
System Model (Class Diagram) Document

제 2 조 NaLang

조원 : 201402387 이동원

201402330 김연훈

201604136 김노은

지도교수: 정상근 (서명)

Document Revision History

REV#	DATE	AFFECTEDSECTION	AUTHOR
1	2020507	Application, View model	김연훈
2	2020507	DataBase	김노은
3	2020507	Fragement	이동원

Table of Contents

1. INTRODUCTION	5
1.1. OBJECTIVE	5
2. CLASS DIAGRAM	6
3. USE CASE와 CLASS 간의 관계	7
3.1. UC: 회원가입	7
3.2. UC: 메모 작성	8
3.3. UC: 메모 수정	9
3.4. UC: 메모 삭제	10
3.5. UC: 태그 편집	11
3.6. UC: 메모 백업	12
3.7. UC: 분석 백업	13
3.8. UC: 분석 요청	14
3.9. UC: 고객센터 문의	15
4. CLASS 명세	16

List of Figure

FIGURE 1 – SYSTEM CLASS DIAGRAM	6
---------------------------------------	---

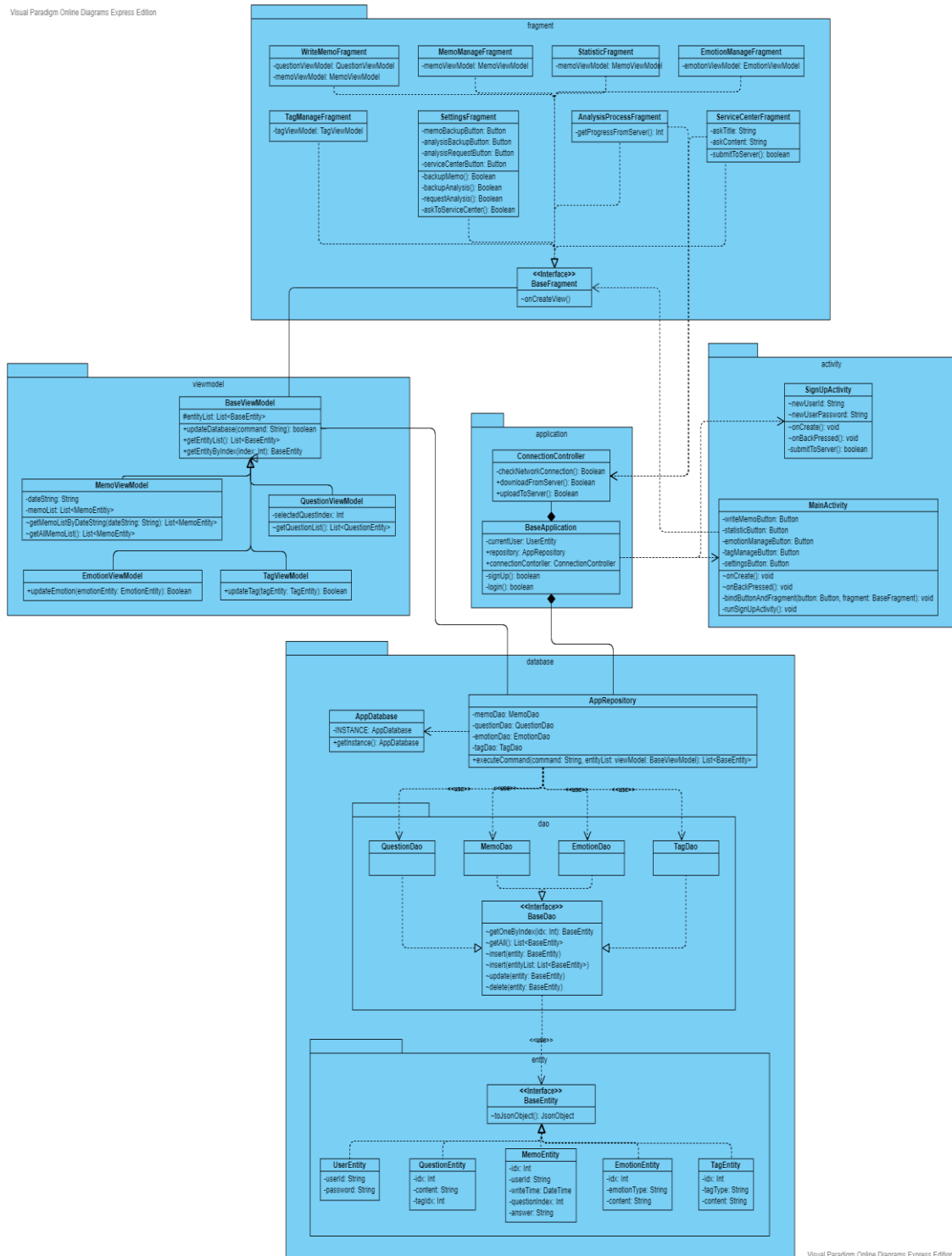
1. Introduction

1.1. Objective

이 문서는 자연어 처리 모델을 활용하여 사용자의 기록들로부터 사용자와 관련된
유의미한 정보들을 추출하여 분석하고 통계로 제시하는 메모 어플리케이션을
구성하는 클래스들 및 각 클래스 내의 함수에 대한 명세를 포함한다.

