

안드로이드 프로그래밍

핵심 5주 과정

# 수업준비 A. 안드로이드 개발 환경 설정

## 1. 자바 설치

윈도우 기준 : [limkydev.tistory.com/61](http://limkydev.tistory.com/61)

맥 OS 기준 :

<http://ishappy.tistory.com/entry/MAC-OS-X-%EC%97%90-JDK-%EC%84%A4%EC%B9%98%ED%95%98%EB%8A%94-%EB%B0%A9%EB%B2%95>

## 2. 안드로이드 스튜디오 설치

<https://m.blog.naver.com/pyj721aa/221275562630>

## 3. **Git Hub** 가입

<https://github.com/>

## 4. **SourceTree** 설치

<https://coding-factory.tistory.com/249>

# 수업준비 B. 안드로이드 프로젝트 개발 환경 설정

“ 안드로이드 스튜디오는 마치  
도서관에서 과제를 하고 있는 학생의 책상 과 같습니다”

“ 왼쪽의 사진에서 처럼 ,  
작업할때 필기도구 , 원고지 , 책상 등이 개발환경 ”



“ 참조하기 위해 , 책장에서 꺼내와서 책상위에 쌓아놓은 책들이 Gradle 입니다 ”

“ 학생 뒤로 보이는 , 어느 누가 읽을지 모르지만 산더미 같이 쌓아놓고 관리하고 있는 책장은 Maven ”

---

“ 우리가 작성한 자바 언어를 컴퓨터가 이해할수 있는 언어로 변환하는 과정을 ‘빌드’ 라고 표현 합니다 ”

“ 이러한 빌드를 도와주는 착하고 똑똑한 도구가 Gradle 입니다 . 요즘 아주 인기 있는 녀석 이죠 ”

“ 이 Gradle 은 우리를 대신해서 컴퓨터 언어로 변환을 담당할 뿐만 아니라 ,  
친절하게 책장에서 우리에게 책을 가져다 주기 까지 해줍니다 . 이 행위를 Link 라고 합니다 ”

# 수업준비 B. 안드로이드 프로젝트 개발 환경 설정

“ 우리 수업을 위한 Gradle 세팅이 가장 첫번째 단계 입니다 ”

1. Project 단위 Gradle 에서 , 어느 책장 (Maven Repository) 을 참조 할 것 인지 지정 해줍니다

```
allprojects {  
    repositories {  
        google()  
        jcenter()  
  
        maven { url 'https://jitpack.io' }  
    }  
}
```

2. Project 를 구성하는 모듈 단위 Gradle 에서 , 어떤 책 (Library) 을 참조할 것 인지 지정 해줍니다.

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"  
  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
  
    implementation 'com.android.support:appcompat-v7:28.0.0'  
    implementation 'com.android.support:design:28.0.0'  
    implementation 'com.android.support:cardview-v7:28.0.0'  
  
    implementation 'com.github.voltsoftdev:root:0.3.0'  
  
    implementation 'com.github.bumptech.glide:glide:3.7.0'  
    implementation 'jp.wasabeef:glide-transformations:2.0.1'  
  
