

MSP Assignment 1

위치 기반 알람

2018년 4월 3일

① 목차

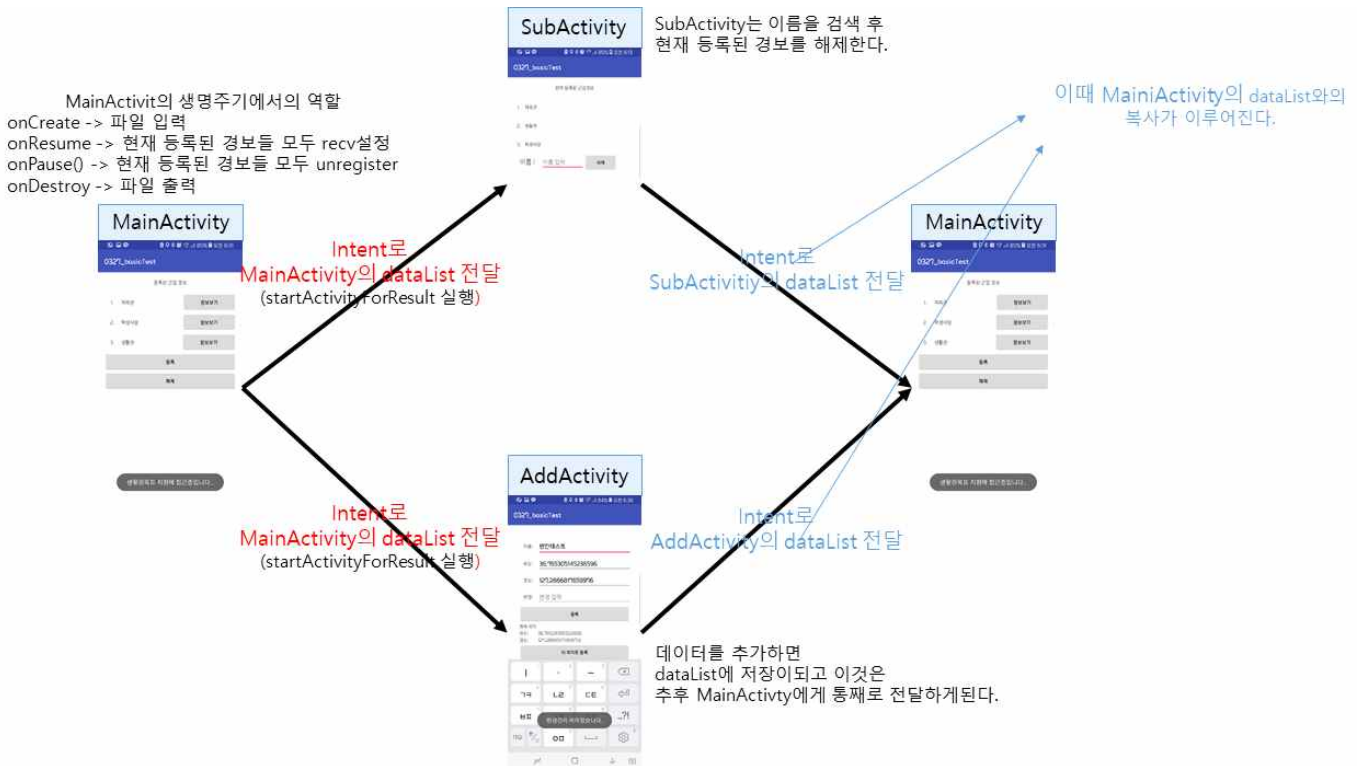
1. 프로그램의 전체 구조 및 로직	3
1.1. 클래스 정의	3
1.2. 프로그램 전체 구조	3
2. 작성한 주요 코드에 대해서 설명	4
2.1. Data Class	4
2.2. MainActivity의 주요 기능과 코드	5
2.3. AddActivity의 주요 기능과 코드	11
2.4. SubActivity의 주요 기능과 코드	13
2.5. 예외처리 및 추가기능	14
2.6. Layout UI	18
3. 과제 수행 소감	21

2. 프로그램의 전체 구조 및 로직

2.1. 클래스 정의

Java 클래스	내용
MainActivity	MainActivity를 정의한 클래스
SubActivity	SubActivity를 정의한 클래스, 이 액티비티 안에서 이름을 검색 후 데이터 요소 삭제 과정이 이루어진다.
AddActivity	AddActivity를 정의한 클래스, 이 액티비티 안에서 이름, 위도, 경도, 반경을 입력하면 데이터 추가 과정이 이루어진다.
AlertReceiver	BroadCast Receiver 정의를 위한 클래스, 위치 기반으로 위치에 가까워지거나 멀어질 때 발생하는 일들을 정의했다.
Data	이름, 위도, 경도, 반경의 데이터구조를 갖는 클래스