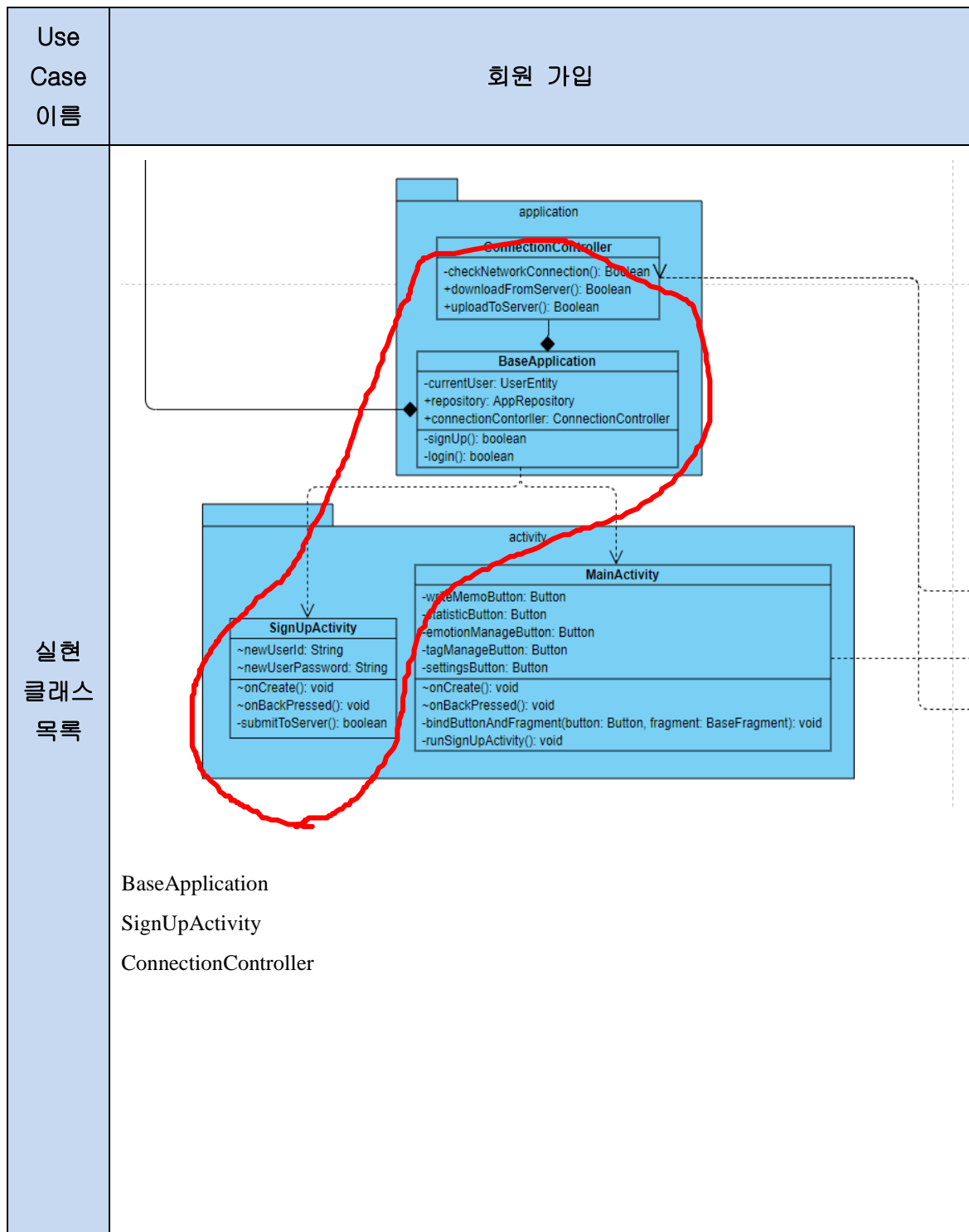
2. Class Diagram

Figure 1 – System Class Diagram



3. Use Case와 Class 간의 관계

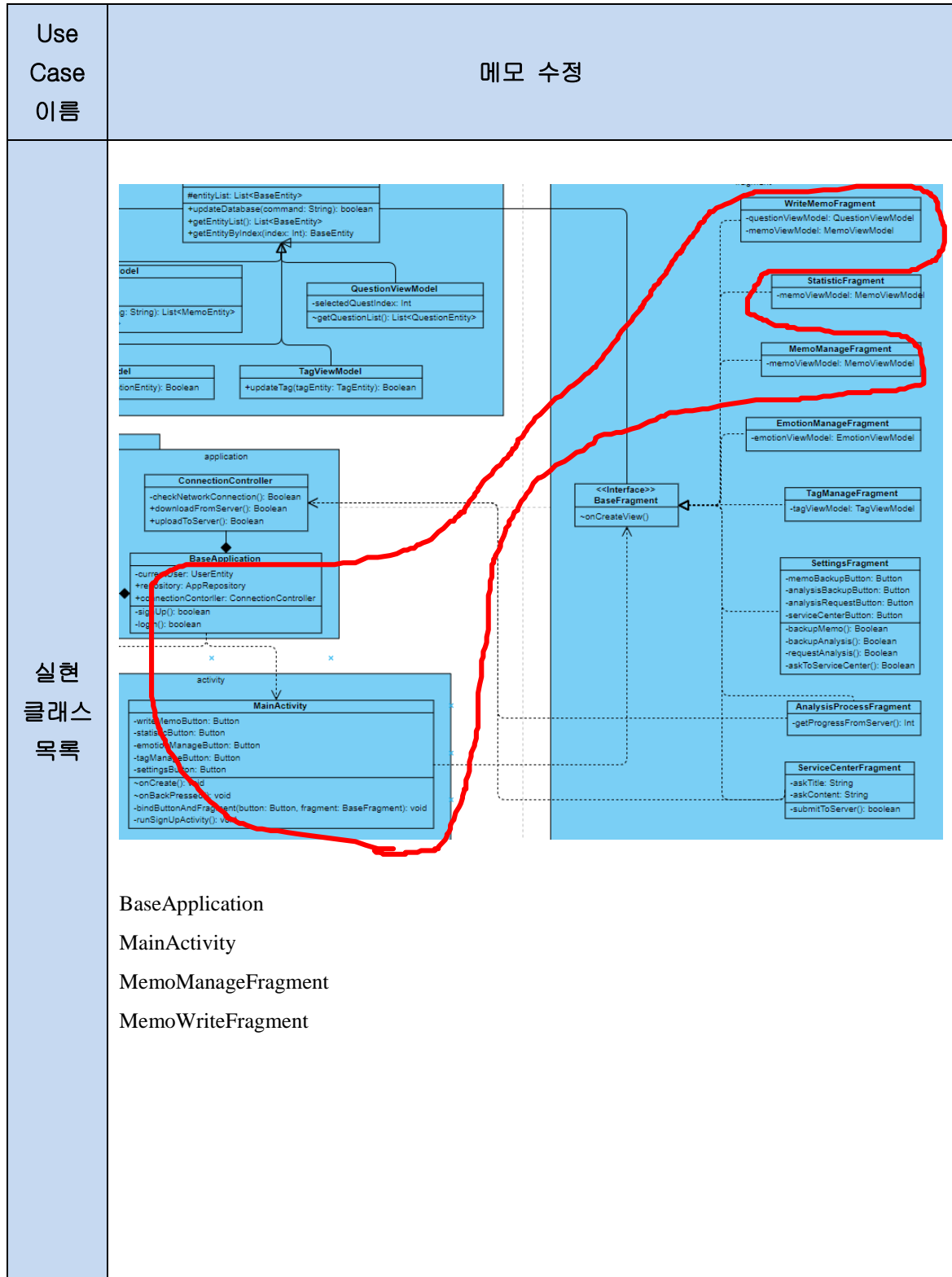
3.1. UC: 회원 가입



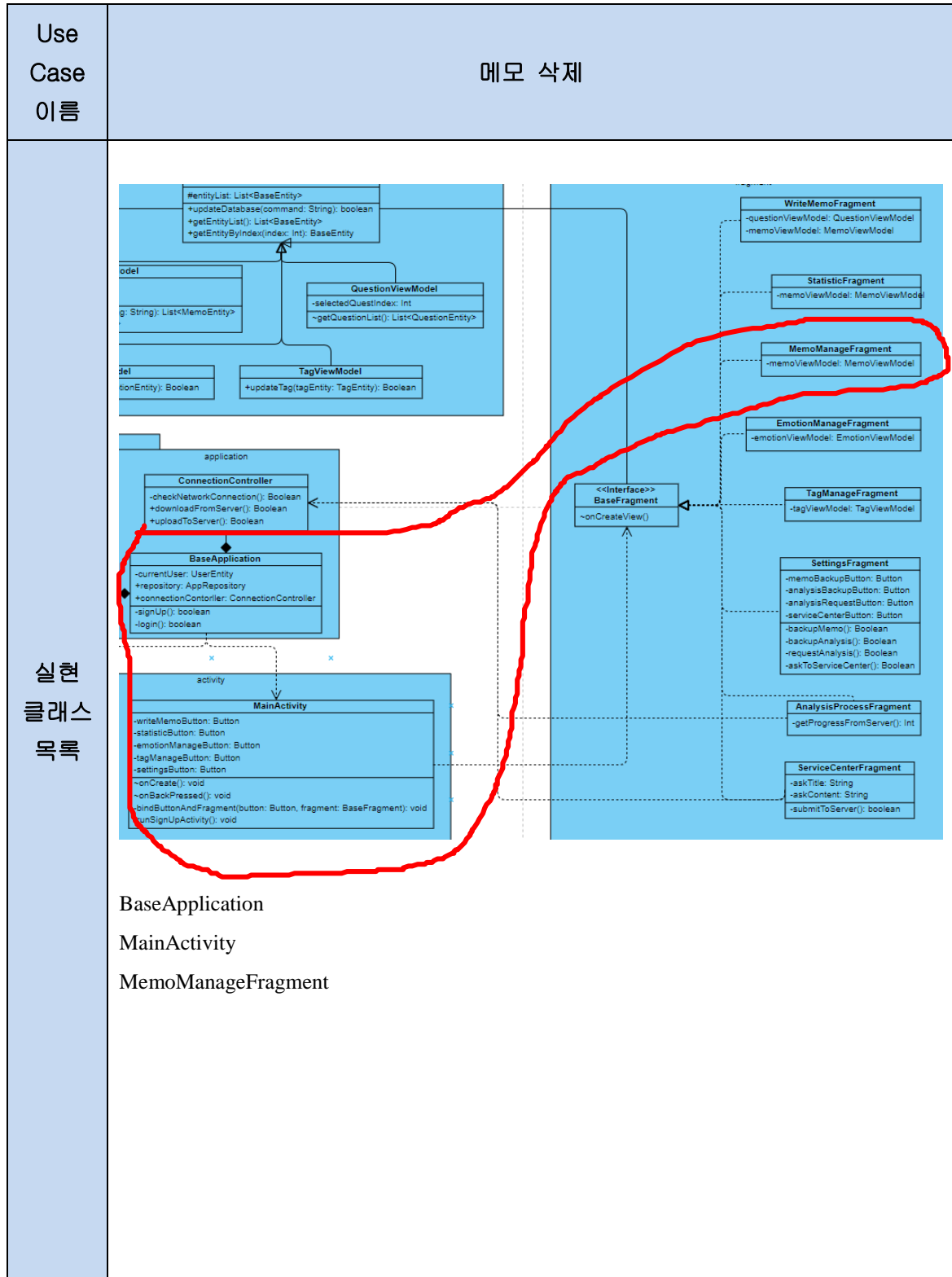
3.2. UC: 메모 작성

Use Case 이름	메모 작성
실현 클래스 목록	<pre> classDiagram class BaseEntity { +updateDatabase(command: String): boolean +getEntityList(): List<BaseEntity> +getEntityByIndex(index: Int): BaseEntity } class QuestionEntity { +selectedQuestIndex: Int ~getList(): List<QuestionEntity> } class TagEntity { +updateTag(tagEntity: TagEntity): Boolean } class BaseViewModel { +g: String: List<MemoEntity> } class QuestionViewModel { ~selectedQuestIndex: Int ~getList(): List<QuestionEntity> } class TagViewModel { +updateTag(tagEntity: TagEntity): Boolean } class ConnectionController { -checkNetworkConnection(): Boolean +downloadFromServer(): Boolean +uploadToServer(): Boolean } class BaseApplication { -currentUser: UserEntity +repository: AppRepository +connectionController: ConnectionController -signUp(): boolean -login(): boolean } class MainActivity { -writeMemoButton: Button -statisticButton: Button -emotionManageButton: Button -tagManageButton: Button -settingsButton: Button ~onCreate(): void ~onBackPressed(): void ~bindButtons(fragment: Button, fragment: BaseFragment): void ~runSignUpActivity(): void } class WriteMemoFragment { -questionViewModel: QuestionViewModel -memoViewModel: MemoViewModel } class StatisticFragment { -memoViewModel: MemoViewModel } class MemoManageFragment { -memoViewModel: MemoViewModel } class EmotionManageFragment { -emotionViewModel: EmotionViewModel } class TagManageFragment { -tagViewModel: TagViewModel } class SettingsFragment { -memoBackupButton: Button -analysisBackupButton: Button -serviceCenterButton: Button -backupMemo(): Boolean -backupAnalysis(): Boolean -requestAnalysis(): Boolean -askToServiceCenter(): Boolean } class AnalysisProcessFragment { -getProgressFromServer(): Int } class ServiceCenterFragment { -askTitle: String -askContent: String -submitToServer(): boolean } <<interface>> class BaseFragment { ~onCreateView() } BaseEntity < -- QuestionEntity BaseEntity < -- TagEntity BaseViewModel < -- QuestionViewModel BaseViewModel < -- TagViewModel BaseApplication o--> ConnectionController MainActivity ..