2017 LINC+ 사업단
사회맞춤형 JAVA 실무인력양성과정

안드로이드 앱 개발 기초

컴퓨터공학부
강승우

7. 파일 IO / 리스트 뷰

2018.02.12

파일 IO

안드로이드에서 데이터 저장하기

- 애플리케이션에서 생성한 데이터를 저장하기 위한 방법
  - 파일 이용
    - 내부 저장소
    - 외부 저장소
  - 공유 프레퍼런스 (Shared Preferences) 이용
    - 애플리케이션 환경 설정 정보
  - 데이터베이스 이용
    - SQLite 데이터베이스
  - 서버 이용
    - 네트워크를 통한 데이터 전송

안드로이드 파일 저장소 영역

- 파일 저장소 종류
  - 내부 저장소 (Internal storage)
  - 외부 저장소 (External storage)

- 파일 저장소의 물리적 구분
  - 초기 내장 메모리와 마이크로 SD 카드와 같은 이동식 저장 장치가 같이 있었던 때는 내장 메모리를 내부 저장소, 이동식 저장 장치를 외부 저장소로 이용
  - 이동식 저장 장치가 없는 경우에도 내장 메모리를 내부와 외부로 나누어 두 가지의 저장소 공간 제공

안드로이드 파일 저장소의 특징

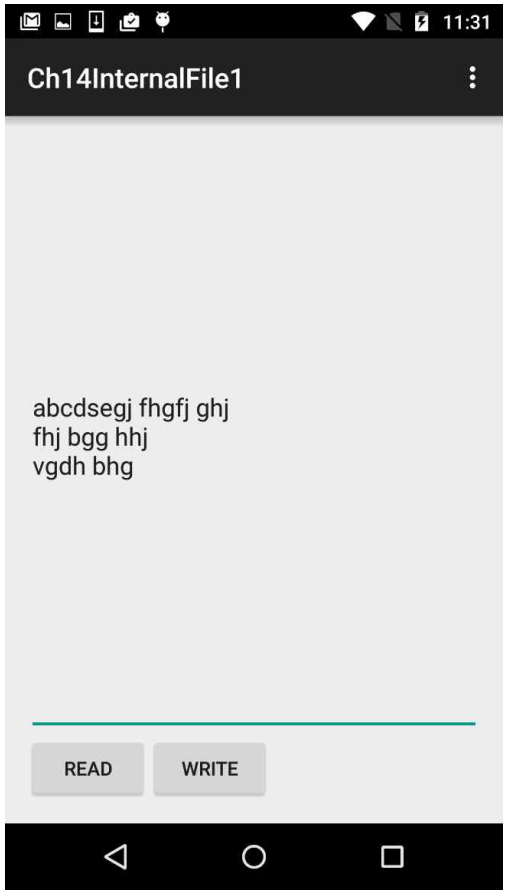
- 내부 저장소
  - 항상 사용 가능
  - 여기에 저장된 파일은 해당 앱에서만 액세스 가능 (앱을 설치하면 저장 공간이 할당됨)
  - 사용자가 앱을 삭제하면 시스템이 내장 저장소에서 앱의 모든 파일을 제거
  - ✓내부 저장소는 사용자와 다른 앱이 자신의 파일에 액세스하는 것을 원치 않을 때 적합
- 외부 저장소
  - 항상 사용 가능한 것은 아님
    - 사용자가 USB 저장소와 같은 외부 저장소를 마운트하고 경우에 따라 기기에서 외부 저장소를 제거할 수 있기 때문
  - 외부 저장소에 저장된 파일은 다른 애플리케이션에서도 접근 가능하고, 사용자에 의해서 변경될 수 있음
  - 외부 저장소를 이용하는 경우에도 특정 애플리케이션이 사적으로 사용하는 파일 생성 가능
    - 사용자가 앱을 삭제하면 getExternalFilesDir()의 디렉터리에 저장한 앱 파일에 한해서 시스템이 제거
    - 사용자가 애플리케이션을 제거하더라도 공용 디렉토리에 저장된 파일은 삭제되지 않음
  - ✓외부 저장소는 액세스 제한이 필요치 않은 파일과 다른 앱과 공유하기 원하는 파일 또는 사용자가 컴퓨터에서 액세스할 수 있도록 허용하는 파일에 적합

