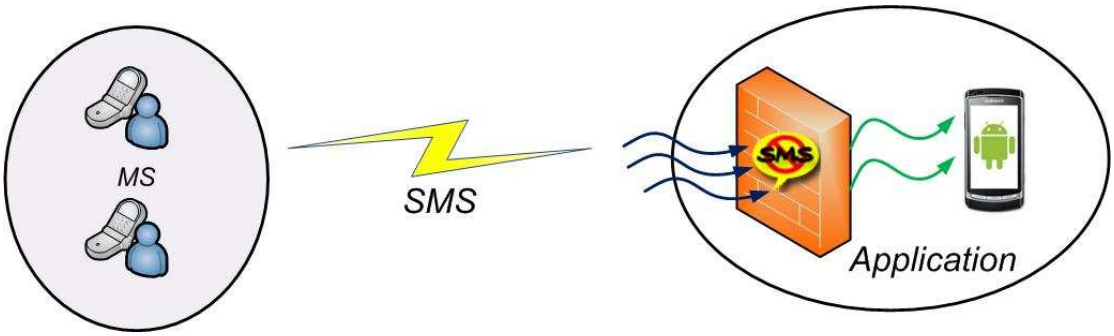
บทที่ 4
โปรแกรมกรองข้อความสั้นบนโทรศัพท์เคลื่อนที่แอนดรอยด์ โดยอาศัยข้อมูลในสมุดโทรศัพท์

4.1 ภาพรวมการทำงานของระบบ



ภาพที่ 4.1 โครงสร้างของโปรแกรมกรองข้อความสั้นบนโทรศัพท์เคลื่อนที่แอนดรอยด์

การทำงานของระบบนี้ได้แบ่งออกเป็น 2 ส่วน ดังภาพที่ 4.1 ซึ่งแต่ละส่วนมีรายละเอียดดังต่อไปนี้

4.1.1 โทรศัพท์เคลื่อนที่ทั่วไป (MS) ทำหน้าที่ส่งข้อความสั้น เพื่อเข้ามายังโทรศัพท์เคลื่อนที่แอนดรอยด์ที่ได้ติดตั้งแอพพลิเคชันการกรองข้อความสั้นโดยอาศัยข้อมูลในสมุดโทรศัพท์

4.1.2 แอพพลิเคชัน ทำหน้าที่กรองข้อความสั้นที่เข้ามา โดยอาศัยข้อมูลในสมุดโทรศัพท์เป็นตัวกรอง

4.2 ข้อกำหนดของโปรแกรมกรองข้อความสั้น

4.2.1 เมื่อมีข้อความสั้นที่ถูกส่งมาจากโทรศัพท์เคลื่อนที่ทั่วไปเข้ามา แอพพลิเคชันจะทำการตรวจสอบว่าเลขหมายที่ส่งข้อความสั้นเข้ามานั้น มีการบันทึกชื่ออยู่ในสมุดโทรศัพท์หรือไม่ หากมีการบันทึก ข้อความสั้นนั้นก็จะเข้ากล่องข้อความเข้าของตัวเครื่องปกติ แต่หากไม่มีชื่อบันทึกอยู่ในสมุดโทรศัพท์ ข้อความสั้นนั้นก็จะถูกเก็บไว้ในแอพพลิเคชัน

4.2.2 การทำงานภายในแอพพลิเคชัน จะเริ่มตั้งแต่เมื่อมีการเปิดแอพพลิเคชัน โดยทำการอ่านข้อมูลจากฐานข้อมูลทั้งหมด แล้วนำมาแสดงในรายการ (list) ซึ่งจะเป็นข้อความสั้นทั้งหมดที่แอพพลิเคชันได้กรองไว้ โดยสามารถเลือกเข้าไปยังแต่ละรายการ เพื่อดูข้อความสั้นที่กรองไว้ของแต่ละเลขหมาย โดยในแต่ละรายการจะสามารถเพิ่มเลขหมายและบันทึกรายชื่อลงในฐานข้อมูลของเครื่องโทรศัพท์ได้ ซึ่งจะเป็นการปรับปรุงสมุดโทรศัพท์ในตัวเครื่อง ทั้งนี้ยังสามารถลบข้อความสั้นที่อยู่ในแอพพลิเคชันได้ด้วย หากไม่ต้องการเก็บข้อความสั้นนั้นไว้

4.3 ขั้นตอนการทำงานของโปรแกรมกรองข้อความสั้น

การทำงานจะเริ่มต้นขึ้น เมื่อโปรแกรมได้ถูกติดตั้ง (Installed) ลงบนโทรศัพท์เคลื่อนที่แอนดรอยด์ และจะทำงานอัตโนมัติเมื่อมีข้อความใหม่เข้ามาที่เครื่องโทรศัพท์ โดยอาศัยกระบวนการและขั้นตอนการทำงานดังนี้

4.3.1 ทำการลงทะเบียนเพื่อเปิดบริการกรองข้อความสั้น สำหรับการพัฒนาแอพพลิเคชันนั้น ๆ จะต้องมีการตั้งค่าการอนุญาต (Permission) เป็นการให้สิทธิต่าง ๆ กับแอพพลิเคชันตามตารางที่ 4.1 ซึ่งจะทำบนไฟล์ AndroidManifest.xml