> WriteMemoFragment MainActivity ..> StatisticFragment MainActivity ..> MemoManageFragment MainActivity ..> EmotionManageFragment MainActivity ..> TagManageFragment MainActivity ..> SettingsFragment MainActivity ..> AnalysisProcessFragment MainActivity ..> ServiceCenterFragment BaseApplication -- > MainActivity BaseFragment <..> WriteMemoFragment BaseFragment <..> StatisticFragment BaseFragment <..> MemoManageFragment BaseFragment <..> EmotionManageFragment BaseFragment <..> TagManageFragment BaseFragment <..> SettingsFragment BaseFragment <..> AnalysisProcessFragment BaseFragment <..> ServiceCenterFragment </pre> <p>The UML class diagram illustrates the architecture for the 'Memo Writing' use case. It features several key components:</p> <ul style="list-style-type: none"> Entities and View Models: At the top left, there are three base entity classes (<code>BaseEntity</code>, <code>QuestionEntity</code>, <code>TagEntity</code>) and their corresponding view models (<code>BaseViewModel</code>, <code>QuestionViewModel</code>, <code>TagViewModel</code>). These are interconnected by solid lines. Application Layer: In the center, the <code>application</code> package contains <code>ConnectionController</code>, <code>BaseApplication</code>, and <code>MainActivity</code>. <code>BaseApplication</code> has a dependency on <code>ConnectionController</code>. Fragment Layer: On the right, the <code>fragment</code> package lists eight fragments: <code>WriteMemoFragment</code>, <code>StatisticFragment</code>, <code>MemoManageFragment</code>, <code>EmotionManageFragment</code>, <code>TagManageFragment</code>, <code>SettingsFragment</code>, <code>AnalysisProcessFragment</code>, and <code>ServiceCenterFragment</code>. Each fragment implements the <code><<interface>> BaseFragment</code>. Dependencies: Dashed arrows show dependencies from <code>MainActivity</code> to each of the eight fragments. Additionally, dashed arrows point from <code>MainActivity</code> back to <code>BaseApplication</code> and <code>ConnectionController</code>. Red Circle Highlight: A prominent red circle encircles the <code>WriteMemoFragment</code> class and its associated dependencies, indicating its primary role in the use case.

