**Implements**

1. Implements

1) DB

(1) AddressDB

**private static** AddressDBHandler *instance*; //유일한 객체

**private static** SQLiteDatabase *db*;  
**private** AddressDBHandler(Context context, String name, SQLiteDatabase.CursorFactory factory, **int** version) //addressDB 생성자  
**private static void** initialize(Context context) //객체 생성, db 열기  
**public static final** AddressDBHandler getInstance(Context context) //addressDB의 객체를 가져옴  
**public void** close() //db를 닫음  
**public void** onCreate(SQLiteDatabase db) /db를 생성하고, me 정보 삽입  
@Override  
**public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) //db upgrade  
**public void** INSERT(String name, String phone, String organization, String email, String memo) //db에 정보 삽입  
**public void** DELETE(String name) //이름으로 db 삭제  
**public** String getName() //이름순으로 db 가져오기  
**public** String getInfo(String name) //이름에 대한 db정보를 가져옴  
**public** String findName(String phone) //전화번호로 이름 정보 찾음  
**public** String findPhone(String name) //이름으로 전화번호 정보 찾음  
**public boolean** checkExist(String name) //이름에 해당하는 데이터가 존재하는지 검색  
**public int** getDataSize () //db의 사이즈 리턴

(2) callDB

**private static** CallListDBHandler *instance*; //유일한 객체  
**private static** SQLiteDatabase *db*;  
**private** CallListDBHandler(Context context, String name, SQLiteDatabase.CursorFactory factory, **int** version) //callDB생성자  
**private static void** initialize(Context context) //instance 초기화하고 db 가져옴  
**public static final** CallListDBHandler getInstance(Context context) //callDB의 객체를 가져옴  
**public void** close() //db를 닫음  
**public void** onCreate(SQLiteDatabase db) //db 없을시 db 생성  
**public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) //db upgrade  
**public void** INSERT(**int** receive, String phone) //db삽입  
**public void** DELETE(String phone) //전화번호로 db 삭제  
**public void** DELETE\_ALL() //전체 db 데이터 삭제  
**public void** DELETE\_BY\_ID(**int** id) //id로 db 삭제  
**public** String getPhone() //전체 phone number, id 획득  
**public int** getReceived (**int** id) //id로 수신여부 확인  
**public** String getLog(String phone) //전화번호로 db에서 관련 information 검색

(3) messaged

**private static** MessageDBHandler *instance*; //유일한 객체  
**private static** SQLiteDatabase *db*;  
**private** MessageDBHandler(Context context, String name, SQLiteDatabase.CursorFactory factory, **int** version) //MassageDB의 생성자  
**private static void** initialize(Context context) //instance 초기화하고 db 가져옴  
**public static final** MessageDBHandler getInstance(Context context) //MessageDB의 객체를 가져옴  
**public void** close() //db를 닫음  
**public void** onCreate(SQLiteDatabase db) //db가 없으면 db 생성  
**public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) //db upgrade  
**public void** INSERT(String caller, String phone, String message) //db에 데이터 삽입  
**public boolean** checkExist(String phone) //data에 해당하는 phone에 해당하는 정보가 존재하는지 확인  
**public void** DELETE\_BY\_ID(**int** id) //id에 해당하는 데이터 삭제  
**public void** DELETE(String phone) //phone에 해당하는 데이터 삭제  
**public** String getPhone() //전화 수신자, 메세지, 아이디 전체를 가져옴  
**public** String getLog(String phone) //해당 수신자가 발신하거나 수신한 기록 모두 가져옴

2) Address

(1) ListViewItem

**private** Drawable **picture**; //리스트뷰 항목의 사진  
**private** String **name**; //리스트뷰 항목의 이름

**public void** setPicture(Drawable picture) //리스트뷰의 항목에 사진 설정  
**public void** setName(String name) //리스트뷰의 항목에 이름 설정  
**public** Drawable getPicture() //리스트뷰에 항목에 존재하는 사진을 가져옴  
**public** String getName() //리스트뷰에 항목에 존재하는 이름을 가져옴

(2) ListViewAdaptor

**private** ArrayList<ListViewItem> **items; //리스트뷰에 이름 저장**

**private** ArrayList<ListViewItem> **filter** = **items**; //리스트뷰에 검색된 이름 저장  
Filter **listFilter**; //필터

**public** ListViewAdaptor() {} //생성자  
**public interface** ListBtnClickListner //아이템의 버튼이 클릭되는지 확인  
**int resourceId**; //리소스

