오픈소스전문프로젝트

Report #01

제출일 2020 3 29

2 조

2016038034 김혁

2018037017 박소연

2018037045 유지원

2017037008 한현지

목차

과제 1

조별로 스마트폰에 내제된 센서들에 대해 조사하고 이들의 복합적 정보를 기반으로 제공할 수 있는 새로운 서비스에 대해 고찰 보고서를 작성하시오

스마트폰 센서 조사

2018037017 박소연

스마트폰에 내장된 센서는 하드웨어 기반 센서와 소프트웨어 기반 센서가 있다. 하드웨어 기반 센서는 디바이스에 내장된 물리적 구성 요소이며, 특정 환경 특성을 직접 측정하여 데이터를 얻는다. 소프트웨어 기반 센서는 하드웨어 기반 센서에서 데이터를 가져오며 가상 센서라 한다.

안드로이드 스마트폰에 존재하는 센서는 크게 세 분류로 나눌 수 있다.

-동작 센서: 동작 센서는 기기의 동작을 모니터링 하는 데 유용하다. X, Y, Z 세 축을 따라 가속력과 회전력을 측정한다. 가속 센서, 중력 센서, 자이로스코프 센서 등이 있다. 가속 센서와 자이로스코프 센서는 하드웨어 기반이다.

-위치 센서: 기기의 물리적 위치를 측정한다. 방향 센서를 포함한다. 지자기 센서와 근접 센서는 하드웨어에 기반한다.

-환경 센서: 환경 센서 4 개는 모두 하드웨어 기반이기 때문에 기기에 내장된 경우에만 사용할 수 있다. 주변의 기온, 압력, 조도, 습도와 같은 환경의 변화를 측정한다. 압력 센서, 온도 센서, 습도 센서, 광도 센서 등이 있다.

1) 조도 센서: 주변 빛의 밝기를 감지해서 밝기에 따라 동작을 조절한다. 스마트폰 화면의 밝기를 자동으로 조절하는 기능도 조도 센서를 이용한 것이다.

2) 가속 센서: X, Y, Z 좌표를 통해 가속도를 측정하고 스마트폰의 움직임 또는 움직이는 물체의 속도를 상세하게 감지한다. 출력 신호를 처리하여 진동이나 충격 등을 측정할 수도 있다.

3) GPS 센서: gps 위성을 통해 현재 위치와 시간을 측정할 수 있고 위치 검색 서비스도 가능하다. 주로 네비게이션/지도 앱에 이용된다.

4) 자이로스코프 센서: 가속 센서에 회전축이 추가된 센서이다. 스마트폰의 기울임이나 회전 등을 측정할 수 있다. 구글 카메라는 파노라마 촬영을 할 때 이 센서를 활용한다.

5) 근접 센서: 물리적 접촉 없이 물체의 존재 여부, 통과, 흐름 등을 감지한다. 사물이 어느 정도 거리에 있는지 감지할 수 있다. 통화 중 화면이 저절로 꺼지는 기능은 근접 센서를 이용한 것이다. 이를 통해 배터리 절약 및 불필요한 터치를 방지 할 수 있다.

6) RGB 센서: 광원의 RGB 밝기를 측정하는 센서이다. 주변 빛에 따라서 색온도가 변하기 때문에 최적화를 통해 색을 원본과 같게 맞춘다. 블루라이트를 낮출 때도 쓰인다.

7) 지문 인식 센서: 지문 굴곡에 따라 강약이 다른 전기 용량을 검출하고 패턴을 알아내는 정전식으로 동작한다. 지문과 접촉하면 인식이 되는 에어리어 방식과 지문센서를 훑는 스와이프 방식이 있다. 잠금화면 해제, 금융 결제 등 보안이 필요한 영역에 주로 활용된다.

8) 모션 센서(복합 센서): 모션 센서는 물체의 움직임과 위치를 인식하는 모든 센서를 일컫는 말이다. 단순한 동작 감지에서부터 표정을 인지하는 세밀한 범위까지 다룬다. 중력 센서, 자이로 센서, 가속 센서 등 모두 모션 센서에 속한다.

9) 지자기 센서: 지구의 자기장을 통해 방위각을 측정한다. X, Y, Z 축 세 방향의 출력 합으로 자기장 방향을 탐지한다. 대표적으로 지자기 센서가 활약하는 앱은 나침반 앱이다. gps 센서와 조합하여 위치기반 앱에 쓰이기도 한다.

10) 방향 센서: X, Y, Z 의 변화하는 회전각을 탐지한다. 안드로이드 2.2 에서 지원이 중단되었기 때문에 현재는 잘 쓰이지 않는다.
11) 온도/습도 센서: 단말기 주변의 온도와 습도를 나타낸다. 온습도 센서가 같이 있는 경우 이슬점과 절대 습도 또한 계산할 수 있다.

12) 압력 센서: 공기의 압력을 hPa,mbar 단위로 측정한다. 고도의 높낮이 인식이 가능하다.

13) 심장 박동 센서: 광학식 펄스를 통해 혈액의 흐름을 파악한다. 이를 분석해 심장 박동수를 알 수 있다.

14) 중력 센서: 중력의 방향과 강도를 나타내는 3D 벡터를 제공한다.