3.3. UC: 메모 수정



3.4. UC: 메모 삭제



3.5. UC: 태그 편집

Use Case 이름	태그 편집
<p>실행 클래스 목록</p>	<p>BaseApplication</p> <p>MainActivity</p> <p>TagManageFragment</p>

3.6. UC: 메모 백업

Use Case 이름	메모 백업
실현 클래스 목록	<pre>classDiagram package application { class ConnectionController { -checkNetworkConnection() Boolean +downloadFromServer() Boolean +uploadToServer() Boolean } class BaseApplication { -currentUser: UserEntity +repository: AppRepository +connectionController: ConnectionController -signUp() boolean -login() boolean } } package activity { class MainActivity { -writeMemoButton: Button -statisticButton: Button -emotionManageButton: Button -tagManageButton: Button -settingsButton: Button -onCreate(): void -onClickPressed(): void -bindButtonAndLayout(button: Button, fragment: BaseFragment): void -runSignUpActivity(): void } } package fragment { class BaseFragment { <<interface>> -onCreateView() } class EmotionManageFragment { -emotionViewModel: EmotionViewModel } class TagManageFragment { -tagViewModel: TagViewModel } class SettingsFragment { -memoBackupButton: Button -analysisBackupButton: Button -analysisRequestButton: Button -serviceCenterButton: Button -backupMemo(): Boolean -backupAnalysis(): Boolean -requestAnalysis(): Boolean -askToServiceCenter(): Boolean } class AnalysisProcessFragment { -getProgressFromServer(): Int } class ServiceCenterFragment { -askTitle: String -askContent: String -submitToServer(): boolean } } ConnectionController --> BaseApplication BaseApplication --> MainActivity BaseApplication --> BaseFragment BaseApplication --> EmotionManageFragment BaseApplication --> TagManageFragment BaseApplication --> SettingsFragment BaseApplication --> AnalysisProcessFragment BaseApplication --> ServiceCenterFragment</pre> <p>BaseApplication</p> <p>ConnectionController</p> <p>MainActivity</p> <p>SettingsFragment</p>

3.7. UC: 분석 백업

Use Case 이름	분석 백업
<p>실현 클래스 목록</p>	<p>BaseApplication</p> <p>ConnectionController</p> <p>MainActivity</p> <p>SettingsFragment</p>

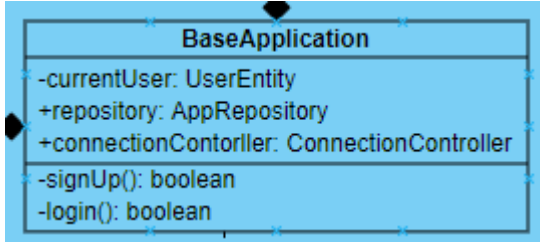
3.8. UC: 분석 요청

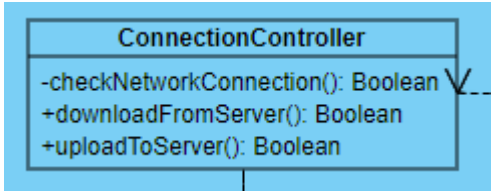
Use Case 이름	분석 요청
실현 클래스 목록	

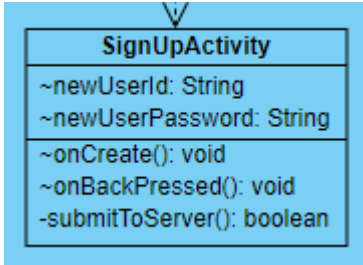
3.9. UC: 고객 센터 문의

Use Case 이름	고객 센터 문의
<p>실현 클래스 목록</p>	<p>BaseApplication</p> <p>ConnectionController</p> <p>MainActivity</p> <p>SettingsFragment</p> <p>ServiceCenterFragment</p>

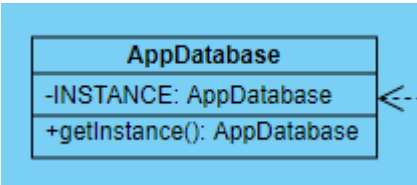
4. Class 명세

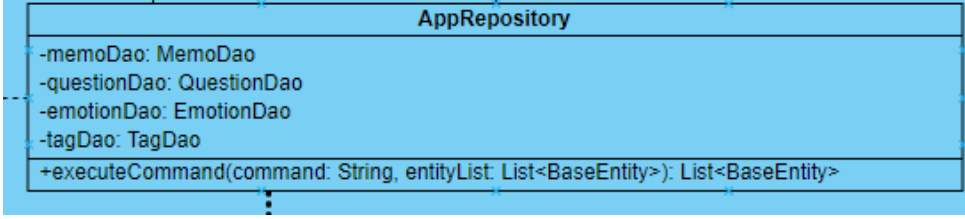
BaseApplication				
Class Diagram				
Responsibility	<p>Application 실행시 Background 의 역할을 하는 클래스이다.</p> <p>사용자 정보, 내장 DB 접근을 위한 저장소, 연결 제어 클래스를 멤버로 갖고 있으며 이 클래스 위에서 여러 Activity 를 실행한다.</p>			
Attribute	Type	Name	Description	
	UserEntity	currentUser	어플리케이션 사용자 객체	
	AppRepository	repository	내장 DB 접근을 위한 저장소	
	ConnectionController	connectionController	서버와의 연결을 위한 객체	
Operation	Return Type	Method Name	Parameter Type	Parameter Name
	boolean	signUp		
	Description	회원가입 화면으로 이동하기 위해 SignUpActivity 를 실행시키는 메소드.		
	Return Type	Method Name	Parameter Type	Parameter Name
	boolean	login		
	Description	어플리케이션의 MainActivity 를 실행시키는 메소드		