2.2. 프로그램 전체 구조



3. 작성한 주요 코드에 대해서 설명

3.1. Data Class

Data

```
// Parcelable을 implements한 이유는 인텐트에서 ArrayList를 전달하기 위함
public class Data implements Parcelable {
    String name;    // 지역 이름
    double lat;     // 위도
    double lon;     // 경도
    float rad;      // 반경

    public Data() {
    }

    public Data(String name, float lat, float lon, int rad) {
        this.name = name;
        this.lat = lat;
        this.lon = lon;
        this.rad = rad;
    }

    protected Data(Parcel in) {
        name = in.readString();
        lat = in.readDouble();
        lon = in.readDouble();
        rad = in.readFloat();
    }

    public static final Creator<Data> CREATOR = new Creator<Data>() {
        @Override
        public Data createFromParcel(Parcel in) {            return new Data(in);        }

        @Override
        public Data[] newArray(int size) {                  return new Data[size];        }
    };

    public String getName() {            return name;        }
    public void setName(String name) {        this.name = name;    }
    public double getLat() {            return lat;        }
    public void setLat(float lat) {        this.lat = lat;    }
    public double getLon() {            return lon;        }
    public void setLon(float lon) {        this.lon = lon;    }
    public float getRad() {            return rad;        }
    public void setRad(int rad) {        this.rad = rad;    }

    // ArrayList를 다른 Activity로 넘겨주기 위한 설정 ( implements Parcelable 때문에)
    @Override
    public int describeContents() {        return 0;    }

    // ArrayList를 다른 Activity로 넘겨주기 위한 설정 ( implements Parcelable 때문에)
    @Override
    public void writeToParcel(Parcel parcel, int i) {
        parcel.writeString(this.name);
        parcel.writeDouble(this.lat);
        parcel.writeDouble(this.lon);
        parcel.writeFloat(this.rad);
    }
}
```

Parcelable 클래스를 implements한 이유는 액티비티 간 List로 전달할 때 필요하기 때문이다.

3.2. MainActivity의 주요 기능과 코드

① 주요 변수들

MainActivity	용도 및 설명
ArrayList<Data> dataList	Data 자료형 리스트, Data 자료형은 이름, 위도, 경도, 반경 정보가 들어있다.
ArrayList<Data> subTempList	Sub activity에서 가져올 데이터 리스트
ArrayList<Data> addTempList	Add activity에서 가져올 데이터 리스트
PendingIntent[] proximityIntentList	3개의 Proximity Alert이 등록될 것이므로 Intnet를 3개 선언했다.
AlertReceiver receive	브로드캐스트를 받을 리시버
LocationManager locManager	ocation Manager로 쓰일 변수
private static final String FILENAME	일 입출력을 위한 파일명 담는 변수

② 액티비티 간의 값 전달하기

앞서 말했듯이 나는 다른 Activity간의 전달을 위해서 startActivityForResult를 썼다.

MainActivity
<pre> public void onClickAdd(View view) { if(dataList.size() == 3) { Toast.makeText(MainActivity.this, "요소가 꽉찼습니다!", Toast.LENGTH_SHORT).show(); return; } //다음 페이지로 화면을 전환 // 화면을 전환할때 사용하는 클래스 그것이 Intent 클래스이다. // 첫번째 파라미터는 이동 전 액티비티, 두번째 파라미터는 이동할 액티비티 // 이 인텐트는 a에서 b로 이동한다는 화면 전환 정보를 가지고 있다. Intent intent = new Intent(MainActivity.this, AddActivity.class); intent.putParcelableArrayListExtra("list1", dataList); // 화면 전환하기!! startActivityForResult(intent, 0); } </pre>

위의 소스는 addAcitivity로 넘어가는 소스이다. 보시다시피 startActivityForResult(결과값을 받아오기 위해)를 썼고, dataList를 통째로 넘겨주었다.

③ 액티비티 간의 값 전달받기

AddActivity나 SubActivity에서 모든 값 리스트들을 전달할텐데 그 값 리스트들을 받아 원래 MainActivity의 데이터 리스트를 초기화한다.

```
MainActivity
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);

    // 아 이 코드가 반드시 들어가야 된다
    // 뒤로가기 또는 비정상적인 종료라면 실행되어야 되는 부분
    if (resultCode != RESULT_OK) {
        Toast.makeText(MainActivity.this, "데이터가 입력이 안된 채로 종료되었습니다.",
        Toast.LENGTH_SHORT).show();
        return;
    }

    //건너온 액티비티의 requestCode에 맞는 분기실행
    switch (requestCode) {
        case 0: // AddActivity에서 0으로 해놓았다.
            // AddActivity에서 실행했던 addList를 통째로 받아와 초기화한다.
            addTempList = intent.getParcelableArrayListExtra("addlist");
            dataList.clear();
            for(int i = 0; i < addTempList.size(); i++ ) {
                dataList.add(addTempList.get(i));
            }
            break;

        case 1: // SubActivity는 1로 해놓았다
            // SubActivity에서 실행했던 addList를 통째로 받아와 초기화한다.
            subTempList = intent.getParcelableArrayListExtra("sublist");
            dataList.clear();
            for(int i = 0; i < subTempList.size(); i++ ) {
                dataList.add(subTempList.get(i));
            }
            break;
        default:
            break;
    }
}
```

여기서 requestCoed가 0이라면 Addactivity에서 Activity가 돌아온 것이고, 1이라면 SubActivity이다. 자세히 보면 그 액티비티에서 전달한 ArrayList<Data>형 자료형을 그대로 받아서 원래 MainActivity의 dataList를 초기화 하는 것을 알 수 있다. 이렇게 구현한 이유는 Add나 SubAcitivity에서 예외처리를 포함한 추가/삭제를 할 때, 모든 데이터 값 리스트들, dataList의 정보가 필요하기 때문에 List로 모든 데이터를 전달하게 했다.