내부 저장소 파일 입출력

- Context class에 정의된 파일 입출력 메소드 이용
  - FileOutputStream openFileOutput(String name, int mode)
  - FileInputStream openFileInput(String name)
  - boolean deleteFile(String name)

- Mode
  - Context.MODE_PRIVATE
    - 다른 애플리케이션 패키지에서 접근할 수 없는 파일을 생성할 때 사용
  - Context.MODE_APPEND
    - 기존 파일의 끝에 데이터를 추가하기 위한 용도로 파일을 열 때 사용
  - Context.MODE_WORLD_READABLE / Context.MODE_WORLD_WRITABLE
    - 다른 애플리케이션이 접근 가능하도록 하는 모드
    - 보안 상의 이유로 deprecated (API level 17 – Android 4.2부터)
    - 사용하지 않는 것이 좋다

텍스트 파일 입출력 예제



- 예제 프로젝트 이름: Ch14InternalFile1

- EditText에 텍스트 입력

- WRITE 버튼을 누르면 지정된 파일 이름을 갖는 파일에 저장
  - String FILENAME = “text.txt”;
  - 예제는 위와 같이 파일 이름을 하드 코딩한 상태

- READ 버튼을 누르면 동일한 이름의 파일을 읽어서 그 내용을 EditText에 설정

파일 쓰기

```
Button writeBtn = (Button)findViewById(R.id.write);
writeBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
            fos.write(edit.getText().toString().getBytes());
            fos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
});
```

- Private 모드로 파일 생성 후 쓰기
- 쓰기 후 close

File IO 관련 API를 사용하는 경우 IOException 처리를 해주어야 함

파일 읽기

- 파일 읽기 위해 파일 오픈
- 버퍼 생성 후 읽기 수행

- 읽은 내용을 화면에 표시하기 위해 EditText 객체에 설정

```
Button readBtn = (Button)findViewById(R.id.read);
readBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            FileInputStream fis = openFileInput(FILENAME);
            byte[] buffer = new byte[fis.available()];
            fis.read(buffer);

            edit.setText(new String(buffer));
            fis.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
});
```

파일 경로

- 안드로이드 내장 메모리 공간
  - 리눅스 파일시스템
    - 안드로이드 시스템 관련한 모든 파일
    - 사용자가 설치한 앱 파일
  - 앱이 설치되면 그 앱에서 사용할 수 있는 별도의 공간이 할당됨

  → 앱 홈 폴더: /data/data/패키지명

- /data/data/패키지명/files
  - openFileInput, openFileOutput, deleteFile 등의 메소드에서 접근하는 폴더
  - Context 클래스에 정의된 getFilesDir() 메소드를 이용해 경로를 얻어올 수 있음

관련 메소드

- FILE getFilesDir()
- String[] fileList()
  - 애플리케이션이 현재 저장한 파일 리스트 반환
- FILE getFileStreamPath(String name)
  - 특정 파일의 경로를 얻어옴
  - /data/data/files/파일명

텍스트 파일 입출력 예제 2

- 예제 프로젝트 이름: Ch14InternalFile2
  - TextFileManager라는 별도의 클래스로 파일 쓰기, 읽기, 삭제를 처리하는 예제
- MainActivity에서는 TextFileManager 객체를 생성하여 이용
- 불러오기, 저장, 삭제 버튼을 선택하였을 때 TextFileManager 객체의 관련 메소드 호출하여 필요한 작업 수행
  - save(), load(), delete() 메소드
- 예제 소스 참고

리스트 뷰

지금까지 사용해 온 레이아웃

- 레이아웃에 들어가는 내용(뷰 항목)이 고정되어 있는 것
  - TextView, Button, EditText, ImageView 등
  - 텍스트 내용을 바꾸거나 이미지를 변경하는 것은 가능