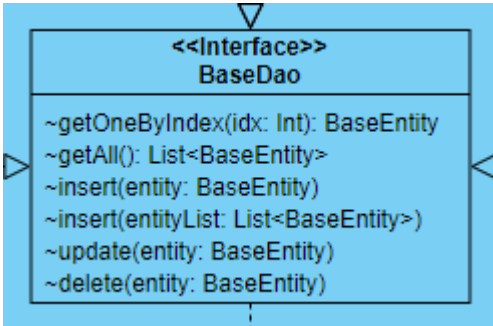
ConnectionController			
Class Diagram	 <pre> classDiagram class ConnectionController { -checkNetworkConnection() Boolean +downloadFromServer() Boolean +uploadToServer() Boolean } </pre>		
Responsibility	<p>어플리케이션이 서버 프로그램과 통신하기 위한 클래스이다. 서버와 통신이 가능한지 체크하는 메소드와 데이터를 송/수신 하는 메소드가 포함되어 있다.</p>		
Attribute	Type	Name	Description
Operation	Return Type	Method Name	Parameter Type
	boolean	checkNetwork Connection	Parameter Name
	Description	어플리케이션이 서버 프로그램과 통신할 수 있는지 확인하는 메소드이다. 서버와의 데이터 송수신이 가능한 경우 true, 불가능한 경우 false 를 리턴한다.	
	Return Type	Method Name	Parameter Type
	boolean	download FromServer	Parameter Name
	Description	서버로부터 다운로드 받는 메소드이다.	
	Return Type	Method Name	Parameter Type
	boolean	upload ToServer	Parameter Name
	Description	서버로 업로드하는 메소드이다.	
	Return Type	Method Name	Parameter Type

SignUpActivity			
Class Diagram	 <pre> classDiagram class SignUpActivity { ~newUserId: String ~newUserPassword: String ~onCreate(): void ~onBackPressed(): void ~submitToServer(): boolean } </pre>		
Responsibility	<p>회원가입을 위한 화면이 출력되는 Activity 이다.</p> <p>사용자의 ID 와 Password 를 입력받아 서버로 전송하는 역할을 한다.</p>		
Attribute	Type	Name	Description
Operation	Return Type	Method Name	Parameter Type
	boolean	onCreate	Parameter Name
	Description	<p>화면으로 출력하기 위해 동작하는 메소드이다.</p> <p>사용자의 아이디와 비밀번호를 입력받기 위한 대화창을 화면에 출력한다.</p>	
	Return Type	Method Name	Parameter Type
	boolean	onBackPressed	Parameter Name
	Description	<p>뒤로가기 버튼을 눌렀을 때의 동작을 위한 메소드이다.</p>	
	Return Type	Method Name	Parameter Type
	boolean	uploadToServer	Parameter Name
	Description	<p>서버로 업로드하는 메소드이다.</p>	
	Return Type	Method Name	Parameter Type

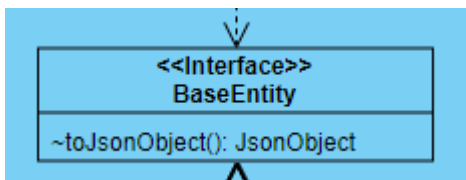
		메모 관리 화면에서 뒤로가기를 누르면 어플리케이션을 종료시킨다.		
	Return Type	Method Name	Parameter Type	Parameter Name
	void	bintButton AndFragment	Button, BaseFragment	button, fragment
	Description	버튼과 Fragment(화면)을 연결하는 메소드이다. onCreate 메소드 내에서 실행되며 화면에 나타나는 각 버튼들을 눌렀을 때 해당 버튼에 맞는 화면이 출력되도록 연결시키는 기능을 한다.		

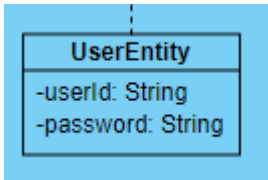
AppDatabase															
Class Diagram	 <pre> classDiagram class AppDatabase { -INSTANCE: AppDatabase +getInstance(): AppDatabase } AppDatabase --> AppDatabase : Singleton </pre>														
Responsibility	<p>어플리케이션의 내장 데이터베이스를 사용하기 위한 클래스이다. 어플리케이션을 처음 사용하는 경우 데이터베이스를 생성하며, 이전에 사용한 적이 있는 경우 데이터베이스를 불러오는 기능을 한다.</p>														
Attribute	Type	Name	Description												
	AppDatabase	INSTANCE	클래스를 싱글톤 객체로 사용하기 위한 필드												
Operation	<table border="1"> <thead> <tr> <th>Return Type</th><th>Method Name</th><th>Parameter Type</th><th>Parameter Name</th></tr> </thead> <tbody> <tr> <td>AppDatabase</td><td>getInstance</td><td></td><td></td></tr> <tr> <td>Description</td><td colspan="3"> 어플리케이션 파일 내에 기존에 사용하던 데이터베이스가 없는 경우 데이터베이스를 생성한 후 그 데이터베이스를 리턴하며, 기존에 사용하던 데이터베이스가 있는 경우 그대로 불러와 리턴한다. </td></tr> </tbody> </table>			Return Type	Method Name	Parameter Type	Parameter Name	AppDatabase	getInstance			Description	어플리케이션 파일 내에 기존에 사용하던 데이터베이스가 없는 경우 데이터베이스를 생성한 후 그 데이터베이스를 리턴하며, 기존에 사용하던 데이터베이스가 있는 경우 그대로 불러와 리턴한다.		
Return Type	Method Name	Parameter Type	Parameter Name												
AppDatabase	getInstance														
Description	어플리케이션 파일 내에 기존에 사용하던 데이터베이스가 없는 경우 데이터베이스를 생성한 후 그 데이터베이스를 리턴하며, 기존에 사용하던 데이터베이스가 있는 경우 그대로 불러와 리턴한다.														

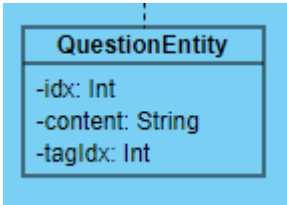
AppRepository			
Class Diagram			
Responsibility	<p>AppDatabase 객체에 접근하기 위해 사용되는 클래스이다.</p> <p>데이터베이스의 테이블마다 테이블의 동작을 위한 DAO 가 있으며 executeCommand 메소드 내에서 DAO 를 사용하여 DB 와 통신한다.</p>		
Attribute	Type	Name	Description
	MemoDao	memoDao	메모 테이블 DAO
	QuestionDao	questionDao	메모의 카테고리(질문) 테이블 DAO
	EmotionDao	emotionDao	메모의 감정 테이블 DAO
	TagDao	tagDao	단어 태그 테이블 DAO
Operation	Return Type	Method Name	Parameter Type
	List<BaseEntity>	execute Command	String, List<BaseEntity>
	Description	<p>1. DB 에 실행할 명령어와 데이터베이스에 갱신할 데이터를 인자로 받아 데이터가 갱신될 테이블에 해당하는 DAO 를 사용하여 데이터베이스 내의 데이터를 갱신한다. 데이터를 저장하는 경우에 리턴값은 Null 이다.</p> <p>2. DB 에 실행할 조회 명령어를 인자로 받아 해당 데이터 타입에 맞는 DAO 를 사용하여 데이터베이스로부터 데이터를 조회한다. 조회된 데이터를 리스트로 리턴한다.</p>	

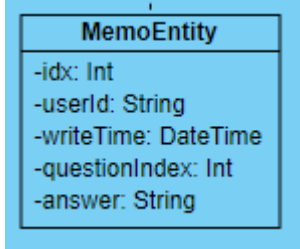
<<Interface>> BaseDao			
Class Diagram			
Responsibility	<p>데이터베이스 테이블에 사용할 수 있는 명령어들이 정의된 인터페이스이다. 테이블에 기본적으로 사용할 수 있는 삽입, 갱신, 삭제, 조회 명령어들을 선언해두었으며 모든 DAO 들은 반드시 이 인터페이스를 상속받아 작성되어야 한다.</p>		
Attribute	Type	Name	Description
Operation	Return Type	Method Name	Parameter Type
	BaseEntity	getOneByIndex	Int
	Parameter Name	idx	
	Description	테이블에서 index 를 이용해 index 에 해당하는 데이터를 검색하여 리턴한다.	
	Return Type	Method Name	Parameter Type
	List<BaseEntity>	getAll	
	Description	해당 테이블의 모든 데이터를 리턴한다.	
	Return Type	Method Name	Parameter Type
	void	insert	BaseEntity
	Parameter Name	entity	
	Description	테이블에 인자로 받은 데이터를 삽입한다.	