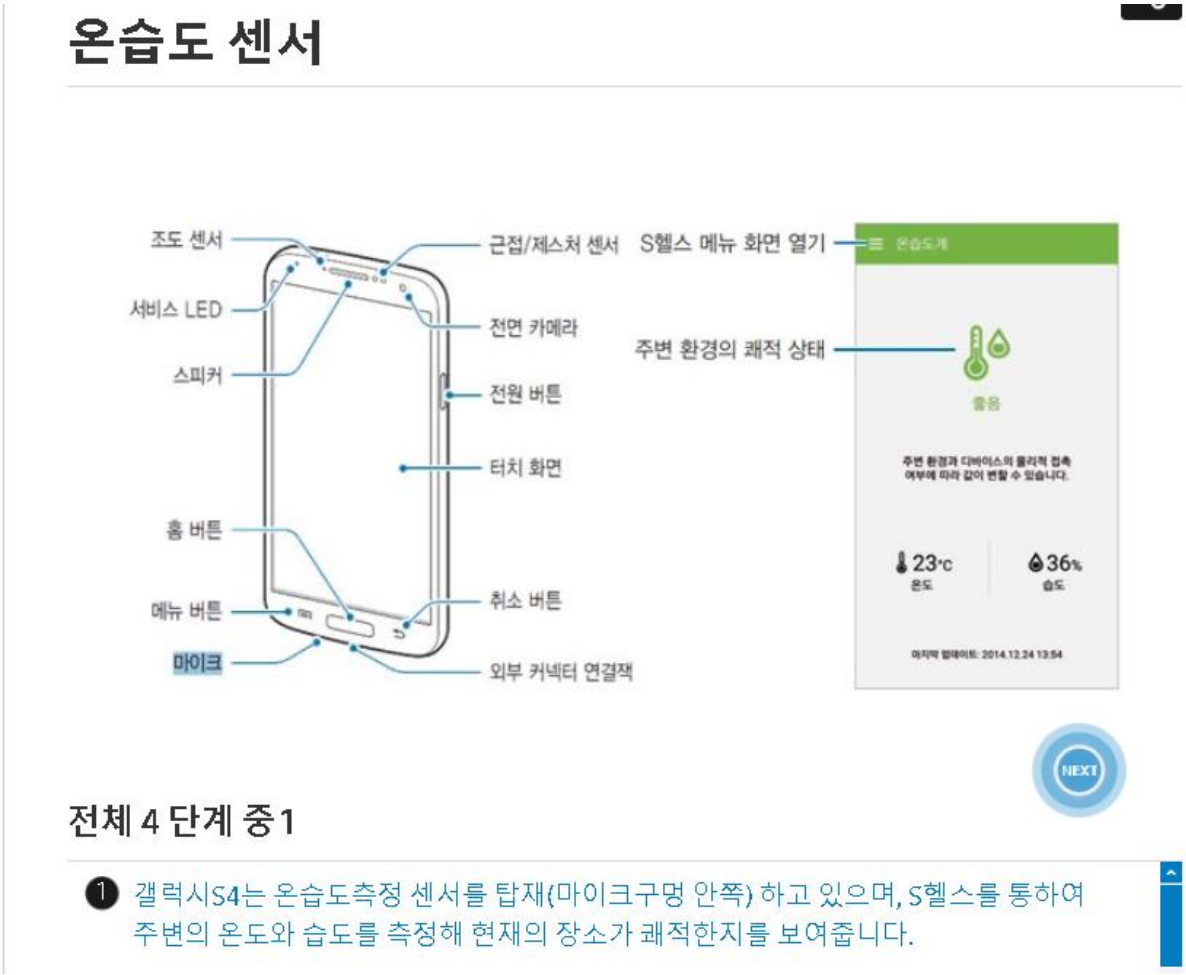
15) 홀 센서: 자기장의 세기에 따라 전압이 변하는 센서이다. 휴대폰의 플립이 닫히기 전에 화면이 저절로 꺼지는 이유는 홀 센서 때문이다.

16) 이미지 센서: 카메라 렌즈를 통해 들어온 빛을 디지털 신호로 변환한다. 이미지 센서가 클수록 고화질의 사진을 얻을 수 있다. 일반적으로 카메라 앱에 쓰인다. 이미지 센서를 두 개 사용하면 듀얼 이미지 센서가 된다. 이를 통해 사진의 심도를 조절하거나 줌 기능을 사용할 수 있다.

17) 터치 센서: 대부분 스마트폰에서 정전식 터치 방식을 사용한다. 전류가 흐르는 액정 유리에 손가락이 닿으면 정전기가 발생한다. 이 정전기를 감지해서 터치스크린을 원하는 대로 조작할 수 있다.

새로운 서비스에 대한 고찰

2018037045 유지원



1. 온습도 조절

사용자가 원하는 온습도를 설정해둔다.

갤럭시 스마트폰에 탑재된 온습도 측정 센서를 이용하여 주변의 온습도를 측정한다.

측정을 토대로 블루투스로 연결된 에어컨을 작동시켜 사용자가 설정한 온습도의 범위에서 넘었는지 체크 후 유지할 수 있도록 한다.

2. 블라인드

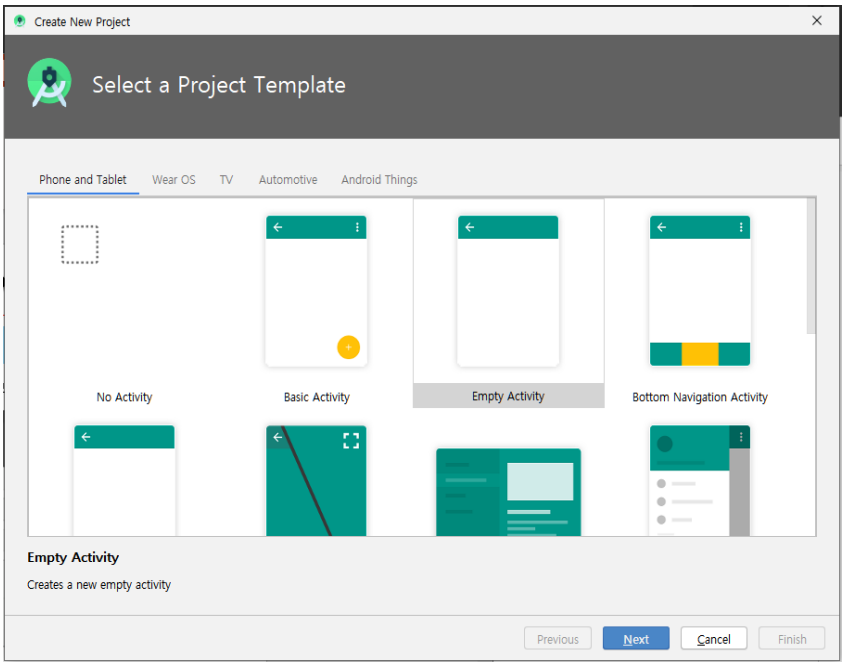
조도 센서를 이용하여 어떤 제스처를 취했을 때 조도 센서가 감지 후 어두우면 블라인드를 올리고 밝으면 블라인드를 내리는 등의 기능을 수행할 수 있도록 한다.