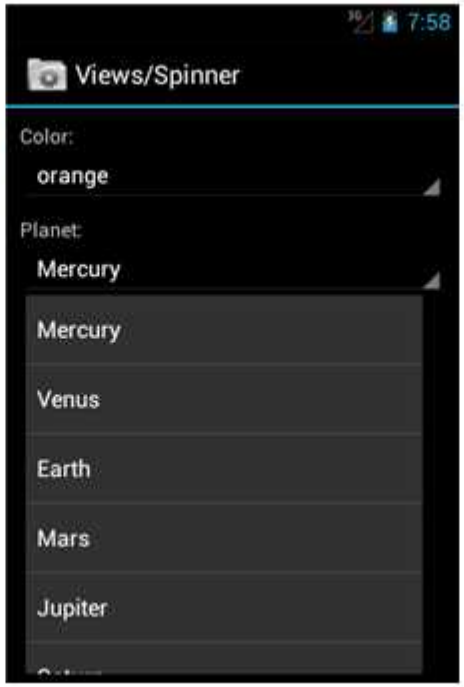
- 하지만, 새로운 TextView 항목을 추가하는 등의 동적인 변경을 하고자 할 때는 어떻게 해야 할까?

어댑터 뷰 (AdapterView)

- 화면에 동적으로 변경되는 컨텐츠를 채울 때 사용하는 뷰
  - 배열, 파일, 데이터베이스에 저장된 데이터를 화면에 표시할 때 유용

ListView

Gallery



Spinner

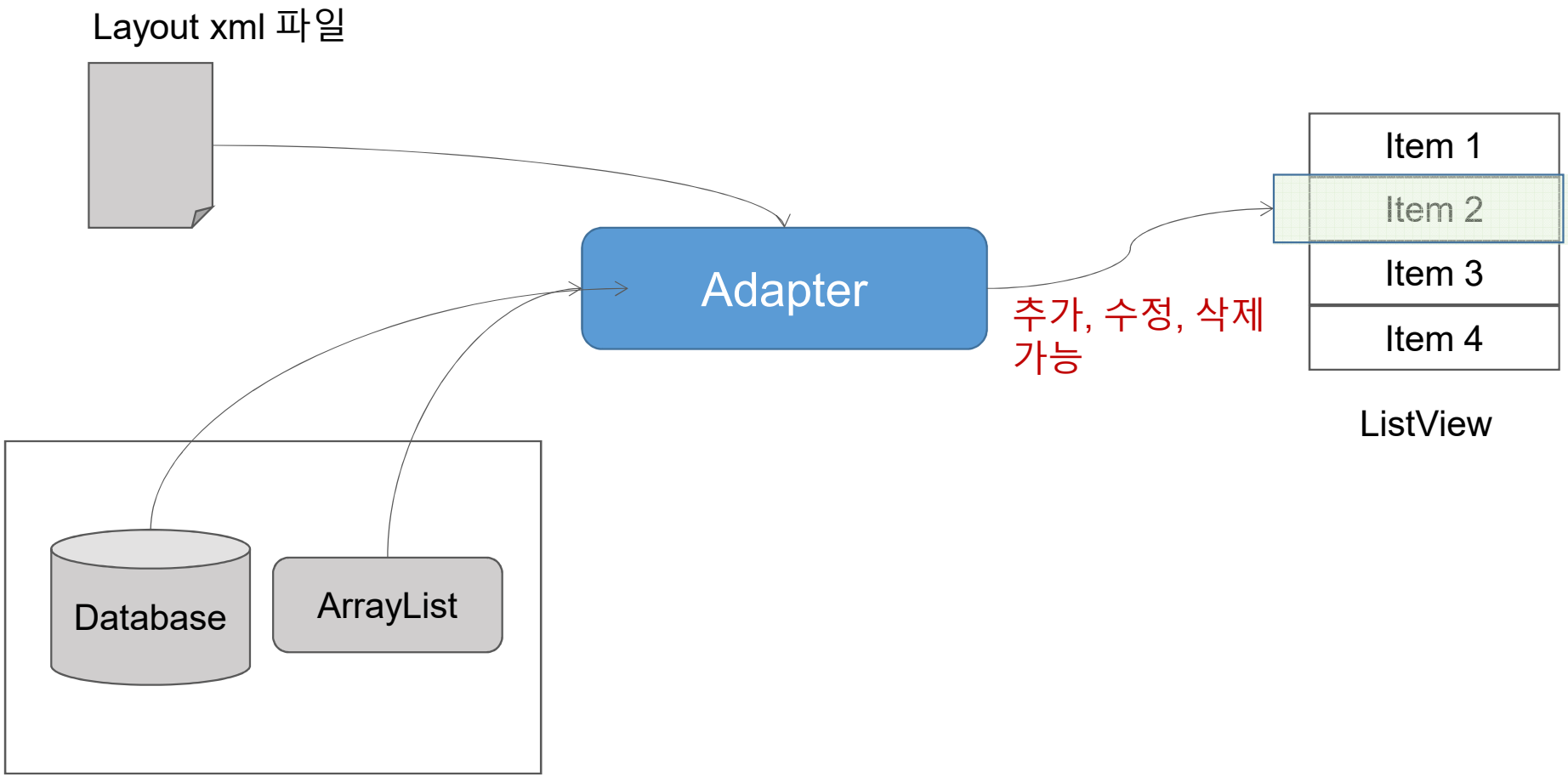
GridView

어댑터 (Adapter)