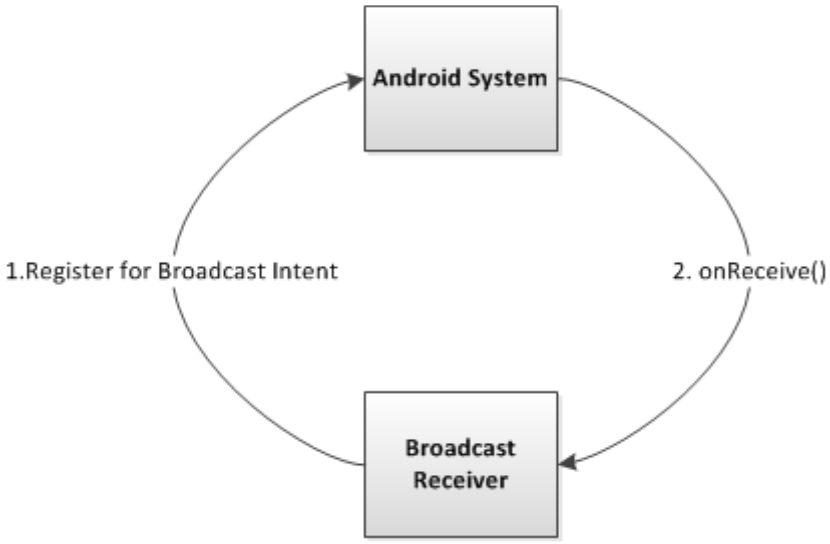
ตารางที่ 4.1 การอนุญาตให้เข้าถึงข้อมูลของข้อความสั้น (SMS)

การอนุญาตที่เกี่ยวกับโทรศัพท์	วัตถุประสงค์
android.permission.RECEIVE_SMS	อนุญาตให้แอพพลิเคชันสามารถตรวจสอบข้อความสั้นที่เข้ามาได้
android.permission.READ_SMS	อนุญาตให้แอพพลิเคชันสามารถอ่านข้อความสั้นที่เข้ามาได้
android.permission.SEND_SMS	อนุญาตให้แอพพลิเคชันสามารถส่งข้อความสั้นได้
android.permission.WRITE_SMS	อนุญาตให้สามารถเขียนข้อความสั้นลงบนแอพพลิเคชันที่สร้างขึ้นได้

ที่มา: Android Developer [online] : เข้าถึง 15 ต.ค. 2554. จาก

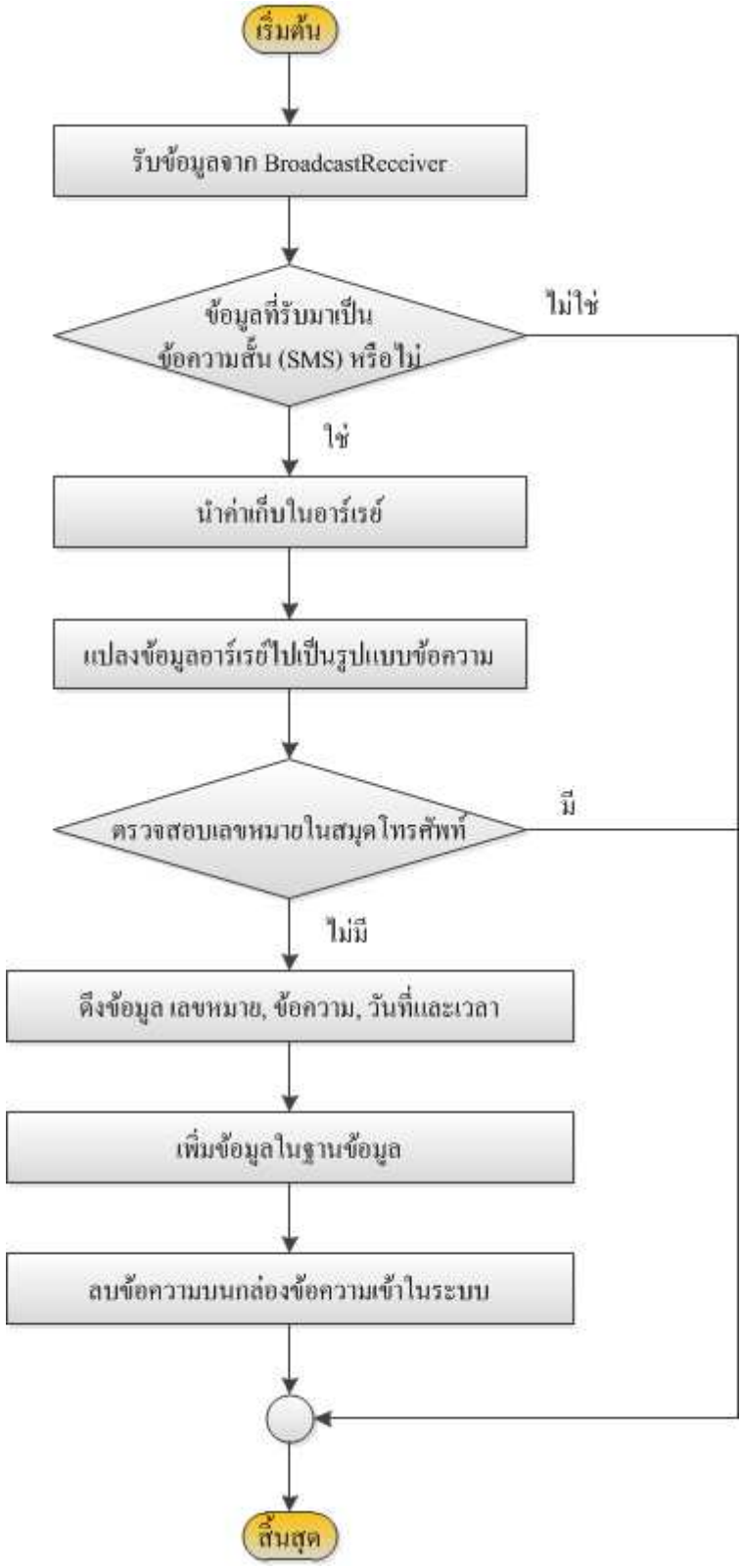
http://developer.android.com/reference/android/Manifest.permission.html

