-String ArrayList 자료구조>사진과 같은 자료구조를 가진 UI를 보며 가고자 하는 장소를 입력할 수 있습니다. 버튼을 생성하기 위해 Parent Data라는 Object ArrayList를 크게 갖고 있으며, 각 elements들은 층별 정보를 담고 있는 String ArrayList들을 포함하고 있습니다.

6. Test

6.1 Test case

Subject	Testing Case	Progress
System Error Testing	사용자가 입력한 정보에 대한 처리 테스팅	Done
	경유지 유무에 대한 테스팅	Done
	비정상 입력에 대한 오류 테스팅	Done
Algorithm Testing	수평 이동 알고리즘의 타당성 테스팅	Done
Other Testing	출발지/목적지 변경 기능 테스팅	Done



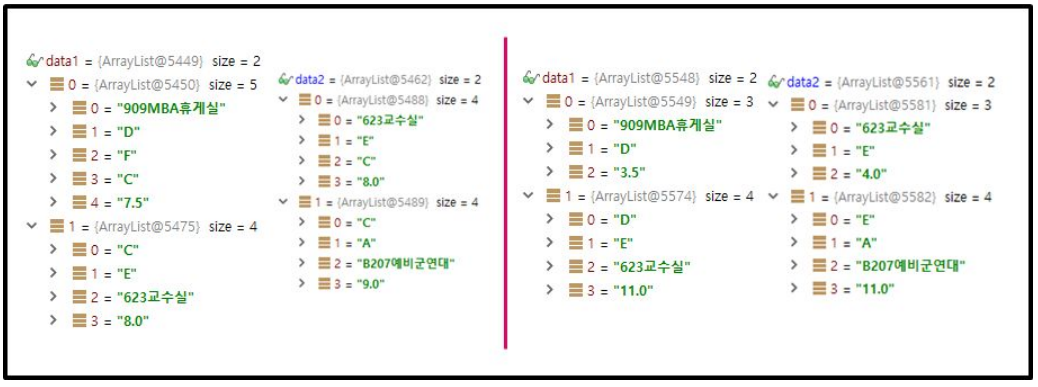
6.2 Test result1

a. 입력했을 때 결과값이 타당하게 나오는 지 비교 (올라가는 경우)

출발지	B205신문사
경유지	102남자화장실
도착지	314공학교육혁신센터

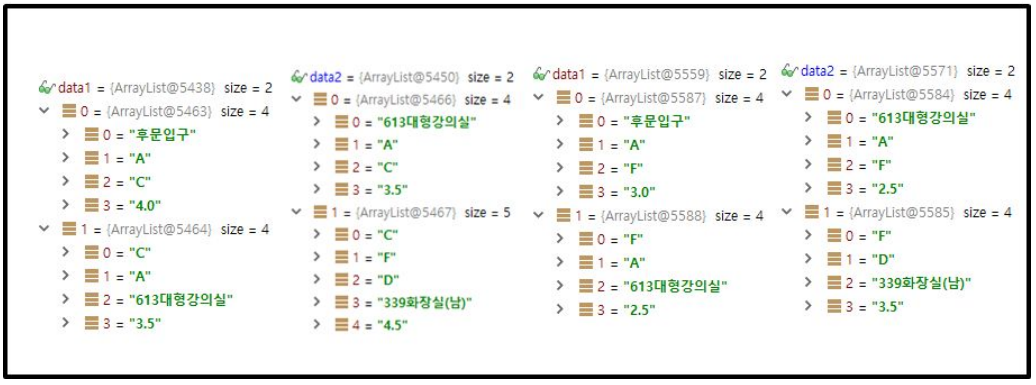
b. 입력했을 때 결과값이 타당하게 나오는 지 비교 (내려가는 경우)

출발지	909MBA휴게실
경유지	623교수실
도착지	예비군연대(B207)



c. 출발지/목적지가 똑같고 경유지가 다를 때 소요시간 비교

출발지	후문입구
경유지	613대형강의실
도착지	339화장실(남)