④ 파일 입력하기

MainActivity

```
// 파일 저장하는 함수
public void outputFile() throws IOException {
    File file = new File(getFilesDir(), FILENAME); // 데이터를 저장할 객체 생성
    FileWriter fw = null; // 파일쓰기 객체
    BufferedWriter bufwrit = null; // 파일 쓰기용 버퍼
    try {
        // open file.
        fw = new FileWriter(file);
        bufwrit = new BufferedWriter(fw);

        for (int i = 0; i < dataList.size(); i++)
        {
            bufwrit.write(dataList.get(i).name + "/" + dataList.get(i).lat + "/" +
                dataList.get(i).lon + "/" + dataList.get(i).rad);
            bufwrit.newLine();
        }
        // 버퍼를 비운다.
        bufwrit.flush();
        Toast.makeText(this, "파일 저장 완료~!", Toast.LENGTH_LONG).show();
    }
    catch (Exception e) {
        e.printStackTrace() ;
    }

    try {
        // 파일을 닫는다.
        if (bufwrit != null) bufwrit.close();
        if (fw != null) fw.close();
    }
    catch (Exception e) {
        e.printStackTrace() ;
    }
}
```

파일을 입력하는 소스이다. FileWrite와 BufferedWriter를 썼으며, BufferedWriter로 쓸 때 슬래시(/)를 이용하여 데이터를 구분했으며, 읽어드릴 때도 슬래시(/)를 이용하여 구분했다..

⑤ 파일 출력하기

MainActivity

```
// 파일 가져오는 함수
public void inputFile() throws IOException {

    File file = new File(getFilesDir(), FILENAME); // 파일 객체를 생성한다
    FileReader fr = null; // 파일을 읽는 객체를 생성한다.
    BufferedReader bufread = null; // 파일 읽기용 버퍼 만든다

    if (file.exists()) // 만약에 파일이 존재한다면?
    {
        try
        {
            // 파일 읽는다.
            fr = new FileReader(file);
            bufread = new BufferedReader(fr);

            String str;
            while ((str = bufread.readLine()) != null)
            {
                String[] data = str.split("/"); // 슬래시로 구분해준다.
                Data temp = new Data();
                temp.name = data[0];
                temp.lat = Double.parseDouble(data[1]);
                temp.lon = Double.parseDouble(data[2]);
                temp.rad = Float.parseFloat(data[3]);

                dataList.add(temp);
            }
            bufread.close();
            fr.close();
            Toast.makeText(this, "파일 읽기 완료~!", Toast.LENGTH_LONG).show();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

위에서 말한대로 BufferedWriter를 썼으므로 BufferedRead로 읽어드리고, 슬래시(/)로 데이터를 구분하여 읽어드린다.

⑥ 모든 데이터 위치 경보 등록

MainActivity
<pre>@Override protected void onResume() { super.onResume(); .. 생략 // 모든 데이터 간에 Alert을 진행한다. for(int i = 0; i < dataList.size(); i++) startAlert(i); } public void startAlert(int index) { // 근접 경보를 받을 브로드캐스트 리시버 객체 생성 및 등록 // 액션이 kr.ac.koreatech.msp.locationAlert인 브로드캐스트 메시지를 받도록 설정 receiver = new AlertReceiver(); IntentFilter filter = new IntentFilter("dis"+index); // action을 filter마다 달리 지정해주어야지 여러 가지 근 접경보를 받을 수 있다. registerReceiver(receiver, filter); // ProximityAlert 등록을 위한 PendingIntent 객체 얻기 Intent intent = new Intent("dis"+index); intent.putExtra("name",dataList.get(index).name); proximityIntentList[index] = PendingIntent.getBroadcast(this, 0, intent, 0); try { // 근접 경보 등록 메소드 // void addProximityAlert(double latitude, double longitude, float radius, long expiration, PendingInt ent intent) // 아래 위도, 경도 값의 위치는 2공학관 420호 창가 부근 locationManager.addProximityAlert(dataList.get(index).lat, dataList.get(index).lon, dataList.get(index).ra d, 20000, proximityIntentList[index]); } catch (SecurityException e) { e.printStackTrace(); } isAlertRegistered = true; }</pre>

위와 같이 반복문을 써서 모든 데이터 요소들이 receiver에 추가하도록 하였다. 이것은 Resume에 돌아가게 했는데 그 이유는 앱이 정지되고 다시 실행했을 때에도 Receiver가 다시 작동해야 하기 때문이다. 즉 나는 onPause에서 모든 ProximityAlert을 정지시키고, onResume이 됐을 때 Alert을 다시 키는 방식으로 했다.

⑦ 모든 데이터 위치 경보 해제

MainActivity
<pre>..//생략 @Override protected void onPause(){ super.onPause(); //-----alert // 앱이 다른 액티비티로 가거나 종료되는 등 화면 벗어날때 Alert들 을 종료해준다. for(int i = 0; i < dataList.size(); i++) exitAlert(i); // 등록되었을 때만 해제하기 위함 try { unregisterReceiver(receiver); } catch (IllegalArgumentException e) { return; } } // 해당하는 proximityIntentList의 근접경보를 해제하는 코드 public void exitAlert(int index) { // 자원 사용 해제 try { if(isLocRequested) { locationManager.removeUpdates(this); // Updates를 더이상 안한다. isLocRequested = false; } if(isAlertRegistered) { locationManager.removeProximityAlert(proximityIntentList[index]); // 해당 proximityIntent를 해제한다. } } catch (SecurityException e) { e.printStackTrace(); } } ..//생략</pre>

exitAlert이라는 함수를 만들고 데이터 갯수만큼 루프를 돌아, proximityIntentList에 있는 모든 경보를 해제한다.

3.3. AddActivity의 주요 기능과 코드

① MainActivity를 통해 값 전달받기

AddActivity
<pre>public void homeMove(View view) { ..//생략 Intent intent = new Intent(); intent.putParcelableArrayListExtra("addlist", _dataList); // ArrayList 보내기 setResult(RESULT_OK, intent); // 안전하게 보냈다 finish();// 현재 액티비티 종료 }</pre>

MainActivity에서 들어온 key값 addlist들을 통해 AddActivity의 _dataList를 초기화했다.

② MainActivity에게 값 전달하기

SubActivity
<pre>public void homeMove(View view) { .. 생략 Intent intent = new Intent(); intent.putParcelableArrayListExtra("addlist", _dataList); // ArrayList 보내기 setResult(RESULT_OK, intent); // 안전하게 보냈다 finish();// 현재 액티비티 종료 }</pre>

MainActivity에게 현재까지 저장된 dataList들을 전달하는 코드이다. putParcelableArray의 키워드를 썼으며, 인텐트를 통해 값을 전달했다.

③ 현재 위도와 경도 표시

AddActivity
<pre>@Override public void onLocationChanged(Location location) { double lat = location.getLatitude(); double lon = location.getLongitude(); tvLat.setText(Double.toString(lat)); // 현재 위도 표시 tvLon.setText(Double.toString(lon)); // 현재 경도 표시 }</pre>

onLocationChanged는 장소 정보가 바뀔 때 실행하는 함수이다. 이 함수 안에 setText 구문을 넣어서 위치가 바뀔 때 바뀌도록 한다. 나머지 AddActivity의 예외처리 부분은 예외처리 및 추가기능에서 설명하도록 하겠다.

④ 데이터 추가 기능

AddActivity