    implementation 'com.google.firebase:firebase-core:16.0.1'  
    implementation 'com.google.firebase:firebase-database:16.0.1'  
}
```

# 1주차 A. 프로그래밍 이란 무엇인가 - (1) 단위



“

4가지 원소 를 이용하여 세상을 구성 했던 것 처럼

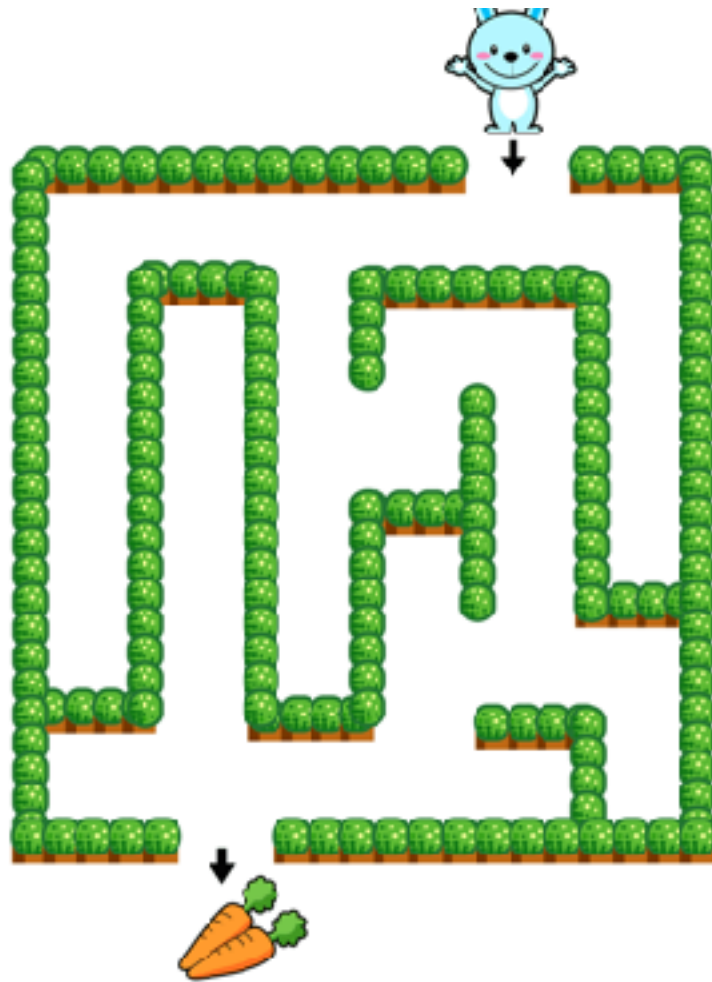
4가지 원시타입 으로 컴퓨터 세상을 구성합니다

”

“ 사람을 구성하는 요소 나이 , 이름 , 성별 , 주소 , 직업 , 키

를 가지고 사람을 만들어봅시다 ”

# 1주차 B. 프로그래밍 이란 무엇인가 - (2) 방법



“

토끼가 벽에 부딪히지 않고 당근까지 무사히 갈 수 있도록

길잡이를 입력해주세요

”

“ 토끼는 길이 열려 있는지 확인 하는 ‘검색’ 동작이 필요합니다  
그리고 움직이는 ‘진행’ 동작이 필요하죠 ”

# 1주차 C. 객체지향 프로그래밍 이란 무엇인가

- 객체지향 프로그래밍 이란 ?

“ 객체를 하나의 단위로서 시스템을 구축 하는 것 ”

- 객체란 무엇인가 ?

“ 객체란 추상적인 개념의 단위 혹은 변화의 단위 이다 ” 혹은

“ 객체란 참조 타입 ”

- 참조타입 이란 무엇인가 ?

“ 힙에 할당되어 있는 메모리 주소를 값으로 가지고 있는 변수 ,

자바 에서는 원시타입을 제외한 모든 변수를 일컫는다 ”

- 객체지향 프로그래밍 설계의 첫걸음 은 무엇인가 ?

“ 구현하고자 하는 시스템을 구성하는 객체 (Model) 들을 정의

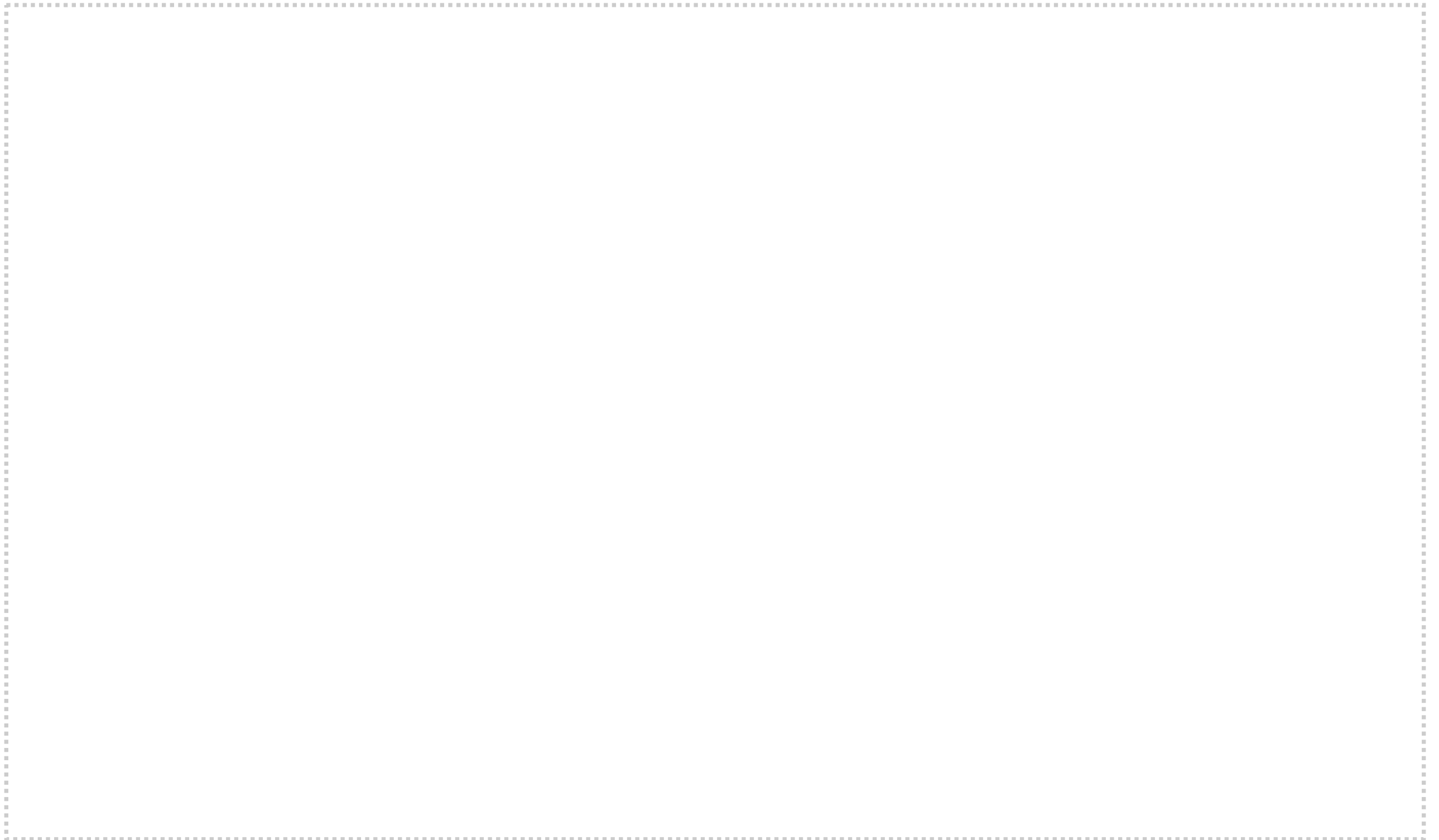
이 객체 들을 화면에 어떻게 보여줄 것인지 설계 (View)

정의한 객체 (Model) 을 구현화 하고 화면 (View) 에 보여주는 역할을 하는 객체

(Controller 혹은 Presenter) 설계 “

# 1주차 C. 객체지향 프로그래밍 설계 첫걸음 MVC

“ 우리가 만들고자 하는 샘플앱을 MVC 로 설계 해봅시다 “





# 1주차 D. 저는 일단 화면 부터 만들고 싶어요

“ 안드로이드 화면은 뷰 와 뷰를 담는 그릇 (레이아웃) 으로 구성 되어 있습니다 ”

“ 안드로이드 레이아웃 중 리니어 레이아웃 (**LinearLayout**) 프레임 레이아웃 (**FrameLayout**) 이 가장 핵심입니다 ”

## 1. 뷰 를 나란히 세로로 (혹은 가로로) 배치 해야 하는 상황

“ 리니어 레이아웃 의 Orientation 속성 VERTICAL 혹은 HORIZONTAL 사용 ”

## 2. 나란히 배치된 뷰들이 자동으로 균등하게 배열되어야 하는 상황

“ 리니어 레이아웃 의 weighSum 과 layout\_weight 을 이용 ”

## 3. 뷰 들을 겹치게 배치 해야 하는 상황

“ FrameLayout 을 이용 , layout\_gravity 를 이용하여 위치 기준점을 설정 . 기준점 설정 이후에는

left , top , right , bottom margin 을 이용해서 배치 ”

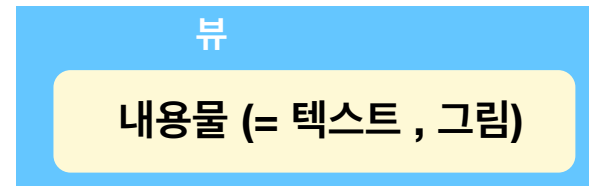
## 4. 뷰 들을 유동적으로 노출 , 미노출을 결정 해야 하는 상황

“ 뷰 의 visibility 를 이용 하여 결정 , 이때의 Visible = 노출 , Invisible = 미노출 (뷰 자체 위치 , 크기는 유지) ,

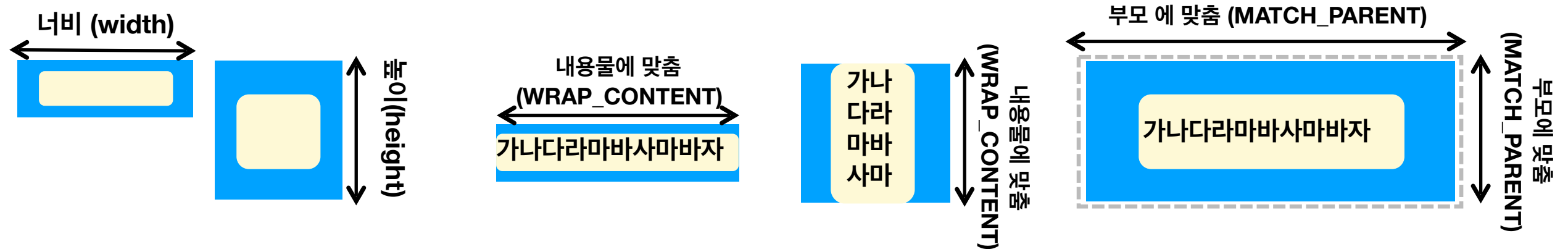
Gone = 뷰 자체를 완전 사라지게 함 ”

# 1주차 E. 그림 일단 화면 에서 뷰 부터 만들고 싶어요

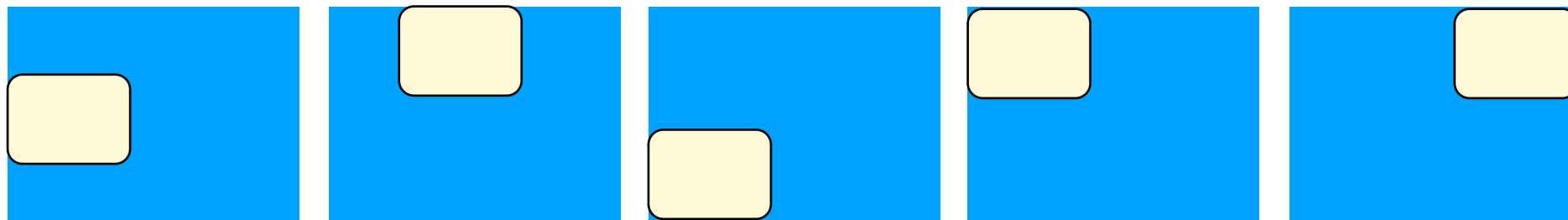
“뷰는 내용물 (Contents) 를 담고 있습니다 ”



“ 뷰를 레이아웃안에 배치 하려면 먼저 너비(width) 와 높이(height) 값이 필요합니다 ”



“ 뷰 안에 있는 내용물 의 위치 (Gravity) 를 배치 할 수도 있습니다 ”



“ 뷰 와 내용물 간에 간격을 패딩 (Padding) 이라고 합니다 ”



# 1주차 F. 이제 뷰를 화면에 담고 싶어요

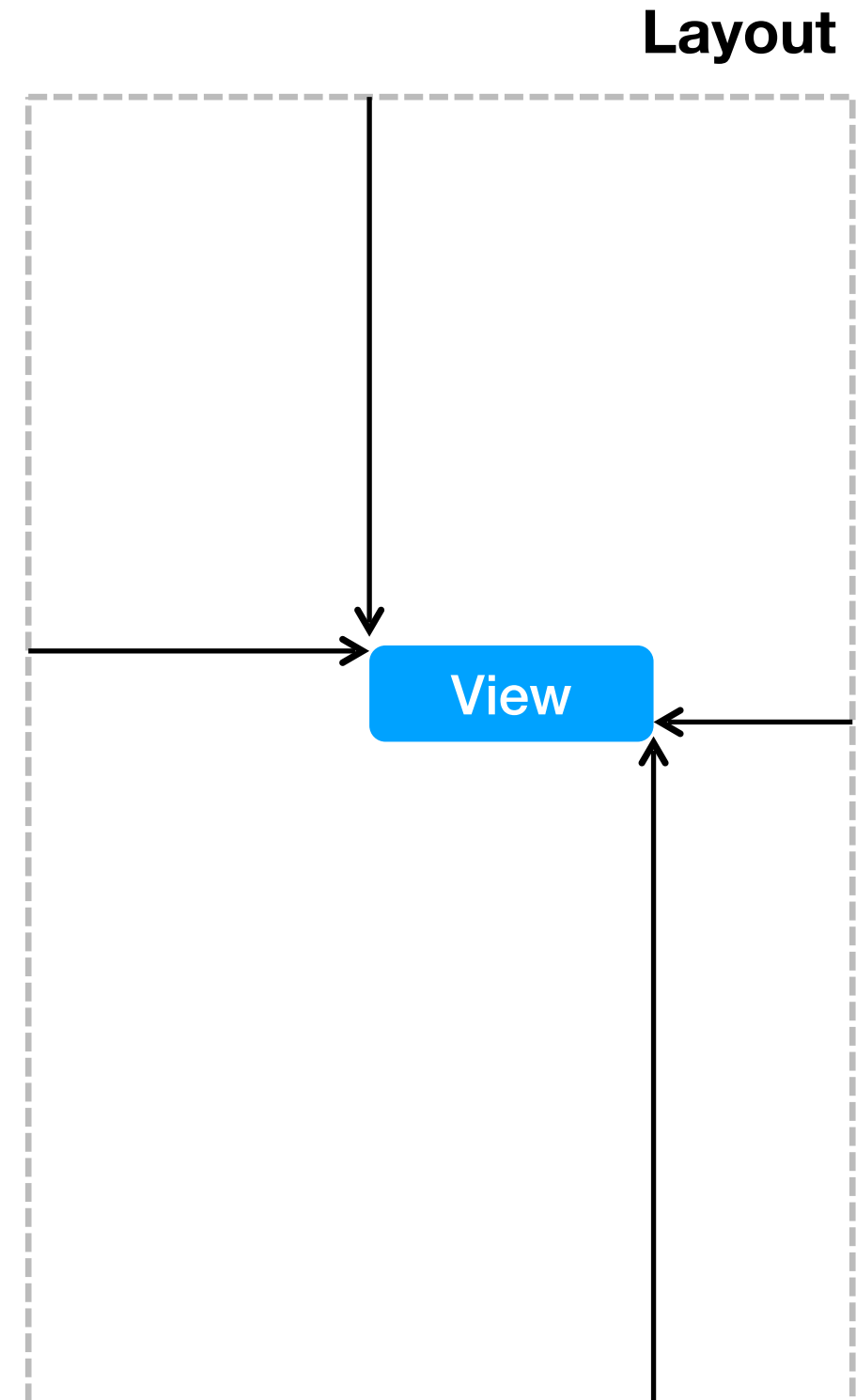
“ 뷰와 레이아웃 간의 간격을 마진 이라고 합니다 ”

“ 레이아웃 안에 뷰의 위치 (Gravity) 를 배치 할 수 있습니다 ”

“ 뷰를 여러 개 세로로 배치 하려면 \_\_\_\_\_Layout\_\_\_\_\_ ”

“ 뷰를 여러 개 가로로 배치 하려면 \_\_\_\_\_Layout\_\_\_\_\_ ”

“ 뷰를 여러 개 겹치게 배치 하려면 \_\_\_\_\_Layout ”

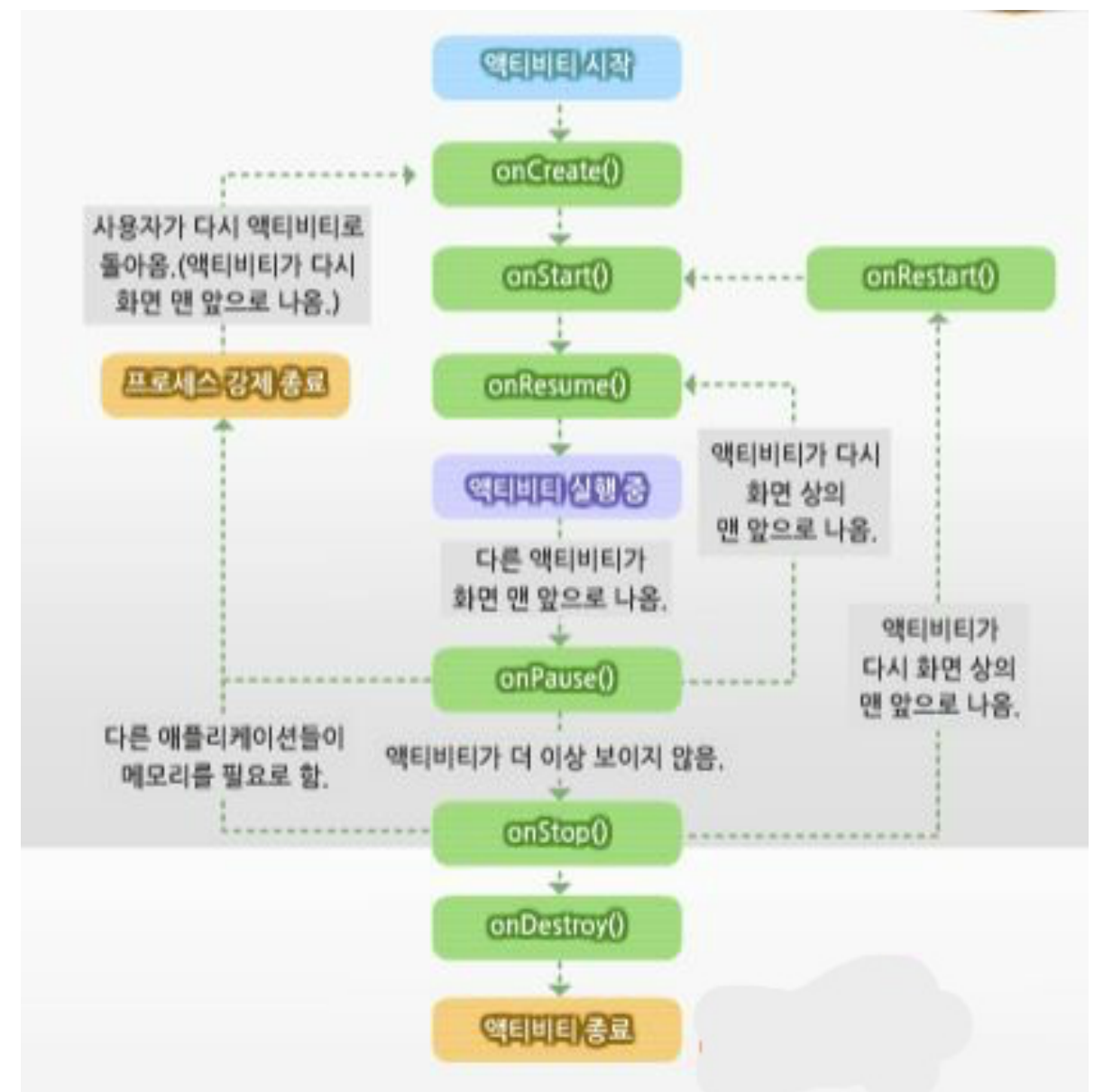


# 1주차 G. 이제 이 화면 을 어떻게 노출 해야하나요 ?

“ 액티비티 = 사용자가 접하는 화면 ”

“ 안드로이드 화면을 어떻게 노출 할지 정하는 법 = 액티비티 라이프 싸이클 을 제어 하는법 ”

“ 액티비티 라이프 싸이클 = 화면 생성 순서 ”



# 1주차 H. 안드로이드 액티비티 외에 이런것도 있어요

구글에서 제공하는 안드로이드 SDK 는 크게 4가지로 나눌 수 있습니다 (4대 컴포넌트)

## 1. 액티비티

“ 안드로이드 어플리케이션 화면 단위 ”

## 2. 서비스

“ 백그라운드 에서 동작하는 메인 스레드 ”

## 3. 콘텐츠 프로바이더

“ 외부 앱에 데이터를 제공 할 수 있는 객체 ”

## 4. 브로드 캐스트 리시버

“ 어플리케이션 내부 및 외부 에서 변화를 수신 하는 객체 ”

# 1주차 1. 안드로이드 액티비티 예제

“ 버튼을 세개 가지고 있는 로그인 페이지를 만들어 봅시다”

공통 요건 (1) :

상단에서 64px 밑에 있는 150px X 150px 크기의 로고뷰를 위치 시킵니다

공통 요건 (2) :

하단에서 35px 위에서 부터 , 버튼 3개를 세로로 배치 시킵니다



# 1주차 I. 안드로이드 액티비티 예제

“ 버튼을 세개를 가지고 있는 로그인 페이지를 만들어 봅시다 ”

1. “ 액티비티 생성전에 , 안드로이드에게 내가 만들려는 액티비티는 이러한 액티비티 라고 알려 줘야 합니다 ”

= 안드로이드 매니페스트 에 어플리케이션 첫 실행 액티비티를 선언 해줍니다

2. “ 보여주고자 하는 화면을 **xml** 로 구성 합니다 ”

= **FrameLayout** 과 **LinearLayout** 을 이용하여 화면을 구성 합니다

3. “ **2번** 에서 생성한 화면 **xml** 을 인트로 액티비티 에 박아줍니다 “

= 액티비티 **onCreate** 에 layout xml 파일을 **setContentView** 를 통해 설정해줍니다.

4. “ 인트로 화면에서 생성한 버튼에 클릭 이벤트 동작을 박아줍니다 ”

= layout xml 파일에서 만든 버튼을 **findViewById** 를 이용하여 찾은후 , **setOnClickListener** 로 설정 합니다

5. “ 이벤트 동작 에는 화면 호출 동작을 넣어줍니다 ”

= **onClick** 안에서 **Toast** 팝업을 노출 합니다



다른 단말기에서 보니까  
제가 의도 했던 대로 화면이 나오지 않아요



# 1주차 J. 안드로이드 해상도 특성

“ 단말기 마다 특성 및 해상도 차이가 큼에 따라 , 이에 따른 개발 어려움이 태생적으로 존재합니다 ”

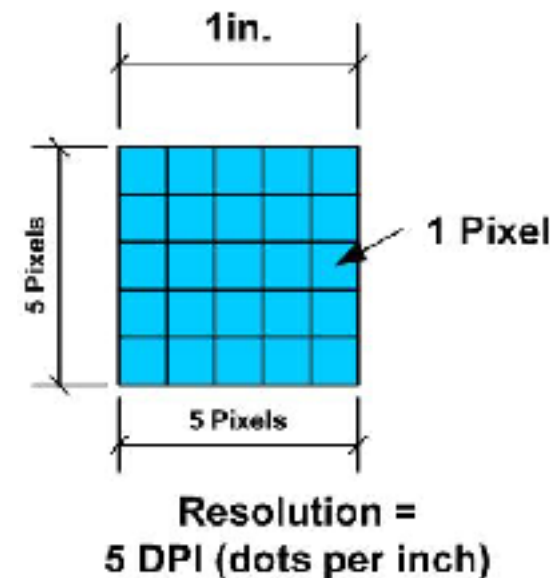
“ 핵심 원인은 , 뷰의 위치 나 크기 단위를 픽셀로 설정하면 안된다는 것 ”

## 1. 해상도란 무엇인가 ?

“ 1인치당 몇개의 픽셀 혹은 도트로 이미지를 표현하는지 에 대한 정도 ”

## 2. 안드로이드 해상도 단위

dpi = “ 1인치 x 1인치 공간 안에 얼마나 많은 픽셀을 표현 가능한지 에 대한 단위 ”



## 3. 안드로이드 해상도 종류

DPI	비율	픽셀
ldpi (120dpi)	0.75x	75px
mdpi (160dpi)	1x (기본 비율)	100px
hdpi (240dpi)	1.5x	150px
xhdpi (320dpi)	2x	200px
xxhdpi (480dpi)	3x	300px
xxxhdpi (640dpi)	4x	400px

# 1주차 K. 안드로이드 해상도 특성 대응 방안

(안드로이드 모든 뷰 는 화면에 노출 되기 전에 너비값과 높이값을 받아서 배치된다)

## 1. “ 리소스에 있는 **Dimension** 파일 활용 ”

수업에서 제공하는 디멘션 가이드 파일을 활용

## 2. “ Dimension 에 맞는 픽셀값을 반환 해주는 안드로이드 **TypedValue API** 사용 ”

안드로이드 TypedValue .. applyDimension 메소드 활용