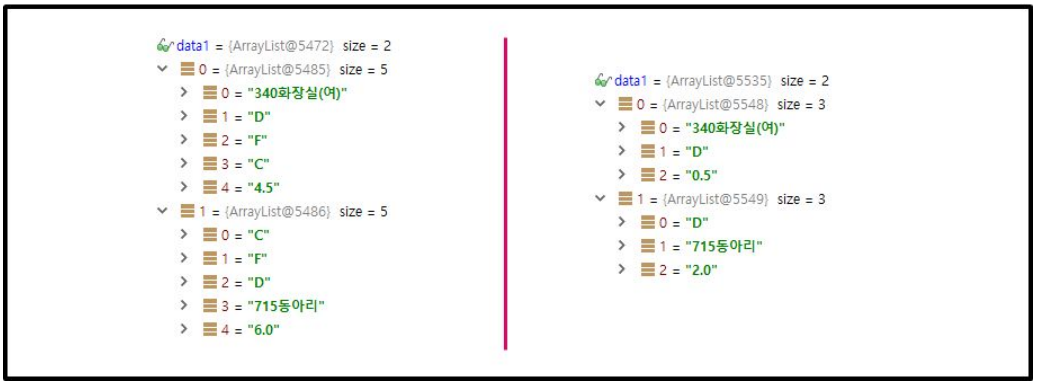


d. 출발지/목적지가 똑같고 경유지가 다를 때 거리 비교

출발지	후문입구
경유지	뚜레주르
도착지	339화장실(남)