```
public void homeMove(View view)
{
    // 예외처리: 이름이 중복되는지 확인한다.
    for(int i = 0; i < _dataList.size(); i++ ) {
        if( _dataList.get(i).name.equals( etName[0].getText().toString() )) {
            Toast.makeText(this, "이미 존재하는 이름입니다.", Toast.LENGTH_SHORT).show();
            return;
        }
    }

    // 예외처리: 모든 입력 칸이 비어있는지 확인하다
    for(int i = 0; i < 4; i++ ) {
        if( etName[i].getText().toString().equals("")) {
            String temp;
            if(i == 0) temp = "이름";
            else if(i == 1) temp = "위도";
            else if(i == 2) temp = "경도";
            else temp = "반경";

            Toast.makeText(this, temp + "칸이 비어있습니다.", Toast.LENGTH_SHORT).show();
            return;
        }
    }

    // 데이터를 add할 temp 데이터 값 초기화
    try {
        tempData.name = etName[0].getText().toString();
        tempData.lat = Float.parseFloat(etName[1].getText().toString());
        tempData.lon = Float.parseFloat(etName[2].getText().toString());
        tempData.rad = Integer.parseInt(etName[3].getText().toString());
    }catch (NumberFormatException e) { // 예외처리: 혹시나 toString하는 과정에서 Format이 잘못됐는지 확인한다.
        Toast.makeText(AddActivity.this, "데이터 형식이 잘못되었습니다", Toast.LENGTH_SHORT).show();
        return;
    }

    _dataList.add(tempData); // 데이터 추가

    Intent intent = new Intent();
    intent.putParcelableArrayListExtra("addlist", _dataList); // ArrayList 보내기
    setResult(RESULT_OK, intent); // 안전하게 보냈다
    finish();// 현재 액티비티 종료
}
```

ArrayList의 add함수를 이용해서 데이터를 추가했다. 추가하기 전에 형변환을 하여 tempData에 넣고, tempData를 add하는 방식으로 하였다. 중간중간의 예외처리에 대한 설명은 뒷부분에서 하겠다.

3.4. SubActivity의 주요 기능과 코드

액티비티 간의 값 전달하기, 받기는 AddActivity와 유사하므로 생략하겠다.

① 데이터 삭제 기능

SubActivity

```
public void onClickSub(View view) {
    boolean DidRemove = false; // 같은 이름이 있다면 루프를 돌때 true로 바뀌질거임
    inputName = EditName.getText().toString();