과제 2

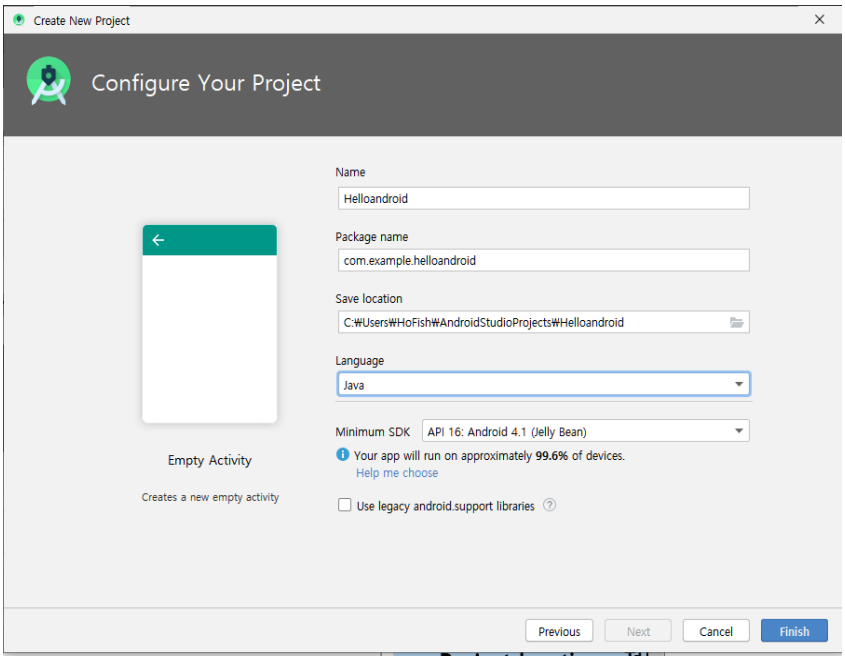
조별로 첨부된 주 교재 내용을 따라 “Hello Android”를 작성하고 코드와 주석, 상세 과정들을 캡처하고 단계별 설명하시오. 섹션 별 또는 보고서 상 담당 조원 이름을 함께 적으시오.

과제 2

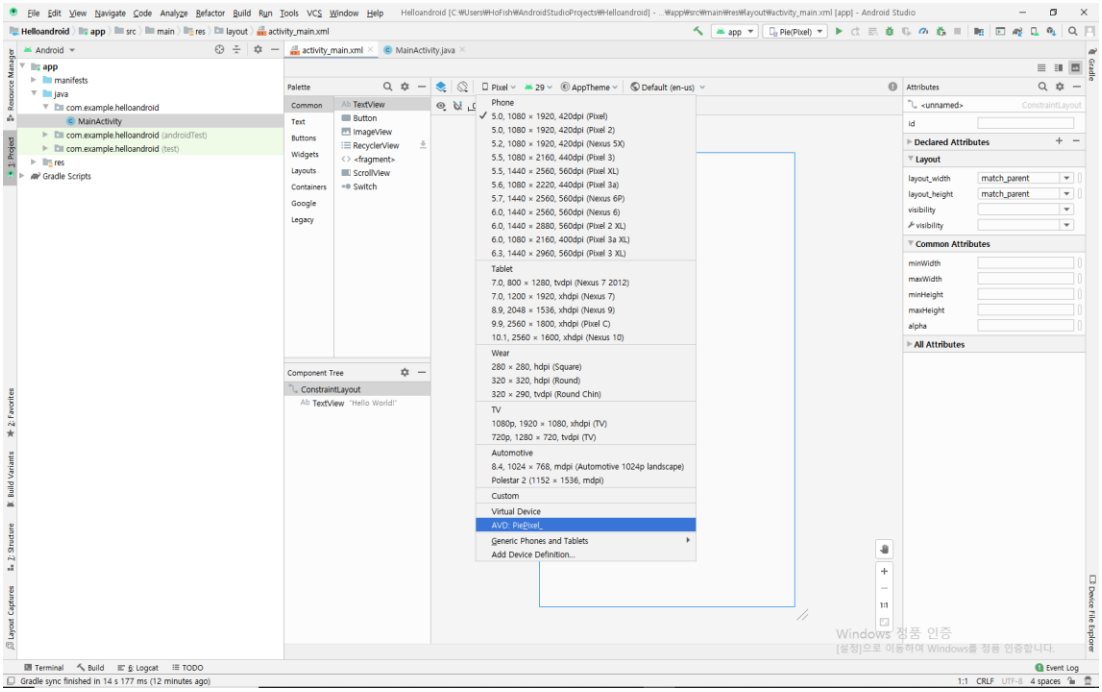
2016038034 김혁



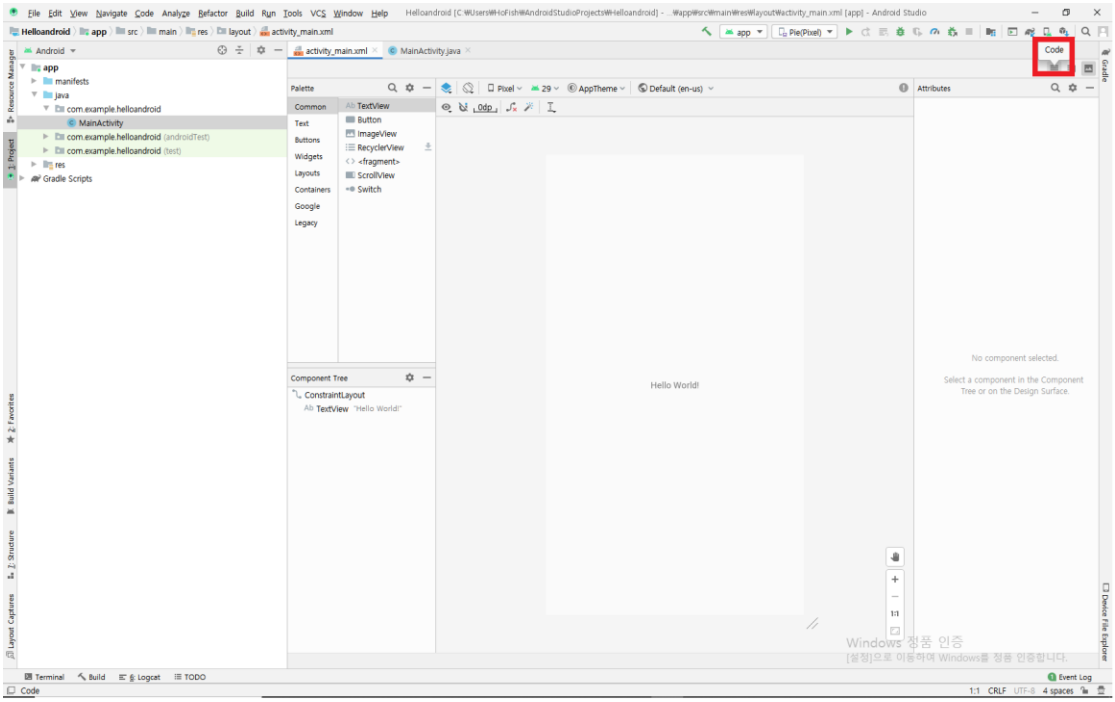
안드로이드 프로젝트를 새로 만들어줍니다 Empty Activity를 사용합니다.