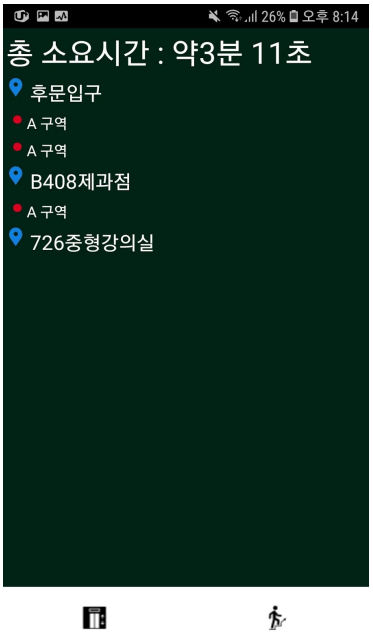
e. 경유지가 존재하지 않는 경우

출발지	310강의실
경유지	X
도착지	917팀플룸



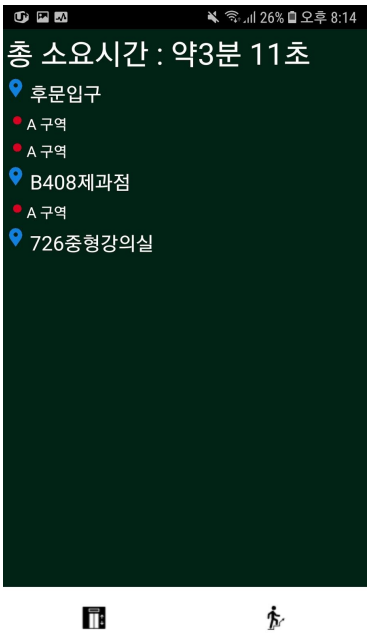
6.3 Test result2

a. 오후 1:30 (Boom Time X) 후문입구 -> B408제과점(뚜레주르) -> 726중형강의실



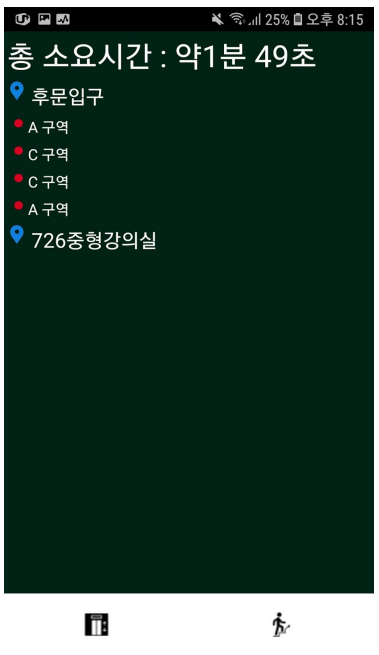
경로선택 엘리베이터 계단

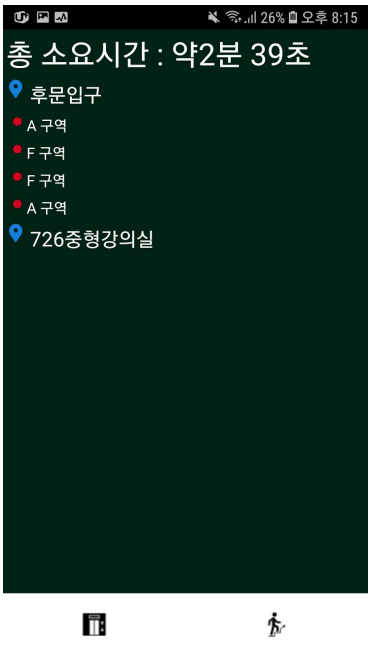
b. 오후12:50 (Boom Time) 후문입구 -> B408제과점(뚜레주르) -> 726중형강의실



경로선택 엘리베이터 계단

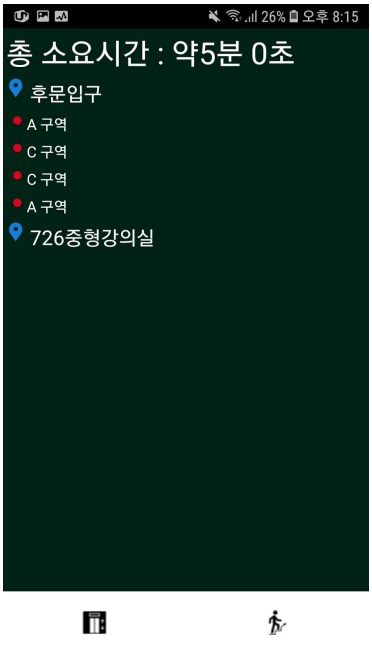
c. 오후 1:30 (Boom Time X) 후문입구 -> 726중형강의실





경로선택 엘리베이터 계단

d. 오후 12:50 (Boom Time) 후문입구 -> 726중형강의실



경로선택 엘리베이터 계단

7. About Result

7.1 Evaluation Process

프로젝트를 진행하기 위해서 우선적으로는 두가지의 데이터가 필요했습니다.

1. 층별 구역간 최단이동 거리
2. 엘리베이터를 사용시 가장 효율적인 엘리베이터 구역 선택 or 계단으로 이동시 해당 위치와 가장 가까운 엘리베이터 구역

이러한 데이터를 효율적으로 저장하고 이를 활용하기 위해서 Excel을 활용하여 두가지의 데이터set을 만들었습니다.

	A	B	C
1	장소	인접장소	거리(cm)
2	A	B507여자화장실	5
3	A	B506남자화장실	5
4	A	B502대형강의실-3	5
5	A	B501대형강의실-1	7
6	A	B5도서관입구	9
7	A	B507여자화장실	5
8	A	B506남자화장실	5
9	A	B502대형강의실-3	5
10	A	B501대형강의실-1	7
11	A	B5도서관입구	9
12	A	G	4
13	G	A	4
14	B507여자화장실	A	5
15	B506남자화장실	A	5
16	B502대형강의실-3	A	5
17	B501대형강의실-1	A	7
18	B5도서관입구	A	9
19	B507여자화장실	A	5

<1번째 DataSet>

305								104		104	89		89	B	E	3
310								17			98		98	A	D	3
311								80		80				A	D	3
312			24											A	D	3
315	49		66											A	D	3
316	59													A	D	3
321		33		33							28		28	A	D	3
410											30		30	A	D	4

<2번째 DataSet>

이를 통하여 저희는 원하는 목표 '310내에 출발지에서 도착지까지 이동시 엘리베이터를 고려하여 최적의 시간정보 제공'을 위한 알고리즘 개발과 데이터 정제를 진행하였습니다. 결과적으로 이를 통하여 다양한 Test결과들과 함께 프로젝트의 목표에 적합한 알고리즘을 구현 할 수 있었습니다.

7.2 Project Evaluation

프로젝트 설계 단계에서 주로 고려야해 할 점은 다음과 같았습니다.

첫째, 시간에 따라 강의실 수강인원의 변동
둘째, 엘리베이터의 운행 예상 횟수 또는 변동
셋째, 시간에 따라서 변동되는 경로

모두 실세계를 반영해야 하는 다이나믹한 요소들이었지만 실세계를 반영하기란 쉽지 않았습니다. 그에 대한 이유는 아래와 같습니다.