    // 예외처리: 만약 데이터가 입력되지 않았다면 빈 것이다._
    if(EditName.getText().toString().equals("")) {
        Toast.makeText(SubActivity.this, "이름을 입력해주세요", Toast.LENGTH_SHORT).show();
        return;
    }

    // loop를 돌면서 해당 이름과 같은 이름을 가진 데이터 리스트 삭제해버리기
    for(int i = 0; i < _dataList.size(); i++) {
        if(_dataList.get(i).name.equals(inputName)) {
            _dataList.remove(i);
            DidRemove = true;
        }
    }

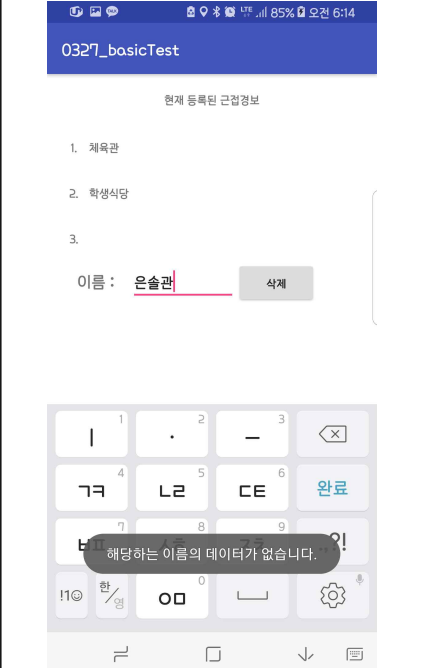
    // 예외처리: 만약 삭제되지 않았다면 해당하는 이름의 데이터가 없는 것이다.
    if(DidRemove == false) {
        Toast.makeText(SubActivity.this, "해당하는 이름의 데이터가 없습니다.", Toast.LENGTH_SHORT).show();
        return;
    }
    else {
        Toast.makeText(SubActivity.this, "삭제완료", Toast.LENGTH_SHORT).show();

        Intent intent = new Intent(SubActivity.this, MainActivity.class );
        intent.putParcelableArrayListExtra("sublist", _dataList); // MainActivity에게 _dataList를 전달한다.
        setResult(RESULT_OK, intent ); // 안전하게 보냈다
        finish();// 현재 액티비티 종료
    }
}
```

이름을 검색하면 해당 이름에 맞는 dataList를 삭제하도록 하였다. ArrayList의 키워드인 remove함수를 이용하였다.

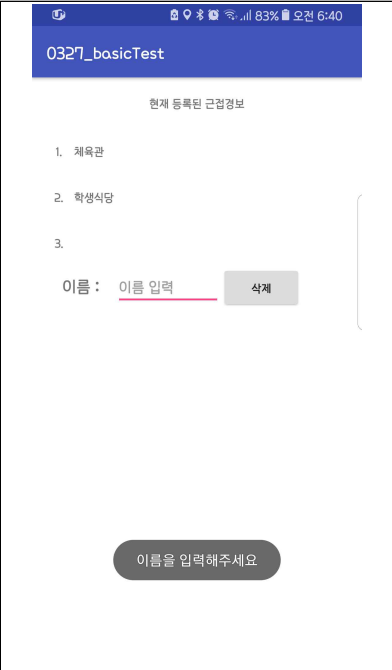
3.5. 예외처리 및 추가기능

① SubActivity 존재하지 않는 이름 예외처리

	<pre>boolean DidRemove = false; // 같은 이름이 있다면 루프를 돌때 true로 바꿔질거임 inputName = EditName.getText().toString(); .. 생략 // loop를 돌면서 해당 이름과 같은 이름을 가진 데이터 리스트 삭제 해버리기 for(int i = 0; i < _dataList.size(); i++) { if(_dataList.get(i).name.equals(inputName)) { _dataList.remove(i); DidRemove = true; } } // 예외처리: 만약 삭제되지 않았다면 해당하는 이름의 데이터가 없는 것이다. if(DidRemove == false) { Toast.makeText(SubActivity.this, "해당하는 이름의 데이터가 없습니다.", Toast.LENGTH_SHORT).show(); return; } .. 생략</pre>
---	--

현재 dataList에 없는 이름을 삭제하려고 할 때 예외처리를 하였다. boolean DidRemove를 선언하여 loop를 돌아 삭제가 완료되면 true로 바꿔게하여 삭제를 했는지, 안했는지 여부로 해당하는 이름의 데이터가 있는 지 없는 지 판단한다.

② SubActivity에서 이름 칸을 빈채로 삭제했을 때 예외처리

	<pre>// 예외처리: 만약 데이터가 입력되지 않았다면 빈 것이다._ if(EditName.getText().toString().equals("")) { Toast.makeText(SubActivity.this, "이름을 입력해주세요", Toast. LENGTH_SHORT).show(); return; }</pre>
---	--

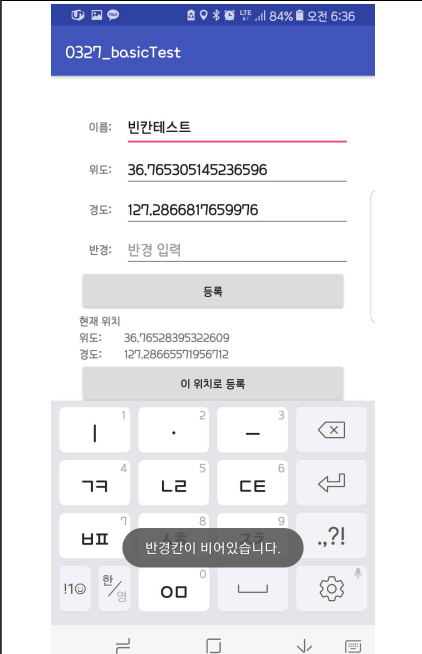
이름 칸의 정보는 EditName에 들어가있는데, 그곳의 getText()가 아무것도 없는 내용 ""이라면 이름이 아직 없다는 뜻이므로, Toast로 입력을 경고를 띄워주었다.

③ MainActivity 요소가 꽉 찼을 때의 예외처리

	<pre> public void onClickAdd(View view) { if(dataList.size() == 3) { Toast.makeText(MainActivity.this, "요소가 꽉 찼습니다!", Toast.LENGTH_SHORT).show(); return; } ..생략 </pre>
---	---

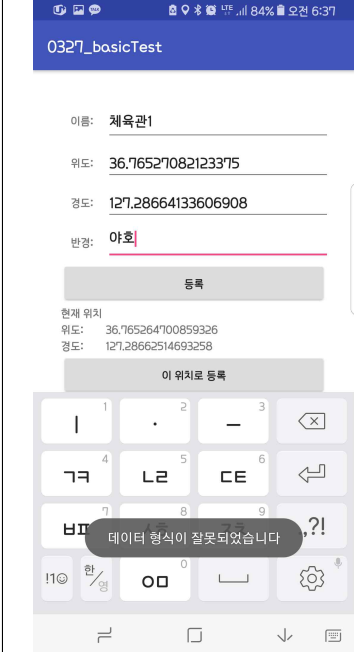
dataList의 요소의 갯수가 3개 이상이면 더이상 등록을 받지 않기로 하였다.

④ AddActivity의 칸이 다 입력하지 않았을 때 예외처리

	<pre> public void homeMove(View view) { ..생략 // 예외처리: 모든 입력 칸이 비어있는지 확인하다 for(int i = 0; i < 4; i++) { if(etName[i].getText().toString().equals("")) { String temp; if(i == 0) temp = "이름"; else if(i == 1) temp = "위도"; else if(i == 2) temp = "경도"; else temp = "반경"; Toast.makeText(this, temp + "칸이 비어있습니다.", Toast.LENGTH_SHORT).show(); return; } } ..생략 </pre>
---	--

칸이 비어있는 것은 getText()가 아무것도 없는 내용 ""이라면 칸이 비어있단 뜻이다. 그래서 모든 요소가 빈칸인 지 검사한다.

⑤ AddActivity의 칸이 데이터 형식이 잘못됐을 때 예외처리