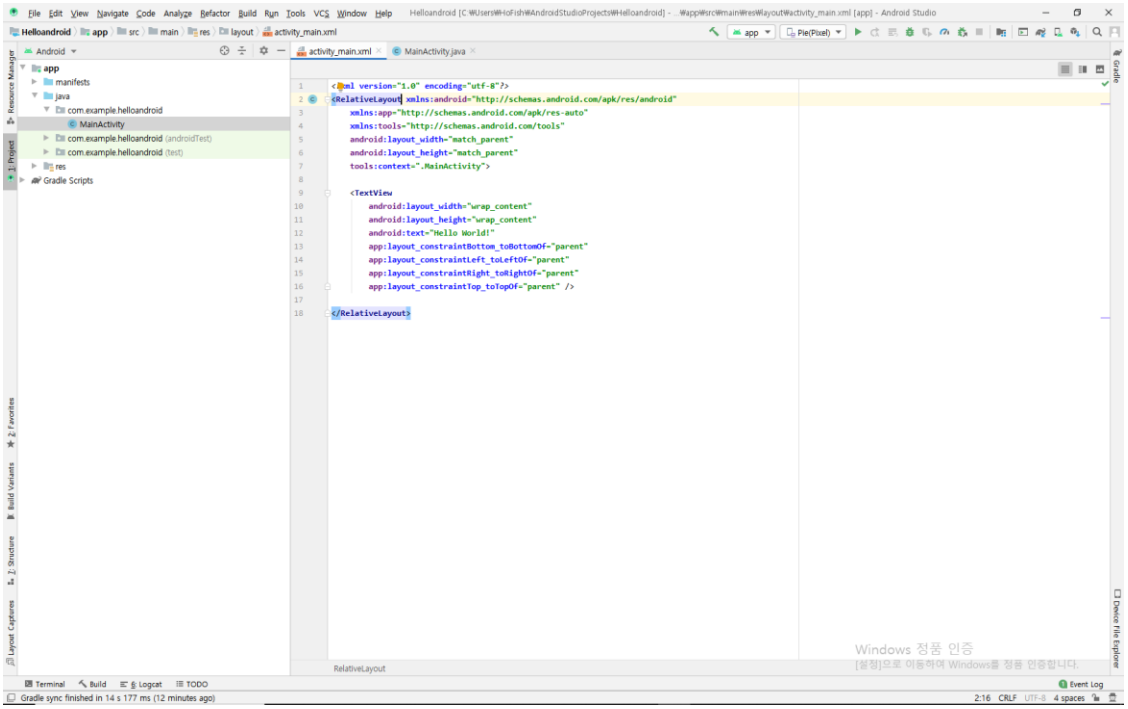
프로젝트 이름은 HelloAndroid 언어는 Java를 선택해줍니다.
설정이 끝났으면 finish를 눌러줍니다.



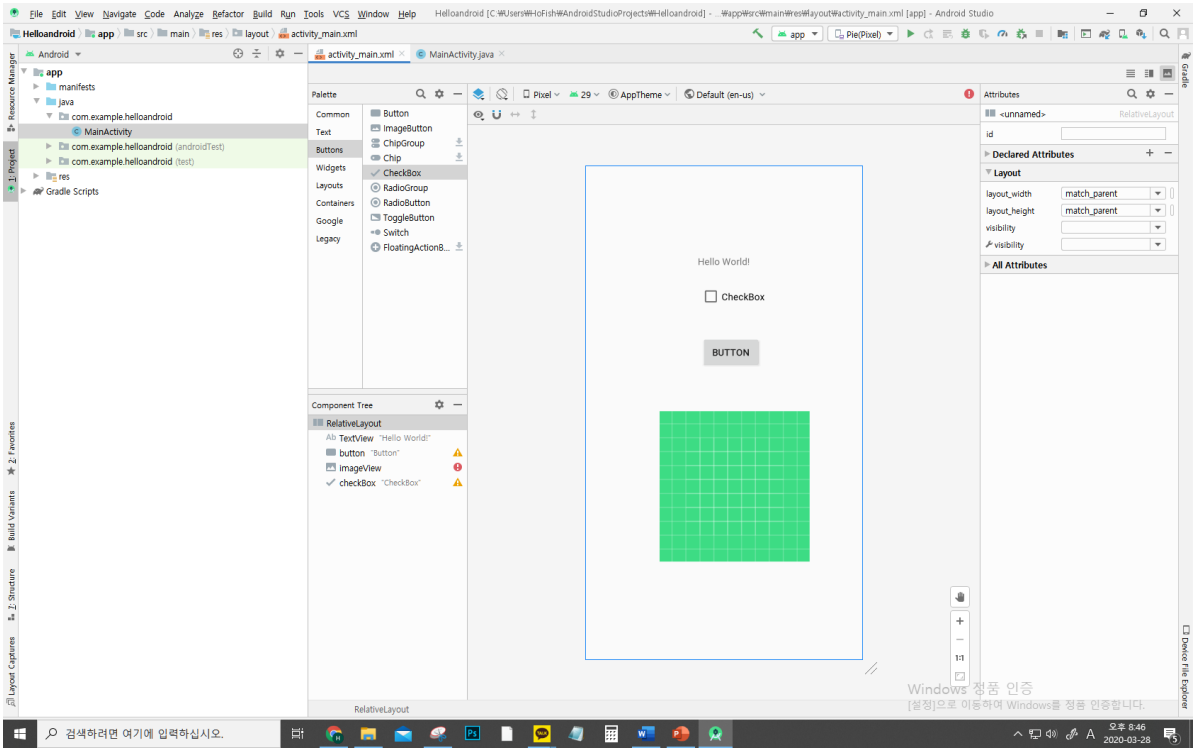
핸드폰의 설정을 만들어놓은 AVD로 설정해줍니다.