**private** ListBtnClickListner **listBtnClickListener**; //리스트 안의 버튼 클릭 확인  
**public int** getCount() //필터된 항목의 사이즈 가져옴  
**public long** getItemId(**int** position) //item의 id를 가져옴  
**public** Object getItem(**int** position) //필터된 항목의 position을 가져옴  
**public void** addItem(Drawable picture, String name) //item을 items에 저장  
**public** View getView(**int** position, View convertView, ViewGroup parent) //리스트 항목의 뷰를 설정  
**public** Filter getFilter() //필터된 항목을 가져옴  
**private class** ListFilter **extends** Filter //Filter를 상속하는 subclass  
**protected** FilterResults performFiltering(CharSequence constraint) //필터된 항목의 결과를 가져옴.  
**protected void** publishResults(CharSequence constraint, FilterResults results) //필터링된 리스트로 리스트뷰의 결과값 수정

3) Message

(1) MessageListItem

**private** Drawable **picture**; // 리스트뷰의 사진  
**private** String **name**; //리스트뷰의 수신자 이름(전화번호/이름)  
**private** String **content**; //리스트뷰의 메세지 내용  
**private int id**; //메세지의 아이디  
**public void** setPicture(Drawable picture) //리스트뷰의 사진 설정

**public void** setName(String name) //리스트뷰의 수신자 이름 설정  
**public void** setContent(String content) //리스트뷰의 메세지 내용 설정   
**public void** setId(**int** id) //리스트뷰 아이템에 id설정  
**public** Drawable getPicture //리스트뷰 아이템에서 사진 가져옴  
**public** String getName() //리스트뷰 아이템에서 수신자 이름 가져옴  
**public** String getContent() //리스트뷰 아이템에서 메세지 내용 가져옴  
**public int** getId() //리스트뷰 아이템에서 id 가져옴

(2) MessageListViewAdaptor

**int resourceId**;  
MessageListViewAdaptor(Context context, **int** resource, ArrayList<MessageListItem> list) //MessageListViewAdaptor의 생성자  
**public** View getView(**int** position, View convertView, ViewGroup parent) //리스트뷰 아이템의 디자인 설정

(3) ShowMessageListItem

**private** Drawable **picture**; // 리스트뷰의 사진  
**private** String **name**; //리스트뷰의 수신자 이름(전화번호/이름)  
**private** String **content**; //리스트뷰의 메세지 내용  
**private int id**; //메세지의 아이디  
**public void** setPicture(Drawable picture) //리스트뷰의 사진 설정

**public void** setName(String name) //리스트뷰의 수신자 이름 설정  
**public void** setContent(String content) //리스트뷰의 메세지 내용 설정   
**public void** setId(**int** id) //리스트뷰 아이템에 id설정  
**public** Drawable getPicture //리스트뷰 아이템에서 사진 가져옴  
**public** String getName() //리스트뷰 아이템에서 수신자 이름 가져옴  
**public** String getContent() //리스트뷰 아이템에서 메세지 내용 가져옴  
**public int** getId() //리스트뷰 아이템에서 id 가져옴

(4) ShowMessageListAdaptor

ShowMessageListViewAdaptor(Context context, **int** resource, ArrayList<ShowMessageListItem> list) //ShowMessageListViewAdaptor의 생성자  
**public** View getView(**int** position, View convertView, ViewGroup parent) //리스트뷰 아이템의 뷰 설정

4) Call

(1) CallListViewItem

**private** Drawable **receive**; //call list view item의 사진 설정  
**private** String **phone**; //phone 번호 혹은 이름 저장  
**public void** setPicture(Drawable picture) //리스트뷰 항목에 사진 설정  
**public void** setPhone(String phone) //리스트뷰 항목에 전화번호 설정  
**public void** setId(**int** id) //리스트뷰 항목에 id 설정  
**public** Drawable getPicture() //리스트뷰 항목에서 사진 얻음  
**public** String getPhone() //리스트뷰 항목에서 전화번호를 가져옴  
**public int** getId() //리스트뷰 항목에서 id를 가져옴

(2) CallListViewAdaptor

**public interface** ListBtnClickListner //리스트뷰의 버튼이 클릭되는지 확인  
**int resourceId**; //resource의 id  
**private** ListViewAdaptor.ListBtnClickListner **listBtnClickListener**; //버튼이 클릭되는지 확인  
CallListViewAdaptor(Context context, **int** resource, ArrayList<CallListViewItem> list) //CallListAdaptor의 생성자

**public** View getView(**int** position, View convertView, ViewGroup parent) //리스트뷰 아이템의 디자인 설정  
**public void** onClick(View v) //버튼이 클릭되는지 확인