```

// 데이터를 add할 temp 데이터 값 초기화
try {
    tempData.name = etName[0].getText().toString();
    tempData.lat = Float.parseFloat(etName[1].getText().toString());
    tempData.lon = Float.parseFloat(etName[2].getText().toString());
    tempData.rad = Integer.parseInt(etName[3].getText().toString());
} catch (NumberFormatException e) { // 예외처리: 혹시나 toString하는 과정에서
    Format이 잘못됐는지 확인한다.
    Toast.makeText(AddActivity.this, "데이터 형식이 잘못되었습니다",
        Toast.LENGTH_SHORT).show();
    return;
}
        
```

데이터를 입력하여 tempData에 넣을 것인데 변환 과정에서 예외가 발생하면 그에 따른 경고를 띄워준다. 변환 과정에서의 오류란, float 형 자료가 들어갈 부분에 문자열이 들어간 경우 등이 있다.

⑥ AddActivity의 현재 위치를 바로 등록하게 하는 기능



```


public void nowRegister(View view) {
    // 이 버튼을 클릭하면 현재의 위도, 경도로 값이 입력된다.
    etName[1].setText(tvLat.getText().toString());
    etName[2].setText(tvLon.getText().toString());
}

// -----

// 아래는 현재 위치를 표시하는 함수
@Override
public void onLocationChanged(Location location) {
    double lat = location.getLatitude();
    double lon = location.getLongitude();
    tvLat.setText(Double.toString(lat)); // 현재 위도 표시
    tvLon.setText(Double.toString(lon)); // 현재 경도 표시
}
        
```

"이 위치로 등록"을 클릭하면 tvLat과 tvLon 즉 현재 위도와 경도를 표시하는 TextView의 문자열을 enName[1], etName[2] 즉 editText부분을 setText함으로써 값을 입력받게 하였다.

⑦ AddActivity에서 현재 존재하는 이름이면 입력받게 하지 않게하는 예외처리




```

public void homeMove(View view)
{
    // 예외처리: 이름이 중복되는지 확인한다.
    for(int i = 0; i < _dataList.size(); i++) {
        if( _dataList.get(i).name.equals( etName[0].getText().toString() )) {
            Toast.makeText(this, "이미 존재하는 이름입니다.", Toast.LENGTH_SHORT).s
how();
            return;
        }
    }
    ..생략
    