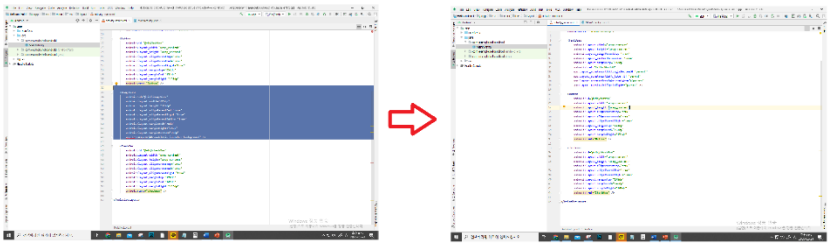
빨간 네모 부분을 보면 아이콘이 세 개 있는데 왼쪽부터 첫 번째는 코드를 보는 것이고, 두번째는 코드와 화면을 보여줍니다. 마지막은 화면만 보여주는 아이콘입니다.



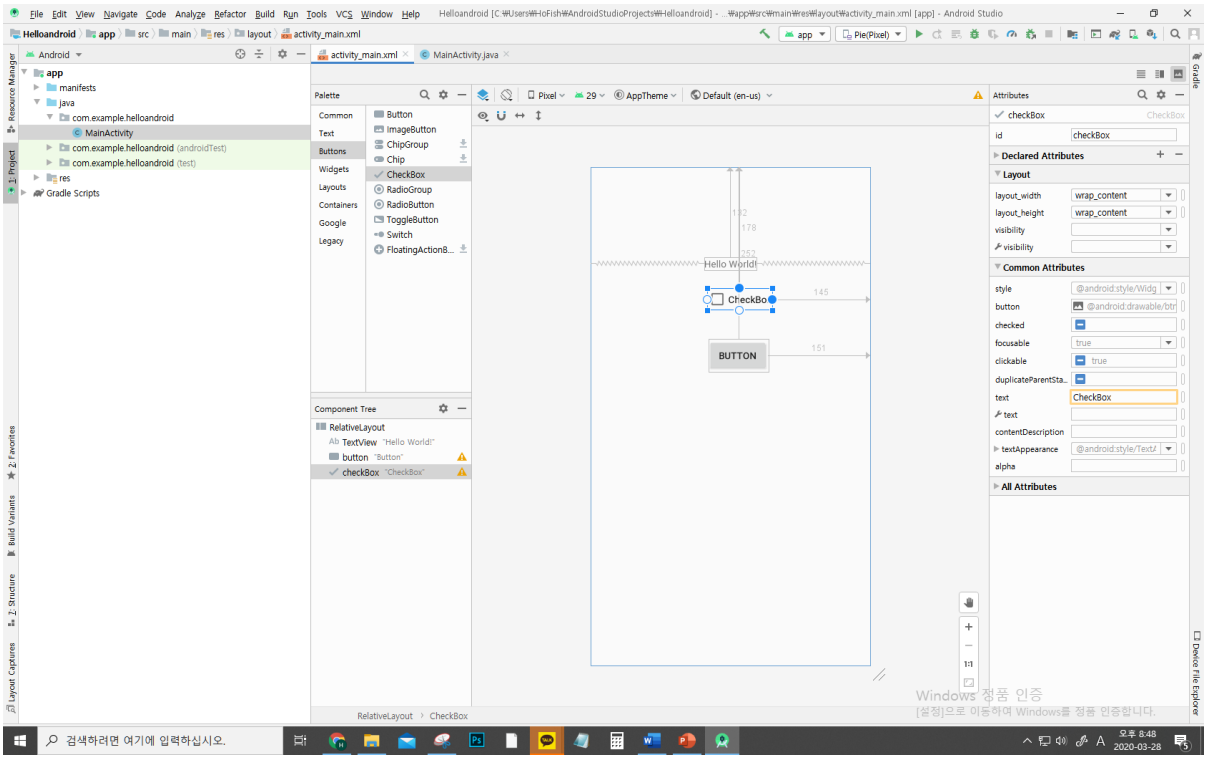
android.support.constraint.ConstraintLayout을 RelativeLayout 구문을 RelativeLayout으로 바꾸었습니다.