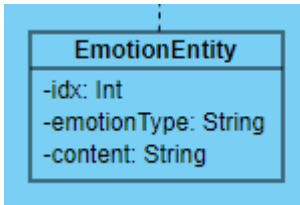
	Return Type	Method Name	Parameter Type	Parameter Name
	void	insert	List<BaseEntity>	entityList
	Description	테이블에 인자로 받은 데이터들을 삽입한다.		
	Return Type	Method Name	Parameter Type	Parameter Name
	void	update	BaseEntity	entity
	Description	테이블에서 인자로 받은 데이터의 index에 해당하는 데이터를 인자로 받은 데이터로 갱신한다.		
	Return Type	Method Name	Parameter Type	Parameter Name
	void	delete	BaseEntity	entity
	Description	테이블에서 인자로 받은 데이터와 일치하는 데이터를 삭제한다.		

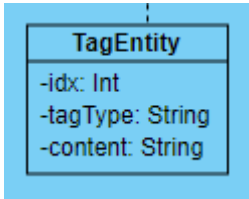
<<Interface>> BaseEntity				
Class Diagram				
Responsibility	<p>데이터베이스의 테이블에 해당하는 인터페이스이다.</p> <p>모든 테이블 클래스는 이 인터페이스를 상속받아 작성되어야 한다.</p> <p>이 인터페이스를 상속받아 정의된 클래스가 코드 내에서 객체로 정의되면 그 객체 하나는 테이블의 튜플 하나를 의미한다.</p>			
Attribute	Type	Name	Description	
Operation	Return Type	Method Name	Parameter Type	Parameter Name
	JsonObject (java imported)	toJsonObject		
	Description	데이터베이스의 모든 데이터들을 JSON 포맷의 문자열 한 줄로 변환하여 리턴한다.		

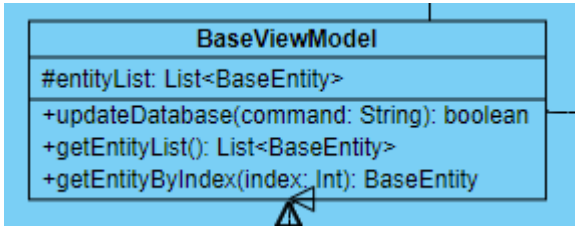
UserEntity			
Class Diagram			
Responsibility	<p>데이터베이스내에 어플리케이션 사용자 테이블을 정의하며, 사용자 테이블은 id 와 password 를 속성으로 갖는다.</p> <p>회원가입 시에 입력받은 아이디와 비밀번호로 이 객체를 생성하여 서버로 전송한다.</p>		
Attribute	Type	Name	Description
	String	userId	사용자의 ID
	String	password	사용자의 비밀번호
Operation			

QuestionEntity			
Class Diagram			
Responsibility	<p>메모를 작성할 때 그 메모에서 제시하는 질문들이 저장될 테이블을 정의하며, 이 질문 테이블은 질문 인덱스, 질문 내용, 질문의 카테고리를 속성으로 갖는다.</p>		
Attribute	Type	Name	Description
	Int	idx	테이블에서 사용될 index
	String	content	질문 내용
	Int	tagIdx	질문 카테고리의 index
Operation			

MemoEntity			
Class Diagram	 <pre> classDiagram class MemoEntity { -idx: Int -userId: String -writeTime: DateTime -questionIndex: Int -answer: String } </pre>		
Responsibility	<p>사용자가 작성한 메모들이 저장될 테이블을 정의하며, 메모 테이블은 메모 index, 작성자 ID, 작성시간, 질문 index, 대답 내용을 속성으로 갖는다. 사용자가 메모를 작성하면 작성한 내용이 이 객체로 정의된다.</p>		
Attribute	Type	Name	Description
	Int	idx	테이블에서 사용될 index
	String	userId	작성한 사람의 사용자 id
	DateTime	writeTime	메모의 작성시간
	Int	questionIndex	메모에서 제시한 질문의 index
	String	answer	질문에 대답한 내용
Operation			

EmotionEntity			
Class Diagram	 <pre> classDiagram class EmotionEntity { -idx: Int -emotionType: String -content: String } </pre>		
Responsibility	<p>사용자가 메모를 작성할 때 메모에 부여될 감정이 저장될 테이블을 정의한다. 감정 테이블은 index, 감정 종류, 감정 설명을 속성값으로 갖는다. 사용자가 감정 편집에서 추가한 감정은 이 객체로 정의된다.</p>		
Attribute	Type	Name	Description
	Int	idx	테이블에서 사용될 index
	String	emotionType	감정의 종류
	String	content	감정에 대한 설명
Operation			

TagEntity			
Class Diagram			
Responsibility	<p>사용자가 메모 분석을 돕기 위해 단어에 부여하는 태그들이 저장되는 테이블을 정의한다. 태그는 'content 는 tagType 이다' 라고 해석하도록 돕는 역할을 한다. 사용자가 태그 편집에서 추가한 태그는 이 객체로 정의된다.</p>		
Attribute	Type	Name	Description
	Int	idx	테이블에서 사용될 index
	String	tagType	태그의 타입
	String	content	단어에 부여할 태그(의미)
Operation			

<<abstract>> BaseViewModel			
Class Diagram			
Responsibility	<p>데이터베이스 내의 데이터와 화면의 출력되는 데이터의 동기화를 보장하기 위한 클래스이다. 또한 데이터베이스 내의 데이터가 화면에서 어떻게 그래픽으로 출력될 것인지 결정되도록 layout 파일과 바인딩된다.</p>		
Attribute	Type	Name	Description

List<BaseEntity>

entityList

데이터들의 변동사항을 저장

Return Type

Method Name

Parameter Type

Parameter Name

List<BaseEntity>

updateDatabase

String

command

Description

entityList 에 저장된 데이터들을 command 에 해당하는 명령어를 이용하여 테이블에 반영한다.