```
... TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, [dp], context.getResources().getDisplayMetrics())
```

## 3. “ 디자인 가이드 기준 에 따라 , 해상도 값 계산 하는 메소드 사용 ”

수업에서 제공하는 해상도 대응 메소드를 참조 혹은 사용

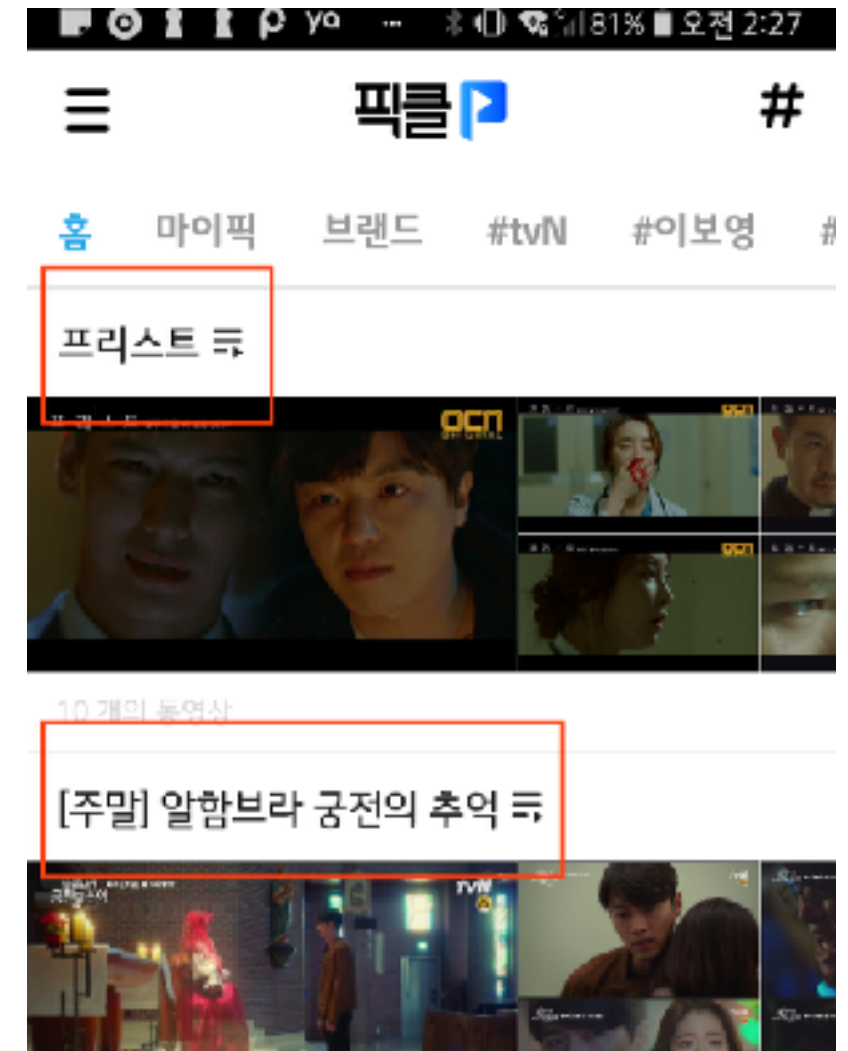
## 4. “ 어떠한 해상도의 단말기에서도 대처 가능하도록 리니어 레이아웃 비율 속성 을 활용 ”

리니어 레이아웃의 layout\_weight 값에 비율값을 설정

# 1주차 L. 실전 안드로이드 레이아웃 구성 과제

“ 매니저 K씨 : 과장님 이 리스트뷰 아이템 텍스트 타이틀 옆에 버튼이 붙어 있는데 , 텍스트 길이에 따라 가변적으로 늘었다가 줄었으면 좋겠어요 ”

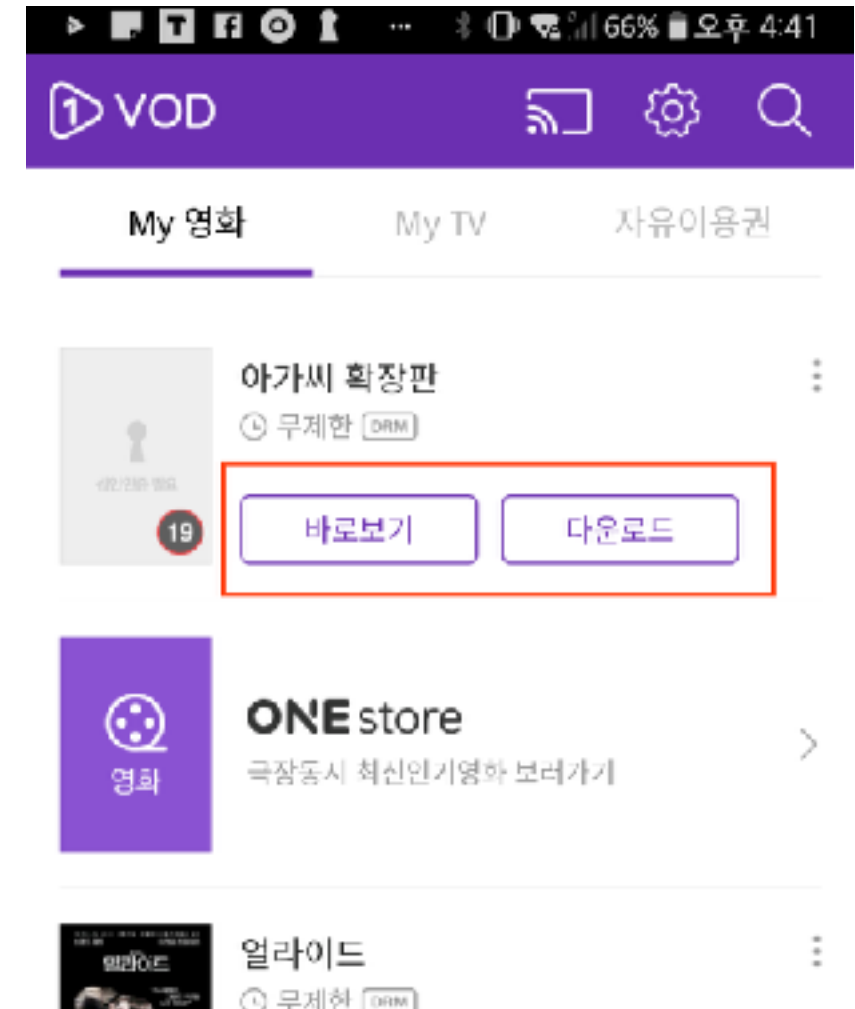
해결방안 :



# 1주차 L. 실전 안드로이드 레이아웃 구성 과제

“ 매니저 K씨 : 과장님 이 리스트뷰 아이템 버튼이 2분할인데 , 특정조건에서는 버튼이 다운로드 버튼만 나오게 하고 싶어요 . 그렇게 구현 가능할까요 ?”

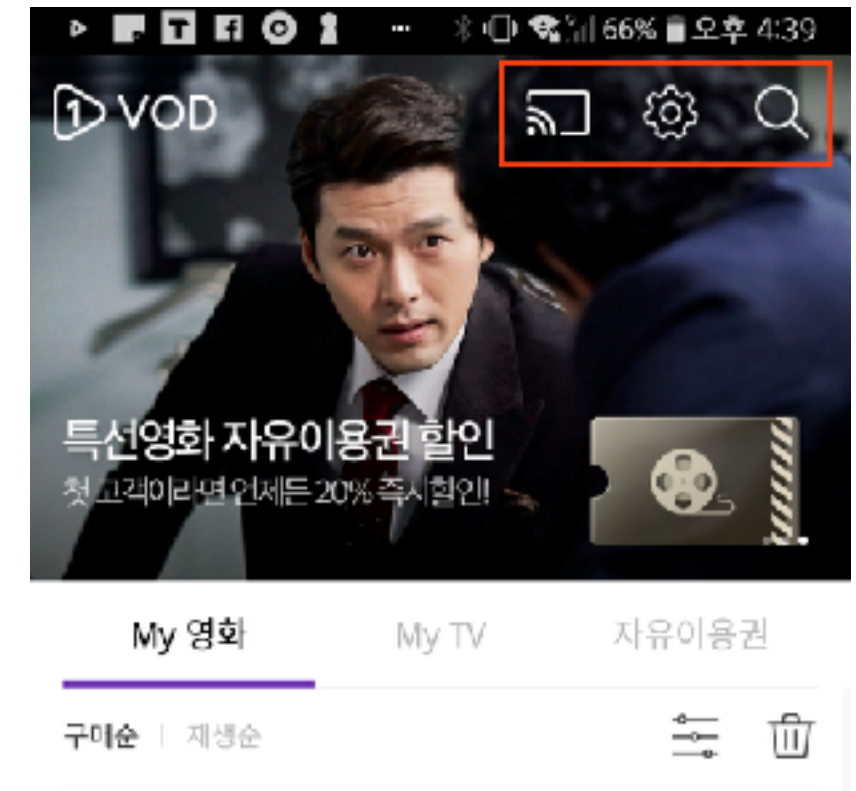
해결방안 :



# 1주차 L. 실전 안드로이드 레이아웃 구성 과제

“ 매니저 K씨 : 과장님 , 상단에 메뉴 버튼 크기가 너무 작아서 그런지 클릭이 잘 안되요 . 클릭 영역을 늘려주세요 ”

해결방안 :



# 1주차 L. 실전 안드로이드 레이아웃 구성 과제

“ 매니저 K씨 : 과장님 이 리스트 뷰 위 , 오른쪽 하단에 플로팅 버튼을 띄우고 싶어요”

해결방안 :



# 1주차 L. 실전 안드로이드 레이아웃 구성 과제

“ 매니저 K씨 : 과장님 어제 만들어 주신 플로팅 버튼 , 리스트 스크롤에 따라 위아래로 움직이게 할 수 있을까요 ?”

해결방안 :



# 〈1주차 과제〉 실전 안드로이드 레이아웃 구성 과제

“로그인 페이지를 수정해 봅시다”

공통 요건 (1) :

1주차 예제로 만든 로그인 화면에서 , 버튼 3개중 위에서 부터 2개를

아이디 패스워드를 입력받는 인풋박스로 교체

공통 요건 (2) :

1주차 예제에서 만든 로그인 화면에서 , 버튼 3개중 마지막 버튼 위치에

버튼 2개를 수평으로 배치 , 각각의 버튼은 “로그인” “회원가입” 버튼 배치





# 〈1주차 과제〉 실전 안드로이드 레이아웃 구성 과제

“ 회원가입 페이지를 만들어 봅시다 ”

공통 요건 (1) :

아이디 , 닉네임 , 패스워드 , 패스워드 확인 입력을 받는 Input 을 포함

공통 요건 (2) :

하단에 입력완료 , 취소 버튼을 포함 시킬것

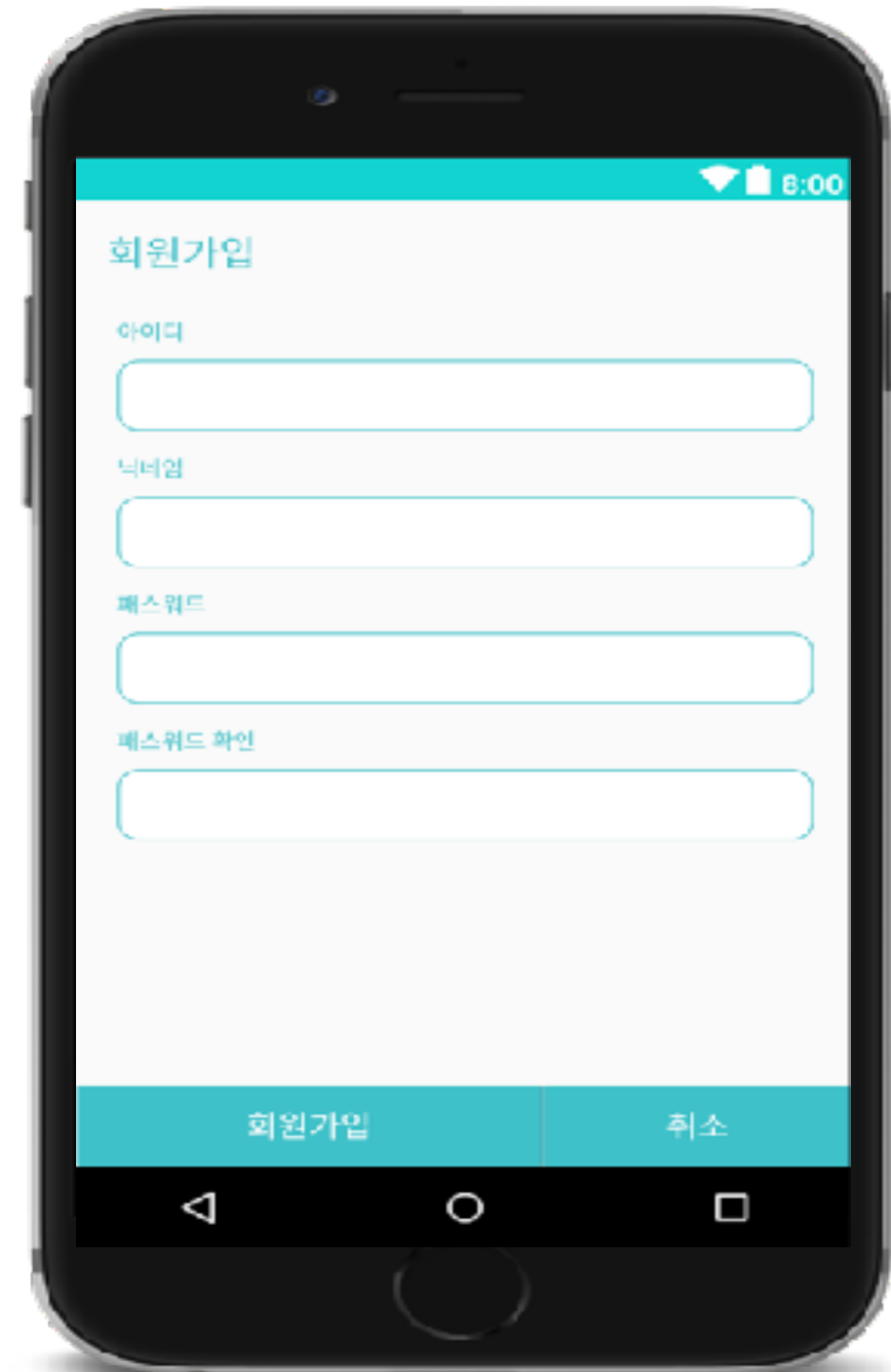
공통 요건 (3) :

입력 받은 두개의 “패스워드 값” 이 서로 일치 할 경우에만 동작 하도록 할것

공통 요건 (4) :

4개의 입력값이 공백인지 아닌지 여부를 확인 후 , 가입 하도록 할 것

아이디는 영어만 입력 가능 하도록 할것



## 2주차 A. 버튼에 동작 입히기 (이론)

“ 빈껍데기 화면 (XML 파일) 에서 버튼을 자바 파일로 가져와서 동작을 입힐겁니다 ”

1. findViewById 를 이용하여 , xml 에서 뷰를 가져 옵니다