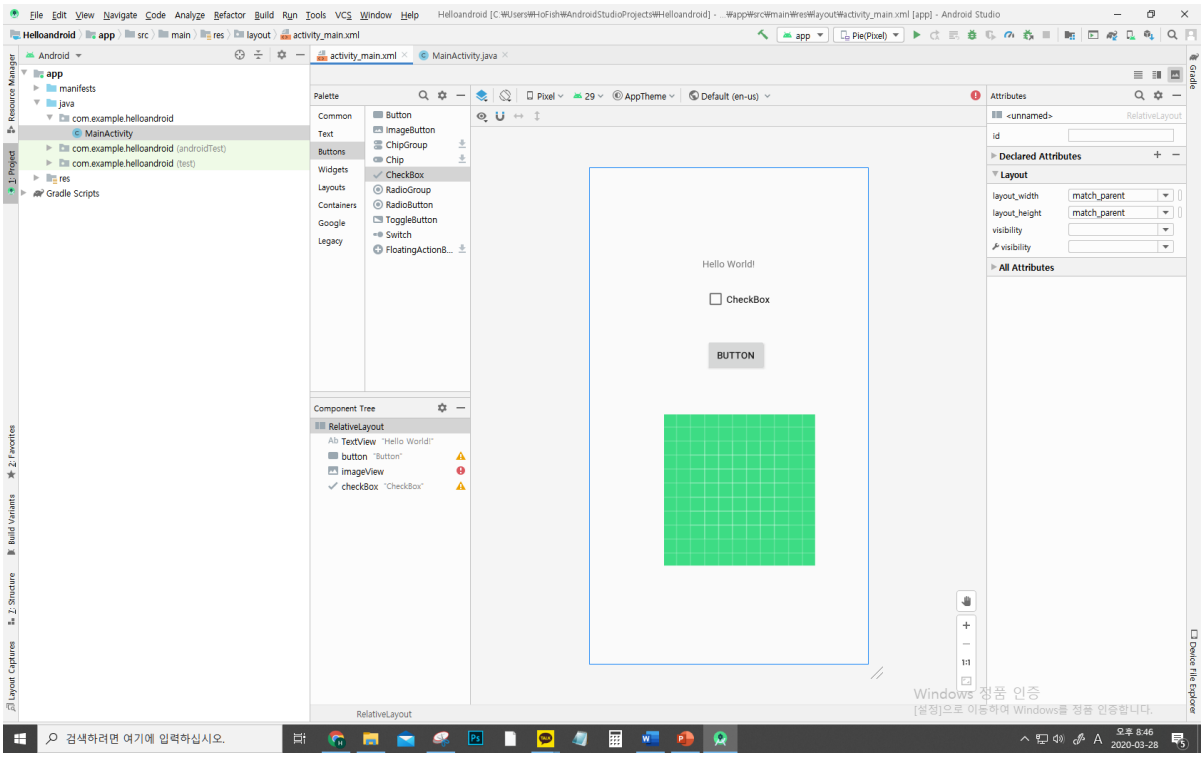


다시 화면으로 돌아가서, 왼쪽의 위젯에서 이미지뷰와 버튼에서 체크박스 버튼과 버튼을 가져왔습니다.

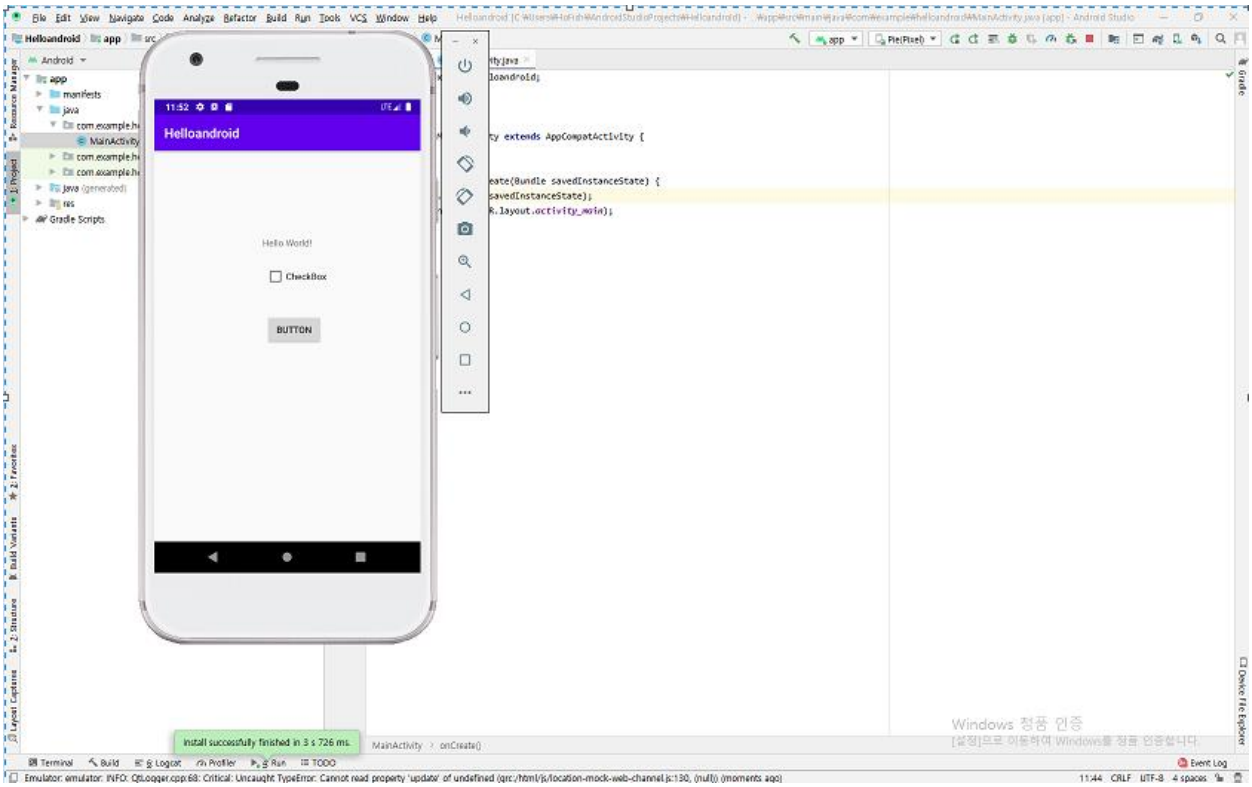


위 처럼 코드 부분으로 돌아가 이미지 뷰의 코드를 없애 줍니다.
그럼 다음과 같이 화면에서의 이미지 뷰도 함께 사라집니다.


Hello World!
CheckBox
BUTTON


Hello World!
CheckBox
BUTTON

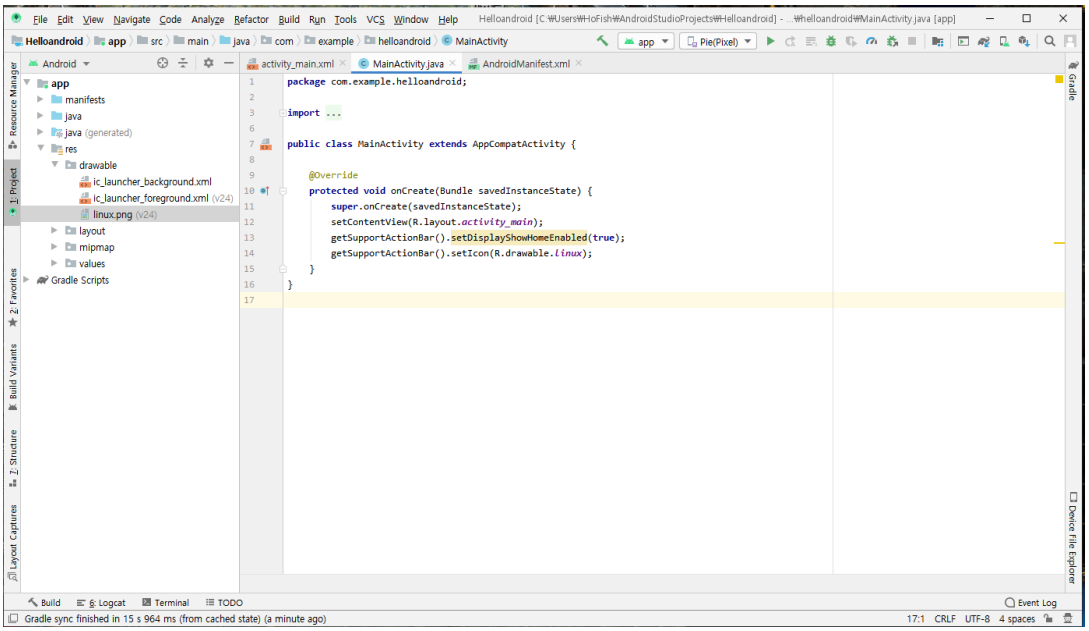
왼쪽의 트리에서 java – com.example.프로젝트이름 - MainActivity 를 더블클릭합니다.
이 파일은 메인 java코드입니다.