Return Type

Method Name

Parameter Type

Parameter Name

List<BaseEntity>

getEntityList

Description

AppRepository 내의 DAO 에서 select 메소드를 수행하여 데이터 객체 리스트를 가져와 화면으로 전달한다.

Return Type

Method Name

Parameter Type

Parameter Name

BaseEntity

getEntityByIndex

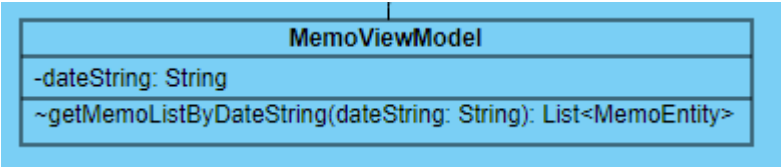
Int

index

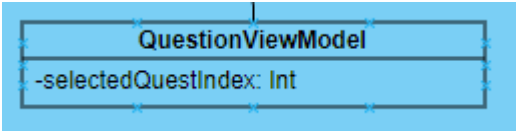
Description

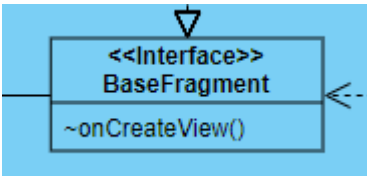
AppRepository 내의 DAO 에서 index 를 인자로 select 메소드를 수행하여 테이블의 index 번째 데이터 객체를 가져와 화면으로 전달한다.

Operation

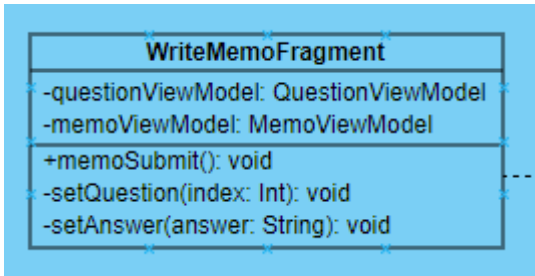
MemoViewModel			
Class Diagram	 <pre> classDiagram class MemoViewModel { -dateString: String ~getMemoListByDateString(dateString: String): List<MemoEntity> } </pre>		
Responsibility	<p>데이터베이스 내의 메모들과 화면에 보여질 메모들의 동기화를 위해 BaseViewModel 을 상속받아 정의된 클래스이다.</p> <p>컴파일 과정에서 메모 데이터가 화면에서 어떻게 보여질지 정의된 memo.xml 파일과 바인딩된다.</p>		
Attribute	Type	Name	Description
	String	DateString	메모 관리 화면에서 선택된 날짜를 임시 저장하는 변수

Operation	Return Type	Method Name	Parameter Type	Parameter Name
	List<MemoEntity>	getMemoList ByDateString	String	dateString
	Description	dateString 에 해당하는 날짜에 작성된 메모들을 AppRepository 에서 불러온다		

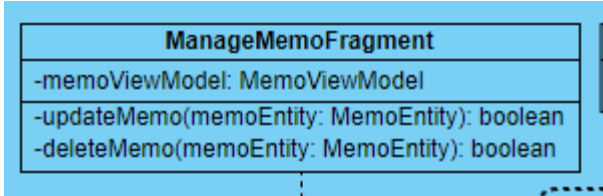
QuestionViewModel			
Class Diagram			
Responsibility	<p>데이터베이스 내에 저장된 질문과 화면에 보여질 질문의 동기화를 위해 BaseViewModel 을 상속받아 정의된 클래스이다.</p> <p>컴파일 과정에서 메모 데이터가 화면에서 어떻게 보여질지 정의된 question.xml 파일과 바인딩된다.</p>		
Attribute	Type	Name	Description
	Int	selectedQuestionIndex	메모 작성 화면 및 메모 관리 필터에서 선택된 질문의 index 를 임시 저장
Operation			

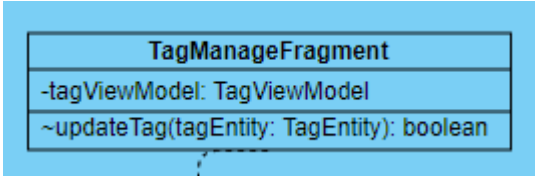
<<interface>> BaseFragment			
Class Diagram			
Responsibility	<p>MainActivity 위에서 출력되는 화면에 해당하는 객체들은 모두 이 인터페이스를 상속받아 작성되어야 한다. 화면이 어떻게 출력될지를 정의하기 위한 onCreateView 메소드가 있으며, 이 메소드 내에서 화면의 레이아웃을 결정하는 xml 파일을 정의해야 한다.</p>		
Attribute	Type	Name	Description

Operation	Return Type	Method Name	Parameter Type	Parameter Name
	void	onCreateView		
	Description	xml 파일로부터 화면의 레이아웃 정보를 읽어와 화면에 출력하는 기능을 한다. 이 함수는 안드로이드 라이브러리 함수이다.		

WriteMemoFragment				
Class Diagram	 <pre> classDiagram class WriteMemoFragment { -questionViewModel: QuestionViewModel -memoViewModel: MemoViewModel +memoSubmit(): void -setQuestion(index: Int): void -setAnswer(answer: String): void } </pre>			
Responsibility	메모 작성 화면을 담당하는 클래스이다. 질문 리스트가 저장된 questionViewModel 과 작성한 메모가 저장될 memoViewModel 이 멤버로 정의되어 있다. 작성 버튼을 누르면 수행될 memoSubmit 메소드가 정의되어 있다.			
Attribute	Type	Name	Description	
	QuestionViewModel	questionViewModel	선택할 수 있는 질문 리스트	
	MemoViewModel	memoViewModel	작성한 메모를 임시저장	
Operation	Return Type	Method Name	Parameter Type	Parameter Name
	void	setQuestion	Int	index
	Description	화면에서 질문을 선택하면 그 질문의 index 를 questionViewModel 에 임시로 저장한다.		
	Return Type	Method Name	Parameter Type	Parameter Name
	void	setAnswer	String	answer
	Description	사용자가 작성한 대답(메모)를 memoViewModel 에 임시로 저장한다.		
	Return Type	Method Name	Parameter Type	Parameter Name
	void	onCreateView		
	Description	xml 파일로부터 화면의 레이아웃 정보를 읽어와 화면에 출력하는 기능을 한다. 이 함수는 안드로이드 라이브러리 함수이다.		

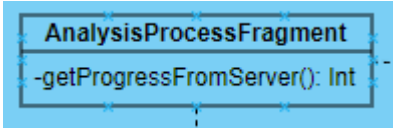
	Return Type	Method Name	Parameter Type	Parameter Name
	void	memoSubmit		
	Description	memoViewModel 에 저장된 임시 메모 데이터가 AppRepository 에 반영되도록 MemoViewModel 의 updateDatabase 메소드를 실행시킨다.		