4.3.2 การใช้งานข้อความสั้น (SMS) ในเฟรมเวิร์คของระบบปฏิบัติการแอนดรอยด์จะมีไลบรารีที่เกี่ยวข้องกับการใช้บริการข้อความสั้น คือคลาส SmsManager จะอยู่ในแพ็คเกจ android.telephony โดยจะรองรับทั้งระบบ GSM และระบบ CDMA รวมถึงการประกาศ Broadcast Receiver เพื่อตรวจจับข้อความสั้นที่ส่งเข้ามา โดยมีวงจรชีวิตของ Broadcast Receiver ตามภาพที่ 4.2



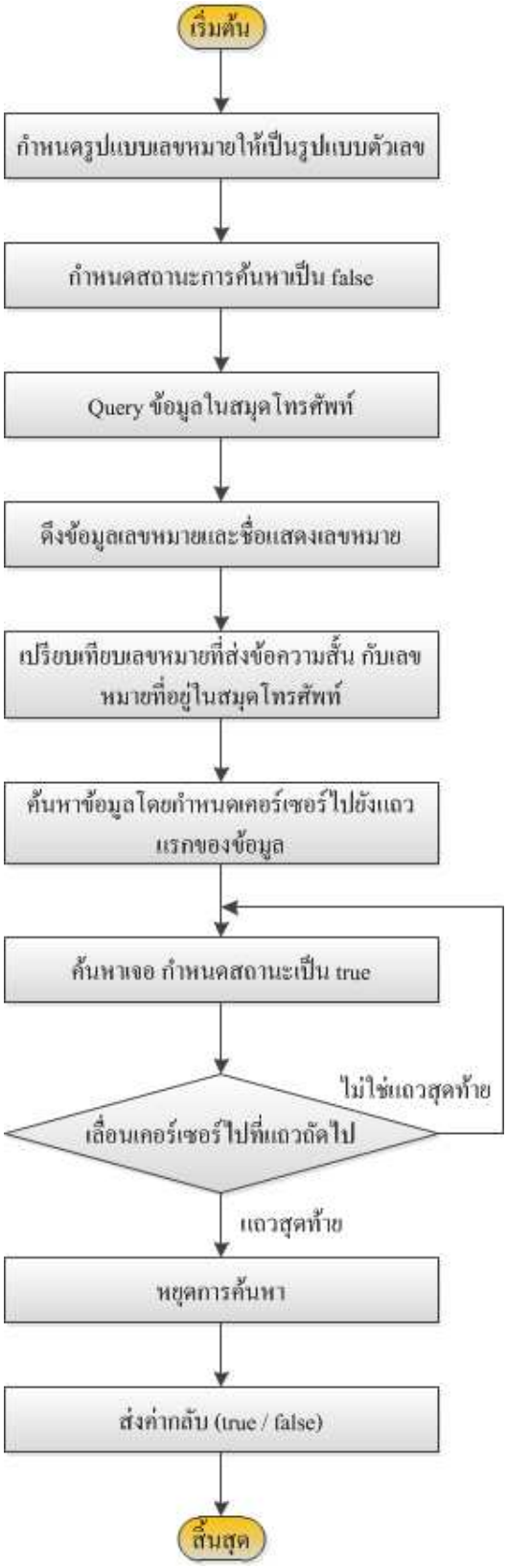
ภาพที่ 4.2 วงจรชีวิตของ Broadcast Receiver

4.3.3 ฟังก์ชัน onReceive() จะถูกเรียกใช้งานเมื่อมีการรับข้อมูลเข้ามา โดยทำการตรวจสอบว่าข้อมูลที่รับมานั้นเป็นข้อมูลประเภทข้อความสั้นหรือไม่ ในการตรวจจับข้อความสั้นที่ถูกส่งเข้ามานั้นจะใช้ค่าของ PDU (Protocol Description Unit) ซึ่งประกอบด้วย ข้อความสั้น ค่าต่าง ๆ ที่ใช้ในการส่งโดยเก็บไว้ในอาร์เรย์ออบเจ็ค (object) และเมธอด createFromPdu() ซึ่งจะใช้แปลงข้อมูลอาร์เรย์ออบเจ็ค ไปเป็น SmsMessage และใช้คำสั่ง getOriginatingAddress() เพื่ออ่านค่าเลขหมายโทรศัพท์ของผู้ส่ง และใช้คำสั่ง getMessageBody() เพื่ออ่านค่าข้อความ หลังจากนั้นจะนำค่าเลขหมายโทรศัพท์ ไปตรวจสอบกับเมธอด isHaveContact() บนคลาส ContactPhone เพื่อเปรียบเทียบเลขหมายระหว่างเลขหมายของข้อความสั้นใหม่ และเลขหมายบนสมุดโทรศัพท์ของระบบ ซึ่งหากตรวจสอบแล้วไม่มี ก็จะนำค่าทั้งเลขหมาย ข้อความ วันที่และเวลา ไปเพิ่มในฐานข้อมูลของแอพพลิเคชัน และเข้าไปลบข้อความสั้นนั้นบนกล่องข้อความเข้าของระบบ โดยใช้เมธอด deleteSMS() บนคลาส SMSReceive ตามภาพที่ 4.3