- Adapter를 사용하여 데이터를 어댑터 뷰에 제공
  - 어댑터는 데이터 소스와 어댑터 뷰 중간에 위치하여 데이터 소스에서 데이터를 읽어서 어댑터 뷰에 공급
  - https://developer.android.com/reference/android/widget/Adapter.html
- ArrayAdapter
  - 배열에서 데이터를 가져오는 어댑터
  - https://developer.android.com/reference/android/widget/ArrayAdapter.html
- SimpleCursorAdapter
  - 데이터베이스에서 데이터를 가져오는 어댑터

➢ TextView와 같은 기본 위젯은 뷰에 직접 데이터를 설정
  - setText() 메소드를 이용했음

어댑터


Layout xml 파일
Adapter
추가, 수정, 삭제
가능
Item 1
Item 2
Item 3
Item 4
ListView
Database
ArrayList

리스트 뷰 (ListView)

- 항목들을 수직 방향의 목록 형태로 보여주는 어댑터 뷰
  - 상하 스크롤이 가능
  - 일반적으로 목록의 한 항목을 선택하여 일정한 작업 수행

리스트 뷰 생성 방법

- 레이아웃 파일에 <ListView> element 선언하기
  - 예제 프로젝트 이름: Ch9ListView

- ListActivity를 상속받는 액티비티로 만들기
  - 리스트 뷰가 레이아웃 화면으로 미리 설정되어 있는 ListActivity 사용
    - https://developer.android.com/reference/android/app/ListActivity.html
  - 예제 프로젝트 이름: Ch9ListView2

리스트 뷰 예제 1 – ListView element

```
public class MainActivity extends AppCompatActivity {
    private ListView m_ListView;
    private ArrayAdapter<String> m_Adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        String[] values = {"하스스톤", "몬스터 헌터", "디아블로", "와우", "리니지", "안드로이드", "아이폰"};

        // Android에서 제공하는 String 문자열 하나를 출력하는 layout으로 어댑터 생성
        m_Adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, values);
        // Xml에서 추가한 ListView의 객체
        m_ListView = (ListView) findViewById(R.id.list);
        // ListView에 어댑터 연결
        m_ListView.setAdapter(m_Adapter);
        // ListView 아이템 터치 시 이벤트를 처리할 리스너 설정
        m_ListView.setOnItemClickListener(onClickListItem);
    }
```

리스트 뷰 예제 1 – ListView element

- 리스트 뷰 항목을 선택 이벤트 처리를 위한 AdapterView.OnItemClickListener 를 구현해야 함

```
// 아이템 터치 이벤트 리스너 구현
private AdapterView.OnItemClickListener onClickListItem = new AdapterView.OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        // 이벤트 발생 시 해당 아이템 위치를 텍스트로 출력
        Toast.makeText(getApplicationContext(), m_Adapter.getItem(position), Toast.LENGTH_SHORT).show();
    }
};
```

리스트 뷰 예제 2 – ListActivity 상속

```
public class MainActivity extends ListActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main);
        String[] values = {"하스스톤", "몬스터 헌터", "디아블로", "와우", "리니지", "안드로이드", "아이폰"};

        // Android에서 제공하는 String 문자열 하나를 출력하는 layout으로 어댑터 생성
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, values);
        // ListView에 어댑터 연결
        setListAdapter(adapter);
    }

    @Override
    protected void onListItemClick(ListView l, View view, int position, long id) {
        String item = (String)getListAdapter().getItem(position);
        Toast.makeText(getApplicationContext(), item + " selected", Toast.LENGTH_SHORT).show();
    }
}
```

- 리스트 뷰 항목을 선택 이벤트 처리를 위해 구현해야 하는 메소드

리스트 뷰의 표준 레이아웃

android.R.layout.