첫째, 중앙대학교 학생 개개인의 시간표를 모두 구할 수 없으며
둘째, 이동 경로 사이의 혼잡도를 완벽히 반영할 수 없으며
셋째, 310관 AP 데이터 또한 매우 부실하여 실세계를 반영하기에는 한계가 있다.

이에 실세계를 최대한 반영할 수 있는 유의미한 가정을 함으로써 프로젝트를 진행했습니다. 저희가 한 가정은 다음과 같습니다.

첫째, 강의 시작 전 20분, 시작 후 10분 (hh:40 ~ hh:10) 사이는 그렇지 않은 시간대보다 혼잡하다.
둘째, 사람이 많아 계단을 내려가지 못하거나 경로가 차단되는 경우는 극히 드물고, 엘리베이터 대기시간을 제외한 이동 시간에 혼잡도는 고려하지 않는다.

셋째, 엘리베이터 대기 횟수는 최대 3회이다.
넷째, 해당 층의 혼잡도는 그 시간대의 해당 층에 존재하는 강의의 수강생 수와 비례한다.
다섯째, 사용자의 개인적인 행위(ex> 제과점에서 커피를 산다.)는 고려하지 않는다. 즉, 이동 소요시간만을 고려한다.

여섯째, 사용자의 개인적인 체력은 고려하지 않는다. 즉, 층수와 상관없이 계단 이동시간은 층별로 동일하다.

위와 같은 가정을 기초로 프로젝트를 진행하였으며 모두 구현단계에서 고려, 적용되었습니다.

7.3 Dynamic elements of the project

그래프에서의 다이나믹 요소란 크게 노드, 가중치(weight)가 있습니다. 저희 프로젝트에서는 각 강의실 또는 편의시설, 엘리베이터, 계단을 노드로 설정했습니다. 그리고 노드 사이의 가중치는 노드와 노드간의 거리이며 거리는 건물 단면도 기준의 cm입니다. 강의실, 편의시설, 계단 노드는 정적(Static)이지만 엘리베이터 노드는 동적(Dynamic)이며 프로젝트의 다이나믹한 요소에 큰 영향을 미쳤습니다. 시간대별 혼잡도에 따라서 Node to Node의 경로는 다이나믹하며 이동요소(엘리베이터, 계단)을 선택할 수 있다는 점에서 다이나믹한 요소가 추가되었습니다.

8. Meeting Documentation

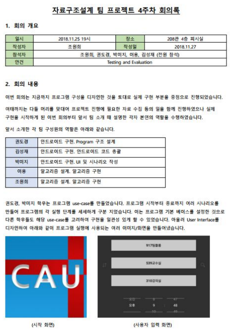
8.1 Meeting Documentation

팀 회의를 할 경우 회의록을 통해 회의 개요(일시, 장소 안건 등)와 내용을 기록했습니다.

회의록에는 당일 누가 어떤 업무를 했고 진척은 어느정도 되었는지 그리고 다음 회의까지 각자 무엇을 준비해야 하는지 등의 내용이 구체적으로 작성되어 있습니다. 회의록을 통해서 팀의 프로젝트 진행 정도를 한눈에 파악하기 쉬어 교수님 그리고 조교님이 참고하기 좋은 자료라고 생각됩니다.

팀 프로젝트 발표 일정이 주차별로 총 5주동안 진행되는것을 고려하여 회의록 또한 주차별로 정리가 되어 있습니다. 실제로 회의를 한 횟수는 더 많지만 항상 안건이 풍부하고 성과가 눈에 띌 정도로 나오는것은 아니기에 주차 별로 핵심 회의 위주로 작성 되었습니다. 회의록은 조원희 학우가 네번 그리고 권도경 학우가 한 번 작성했으며 팀 구성원들과 함께 해당 회의 내용을 요약하고 정리하는 방식으로 진행되었습니다.

주차별 회의 자료는 Github에서 확인해보실 수 있습니다.



자료구조설계 팀 프로젝트 4주차 회의록

1. 회의 개요

일시	2018.11.25 19시	장소	208관 4층 피시실
작성자	조원희	작성일	2018.11.27
참석자	조원희, 권도경, 박미지, 이용, 김성재 (전원 참석)		
안건	Testing and Evaluation		

2. 회의 내용

이번 회의는 지금까지 프로그램 구성을 디자인한 것을 토대로 실제 구현 부분을 중점으로 진행되었습니다.