```
Button sampleButton = findViewById(R.id. XML 에서의 뷰 아이디
```

2. 가져온 뷰에 setOnClickListener 를 이용하여 동작 지정 합니다

```
sampleButton.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        sampleButton 을 눌렀을때 수행할 동작을 지정 !!
    }
});
```

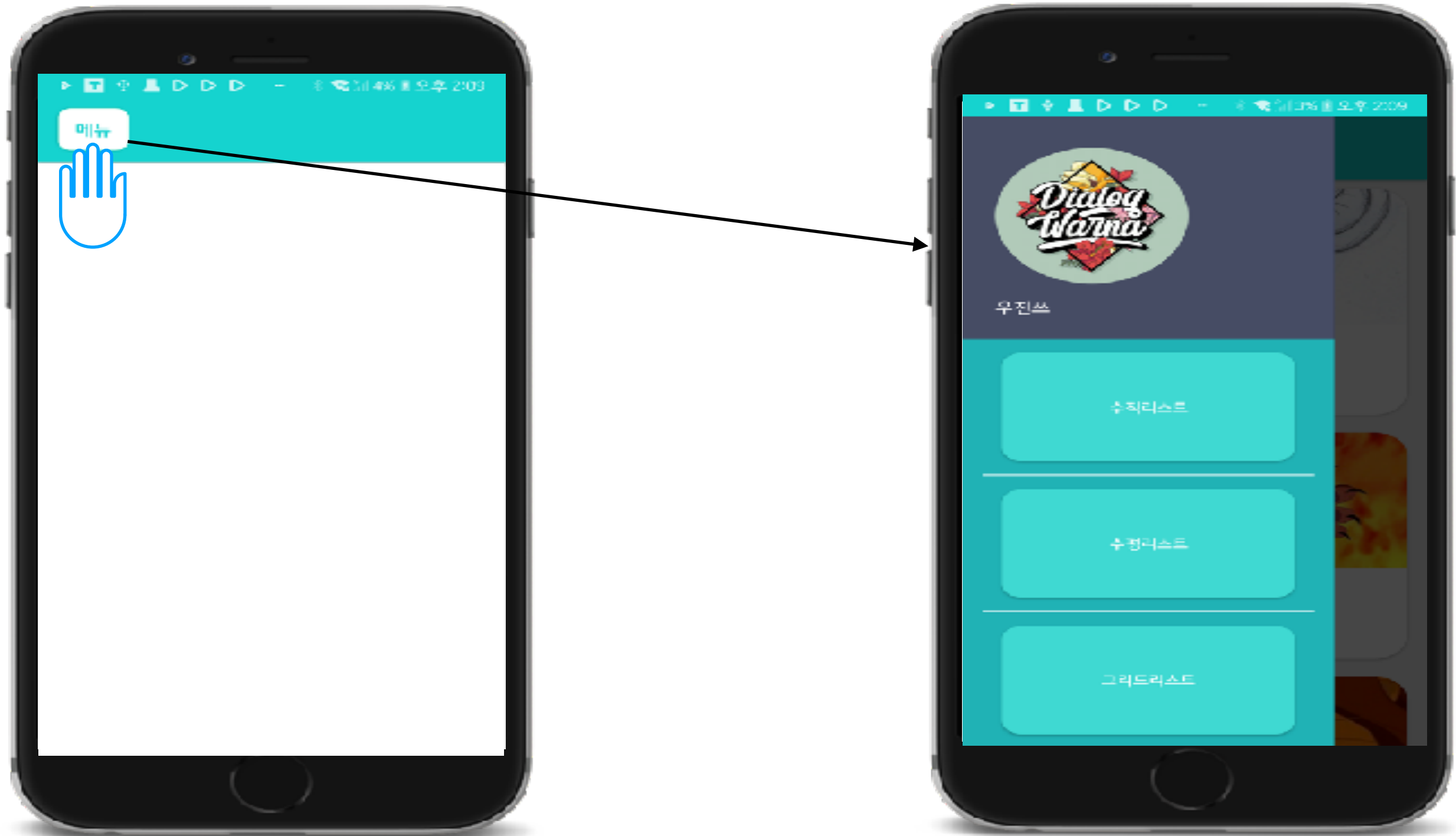
(참고) 다른 화면으로 이동하는 코딩

```
Intent intent = new Intent( 현재 머물고 있는 화면 .this, 이동하고자 하는 화면 .class);
startActivity(intent);
```

## 2주차 B. 버튼에 동작 입히기 (첫번째 예제)



## 2주차 C. 버튼에 동작 입히기 (두번째 예제)



## 3주차 A. 리스트 뷰 , 기획

“ 리스트를 기획 해봅시다 , 어떤 리스트를 만들고 싶나요 ? “

“ 리스트 의 아이템뷰 마다 표현해야할 정보 객체를 먼저 설계 합니다 “

“ 설계가 끝난 정보 객체를 , 아이템뷰 에서 어떻게 표현 할것인지 설계합니다 “



### 3주차 B. 심플리스트뷰 를 이용하여 메인화면에 리스트 붙이기 (예제)

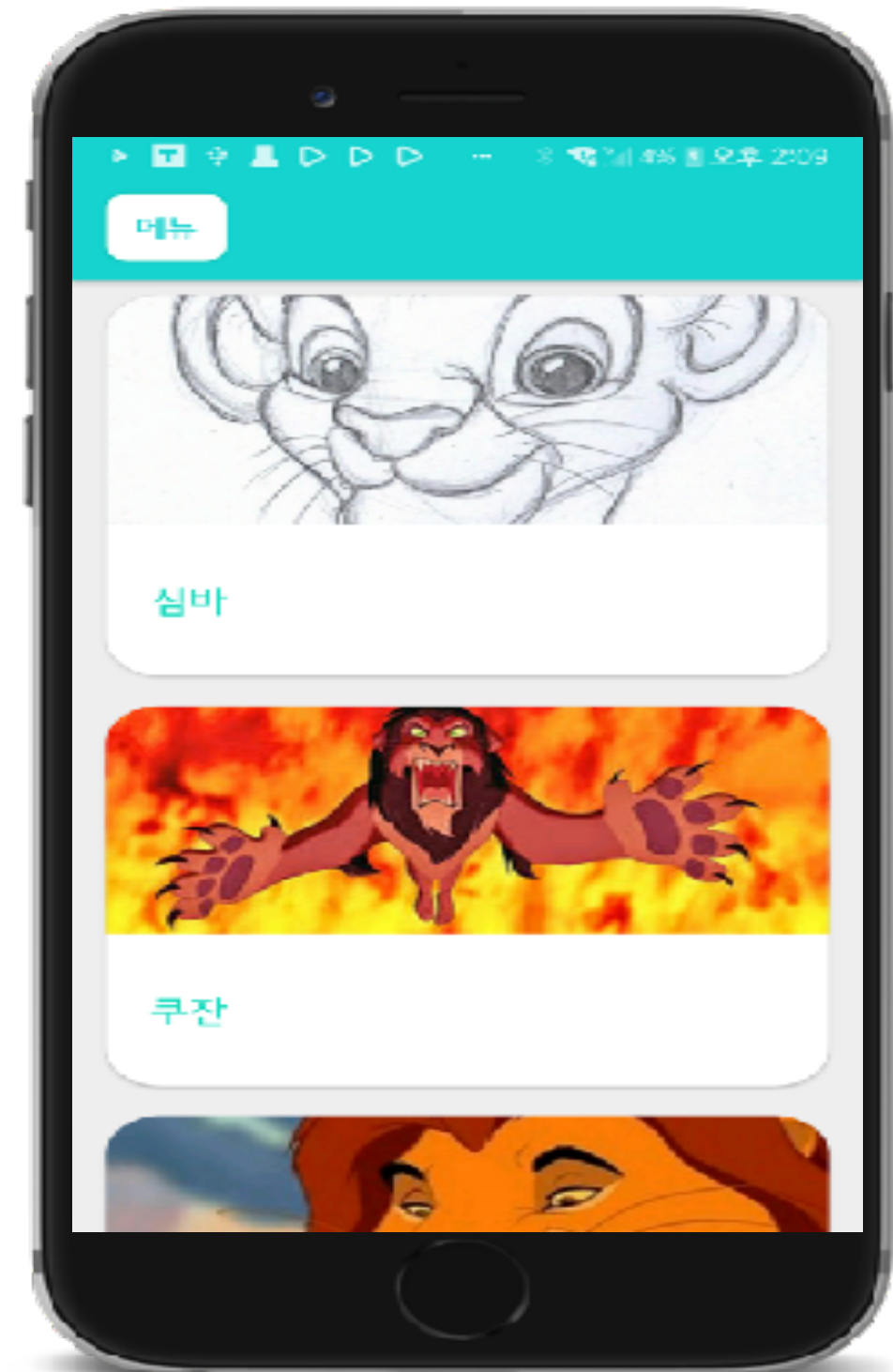
“ 서로 다른 아이템뷰를 가지고 있는 리스트뷰를 만들어 봅시다”

공통 요건 (1) :

서로 다른 유형의 아이템뷰 2개를 가지고 있는 리스트뷰 이어야 합니다

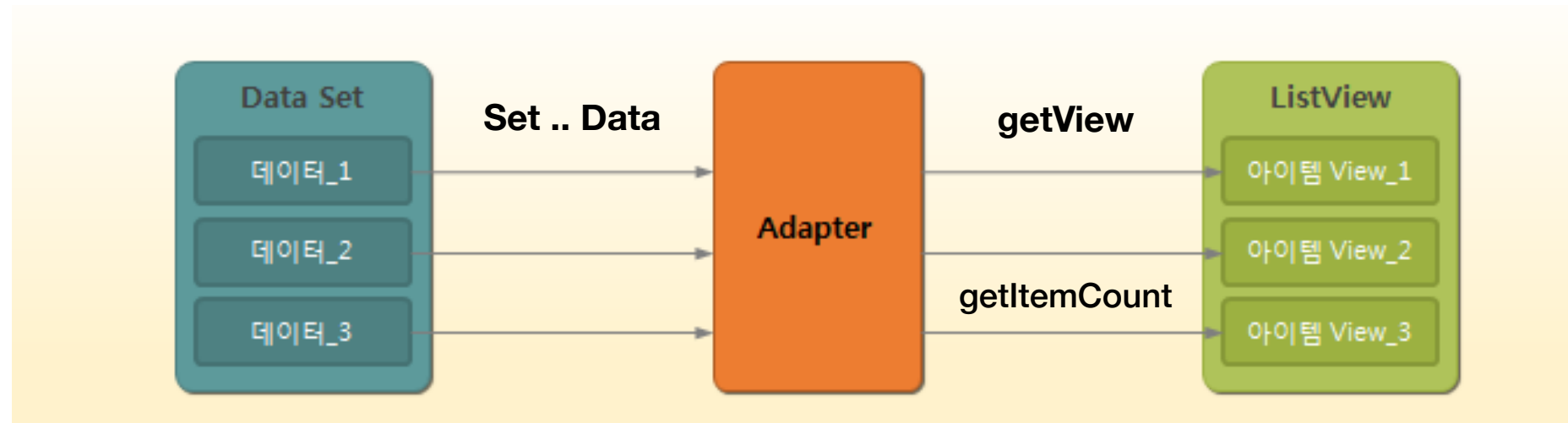
공통 요건 (2) :

각각의 아이템뷰에는 이미지뷰가 하나씩 있어야 합니다



## 3주차 B. 리스트 뷰 , 설계

“ 리스트뷰 를 만드는 과정을 한눈에 볼수 있는 구조도 입니다 “



“ 위 그림에 맞춰서 , 내가 만들고자 하는 리스트뷰 구조도를 그려 봅시다”

A large dashed rectangular box intended for drawing the ListView structure diagram.

# 3주차 C. 리스트 뷰 , 그래서 어떻게 만드는 겁니까 ?

## 1. “ 화면 레이아웃 xml 안에 리스트뷰 를 배치 합니다 ”

= 내가 노출 하고자 하는 액티비티 혹은 프래그먼트 레이아웃 xml 파일안에 리스트뷰를 배치

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</FrameLayout>
```