레이아웃 ID	설명
simple_list_item_1	하나의 텍스트 뷰 사용
simple_list_item_2	두개의 텍스트 뷰 사용
simple_list_item_checked	항목당 체크 표시
simple_list_item_single_choice	한 개의 항목만 선택
simple_list_item_multiple_choice	여러 개의 항목 선택 가능

- 앞의 예제에서 사용한 레이아웃

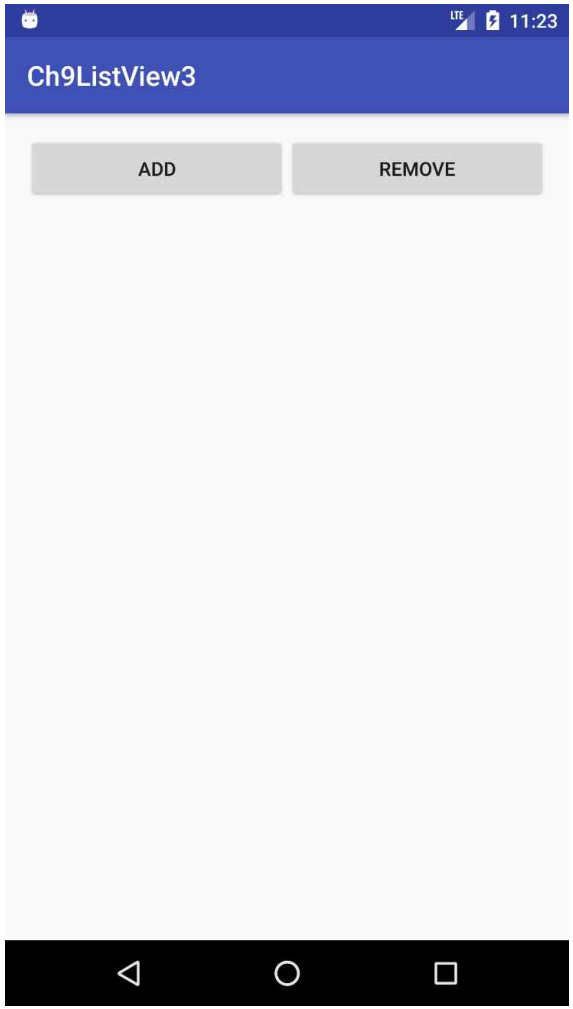
리스트 뷰 항목 추가/삭제 예제

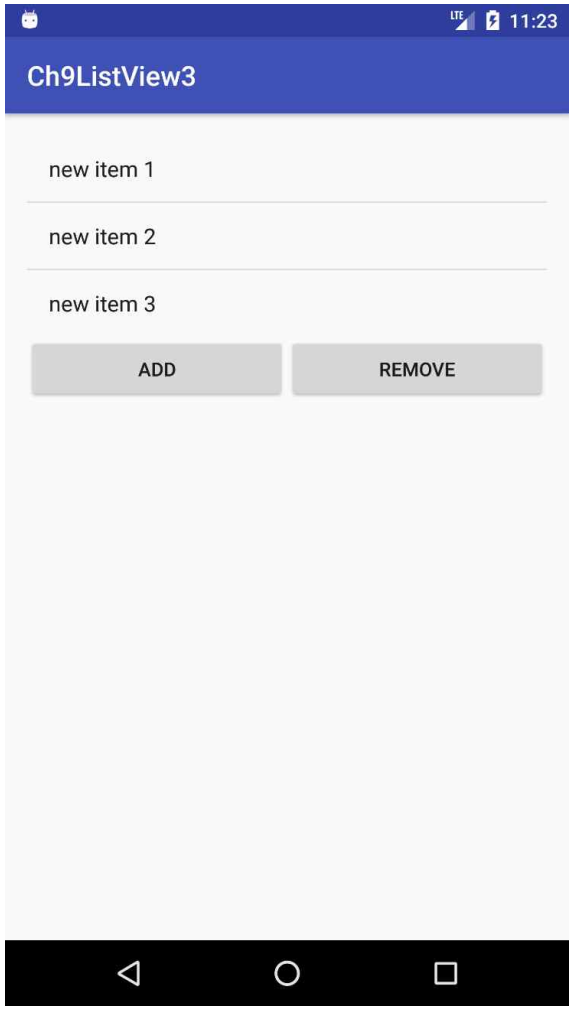
- 예제 프로젝트 이름
  - Ch9ListView3
  - Add, Remove 버튼을 이용하여 항목 추가, 삭제
- 관련 API
  - ArrayAdapter의 add, remove 메소드 사용
  - void add(T object)
  - void remove(T object)