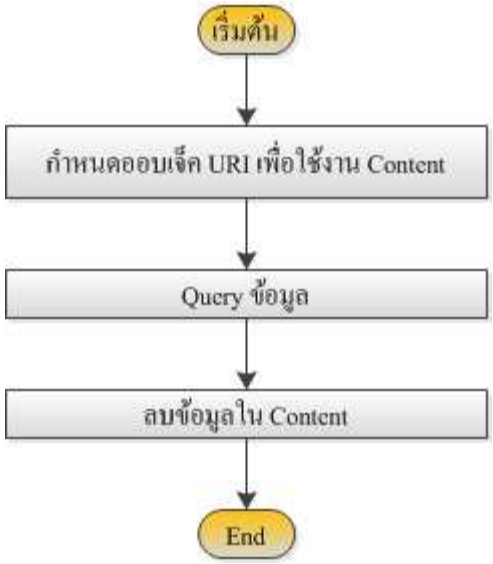


ภาพที่ 4.3 ผังงานการรับและตรวจสอบข้อความสั้น

สำหรับการตรวจสอบเลขหมายในสมุดโทรศัพท์ ตามภาพที่ 4.4 ได้แยกย่อยการทำงานออกมา โดยสร้างคลาส isHaveContact() เป็นตัวตรวจสอบเลขหมายของข้อความสั้นนั้น ๆ กับเลขหมายหรือรายชื่อบนสมุดโทรศัพท์ของระบบ โดยใช้รูปแบบ PhoneNumberUtils.formatNumber เพื่อเรียกเครื่องมือสำหรับจัดการสมุดโทรศัพท์และกำหนดรูปแบบของตัวเลข โดยกำหนดสถานะเริ่มต้นเป็น false หลังจากนั้นจะใช้เคอร์เซอร์ ในการสืบค้น (query) ข้อมูล โดยกำหนด Phone.NUMBER เป็นหมายเลขโทรศัพท์ และ Phone.DISPLAY_NAME เป็นชื่อที่แสดงแทนเลขหมายนั้น ๆ โดยในการสืบค้นนั้นจะย้ายเคอร์เซอร์ไปยังแถวแรกของข้อมูลแล้วเปรียบเทียบข้อมูล หากตรงกันจะส่งค่ากลับเป็น true และย้ายเคอร์เซอร์ไปยังตำแหน่งถัดไป ซึ่งหากแถวนั้นเป็นแถวสุดท้าย ก็จะหยุดการค้นหา และจะส่งค่ากลับของสถานะการค้นหา และเมื่อได้ทำการเพิ่มข้อมูลลงในฐานข้อมูลแล้ว จะต้องให้แอพพลิเคชันเข้าไปลบข้อความสั้นนั้นบนกล่องข้อความเข้าของระบบ โดยใช้เมธอด deleteSMS() บนคลาส SMSReceive โดยจะเข้าไปทำงานอัตโนมัติ ผ่านการเรียกใช้ URI สำหรับการเข้าถึงข้อมูลใน Content Provider (“content://sms/inbox”) และจะใช้คำสั่ง query ใน URI โดยมีค่า ID, thread_id, date, body และจะเข้าไปลบในส่วนของ conversations โดยอ้างอิงจากเลขหมายนั้น ๆ ตามภาพที่ 4.5



ภาพที่ 4.4 ผังงานการตรวจสอบบนสมุดโทรศัพท์ของระบบ



ภาพที่ 4.5 ผังงานการลบข้อความสั้นบนกล่องข้อความของระบบ

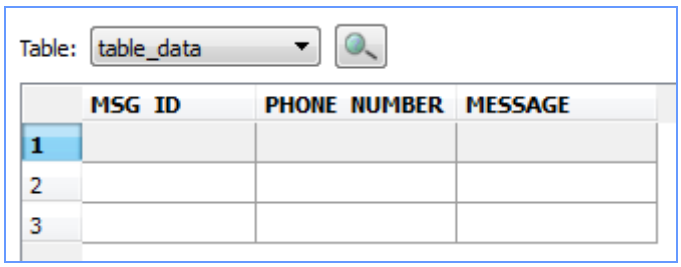
2.3.4 การใช้งานฐานข้อมูล SQLite (เจมส์ สตีล และ เนลสัน โต, 2554) โดยในระบบปฏิบัติการแอนดรอยด์นั้น มีการติดตั้งระบบฐานข้อมูล SQLite มาด้วยแล้ว ซึ่ง SQLite มีลักษณะเป็นฐานข้อมูลเชิงสัมพันธ์ โดยสามารถใช้คำสั่ง SQL ในการสืบค้นข้อมูลได้ แอพพลิเคชันที่ใช้งาน SQLite นั้นจะมีอินสแตนซ์ของฐานข้อมูลเป็นของตัวเอง และสามารถเข้าถึงได้จากแอพพลิเคชันเท่านั้น ในส่วนของแอพพลิเคชันนี้ มีการใช้งาน SQLite ทางด้าน สร้างฐานข้อมูล เพิ่มข้อมูล เรียกดูข้อมูล ลบข้อมูล โดยมีรายละเอียดดังนี้

4.3.4.1 การสร้างฐานข้อมูล มีการประกาศตัวแปร เพื่อใช้เก็บตารางข้อมูล โดยระบุชื่อตารางฐานข้อมูลและประกาศตัวแปรชื่อของฐานข้อมูล โดยระบุชื่อไฟล์ของฐานข้อมูล โดยทำการสร้างฐานข้อมูลภายใต้ชื่อฐานข้อมูลที่ได้ประกาศไว้ โดยใช้รูปแบบคำสั่งตามภาพที่ 4.6

```
public abstract SQLiteDatabase openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory)
```