## 2. “ 화면 (액티비티 혹은 프래그먼트) 에서 내가 배치한 리스트뷰를 찾아옵니다 ”

= 액티비티 혹은 프래그먼트 에서 findViewById 로 리스트뷰를 찾아옵니다

```
private ListView    mListview;

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    setContentView(R.layout.list_activity);

    mListview = findViewById(R.id.listView);
}
```



# 3주차 C. 리스트 뷰 , 그래서 어떻게 만드는 겁니까 ?

## 3. “ 리스트로 표현할 아이템 (=객체) 리스트를 생성 ”

= 아이템 정보를 담은 객체 리스트를 ArrayList 혹은 LinkedList 로 만들어줍니다

```
ArrayList<여러분이 표현하고자 하는 객체 클래스> tArrayList = new ArrayList<>();  
  
for (int i = 0 ; i < 100 ; i ++)  
{  
    "객체 클래스" 객체 = new "객체 클래스"();  
  
    tArrayList.add(객체);  
}
```

## 4. “ 리스트 안에서 아이템뷰 를 생성할 어댑터를 생성 “

```
public class SampleAdapter extends BaseAdapter  
{  
    @Override  
    public int getCount()  
    {  
        return 0;  
    }  
  
    @Override  
    public Object getItem(int position)  
    {  
        return null;  
    }  
  
    @Override  
    public long getItemId(int position)  
    {  
        return 0;  
    }  
  
    @Override  
    public View getView(int position, View convertView, ViewGroup parent)  
    {  
        return null;  
    }  
}
```

# 3주차 C. 리스트 뷰 , 그래서 어떻게 만드는 겁니까 ?

## 5. “ 리스트뷰에 어댑터를 연결 ”

```
SampleAdapter sampleAdapter = new SampleAdapter();  
mListView.setAdapter(sampleAdapter);
```

## 6. “ 자 이제 , 모든걸 끝냈으니 어댑터에게 일을 하라고 알려준다 (notify) ”

= 어댑터에 담은 정보를 화면에 노출 하기 위해서 notifyDataSetChanged 를 호출 합니다

```
sampleAdapter.notifyDataSetChanged();
```



리스트 가 나오지 않아요

## 3주차 D. 리스트 뷰 핵심 , 어댑터 는 어떻게 완성 하나요?

“ 어댑터 안에서 가장 중요한 , getCount 와 getView 메소드를 완성 하고 , setItemList 를 추가 해야 합니다 ”

```
public class SampleAdapter extends BaseAdapter
{
    @Override
    public int getCount()
    {
        return 0; // (중요) 몇개의 아이템을 보여줄지 리스트뷰에게 알려줍니다
    }

    @Override
    public Object getItem(int position)
    {
        return null; // 몇번째에 있는 아이템을 반환 해준다
    }

    @Override
    public long getItemId(int position)
    {
        return 0; // 포지션에 맞는 아이템의 아이디값을 반환 해준다
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent)
    {
        return null; // (중요) 포지션에 맞는 뷰를 만들어서 반환 해준다
    }
}
```

# 3주차 D. 리스트 뷰 핵심 , 어댑터 는 어떻게 완성 하나요?

## 1. “어댑터 안에 아이템 리스트를 저장 할 수 있도록 해줍니다”

```
public class SampleAdapter extends BaseAdapter
{
```

```
private ArrayList<Member> itemArrayList;
```

```
public void setItemList(ArrayList<Member> list) {
    this.itemArrayList = list;
}
```



```
@Override
public int getCount()
{
    return 0; // (중요) 몇개의 아이템을 보여줄지 리스트뷰에게 알려줍니다
}
```

```
@Override
public Object getItem(int position)
{
    return null; // 몇번째에 있는 아이템을 반환 해준다
}
```

```
@Override
public long getItemId(int position)
{
    return 0; // 포지션에 맞는 아이템의 아이디값을 반환 해준다
}
```

```
@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    return null; // (중요) 포지션에 맞는 뷰를 만들어서 반환 해준다
}
}
```

## 3주차 D. 리스트 뷰 핵심 , 어댑터 는 어떻게 완성 하나요?

2. “ 리스트의 아이템을 몇 개를 표시할 것 인지 , **getItemCount** 메소드 로 명시합니다 ”

```
@Override
public int getCount()
{
    return (itemArrayList == null ? 0 : itemArrayList.size());
}
```

3. “ 리스트의 아이템을 뷰에 세팅 해주는 **getView** 메소드로 명시합니다”

```
@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    convertView = LayoutInflater.from(mContext).inflate(R.layout.list_item_xml , parent , false);

    // ... Data 를 뷰에 세팅 해준다

    ....

    return convertView;
}
```



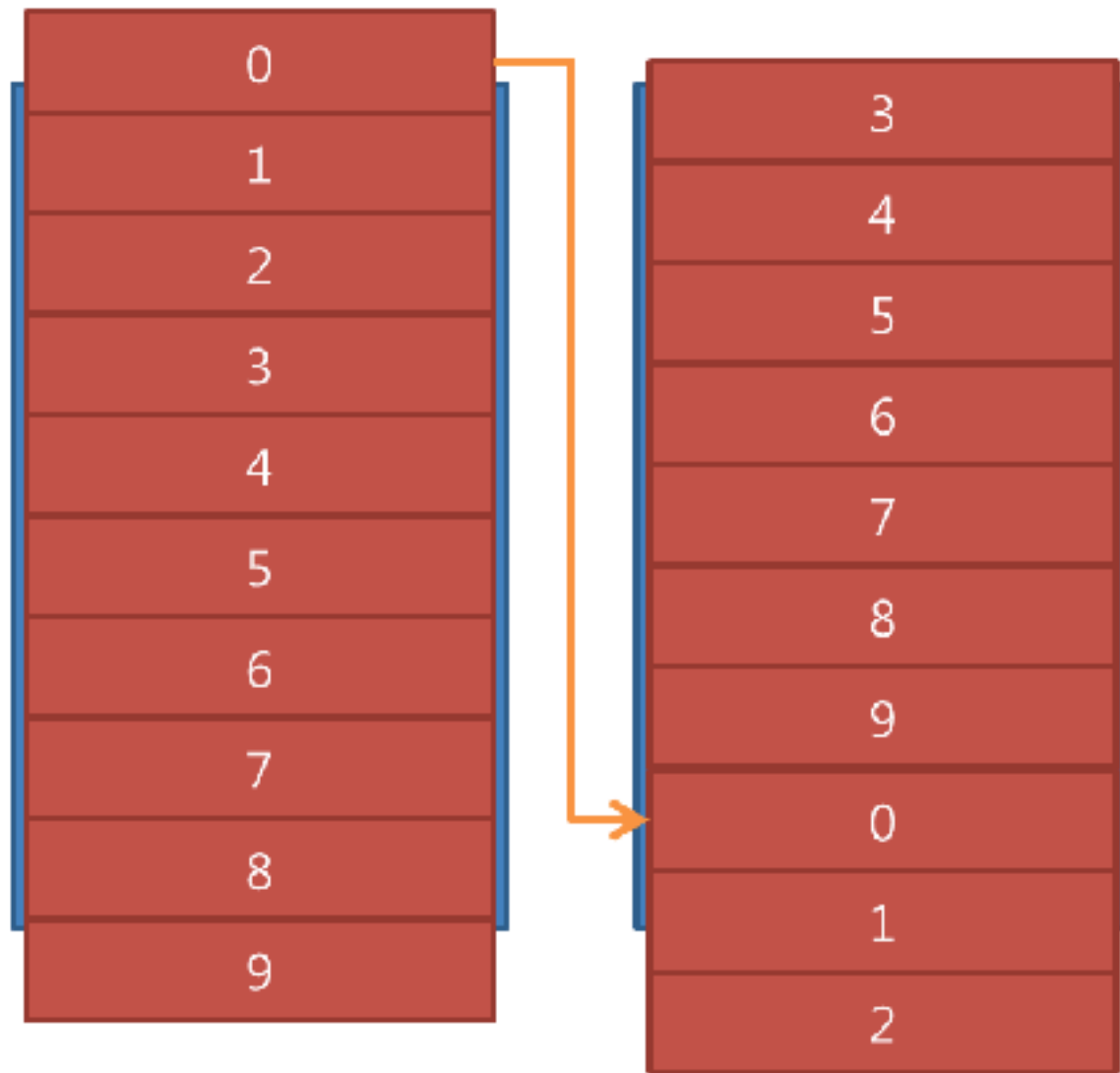
리스트 가 스크롤 하다보면 스크롤 이전

사라진 아이템이 다시 보여요

이거 왜이러는거죠 ?

### 3주차 E. 리스트 뷰 , 왜 아이템 뷰 를 재활용 하는건가요 ?

“ 리스트뷰는 그림 과 같이 스크롤 되서 사라진 뷰를 다시 재활용 합니다 ”



“ 위와 같이 뷰를 재활용 하지 않으면 , 단말기 메모리 가 부족하여 앱이 멈추거나 종료 됩니다 ”

“ 이 때문에 , 리스트 뷰 성능 향상을 위해 뷰 홀더 를 사용 해야 합니다 ”



## 3주차 F. 리스트 뷰 , 뷰홀더 이걸 뭐고 왜 쓰는거죠 ?

“ 리스트뷰가 처음 화면에 나올때는 . 아무것도 하지 않았기 때문에 아이템뷰를 생성 합니다 ”

“ getView 를 통해 , 생성한 아이템뷰를 반환 해줍니다 . 그럼 이제 이 아이템뷰가 화면에 노출 됩니다 ”

“ 스크롤 해서 올리다 보면 이제 , 처음 생성한 아이템 뷰는 화면에서 사라집니다 ”

“ 동시에 , 새로운 아이템뷰가 화면에 나올려고 대기하고 있죠 ”

“ 이때 안드로이드는 화면에서 사라졌던 뷰를 우리에게 재활용 해서 쓰라고 , 다시 돌려줍니다 ”

“ 그럼 우리는 이제 , 재활용 된 뷰를 쓸 수 있게 됩니다 ”

“ 어찌됐든 뷰를 리스트뷰 아이템 순서에 맞게 , 데이터를 뷰에 다시 세팅 해줘야겠죠 ? ”

“ 그럼 이전에 했던것처럼 , 뷰 에서 findViewById 로 텍스트뷰 나 버튼뷰를 찾아와야 할겁니다 ”

“ 여기서 이 findViewById 는 사실 , 안드로이드 입장에선 진상 메소드 입니다 . 부하가 많이 들거든요 ”

“ 이 findViewById 메소드를 다시 호출 하지 않기 위해서 우리는 뷰홀더를 씁니다 ”

## 3주차 G. 리스트 뷰 , 뷰홀더 그럼 어떻게 만들어야 하나요?

“ 뷰홀더 클래스를 만들어 줍니다 . 이 클래스는 static 이어야 합니다 “

```
static class ViewHolder
{
```

“ 그리고 이 뷰홀더 클래스 안에 , 우리가 필요한 각종 뷰를 선언 해 줍시다 ”

```
static class ViewHolder
{
    static public Button mButton0;
    static public Button mButton1;
    ...

    static public TextView mTextView0;
    static public TextView mTextView1;
    ...
}
```

“ 우리는 이 뷰홀더를 다시 재사용해서 쓸거니까 , 처음 생성시에는 넣어줘야겠죠 ? ”

```
if (convertView == null)
{
    convertView = LayoutInflater.from(mContext).inflate(R.layout.list_item_xml , parent , false);

    viewHolder = new ViewHolder(convertView);
    viewHolder.mButton0 = convertView.findViewById(R.id.button0);
    viewHolder.mButton1 = convertView.findViewById(R.id.button1);

    convertView.setTag(viewHolder);
}
```

“ 재사용 된 뷰를 안드로이드로 부터 다시 받으면 이제는 findViewById 가 아닌 , 뷰 홀더 를 이용 합니다 ”

```
ViewHolder viewHolder;

if (convertView != null)
{
    viewHolder = (ViewHolder) convertView.getTag();
}
```