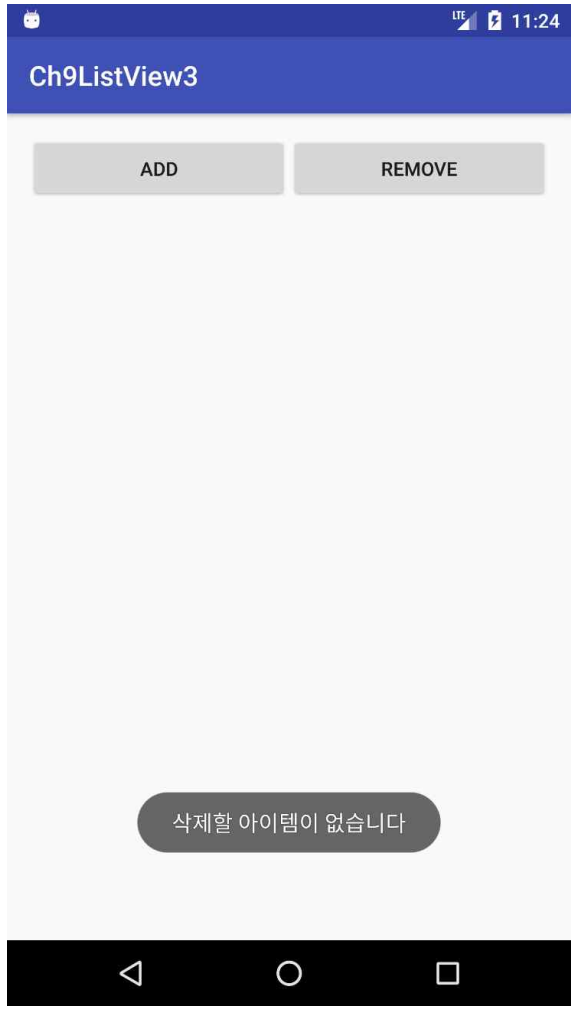
  - 추가, 삭제를 하려면 ListView로 보여지는 아이템은 정적 배열 객체가 아닌 List 객체여야 함
    - ArrayAdapter(Context context, int resource, T[] objects)
    - ArrayAdapter(Context context, int resource, List<T> objects)

https://developer.android.com/reference/android/widget/ArrayAdapter.html

리스트 뷰 항목 추가/삭제 예제







```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ArrayList<String> values = new ArrayList<>();

    // Android에서 제공하는 String 문자열 하나를 출력하는 layout으로 어댑터 생성
    m_Adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, values);

    // Xml에서 추가한 ListView의 객체
    m_ListView = (ListView) findViewById(R.id.list);

    // ListView에 어댑터 연결
    m_ListView.setAdapter(m_Adapter);

    // ListView 아이템 터치 시 이벤트를 처리할 리스너 설정
    m_ListView.setOnItemClickListener(onClickListItem);
}
```

values를 String 객체를 담는 ArrayList 객체로 선언

이 부분은 앞의 Ch9ListView 예제와 동일

```
public void onClick(View view) {
    int count;
    count = m_Adapter.getCount();

    if(view.getId() == R.id.add) {
        // add 버튼 클릭한 경우 리스트의 마지막에 새 아이템 추가
        // adapter에 아이템 추가
        m_Adapter.add("new item " + Integer.toString(count + 1));
    } else if(view.getId() == R.id.remove) {
        // remove 버튼 클릭한 경우 리스트의 마지막 아이템을 삭제
        // 삭제할 아이템이 없으면 메시지 출력 후 종료
        if (count == 0) {
            Toast.makeText(getApplicationContext(), "삭제할 아이템이 없습니다", Toast.LENGTH_SHORT).show();
            return;
        }
        // 리스트의 마지막 아이템을 얻음
        String item = m_Adapter.getItem(count - 1);
        // 해당 아이템을 adapter에서 삭제
        m_Adapter.remove(item);
    }
}
```