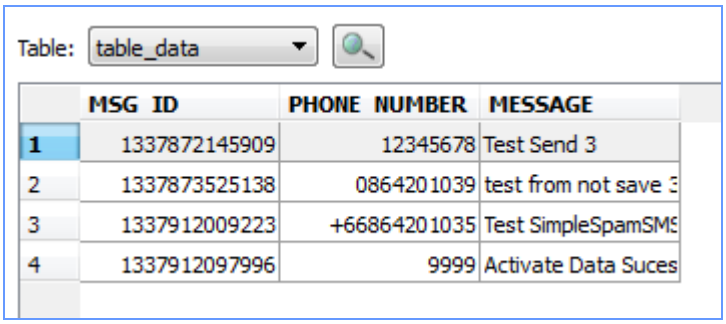
ภาพที่ 4.6 คำสั่ง openOrCreateDatabase

ในการสร้างฐานข้อมูลจะใช้คำสั่ง execSQL เพื่อสร้างตาราง โดยกำหนดชื่อตาราง และฟิลด์ของข้อมูลตามภาพที่ 4.7



ภาพที่ 4.7 การสร้างตารางฐานข้อมูลและการบันทึกข้อมูล

4.3.4.2 การเพิ่มข้อความสั้นลงในฐานข้อมูล จะใช้คำสั่ง ContentValues.put เพื่อนำค่าเข้าไปยังฟิลด์ของข้อมูล MSG_ID, PHONE_NUMBER, MESSAGE ที่สร้างไว้ และใช้คำสั่ง insert() โดยการอ้างอิงตารางข้อมูล และนำค่าข้อมูลที่อยู่ใน ContentValue ไปเพิ่มข้อมูลลงในฐานข้อมูล ตามภาพที่ 4.8



ภาพที่ 4.8 การเพิ่มข้อมูลลงในฐานข้อมูล

4.3.4.3 การเรียกดูข้อมูล (getDataSMS()) เพื่อแสดงข้อมูลเลขหมาย จำนวนข้อความสั้นต่อเลขหมาย และข้อความสั้น โดยใช้คำสั่ง openOrCreateDatabase() เพื่อเข้าถึงฐานข้อมูล และใช้คำสั่ง SQLiteDatabase.rawQuery() ด้วย MSG_ID, PHONE_NUMBER, MESSAGE บนตารางข้อมูล ด้วยกลุ่มของ PHONE_NUMBER และเรียกใช้คลาส getSizeDatailSMS เพื่อตรวจสอบและนับจำนวนข้อความสั้นต่อเลขหมายนั้น ๆ

4.3.4.4 การเรียกดูรายละเอียดของข้อมูล (getDataDetailSMS()) เพื่อแสดงข้อมูลข้อความสั้น โดยใช้คำสั่ง openOrCreateDatabase() เพื่อเข้าถึงฐานข้อมูล และใช้คำสั่ง SQLiteDatabase.rawQuery() ด้วย MSG_ID, PHONE_NUMBER, MESSAGE บนตารางข้อมูลด้วย PHONE_NUMBER

4.3.4.5 การลบข้อความ จะเป็นการลบข้อความ 1 ข้อความ บนเลขหมายนั้น ๆ โดยใช้คำสั่ง SQLiteDatabase.delete() ที่ตารางของฐานข้อมูล ด้วยรหัส MSG_ID ในการอ้างอิงเพื่อลบ

4.3.4.6 การลบข้อความทั้งหมดต่อเลขหมายนั้น ๆ โดยใช้คำสั่ง SQLiteDatabase.delete() ที่ตารางของฐานข้อมูล ด้วย PHONE_NUMBER ในการอ้างอิงเพื่อลบข้อความทั้งหมดบนเลขหมายนั้น ๆ

4.3.5 การกำหนดการทำงานของแอคทิวิตี้ต่าง ๆ บนคลาส MainActivity โดยมีการกำหนดแอตทริบิวต์ที่เกี่ยวข้องกับคลาส และมีการทำงานของเมธอดต่าง ๆ ดังนี้

4.3.5.1 เมธอด onCreate() จะมีการเรียกใช้เมื่อแอคทิวิตี้เริ่มทำงานและจะมีการเรียกใช้เลเอาท์โดยการนำไฟล์ main.xml มาวาดเป็นหน้าจอของแอพพลิเคชัน และมีการสร้างออบเจ็กต์ชื่อ smsReceive บนคลาส SMSReceive(), ออบเจ็กต์ชื่อ db บนคลาส DataBase(), ออบเจ็กต์ชื่อ conP บนคลาส ContactPhone() และมีการเรียกใช้เมธอด isPhoneContact() บนคลาส ContactPhone() และทำการหารหัส (id) จากไฟล์ list_sms.xml ที่ผ่านกระบวนการ (process) บนเมธอด onCreate() รวมถึงการเรียกค่าจากเมธอด getDataSMS() บนคลาส DataBase มาเก็บบน vData ตามภาพที่ 4.9

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    smsReceive = new SMSReceive();
    db = new DataBase(this);
    ContactPhone conP = new ContactPhone(this);
    conP.isPhoneContact();

    list_sms = (List)findViewById(R.id.list_sms);
    list_sms.setOnItemClickListener(this);
    list_sms.setOnItemLongClickListener(this);
    vData = db.getDataSMS();
    if(vData.size() != 0) list_sms.setListAdapter(vData);
}
```

ภาพที่ 4.9 การทำงานของเมธอด onCreate()

4.3.5.2 เมธอด onItemClick() จะเป็นการเรียกกลับเมธอดเมื่อมีการเลือกรายการใน AdapterView โดยจะเชื่อมแอคทิวิตี้ต่าง ๆ และส่งผ่านข้อมูลกันระหว่างแอคทิวิตี้โดยใช้ Intent โดยทำหลังจากมีการเลือก แล้วจึงไปทำที่คลาส DetailActivity และสั่งเริ่มการทำงาน ตามภาพที่ 4.10