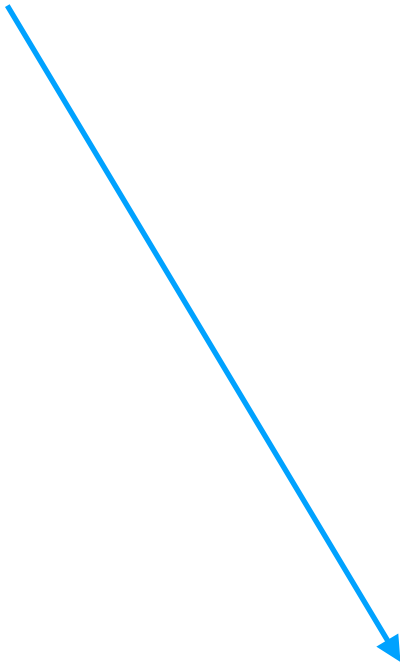
## 3주차 G. 리스트 뷰 , 뷰홀더 그림 어떻게 만들어야 하나요?

```
@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    convertView = LayoutInflater.from(mContext).inflate(R.layout.list_item_xml , parent , false);

    // ... Data 를 뷰에 세팅 해준다

    ....

    return convertView;
}
```



```
@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    ViewHolder viewHolder;

    if (convertView == null)
    {
        convertView = LayoutInflater.from(mContext).inflate(R.layout.list_item_xml , parent , false);

        viewHolder = new ViewHolder(convertView);
        viewHolder.mButton0 = convertView.findViewById(R.id.button0);
        viewHolder.mButton1 = convertView.findViewById(R.id.button1);

        convertView.setTag(viewHolder);
    }
    else
    {
        viewHolder = (ViewHolder) convertView.getTag();
    }

    // ... Data 를 뷰에 세팅 해준다

    ....

    return convertView;
}
```

# <여기서 잠깐> 제네릭 이란게 있어요

“ 제네릭 **Generic** 이란 , 클래스 혹은 메소드 내부에서 사용할 데이터 타입을 외부에서 지정하는 기법 ”

“ 제네릭을 사용하기 전 메소드 ”

```
public View find(int viewResId)
{
    return mChildViewArray.get(viewResId);
}
```

“ 제네릭을 사용 한 후 메소드 ”

```
public <V extends View> V find(int viewResId)
{
    View view = mChildViewArray.get(viewResId);

    if (view == null)
    {
        view = itemView.findViewById(viewResId);
        mChildViewArray.put(viewResId , view);
    }

    return (V) view;
}
```

“ 제네릭을 사용하면 , 타입에 관계 없이 자유자재로 객체를 저장하고 쓸 수 있습니다 ”



갑자기 왜 제네릭을 설정하시는거죠 ?

### 3주차 H. 리스트 뷰 , 제네릭을 이용하여 컴포짓 뷰홀더 를 사용합시다

“ 우리가 일반 뷰홀더를 쓸때는 이러 했습니다 ”

```
static class ViewHolder {  
    static public Button mButton0;  
    static public Button mButton1;  
    ...  
    static public TextView mTextView0;  
    static public TextView mTextView1;  
    ...  
}
```

“ 이러한 뷰홀더는 , 리스트뷰 마다 어댑터마다 새로 생성해서 써야 한다는 번거로움이 있습니다 ”

```
public class CompositeViewHolder {  
    private SparseArray<View> mChildViewArray;  
    @SuppressWarnings("unchecked")  
    public <V extends View> V find(int viewResId)  
    {  
        View view = mChildViewArray.get(viewResId);  
        if (view == null)  
        {  
            view = LayoutInflater.findViewById(viewResId);  
            mChildViewArray.put(viewResId, view);  
        }  
        return (V) view;  
    }  
}
```

“

이미 저장된 뷰 리스트에 내가 찾고자 하는 뷰가 없다면

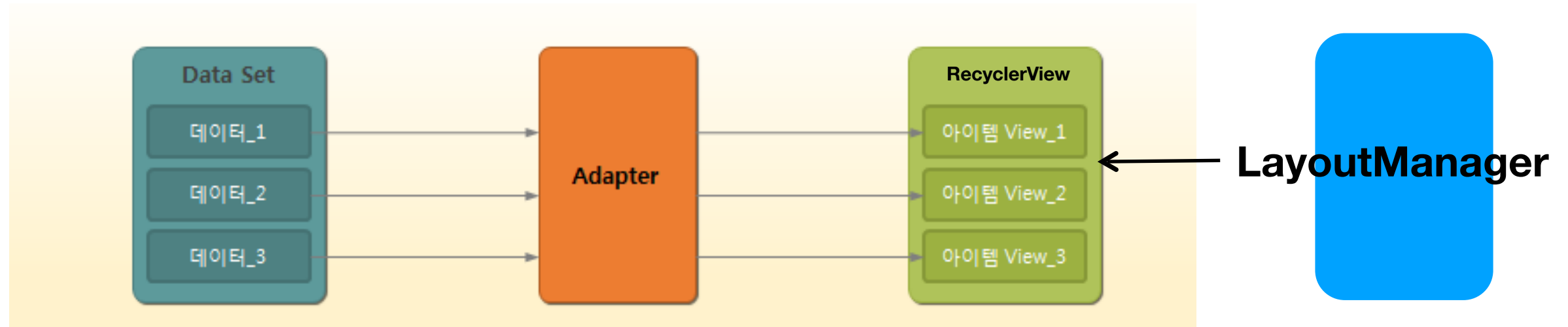
findViewById 로 뷰를 찾아 옵니다

이후 이렇게 한번 찾아온 뷰는  
간편하게 다시 불러서 가져오기 위해 리스트에 저장합니다

”

### 3주차 1. 구글에서 이제 부터는 리싸이클러뷰 쓰라고 하네요

“ 이전에 봤던 리스트뷰에 비해 , **LayoutManager** 라는 녀석이 추가 되었어요 “



“ 리스트뷰에게 이제까지 뷰홀더는 필수 요소가 아니었지만 , 이는 성능상 큰 차이가 있습니다”

“ 이에 따라 , 구글은 뷰홀더를 필수 요소로 사용 할 수 있게 , **RecyclerView** 제공 하게 되었습니다 ”

“ 뿐만 아니라 , 각종 리스트 뷰 애니메이션 및 라이브러리로 존재하던 **API** 들을 기본으로 제공할 수 있게 해주었습니다 ”

# 3주차 J. 리스트뷰 와 리사이클러뷰 , 무엇이 다른거죠 ?

1. 뷰 홀더 가 이제는 필수가 되었다

2. 아이템 뷰를 포지션에 따라 , 각각 다른 뷰 를 생성 할 수 있는 기능이 제공 되었다

“ 안드로이드 에서 RecyclerView.ItemDecoration 를 제공 ”

2. 아이템 뷰에 대한 애니메이션 동작을 처리 할 수 있는 객체 가 제공 되었다

“ 안드로이드 에서 RecyclerView.ItemAnimator 를 제공 ”

3. 같은 어댑터 클래스 를 사용 하더라도 , 아이템뷰 처리를 다르게 할 수 있도록 기능이 제공 되었다

“ 포지션에 따라 아이템 타입을 별도로 지정 , 서로 다른 뷰를 create 하고 다른 데이터를 bind 할 수 있게 되었다 ”

4. 어댑터 업데이트 알림 동작을 세분화 할 수 있게 해주었다

“ 아이템 삽입시 = notifyItemInserted() / 아이템 제거시 = notifyItemRemoved() / 아이템 교환시 notifyItemChanged ”



# 3주차 K. 리사이클러뷰 , 그래서 어떻게 만드는 겁니까 ? (연결)

1. “ 리사이클러 뷰를 사용하기 위해서 , **Recycler** 뷰 **API** 를 구글에서 가져옵니다 ”

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'

    implementation 'com.android.support:cardview-v7:26.1.0'
    implementation 'com.android.support:recyclerview-v7:26.1.0'
}
```

2. “ 내가 노출 하고자 하는 화면 **XML** 파일 안에 리사이클러뷰 를 배치합니다”

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recycler_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ddccdd"
/>
```

3. “ 내가 노출 하고자 하는 화면 (액티비티 혹은 프래그먼트) 에서 리사이클러뷰 를 찾아오고 , 레리아웃 매니저를 연결해줍니다 ”

```
LinearLayoutManager layoutManager = new LinearLayoutManager(this);
```

```
RecyclerView recyclerView = findViewById(R.id.recycler_view);
recyclerView.setHasFixedSize(true);
recyclerView.setLayoutManager(layoutManager);
```

# 3주차 K. 리사이클러뷰 어댑터 , 그래서 어떻게 만드는 겁니까 ? -1

1. “ 우리가 만들었던 컴포짓 뷰홀더를 <> 안에 넣고 **RecyclerView.Adapter** 를 상속받아 어댑터를 만듭니다 ”

```
public class NewAdapter extends RecyclerView.Adapter<CompositeViewHolder>
{
```

2. “ 리스트 몇번째 아이템 마다 , 내가 노출 하고자 하는 아이템뷰가 다르다면 **getItemViewType** 메소드를 명시 합니다 ”

```
@Override
public int getItemViewType(int position)
{
    return A (or B ...)
}
```

3. “ 아이템뷰의 타입에 따라 , **XML** 파일을 로드해서 레이아웃 파일 을 반환 해줍니다 ”

```
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
{
    switch(viewType)
    {
        case A :
            View v = inflater.inflate(parent.getContext()).inflate(R.layout.A, parent, false);
            break;

        case B :
            View v = inflater.inflate(parent.getContext()).inflate(R.layout.B, parent, false);
            break;

        ...
    }

    return new CompositeViewHolder(v);
}
```

## 3주차 K. 리싸이클러뷰 어댑터 , 그래서 어떻게 만드는 겁니까 ? -2

4. “이제 아이템 타입과 리스트 포지션에 맞는 데이터를 가져와서 아이템뷰를 꾸며줍니다 ”

```
@Override
public void onBindViewHolder(CompositeViewHolder holder, int position)
{
    Data data = getItem(position);

    int itemType = getItemViewType(position);

    switch(itemType)
    {
        case A :

            ImageView imageView = holder.find(R.id.imageA);
            imageView.setBackground(data.resource);

            ...

            break;

        case B :

            TextView textView = holder.find(R.id.textA);
            textView.setText(data.name);

            ...

            break;
    }
}
```

5. “리스트뷰 와 마찬가지로 몇 개의 아이템을 리스트로 표시 할것 인지 , **getItemCount** 메소드로 명시 합니다 ”

```
@Override
public int getItemCount()
{
    ...
    return foodInfoArrayList.size();
}
```

### 3주차 심화1. 객체지향 프로그래밍 디자인 패턴

“ 어떠한 새로운 요건이 주어지더라도 유연하게 대처 가능해진다 ”

“ 지저분한 개발을 막을 수 있다 “

“ 기존 코드는 유지하면서 새로운 기능을 덧붙일수 있다 ”

# 3주차 심화1. 객체지향 프로그래밍 디자인 패턴

## 1. 전략 패턴 ( **Strategy Pattern** )

“ 상황에 따라 유동적으로 동작 할 수 있도록 해주는 알고리즘 객체를 만드는 패턴 ”

## 2. 데코레이터 패턴 ( **Decorator Pattern** )

“ 기존에 쓰던 객체 자체는 수정하지 않으면서 상황에 맞는 기능을 추가 한 새로운 객체를 만드는 패턴 ”

## 3. 컴포짓 패턴 ( **Composite Pattern** )

“ 하나의 객체가 가질수 있는 멤버변수를 제한하지 않는 패턴 ”

## 4. 템플릿 메소드 패턴 ( **Template Method Pattern** )

“ 부모 클래스 가 해야할 일을 자식 클래스 에게 미루는 패턴 ”

## 5. 옵저버 패턴 ( **Observer Pattern** )

“ 변화가 생기는 객체에 대해 절대적으로 의존 하도록 하는 패턴 ”

# 3주차 심화1. 리스트뷰 에서 리사이클러뷰 로의 유연한 전환

이슈 상황 “ 기존 프로젝트에서 잘 사용하고 있는 리스트뷰 어댑터를 리사이클러뷰 어댑터로 바꿔야 하는 상황 “

“ 기존 리스트뷰 어댑터의 핵심 알고리즘은 getView 안에 있습니다 ”

```
@Override  
public View getView(int position, View convertView, ViewGroup parent)  
{  
    ....  
}
```

“ 기존 어댑터가 문제 없이 동작 하는 데도 , 리사이클러뷰 어댑터로 바꿔야 한다면 . 새로 만들어야 할까요 ? ”

“ 이때 , 기존 어댑터 기능을 그대로 가지고 있는 리사이클러뷰 어댑터를 새로 만든다면

그 수많은 시간과 노력 , 뿐만 아니라 이후에 따라올 사이드 이펙트 이슈 감당 할 수 있을까요 ? ”

“ 우리는 , 기존 리스트뷰 어댑터를 감싸는 데코레이터 어댑터를 만들 으로서 이슈를 쉽고 빠르게 해결 할수 있습니다 ”