```

이름을 중복 처리하는 과정은 당연히 모든 dataList에 있는 name과 문자열이 같은지 확인하면 된다.

⑧ MainActivity에서 값이 잘 전달됐는지 확인 및 정보보기

내가 이 기능을 추가로 구현한 이유는 AddActivity에서 전달한 dataList들을 MainActivity에서 값을 확인하고 싶어서 따로 구현하였다. 왼쪽화면과 같이 등록을 해서 [정보보기]버튼을 클릭하면 Toast 메시지가 뜨도록 하였다.

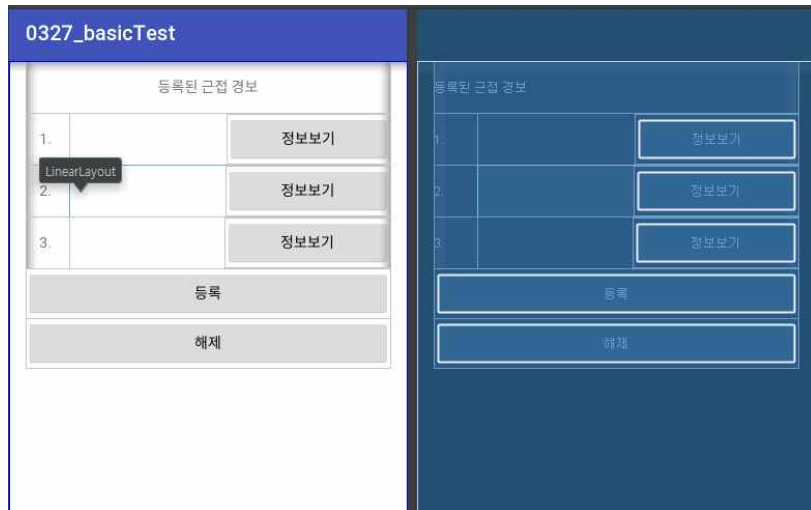


```

public void ClcikInfo(View view) {
    int i = -1;
    switch(view.getId()) {
        case R.id.IdInfo1 : // 첫 번째 번호의 정보보기, 그 원소의 갯수가 충분하
다면 그 번호 반환
            if(dataList.size() >= 1) i = 0;
            break;
        case R.id.IdInfo2 : // 두 번째 번호의 정보보기
            if(dataList.size() >= 2) i = 1;
            break;
        case R.id.IdInfo3 :// 세 번째 번호의 정보보기
            if(dataList.size() >= 3) i = 2;
            break;
    }
    if( i == -1 )
        Toast.makeText(getApplicationContext(),"정보가 없습니다.", Toast.LENG
TH_LONG).show();
    else
        Toast.makeText(getApplicationContext(),
            "이름: " + dataList.get(i).name + "\n위도: " + dataList.get(i).la
t + "\n경도: " + dataList.get(i).lon + "\n반경: " + dataList.get(i).rad, Toast.L
ENGTH_LONG).show();
    }
    
```

3.6. Layout UI

① MainActivity UI



이 MainActivity의 간략화된 코드는 다음과 같다.

```
<LinearLayout (vertical).. 생략>

    <LinearLayout (horizontal속성).. 생략>
        <TextView .. 생략 />
    </LinearLayout>

    <LinearLayout (horizontal속성).. 생략>

        <TextView .. 생략/>
        <TextView .. 생략/>
        <Button .. 생략/>

    </LinearLayout>

    <LinearLayout (horizontal속성).. 생략>

        <TextView .. 생략/>
        <TextView .. 생략/>
        <Button .. 생략/>

    </LinearLayout>

    <LinearLayout (horizontal속성).. 생략>

        <TextView .. 생략/>
        <TextView .. 생략 />
        <Button .. 생략/>

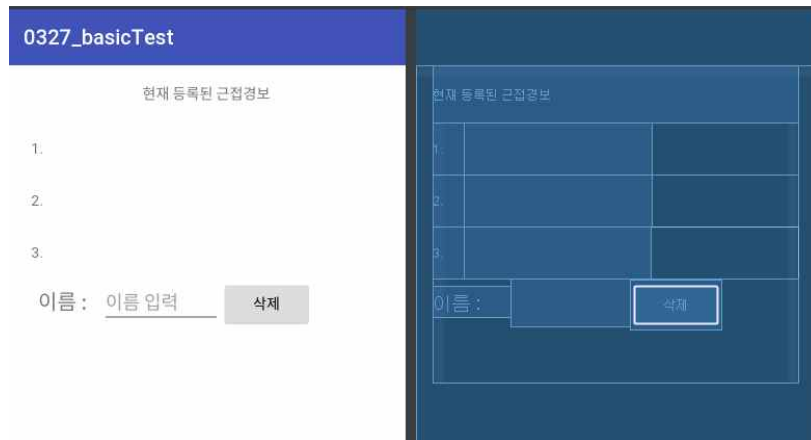
    </LinearLayout>

    <Button .. 생략/>
    <Button .. 생략/>

</LinearLayout>
```

위의 코드 표에 보듯이 먼저 LinearLayout의 Vertical 속성으로 여러 행의 또 다른 horizontal 속성을 가진 LinearLayout을 만들었다. 즉 하나의 LinearLayout에 여러 행 LinearLayout을 만든 셈이다. 이런 식으로 구현해서 칸에 알맞게 디자인 하였다.

② SubActivity UI



이 SubActivity의 간략화된 코드는 다음과 같다.