```
public void onItemClick(AdapterView<?> AV, View v, final int position, long arg3) {
    DataElement dataE = vData.elementAt(position);
    Intent intent = new Intent(this,DetailActivity.class);
    intent.putExtra("PHONE_NUMBER",dataE.getPhoneNumber());
    startActivity(intent);
}
```

ภาพที่ 4.10 การทำงานของเมธอด onItemClick()

4.3.5.3 เมธอด onItemLongClick() จะเป็นการเรียกกลับเมธอดเมื่อมีการเลือกโดยการกดค้างไว้บนรายการใน AdapterView และจะกำหนด view ที่ใช้งาน Context menu ด้วย

```
public boolean onItemLongClick(AdapterView<?> AV, View v, final int position,long arg3) {
    registerForContextMenu(list_sms);

    return false;
}
```

ภาพที่ 4.11 การทำงานของเมธอด onItemLongClick()

4.3.5.4 เมธอด onCreateContextMenu() จะเป็นการสร้างเมนูหลัก เมื่อมีการสัมผัสหน้าจอ โดยมีการสร้างเมนูขึ้นมา 2 เมนูคือ เมนู “Add Contact” เพื่อเพิ่มรายชื่อลงบนสมุดโทรศัพท์บนโทรศัพท์เคลื่อนที่ โดยมีรหัส (id) = 0 และเมนู “Delete Contact” เพื่อลบเลขหมายนี้ออกจากฐานข้อมูลบนแอพพลิเคชัน โดยมีรหัส (id) = 1 ตามภาพที่ 4.12

```
public void onCreateContextMenu(ContextMenu menu, View v,ContextMenuInfo menuInfo)
{
    super.onCreateContextMenu(menu, v, menuInfo);
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)menuInfo;
    DataElement dataE = vData.elementAt(info.position);
    menu.setHeaderTitle(dataE.getPhoneNumber());
    menu.add(0, 0, 0, "Add Contact");
    menu.add(0, 1, 0, "Delete Contact");
}
```

ภาพที่ 4.12 การทำงานของเมธอด onCreateContextMenu()

4.3.5.5 เมธอด onContextItemSelected() จะทำงานเมื่อมีการเลือกรายการเมนู โดยหากเลือกรหัส (id) = 0 ซึ่งก็คือเมนู “Add Contact” จะใช้ Intent เพื่อเชื่อมไปยังแอคทิวิตี้ ACTION เพื่อ

เพิ่มข้อมูล และสั่งให้ ActivityForResult ทำงานต่อ หรือหากเลือกรหัส (id) = 1 ซึ่งก็คือเมนู “Delete Contact” จะมีข้อความปรากฏขึ้นบนแถบไตเติ้ลว่า “DELETE SMS MESSAGE” และเนื้อหาข้อความ “Do you want to delete message?” โดยมีปุ่ม “OK” และ “CANCEL” ทั้งนี้ได้ตั้งค่าให้ปุ่ม “OK” เป็น positive ซึ่งจะคอยฟังเหตุการณ์เมื่อมีการคลิก โดยเมื่อมีการคลิกปุ่ม “OK” จะเป็นการเรียกใช้เมธอด deleteDataNumberSMS() บนคลาส DataBase เพื่อลบข้อความสั้นบนเลขหมายนั้นทั้งหมด และมีการปรับปรุงข้อมูลโดยการเรียกใช้เมธอด getDataSMS() บนคลาส DataBase แล้วมาเก็บไว้ที่ตัวแปร vData เพื่อปรับปรุงรายการใหม่ แต่หากเลือกปุ่ม “CANCEL” ซึ่งตั้งค่าเป็น Negative ไว้ ซึ่งเมื่อมีการคลิกก็จะนำรายการทั้งหมดแสดงไว้เหมือนเดิม