ManageMemoFragment				
Class Diagram				
Responsibility	메모 관리 화면을 담당하는 클래스이다. 보여지는 메모들 중에서 메모를 선택하여 수정하거나 삭제할 수 있다.			
Attribute	Type	Name	Description	
	MemoViewModel	memoViewModel	메모들이 임시저장된 객체	
Operation	Return Type	Method Name	Parameter Type	Parameter Name
	boolean	updateMemo	MemoEntity	memoEntity
	Description	WriteMemoFragment 화면이 출력되도록 하고 memoEntity 의 questionIndex 와 answer 를 인자로 WriteMemoFragment 의 setQuestion 과 setAnswer 를 실행하여 화면에 메모가 자동입력되어 있도록 하며 WriteMemoFragment 의 memoSubmit 메소드를 실행하여 데이터베이스에 저장되도록 한다. 저장이 완료되면 true, 오류 또는 취소시 false 리턴		
	Return Type	Method Name	Parameter Type	Parameter Name
	boolean	deleteMemo	MemoEntity	memoEntity
	Description	선택된 메모가 삭제되도록 memoViewModel 내의 updateDatabase 메소드를 delete 명령어와 memoEntity 를 인자로 하여 실행시킨다.		

TagManageFragment			
Class Diagram	 <pre> classDiagram class TagManageFragment { -tagViewModel: TagViewModel ~updateTag(tagEntity: TagEntity): boolean } </pre>		
Responsibility	<p>태그 관리 화면을 담당하는 클래스이다.</p> <p>tagViewModel로부터 태그 목록을 불러와 화면에 출력하고 태그를 선택하여 편집할 수 있다.</p>		
Attribute	Type	Name	Description
Operation	Return Type	Method Name	Parameter Type
	boolean	updateTag	TagEntity
	Description	<p>편집된 태그가 AppRepository에 반영되도록 tagViewModel의 updateDatabase 메소드를 실행시킨다.</p>	

SettingsFragment															
Class Diagram	<div><div>SettingsFragment</div><div><div>-memoBackupButton: Button</div><div>-analysisBackupButton: Button</div><div>-analysisRequestButton: Button</div><div>-serviceCenterButton: Button</div></div><div><div>-backupMemo(): Boolean</div><div>-backupAnalysis(): Boolean</div><div>-requestAnalysis(): Boolean</div><div>-askToServiceCenter(): Boolean</div></div></div>														
Responsibility	설정 화면을 담당하는 클래스이다. 설정 화면에서는 메모 백업, 분석 백업, 분석 요청, 고객센터 이동을 수행할 수 있다.														
Attribute	Type	Name	Description												
	Button	memoBackupButton	메모 백업 버튼												
	Button	analysisBackupButton	분석 백업 버튼												
	Button	analysisRequestButton	분석 요청 버튼												
	Button	serviceCenterButton	고객 센터 버튼												
Operation	<table><tr><td>Return Type</td><td>Method Name</td><td>Parameter Type</td><td>Parameter Name</td></tr><tr><td>boolean</td><td>backupMemo</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">ConnectionController의 uploadToServer 메소드를 실행시켜 AppRepository에 있는 메모 리스트를 서버로 전송한다.</td></tr></table>			Return Type	Method Name	Parameter Type	Parameter Name	boolean	backupMemo			Description	ConnectionController의 uploadToServer 메소드를 실행시켜 AppRepository에 있는 메모 리스트를 서버로 전송한다.		
	Return Type	Method Name	Parameter Type	Parameter Name											
	boolean	backupMemo													
	Description	ConnectionController의 uploadToServer 메소드를 실행시켜 AppRepository에 있는 메모 리스트를 서버로 전송한다.													
	<table><tr><td>Return Type</td><td>Method Name</td><td>Parameter Type</td><td>Parameter Name</td></tr><tr><td>boolean</td><td>backupAnalysis</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">ConnectionController의 uploadToServer 메소드를 실행시켜 AppRepository에 있는 분석과 관련된 데이터들을 서버로 전송한다.</td></tr></table>			Return Type	Method Name	Parameter Type	Parameter Name	boolean	backupAnalysis			Description	ConnectionController의 uploadToServer 메소드를 실행시켜 AppRepository에 있는 분석과 관련된 데이터들을 서버로 전송한다.		
	Return Type	Method Name	Parameter Type	Parameter Name											
	boolean	backupAnalysis													
	Description	ConnectionController의 uploadToServer 메소드를 실행시켜 AppRepository에 있는 분석과 관련된 데이터들을 서버로 전송한다.													
	<table><tr><td>Return Type</td><td>Method Name</td><td>Parameter Type</td><td>Parameter Name</td></tr><tr><td>boolean</td><td>requestAnalysis</td><td></td><td></td></tr><tr><td>Description</td><td colspan="3">ConnectionController의 uploadToServer 메소드를 실행시켜 분석 요청을 서버로 전송한다. 이미 요청되어 분석이 수행중인 경우</td></tr></table>			Return Type	Method Name	Parameter Type	Parameter Name	boolean	requestAnalysis			Description	ConnectionController의 uploadToServer 메소드를 실행시켜 분석 요청을 서버로 전송한다. 이미 요청되어 분석이 수행중인 경우		
	Return Type	Method Name	Parameter Type	Parameter Name											
	boolean	requestAnalysis													
	Description	ConnectionController의 uploadToServer 메소드를 실행시켜 분석 요청을 서버로 전송한다. 이미 요청되어 분석이 수행중인 경우													

	AnalysisProcessFragment 화면으로 전환시킨다		
	Return Type	Method Name	Parameter Type
	boolean	askToServiceCenter	
Description		고객센터 화면으로 전환시킨다	

AnalysisProcessFragment			
Class Diagram	 <pre> classDiagram class AnalysisProcessFragment { -getProgressFromServer() Int } </pre>		
Responsibility	메모 분석이 서버에서 진행중인 경우 출력되는 화면을 담당하는 클래스 서버로부터 진행도를 불러와 화면에 출력한다.		
Attribute	Type	Name	Description
Operation	Return Type	Method Name	Parameter Type
	int	getProgressFromServer	
	Description	ConnectionController 내의 downloadFromServer 메소드를 실행하여 분석의 진행도를 불러와 반환한다.	

ServiceCenterFragment

Class Diagram	<div><div>ServiceCenterFragment</div><div><div>-askTitle: String</div><div>-askContent: String</div><div>-submitToServer(): boolean</div></div></div>			
Responsibility	고객센터 화면(문의)을 담당하는 클래스. 질문과 제목을 작성하고 서버에 접수시킬 수 있다.			
Attribute	Type	Name	Description	
	String	askTitle	고객센터 문의 제목	
	String	askContent	고객센터 문의 내용	
Operation	Return Type	Method Name	Parameter Type	Parameter Name
	boolean	submitToServer		
	Description	화면에서 입력한 고객센터 문의를 ConnectionController 의 uploadToServer 메소드를 시행 시켜 서버로 전송한다.		

유튜브 링크

https://youtu.be/TDsa_YSUv8M

깃허브 링크

https://github.com/yeonhunkim/Reminder/tree/master/software_engineering