상단 메뉴에서 Run -> Run'app' 을 눌러 가상 AVD 화면을 띄웁니다.
PPT내용에서는 Choose Device 창이 떴지만 최신버전이라 그런지 팝업창이 나오지 않았습니다.

제가 사용할 아이콘입니다. Png로 다운받아줍니다.

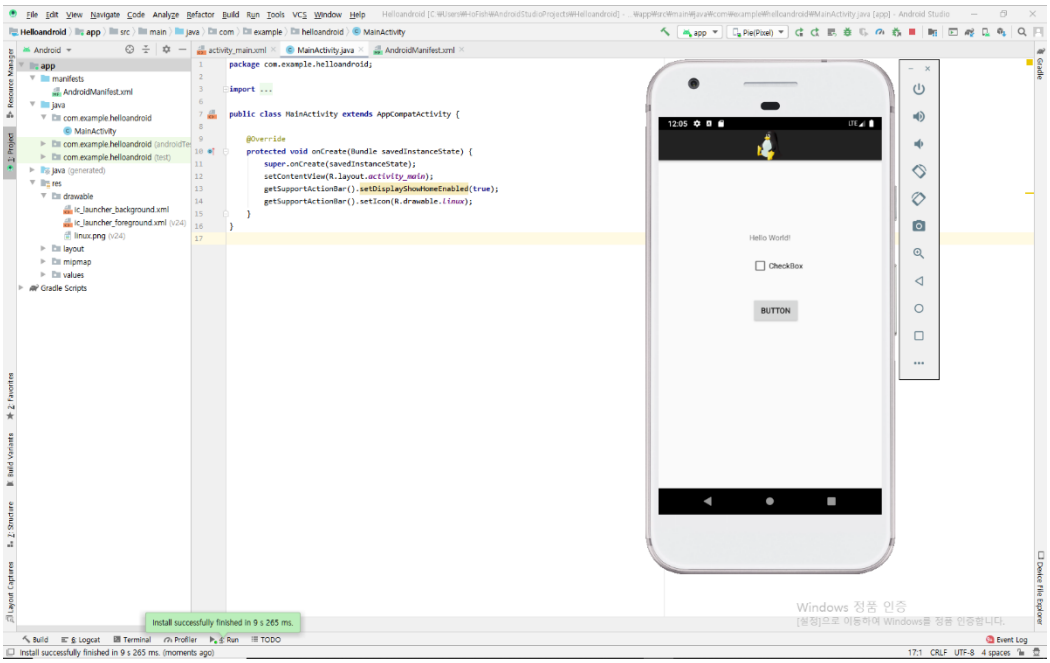
이 아이콘을 끌어다가, 왼쪽 트리의 app -> res -> drawble 안에 넣어줍니다.



그리곤 메인 java코드인 MainActivity로 돌아가 화면 처럼
GetSupportActionbar().setDisplayShowHomeEnable(true);
GetSupportActionbar().setIcon(R.drawable.이미지파일이름);
두 구문을 추가해줍니다.
첫 번째 구문은 앱의 상단 바에 아이콘을 표시 할 수 있게 해줍니다.
두 번째 구문은 앱의 상단 바에 표시할 아이콘이 무엇인지 가르쳐줍니다.

이번엔 왼쪽 트리에서 app -> manifests -> AndroidManifest.xml을 더블 클릭해서 열고
11행의 android:theme 부분을 다음과 같이 수정해 줍니다.
Android:theme = “@style/AppTheme”
Android:theme = “@style/Theme.AppCompat.Light.DarkActionBar”
이 구문의 수정은 앱의 상단 바의 색을 검은색으로 바꾸어 줍니다.

위와 같이 설정하면 아래와 같은 화면을 볼 수 있습니다.


package com.example.helloandroid;
import ...
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
getSupportActionBar().setDisplayShowHomeEnabled(true);
getSupportActionBar().setIcon(R.drawable.linux);
}
}
Hello World!
CheckBox
BUTTON
Install successfully finished in 9 s 265 ms.

과제 3

Slide 68 의 “직접 풀어 보기 2-3”에서 홈페이지 열기까지를 수행하고 1 과 같은 결과 보고서를 작성하시오. (전화 걸기, 갤러리 열기 수행 시 추가 점수)

2018037017 박소연

2017037008 한현지

▶ 직접 풀어보기 2-3

다음 그림과 같이 버튼 4개를 만들고 각 버튼을 클릭하면 필요한 내용이 작동되는 FourButton 프로젝트를 작성하라. 각 버튼은 다른 색상으로 변경한다.