```
public boolean onContextItemSelected(MenuItem item)
{
    final AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)item.getMenuInfo();
    Log.w("","info.position = "+info.position);
    if(item.getItemId() == 0)
    {
        DataElement dataE = vData.elementAt(info.position);
        Intent intent = new Intent(Intent.ACTION_INSERT);
        intent.setType(ContactsContract.Contacts.CONTENT_TYPE);
        intent.putExtra(ContactsContract.Intents.EXTRA_FORCE_CREATE, true);
        intent.putExtra(ContactsContract.Intents.Insert.PHONE,dataE.getPhoneNumber());
        Phone_numberAddSMS = dataE.getPhoneNumber();
        int PICK_CONTACT = 100;
        startActivityForResult(intent, PICK_CONTACT);
    }
    else if(item.getItemId() == 1) {
        AlertDialog.Builder ad = new AlertDialog.Builder(this);
        ad.setTitle("DELETE SMS MESSAGE");
        ad.setMessage("Do you want to delete message?");
        ad.setPositiveButton( "OK", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                DataElement dataE = vData.elementAt(info.position);
                db.deleteDataNumberSMS(dataE.getPhoneNumber());
                vData = db.getDataSMS();
                list_sms.setListAdapter(vData);
            }
        });
        ad.setNegativeButton("CANCEL",new DialogInterface.OnClickListener(){
            public void onClick(DialogInterface dialog, int id) {
            }
        });
        ad.show();
    }
    return true;
}
```

ภาพที่ 4.13 การทำงานของเมธอด onContextItemSelected()

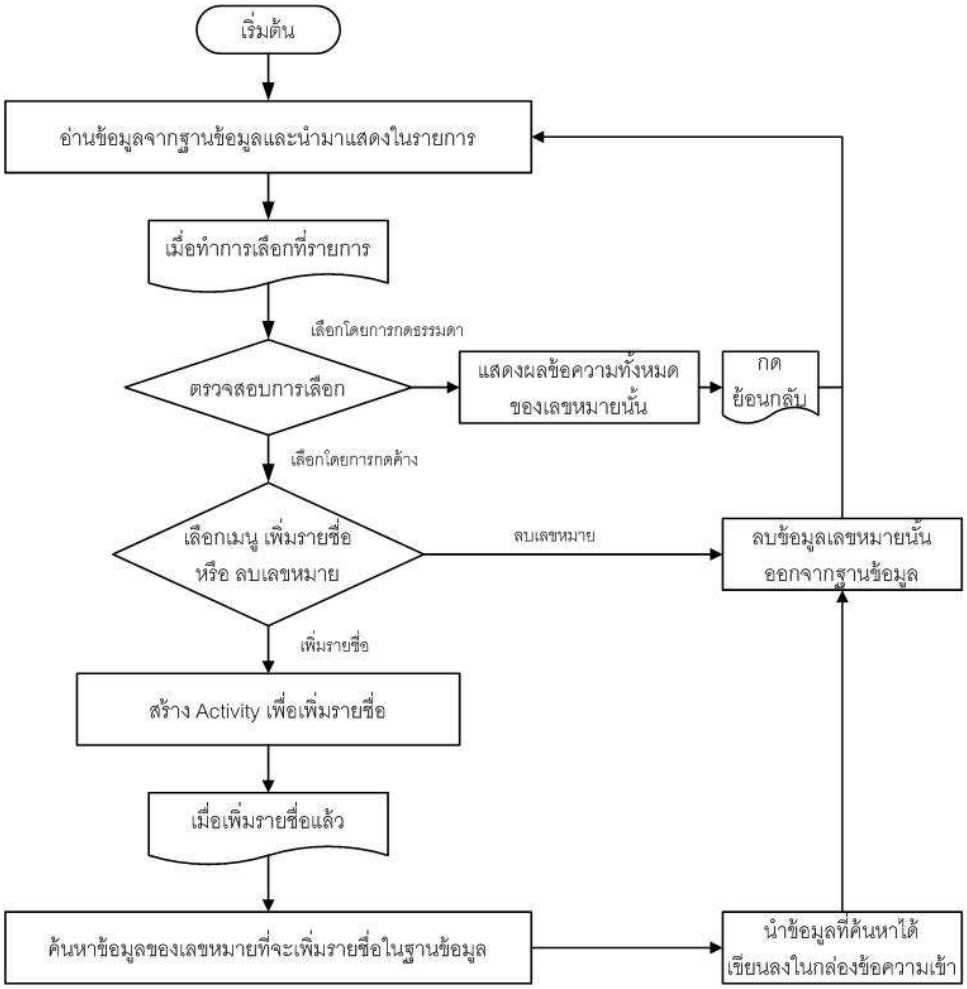
4.3.5.6 เมธอด onActivityResult() จะทำงานต่อจากเมธอด onContextItemSelected() ซึ่งเกิดจากการเลือกเมนู “Add Contact” โดยทำการกำหนดออบเจ็กต์ cp บนคลาส ContactPhone() และเรียกไปยังเมธอด getDataDetailSMS() เพื่อเรียกข้อมูล บนคลาส DataBase และนำค่าไปเก็บไว้ที่ vdata และตรวจสอบจำนวนข้อความสั้นทั้งหมดบนเลขหมายนั้น แล้วนำไปเขียนบนสมุด

โทรศัพท์โดยผ่านเมธอด writeSMS() บนคลาส ContactPhone() ซึ่งมีข้อมูล เลขหมายโทรศัพท์ ข้อความสั้น รหัสข้อความสั้น จากคลาส DataElement() เมื่อเขียนข้อความสั้นลงบนกล่องข้อความเข้าบนเครื่องแล้ว หลังจากนั้นจะเรียกใช้เมธอด deleteDataNumberSMS() บนคลาส DataBase เพื่อลบออกจากแอพพลิเคชัน และปรับปรุงรายการบนฐานข้อมูลแอพพลิเคชัน