```
<LinearLayout (vertical) .. 생략 >

    <TextView .. 생략 />

    <LinearLayout (horizontal) .. 생략>

        <TextView .. 생략 />
        <TextView .. 생략/>

    </LinearLayout>

    <LinearLayout (horizontal) .. 생략>

        <TextView .. 생략/>
        <TextView .. 생략/>

    </LinearLayout>

    <LinearLayout (horizontal) .. 생략>

        <TextView .. 생략/>
        <TextView .. 생략/>

    </LinearLayout>

    <LinearLayout (horizontal) .. 생략>

        <TextView .. 생략 />
        <EditText .. 생략 />
        <Button .. 생략 />

    </LinearLayout>

</LinearLayout>
```

SubActivity도 마찬가지로 하나의 Vertical LinearLayout을 생성한 뒤 여러개의 horizontal LinearLayout을 만들었다.

③ AddActivity UI



AddActivity UI이 이다. 아무래도 표시하는 view들이 많아서 요소들이 다른 Activity보다 많다.

```
<LinearLayout (vertical) .. 생략 >

    <LinearLayout (horizontal) .. 생략>
        <TextView .. 생략/>
        <EditText .. 생략/>
    </LinearLayout>

    <LinearLayout (horizontal) .. 생략>
        <TextView .. 생략/>
        <EditText .. 생략/>
    </LinearLayout>

    <LinearLayout (horizontal).. 생략>
        <TextView .. 생략/>
        <EditText .. 생략/>
    </LinearLayout>

    <LinearLayout (horizontal) .. 생략>
        <TextView .. 생략/>
        <EditText .. 생략/>
    </LinearLayout>

    <Button .. 생략 />

    <LinearLayout (horizontal) .. 생략">
        <TextView .. 생략 />
    </LinearLayout>

    <LinearLayout (horizontal) .. 생략>
        <TextView .. 생략/>
        <TextView .. 생략/>
    </LinearLayout>

    <LinearLayout (horizontal) .. 생략>
        <TextView .. 생략/>
        <TextView .. 생략/>
    </LinearLayout>

    <Button .. 생략/>
</LinearLayout>
```

AddActivity도 마찬가지로 하나의 Vertical Layout에 여러 개의 Horizontal Layout을 넣었다. 위의 3개의 Activity를 이런 식으로 구현한 이유는 view들을 하나의 행으로 놓고 만들면 구현하기가 편리하기 때문이다

다.

4. 과제 수행 소감

처음으로 모바일 시스템 프로그래밍 과제를 해보았다. 난 모바일프로그래밍 교과목을 강승우 교수님의 수업으로 듣지 않은 터라 안드로이드 스튜디오의 '안'도 몰랐다. 처음부터 따라가기 급급했고, 복습은 물론 이거니와 기초 안드로이드 스튜디오 강의를 인터넷에서 찾아듣기도 했었다. 그런 나에게 첫 과제, 위치 기반 알람이 첫 과제였다.

처음엔 막막했다. 수많은 설정 코드들과 시스템 제어 코드들.. 안드로이드 스튜디오에 아직 적응하지 못한 나는 솔직히 힘들었다. 그래도 까짓것 해보자라는 마음이 생겼다. 그 이유는 내가 가진 핸드폰에서 실제로 동작하는 모습이 신기했기 때문이다. 그래서 처음 UI부터 차근차근 데이터 저장하고, Activity 구성하는 것까지 해보았다.

그런데 나는 Activity 간 데이터 전달에서 막혔다. Activity 간 데이터 전달은 intent와 Bunddle 등 여러 가지 키워드들이 있었고, 데이터 전달하려면 startActivity로 다른 Activity를 실행하면 안된다는 것을 알았다.(startActivityForResult로 해야했다) 물론 그렇게 배워가니까 재밌었다. 그것을 깨닫는데까지 거의 몇 일째 컴퓨터를 붙잡고 있었던 것까지 포함해서 말이다.

그뿐만이 아니었다. 여러 지역의 알람을 받기 위해선 IntentFilter의 key값도 달라야했고, proximityIntent도 각각 생성해줘야 했음을.. 여러 가지 시행착오 끝에 과제를 완성할 수 있었다.

이번 과제를 하면서 기억남는 건 내가 이 과제를 하면서 방에서 프로그램만 짜는게 아니라 집 밖으로 나와 열심히 돌아다녔다는 것. 계속 앉아 있는 것 보다 여기저기 돌아다니면서 값도 확인해보고 뭐가 문제인지 고심하면서 하루를 보낸 것이 의미있지 않은가. 물론 새벽 아침 해 뜰 때까지 그런 짓을 반복했지만 말이다. 무튼 재밌었다. :)

- 끝