# 3주차 심화1. 리스트뷰 에서 리사이클러뷰 로의 유연한 전환

## “기존 리스트뷰 어댑터를 감싸는 데코레이터 어댑터 ”

```
/**
 * .. 기존 (v4.0.4 이하) 기존 Listview Adapter > RecyclerView Adapter 로 migration 하기 위한 레고 bricks 이펙터
 *
 * @author wonzi
 *
 * @param <T>
 */
public class MainAdapterDecorator<T> extends ArrayAdapter<T> extends RecyclerView.Adapter<AdapterViewHolder>

    ...

    private T mRootAdapter;

    ...

    @Override
    public int getItemCount() {
        return (mRootAdapter.getItemCount() == 0 ? HEADERS_ITEMCOUNT_DEFAULT : mRootAdapter.getItemCount() + 4);
    }

    @Override
    public AdapterViewHolder or RecyclerViewHolder(ViewGroup parent, int viewType) {
        ....

        View view = mRootAdapter.onCreateItemViewLayout(parent);

        AdapterViewHolder recyclerViewHolder = new AdapterViewHolder(view);
        recyclerViewHolder.setParentViewGroup(parent);

        return recyclerViewHolder;
    }
}
```

```
@Override
public void onBindViewHolder(AdapterViewHolder holder, int position)
{
    ....

    ViewGroup parent = holder.getParentViewGroup();
    View convertView = holder.getItemView();

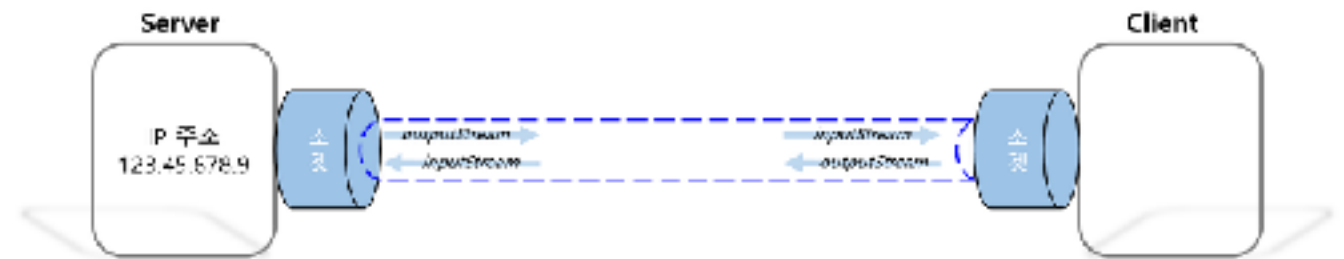
    mRootAdapter.getView(position, convertView, parent);
}
```

## 4주차 A. 네트워크 통신 에 대한 이해

네트워크 통신은 크게 “비연결 지향 통신” 과 “연결 지향 통신” 으로 구분 할 수 있습니다

### 1. 연결 지향 통신

“ 통신을 하지 않는 동안에도 , 통신을 위한 채널을 계속 유지 하는 통신 “



### 2. 비연결 지향 통신

“ 요청이 있을때만 연결 하고 , 요청에 대한 응답이 끝나면 연결을 해제 하는 통신 “





# 4주차 B. HTTP 통신 구현하기

Http URL Connection 을 구현은 크게 아래와 같이 구분 되어 집니다



## 1. HTTP 메소드 방식을 지정

“ HTTP 통신은 크게 POST GET DELETE PUT 방식으로 나누어져 있습니다 “

## 2. HTTP 요청시 , 서버에서 필요한 인자값 전달

“ 우리가 원하는 응답을 얻으려면 , 서버에서 요구하는 요청값이 있습니다 “

“ HTTP 메소드 방식에 따라 , 요청값을 어떻게 전달할지 방법도 나누어집니다 “

## 3. HTTP 응답을 받으면 , 유효한 값인지 검사

“ 서버로 부터 응답값을 받으면 , 먼저 HTTP 응답값으로 올바른 값인지 검사 부터 해야 합니다 “

## 4. HTTP 응답 유형에 따라 , 우리가 원하는 데이터 형으로 변환 (=파싱)

“ 서버에서 클라이언트로 전달해주는 데이터 방식은 유형화 되어 있고 , 우리는 이를 따라야 합니다 “

“ 이 유형화 된 방식에는 보통 JSON 과 XML 방식이 있습니다 ”

# 4주차 C. HTTP 통신으로 전달 받은 데이터 파싱 하기

서버로 부터 넘겨 받는 응답값은 유형화 되어 있는데 , 대부분 JSON 방식과 XML 방식을 따르고 있습니다

XML	JSON
<pre>&lt;empinfo&gt;   &lt;employees&gt;     &lt;employee&gt;       &lt;name&gt;James Kirk&lt;/name&gt;       &lt;age&gt;40&lt;/age&gt;     &lt;/employee&gt;     &lt;employee&gt;       &lt;name&gt;Jean-Luc Picard&lt;/name&gt;       &lt;age&gt;45&lt;/age&gt;     &lt;/employee&gt;     &lt;employee&gt;       &lt;name&gt;Wesley Crusher&lt;/name&gt;       &lt;age&gt;27&lt;/age&gt;     &lt;/employee&gt;   &lt;/employees&gt; &lt;/empinfo&gt;</pre>	<pre>{ "empinfo" :   [     {       "employees" : [         {           "name" : "James Kirk",           "age" : 40,         },         {           "name" : "Jean-Luc Picard",           "age" : 45,         },         {           "name" : "Wesley Crusher",           "age" : 27,         }       ]     }   ] }</pre>

## 1. JSON 데이터 파싱 방법


“ 넘겨받은 응답값을 그대로 이용해 , JSON 객체를 만들거나 GSON 라이브러리를 이용 합니다 “

## 2. XML 데이터 파싱 방법

“ 도큐먼트 안에 노드리스트로 인식후 , 각각의 노드를 하나씩 읽으면서 파싱한다“

## 4주차 D. JSON 으로 데이터 파싱 하기

```
1
{
  "contacts": [
    {
      "id": "c200",
      "name": "Ravi Tanada",
      "email": "ravi@gmail.com",
      "address": "xx-xx-xxxx, x - street, x - country",
      "gender": "male",
      "phone": {
        "mobile": "+91 0000000000",
        "home": "00 000000",
        "office": "00 000000"
      }
    },
    {
      "id": "c201",
      "name": "Johnny Depp",
      "email": "johnny_depp@gmail.com",
      "address": "xx-xx-xxxx, x - street, x - country",
      "gender": "male",
      "phone": {
        "mobile": "+91 0000000000",
        "home": "00 000000",
        "office": "00 000000"
      }
    },
    .
    .
    .
  ]
}
```



```
JSONObject jsonObj = new JSONObject(jsonStr);

// Getting JSON Array node
JSONArray contacts = jsonObj.getJSONArray("contacts");

// looping through All Contacts
for (int i = 0; i < contacts.length(); i++) {
    JSONObject c = contacts.getJSONObject(i);

    String id = c.getString("id");
    String name = c.getString("name");
    String email = c.getString("email");
    String address = c.getString("address");
    String gender = c.getString("gender");

    // Phone node is JSON Object
    JSONObject phone = c.getJSONObject("phone");
    String mobile = phone.getString("mobile");
    String home = phone.getString("home");
    String office = phone.getString("office");
}
```

## 4주차 E. JSON 으로 데이터 파싱 할때 , GSON 이용하기

```
1
{
  "contacts": [
    {
      "id": "c200",
      "name": "Ravi Tanada",
      "email": "ravi@gmail.com",
      "address": "xx-xx-xxxx, x - street, x - country",
      "gender": "male",
      "phone": {
        "mobile": "+91 0000000000",
        "home": "00 000000",
        "office": "00 000000"
      }
    },
    {
      "id": "c201",
      "name": "Johnny Depp",
      "email": "johnny_depp@gmail.com",
      "address": "xx-xx-xxxx, x - street, x - country",
      "gender": "male",
      "phone": {
        "mobile": "+91 0000000000",
        "home": "00 000000",
        "office": "00 000000"
      }
    },
    .
    .
    .
  ]
}
```

// Gradle 에 Gson 라이브러리 추가  
// implementation 'com.google.code.gson:gson:2.8.5'

Gson gson = new GsonBuilder().create();

...  
// (1) JsonObject > 객체  
Result result = gson.fromJson(jsonString, Result.class);  
  
// (2) JsonArray > 객체 리스트로  
Type listType = new TypeToken<List<Contact>>().getType();  
List<Contact> posts = gson.fromJson(jsonString, listType);

# 4주차 F. XML 으로 데이터 파싱 하기

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
  <item>
    <id>1</id>
    <name>Margherita</name>
    <cost>155</cost>
    <description>Single cheese topping</description>
  </item>
  <item>
    <id>2</id>
    <name>Double Cheese Margherita</name>
    <cost>225</cost>
    <description>Loaded with Extra Cheese</description>
  </item>
  <item>
    <id>3</id>
    <name>Fresh Veggie</name>
    <cost>110</cost>
    <description>Onion and Crisp capsicum</description>
  </item>
  <item>
    <id>4</id>
    <name>Peppy Paneer</name>
    <cost>155</cost>
    <description>Paneer, Crisp capsicum and Red pepper</description>
  </item>
  <item>
    <id>5</id>
    <name>Mexican Green Wave</name>
    <cost>445</cost>
    <description>Onion, Crisp capsicum, Tomato with mexican herb</description>
  </item>
</menu>
```

```
Document doc = parser.getDomElement(xml); // getting DOM element

NodeList nl = doc.getElementsByTagName(KEY_ITEM);

// looping through all item nodes <item>
for (int i = 0; i < nl.getLength(); i++) {
    String name = parser.getValue(e, KEY_NAME); // name child value
    String cost = parser.getValue(e, KEY_COST); // cost child value
    String description = parser.getValue(e, KEY_DESC); // description child val
}
```

```
public Document getDomElement(String xml) {
    Document doc = null;
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    try {

        DocumentBuilder db = dbf.newDocumentBuilder();

        InputStream is = new InputStream();
        is.setCharacterStream(new StringReader(xml));
        doc = db.parse(is);
    }
}
```

```
public String getValue(Element item, String str)
{
    NodeList n = item.getElementsByTagName(str);
    return this.getElementValue(n.item(0));
}

public final String getElementValue( Node elem )
{
    Node child;
    if (elem != null)
    {
        if (elem.hasChildNodes())
        {
            for (child = elem.getFirstChild(); child != null; child = child.getNextSibling())
            {
                if (child.getNodeType() == Node.TEXT_NODE)
                {
                    return child.getNodeValue();
                }
            }
        }
    }
    return "";
}
```

# 4주차 G. 비동기 통신 구현 실습

안드로이드 에서 비동기 통신 을 구현하는데는 아래와 같은 3가지 방법이 있습니다

1. AsyncTask 를 이용
2. Thread 를 생성후 응답 값을 핸들러를 통해 메인스레드로 전달하는 방법
3. Retrofit 이라는 외부 라이브러리 이용

# 4주차 F 비동기 통신 구현체 를 실제 적용

오늘 배운 비동기 통신 을 이전에 만들었던 화면에 적용 해 봅시다

1. 로그인 화면에 적용
2. 회원가입 화면에 적용
3. 리스트 데이터 조회 후 , 리스트 뷰 에 노출

# 4주차 보충. FireBase Database 실시간 데이터베이스 이용하기

데이터를 서버에 밀어넣기 (POST)

```
DatabaseReference ref = FirebaseDatabase.getInstance().getReference();
```

```
FirebaseDBRequest<내용물객체의 클래스> request = new FirebaseDBRequest<>();
```

```
request.setReference(ref); // 받는사람 주소 기입 (예: 경기도 광명시 철산동)
```

```
request.addTargetChild("....."); // 받는 사람 상세 주소 (예: 도덕파크 000동 0000호)
```

```
request.setPostInstance(편지 내용물 객체); // 편지에 내용 넣기
```

```
request.setTargetClass(내용물객체의 클래스.class); // 편지 내용물 클래스 지정
```

```
request.setType(RequestType.POST); // 편지 타입 지정 (내용을 디비에 삽입 하는 것이니, post)
```

```
request.setResponseListener(회신 받고자 하는 위치);
```

```
RequestHandler.getInstance().handle(request); // 편지 배달원에게 편지 전달
```

.....

```
@Override
```

```
public void onResponseListen(BaseResponse response)
```

```
{
```

```
    if (response instanceof FirebaseDBResponse)
```

```
    {
```

```
        FirebaseDBResponse<내용물객체의 클래스> responseData = (FirebaseDBResponse<내용물객체의 클래스>) response;
```

```
        내용물 = responseData.getFirstResult();
```

```
        if (responseData.isResponseSuccess())
```

```
        {
```

```
        }
```

```
    }
```

```
}
```



# 4주차 보충. FireBase Database 실시간 데이터베이스 이용하기

데이터를 서버에서 가져오기 (GET)

```
DatabaseReference ref = FirebaseDatabase.getInstance().getReference();

FirebaseDBRequest<내용물객체의 클래스> request = new FirebaseDBRequest<>();

request.setReference(ref); // 받는사람 주소 기입 (예: 경기도 광명시 철산동)
request.addTargetChild("....."); // 받는 사람 상세 주소 (예: 도덕파크 000동 0000호)
request.setKey(".....") // 가져오고자 하는 데이터의 조건 필드
request.setValue(".....") // 가져오고자 하는 데이터 조건 필드의 값
request.setTargetClass(내용물객체의 클래스.class); // 편지 내용물 클래스 지정
request.setType(RequestType.GET); // 편지 타입 지정 (내용을 디비에 삽입 하는 것이니, post)
request.setResponseListener(회신 받고자 하는 위치);

RequestHandler.getInstance().handle(request); // 편지 배달원에게 편지 전달
```

.....