```
protected void onActivityResult(int requestCode, int resultCode,Intent data) {
    Log.w("","---ONACTIVITY RESULT---"+requestCode+":"+resultCode);
    Log.w("","data1 = "+data);
     if (requestCode == 100) {
         if (resultCode == RESULT_OK)
         {
             ContactPhone cp = new ContactPhone(this);
             Vector<DataElement> vdata = db.getDataDetailSMS(Phone_numberAddSMS);
             for (int i = 0; i < vdata.size(); i++) {
                 DataElement dataE = vdata.elementAt(i);
                 cp.writeSMS(dataE.getPhoneNumber(), dataE.getMessage(),dataE.getMessageID());
             }
             db.deleteDataNumberSMS(Phone_numberAddSMS);
             vData = db.getDataSMS();
             list_sms.setListAdapter(vData);
         }
     }
}
```

ภาพที่ 4.14 การทำงานของเมธอด onActivityResult()

และขั้นตอนการทำงานทั้งหมดเมื่อเปิดแอพพลิเคชัน สามารถจัดการทำงานได้ตามภาพที่ 4.15



ภาพที่ 4.15 ขั้นตอนการทำงานของโปรแกรมภายในเมื่อเปิดโปรแกรม

4.3.6 ความสัมพันธ์ระหว่างคลาส แสดงตามภาพที่ 4.16

4.3.6.1 คลาส MainActivity ทำหน้าที่ควบคุมและเชื่อมต่อกับส่วนติดต่อผู้ใช้ เช่น จะทำงานเมื่อมีการคลิก คลิกค้าง เลือกไอเทม (item) หรือเมนูย่อย เป็นต้น

4.3.6.2 คลาส DetailActivity ทำหน้าที่รับ Intent มาจาก คลาส MainActivity และเชื่อมต่อกับส่วนติดต่อผู้ใช้ โดยการเลือกไอเทมค้างไว้ และทำงานตามกล่องโต้ตอบที่ได้ตั้งค่าไว้

4.3.6.3 คลาส SMSReceive ทำหน้าที่ดักจับ service ตามที่ได้กำหนด event ไว้ และทำหน้าที่ตรวจสอบกับสมุดโทรศัพท์ และลบข้อความสั้นอัตโนมัติ

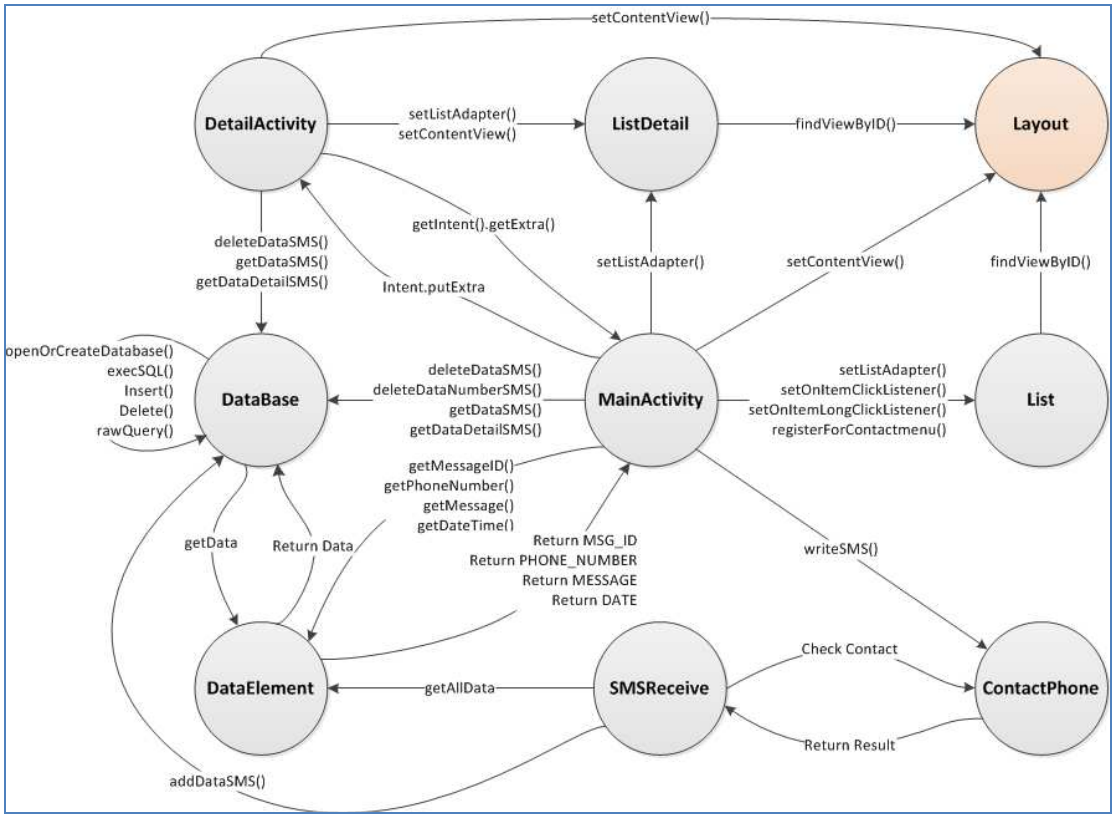
4.3.6.4 คลาส ContactPhone ทำหน้าที่ค้นหาข้อมูลในสมุดโทรศัพท์และนำมาตรวจสอบกับเลขหมายที่เข้ามา ว่ามีอยู่หรือไม่ และทำการเขียนข้อความสั้นลงบนระบบแอนดรอยด์ เมื่อมีการเรียกใช้ activity เพิ่มรายชื่อติดต่อ

4.3.6.5 คลาส Database ทำหน้าที่จัดการกับฐานข้อมูลทั้งหมดบน SQLite เช่น การสร้างฐานข้อมูล การเพิ่มข้อมูล การลบข้อมูล และการค้นหาข้อมูล โดยจะเชื่อมต่อกับคลาส DataElement

4.3.6.6 คลาส DataElement ทำหน้าที่เก็บข้อมูลจากฐานข้อมูล และเชื่อมต่อกับคลาส Database

4.3.6.7 คลาส List ทำหน้าที่แสดงผลข้อมูล โดยการสืบทอดจากคลาส ListView โดยแสดงผลข้อมูล เลขหมาย ข้อความ และ วันที่เวลา

4.3.6.8 คลาส ListDetail ทำหน้าที่แสดงผลข้อมูลเช่นเดียวกับคลาส List โดยการสืบทอดจากคลาส ListView โดยแสดงผลข้อมูล ข้อความ และ วันที่เวลา



ภาพที่ 4.16 ความสัมพันธ์ระหว่างคลาส