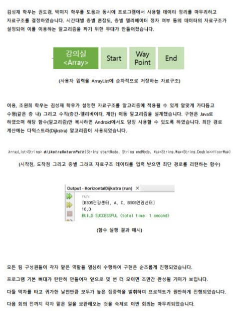
여태까지는 다들 머리를 맞대어 프로젝트 진행에 필요한 자료 수집 등의 일을 함께 진행하였으나 실제 구현을 시작하게 된 이번 회의부터 앞서 팀 소개 때 설명한 각자 본연의 역할을 수행하였습니다.

앞서 소개한 각 팀 구성원의 역할은 아래와 같습니다.

권도경	안드로이드 구현. Program 구조 설계
김성재	안드로이드 구현. 안드로이드 코드 총괄
박미지	안드로이드 구현. UI 및 시나리오 작성
이용	알고리즘 설계. 알고리즘 구현
조원희	알고리즘 설계. 알고리즘 구현

권도경, 박미지 학우는 프로그램 use-case를 만들었습니다. 프로그램 시작부터 종료까지 여러 시나리오를 만들어 프로그램의 각 실행 단계를 세세하게 구분 지었습니다. 이는 프로그램 기본 베이스를 설정한 것으로 다른 학우들도 해당 use-case를 고려하여 구현을 일관성 있게 할 수 있었습니다. 아울러 User Interface를 디자인하여 아래와 같이 프로그램 실행에 사용되는 여러 이미지/화면을 만들어냈습니다.

(시작 화면) (사용자 입력 화면)



김성재 학우는 권도경, 박미지 학우를 도움과 동시에 프로그램에서 사용할 데이터 정리를 마무리하고 자료구조를 결정하였습니다. 시간대별 층별 혼잡도, 층별 엘리베이터 정차 여부 등의 데이터의 자료구조가 설정되어 이를 이용하는 알고리즘을 짜기 위한 무대가 만들어졌습니다.

강의실 <Array> | Start | Way Point | End

(사용자 입력을 ArrayList에 순차적으로 저장하는 자료구조)

이용, 조원희 학우는 김성재 학우가 설정한 자료구조를 알고리즘에 적용될 수 있게 알맞게 가다듬고 수평(같은 층 내) 그리고 수직(층간-엘리베이터, 계단) 이동 알고리즘을 설계했습니다. 구현은 Java로 하였으며 해당 함수(알고리즘)만 복사하면 Android에서도 당장 사용할 수 있도록 하였습니다. 최단 경로 계산에는 다익스트라(Dijkstra) 알고리즘이 사용되었습니다.

```
ArrayList<String> dijkstraReturnPath(String startNode, String endNode, Map<String,Map<String,Double>>floorMap)
```

(시작점, 도착점 그리고 층별 그래프 자료구조 데이터를 입력 받으면 최단 경로를 리턴하는 함수)

```
Output - HorizontalDijkstra (run)
run:
[B305건강센터, A, C, B306인권센터]
10,0
BUILD SUCCESSFUL (total time: 1 second)
```

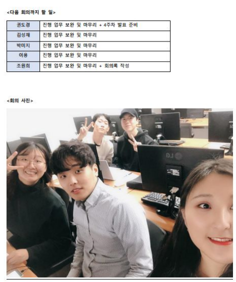
(함수 실행 결과 예시)

모든 팀 구성원들이 각자 맡은 역할을 열심히 수행하여 구현은 순조롭게 진행되었습니다.

프로그램 기본 뼈대가 탄탄히 만들어져 앞으로 몇 번 더 모이면 조만간 완성될 기미가 보입니다.

다들 막차를 타고 귀가한 날인만큼 모두가 높은 집중력을 발휘하여 프로젝트가 원만하게 진행되었습니다.

다음 회의 전까지 각자 맡은 일을 보완해오는 것을 숙제로 이번 회의는 마무리되었습니다.



<다음 회의까지 할 일>

권도경	진행 업무 보완 및 마무리 + 4주차 발표 준비
김성재	진행 업무 보완 및 마무리
박미지	진행 업무 보완 및 마무리
이용	진행 업무 보완 및 마무리
조원희	진행 업무 보완 및 마무리 + 회의록 작성

<회의 사진>

(회의록 작성 예시)