```
@Override
public void onResponseListen(BaseResponse response)
{
    if (response instanceof FirebaseDBResponse)
    {
        FirebaseDBResponse<내용물객체의 클래스> responseData = (FirebaseDBResponse<내용물객체의 클래스>) response;

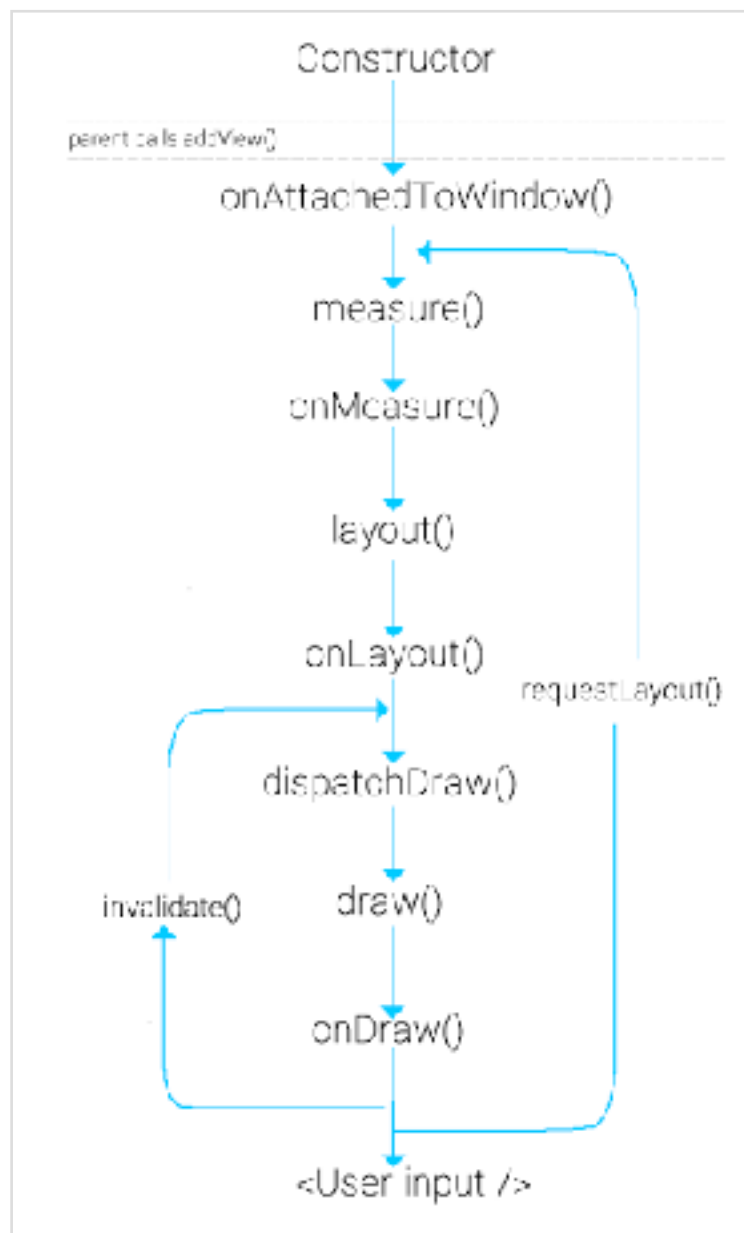
        내용물 = responseData.getFirstResult();

        if (responseData.isResponseSuccess())
        {
        }
    }
}
}
```

## 5주차 A. 뷰란 무엇인가

“뷰란, 화면에 보이는 레이아웃의 단위입니다”

“이러한 뷰가 생성되고 화면에 노출 되는 순서는 아래와 같습니다”



“뷰가 윈도우가 붙으면 호출 됩니다”

“뷰의 너비, 높이가 결정 될때 호출 됩니다”

“뷰의 위치 (Left, Top, Right, Bottom) 가 결정 될때 호출 됩니다”

“뷰를 윈도우에 그릴때 호출 됩니다 (= 뷰의 노출시기)”

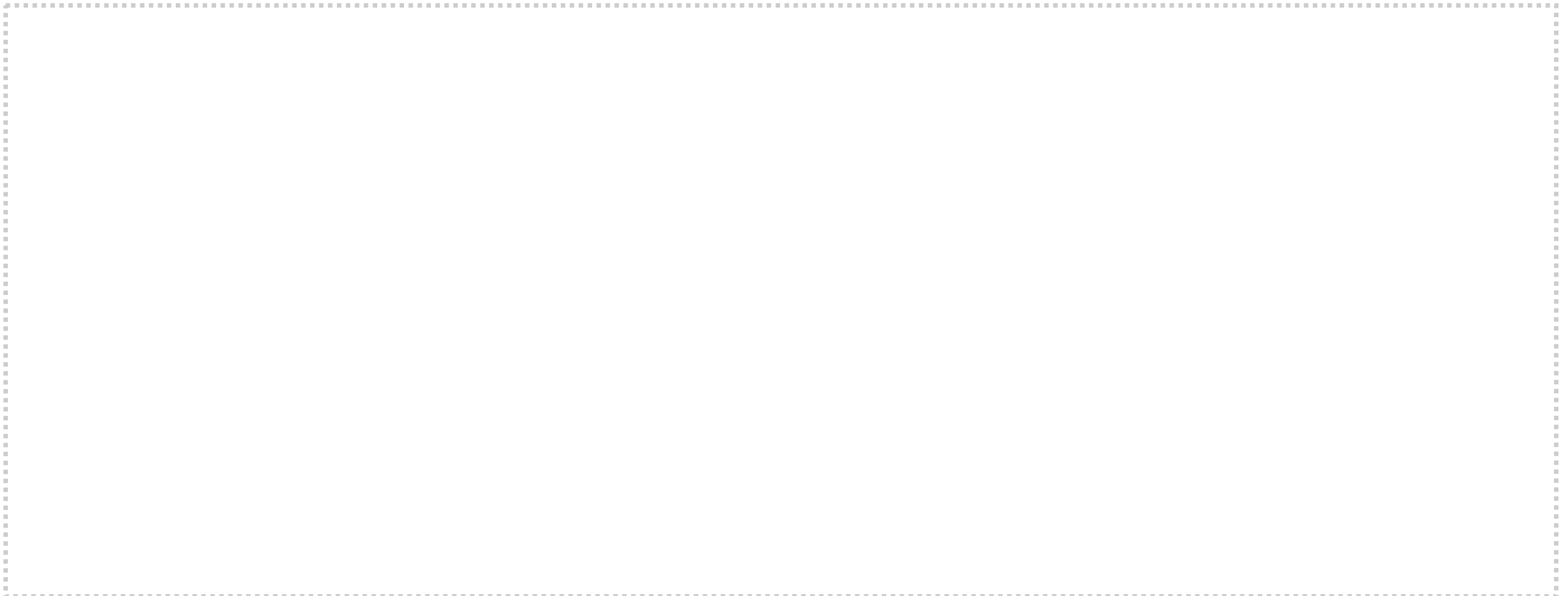
## 5주차 B. 커스텀 뷰 , 왜 만들어야 하나요

“ 구글에서 만들어준 안드로이드 라는 녀석이 , 우리가 필요한 모든 뷰를 다 담고 제공해준다면 ”

“ 그렇다면 , 우리는 이번 주차에 사실 배울 내용이 없습니다 ”

“ 하지만 . 화면마다 우리가 필요한 뷰 중에서는 우리가 스스로 만들어야 하는 뷰가 생기기 마련 입니다 ”

“ 일단 , 우리가 스스로 만들어야 할 뷰는 무엇이 있을까요 , 기획 부터 합시다 ”



## 5주차 C. 커스텀 뷰 , 어떻게 만들어야 하나요

“ 커스텀뷰 = 내가 스스로 만든 뷰 , 제작 의 핵심은 **onDraw** 메소드 다루기 입니다 ”

“ 커스텀뷰 제작 , 예제로 감을 잡아 봅시다 ”

```
public class CustomView extends View
{
    public CustomView(Context context)
    {
        super(context);
        init(context, null);
    }

    public CustomView(Context context, @Nullable AttributeSet attrs)
    {
        super(context, attrs);
        init(context, attrs);
    }

    public CustomView(Context context, @Nullable AttributeSet attrs, int defStyleAttr)
    {
        super(context, attrs, defStyleAttr);
        init(context, attrs);
    }

    private void init(Context context, @Nullable AttributeSet attrs)
    {
    }

    @Override
    protected void onDraw(Canvas canvas)
    {
        super.onDraw(canvas);
    }
}
```

# 5주차 C. 커스텀 뷰 , 어떻게 만들어야 하나요

“ 커스텀뷰 = 내가 스스로 만든 뷰 , 제작 의 핵심은 **onDraw** 메소드 다루기 입니다 ”

“ 커스텀뷰 제작 , 예제로 감을 잡아 봅시다 ”

## 1. onDraw() 메소드 안에서 선 , 사각형 , 원 을 그려봅시다

```
@Override
protected void onDraw(Canvas canvas)
{
    super.onDraw(canvas);

    Paint paint=new Paint();
    paint.setAntiAlias(true); // 확대해도 선이 울퉁불퉁하지 않고 매끈하게 설정
    paint.setColor(Color.BLUE); // 선색 글자색 설정

    // 1.선그리기
    canvas.drawLine(10, 10, 300, 10, paint); // canvas.drawLine(시작x좌표,시작y좌표,끝x좌표,끝y좌표,색 모양 설정된 paint)
    paint.setColor(Color.RED); // 선색
    paint.setStrokeWidth(5); // 선두께 설정 (기본은 1 )
    canvas.drawLine(10, 30, 300, 30, paint); // canvas.drawLine(시작x좌표,시작y좌표,끝x좌표,끝y좌표,색 모양 설정된 paint)

    // 2.사각형그리기
    paint.setColor(Color.MAGENTA); // 선색면색
    paint.setStrokeWidth(0); // 선두께
    paint.setStyle(Paint.Style.FILL); // 면이 채워진 사각형 설정
    Rect rect1=new Rect(10,50,10+100,50+100); // (사각형 시작x좌표,사각형시작y좌표,사각형 끝x좌표, 사각형 끝 y좌표)
    canvas.drawRect(rect1, paint); // 화면세 사각형 그리기.(사각형 좌표가 저장된 객체,사각형 모양설정 paint 객체 )

    // 3.선만 있는 사각형 그리기
    paint.setStyle(Paint.Style.STROKE); // 선만있는 사각형
    Rect rect2=new Rect(130,50,130+100,50+100); // (시작X,시작Y,끝X,끝Y)
    canvas.drawRect(rect2, paint); // 사각형그리기 메서드(사각형좌표,모양설정Paint)

    // 4.모서리 둥근 사각형 그리기
    RectF rect3=new RectF(250,50,250+100,50+100); // (시작X,시작Y,끝X,끝Y)
    canvas.drawRoundRect(rect3,20,20,paint); // 모서리둥근사각형메서드 그리기 ( 사각형 좌표,가로둥글기,세로둥글기,paint ) ;

    // 5.원그리기
    canvas.drawCircle(60, 220, 50, paint); // 원그리기 메서드(시작X, 시작Y, 반지름 ,paint);
}
```

# 5주차 C. 커스텀 뷰 , 어떻게 만들어야 하나요

“ 커스텀뷰 = 내가 스스로 만든 뷰 , 제작 의 핵심은 onDraw 메소드 다루기 입니다 ”

“ 커스텀뷰 제작 , 예제로 감을 잡아 봅시다 ”

2. onDraw() 에서 그린 도형 위에 , 텍스트를 그려 봅시다

```
private Paint mPaint;

@Override
protected void onDraw(Canvas canvas)
{
    super.onDraw(canvas);

    if (mPaint == null)
    {
        mPaint = new Paint();
    }

    mPaint.setAntiAlias(true);
    mPaint.setColor(Color.LTGRAY);
    mPaint.setStyle(Paint.Style.FILL);

    Rect rect2 = new Rect(10,10,10+200,10+100);

    canvas.drawRect(rect2, mPaint);

    mPaint.setTextSize(26);
    mPaint.setTextScaleX(1.0f);

    canvas.drawText("텍스트그리기 샘플", 10, (10 + 100), mPaint);
}
```

# 5주차 C. 커스텀 뷰 , 어떻게 만들어야 하나요

“ 커스텀뷰 = 내가 스스로 만든 뷰 , 제작 의 핵심은 onDraw 메소드 다루기 입니다 ”

“ 커스텀뷰 제작 , 예제로 감을 잡아 봅시다 ”

3. Xml 에서 입력 받은 값을 onDraw() 에 적용 해 봅시다

“로그인 페이지에 사용할 커스텀 입력뷰를 만들어 봅시다”

```
<declare-styleable name="sampleAttribute">

    <attr name="sampleAttribute1" format="string" />
    <attr name="sampleAttribute2" format="integer" />
    <attr name="sampleAttribute3" format="float" />
    <attr name="sampleAttribute4" format="fraction" />
    <attr name="sampleAttribute5" format="color" />
    <attr name="sampleAttribute6" format="boolean" />
    <attr name="sampleAttribute7" format="dimension" />
    <attr name="sampleAttribute8" format="flags" />
    <attr name="sampleAttribute9" format="reference" />

    <attr name="sampleAttribute10" format="enum">
        <enum name="email" value="0"/>
        <enum name="password" value="1"/>
        <enum name="shortMessage" value="2"/>
        <enum name="longMessage" value="3"/>
    </attr>
</declare-styleable>
```

**5주차 H. 우리가 필요한 커스텀 뷰 , 이제 만들어 봅시다**





# 5주차 심화. 커스텀 뷰에 애니메이션 동작을 주는 방법

안드로이드 에서 쓰이는 애니메이션 구현 방법은 크게 아래와 같이 구분 할 수 있습니다

## 1. Handler 를 이용한 딜레이 동작

“ 딜레이 시마다 , 뷰에 속성값을 조절 함으로서 애니메이션 동작 “

## 2. AnimationDrawable 이용

“ 디자이너가 전달 해주는 애니메이션 여러장을 딜레이로 노출 혹은 lottie 라이브러리 이용 “

## 3. ObjectAnimator 를 이용

“ NineOldAnimation 의 오브젝트 애니메이터를 이용하여 , 애니메이션 동작 조합 “

## 4. PropertyAnimator 이용

“ 밀리세컨즈 시간 마다 어떻게 동작 할것인지 , 개발자가 명시 하는 애니메이션 동작 “