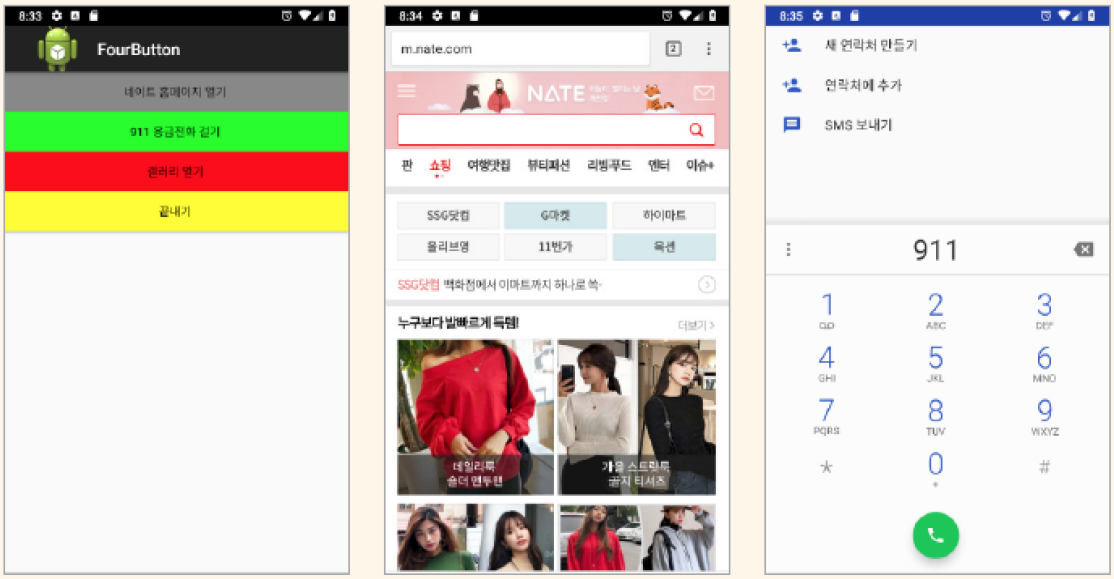
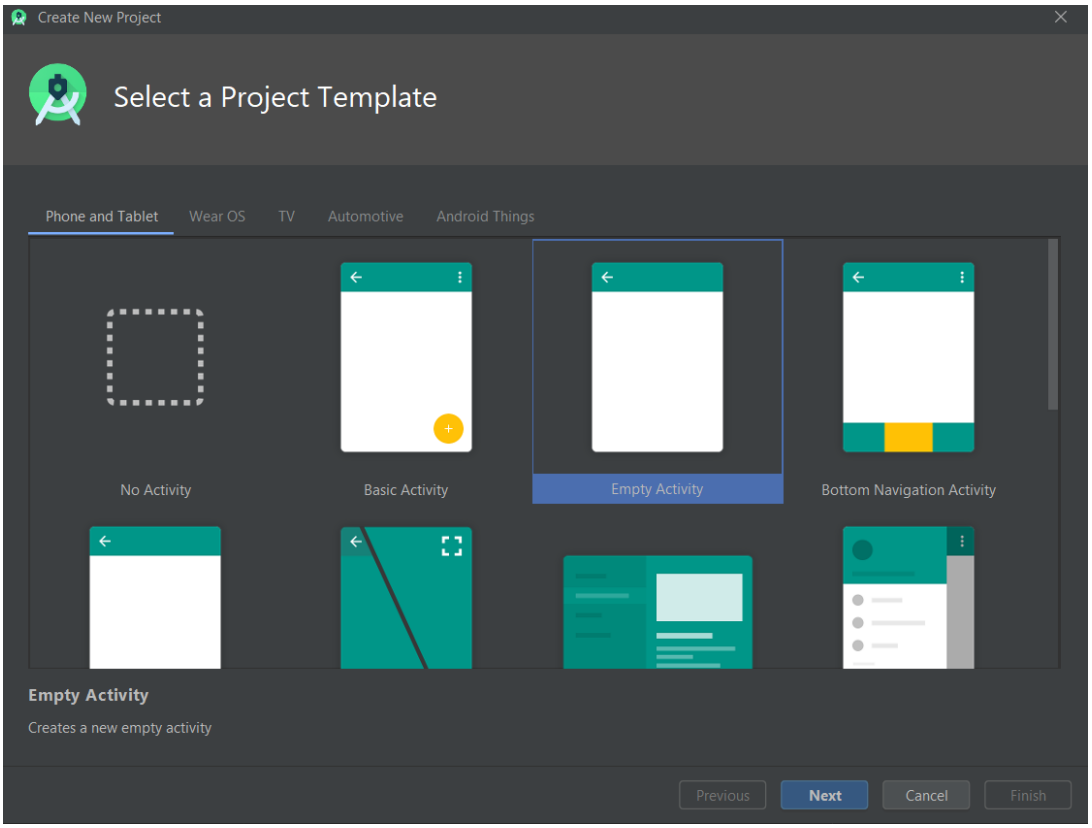
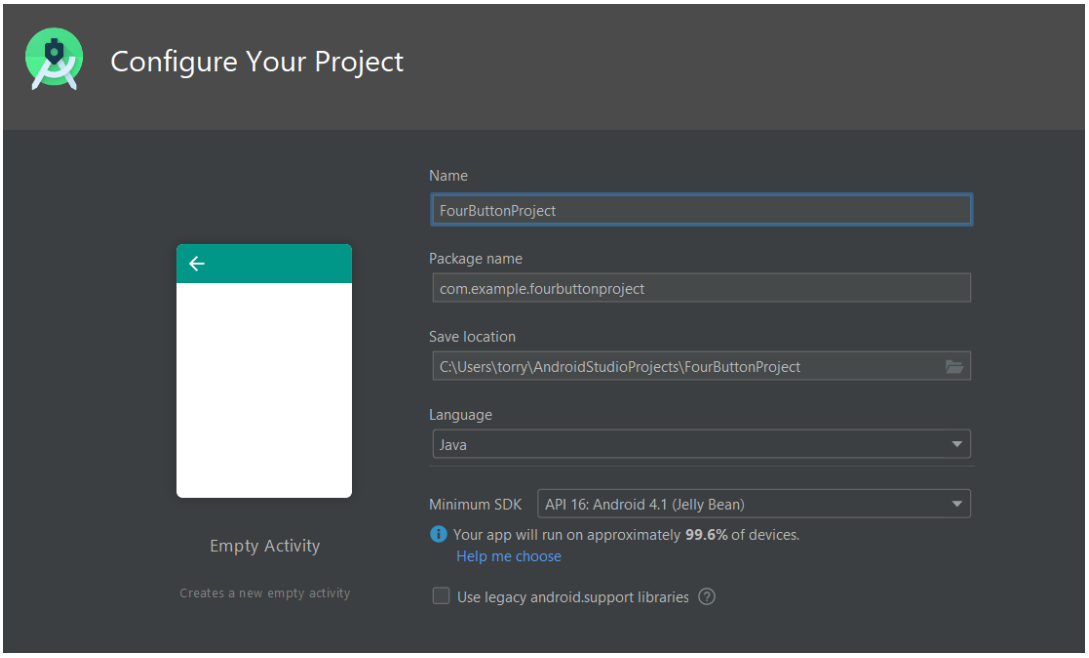


그림 2-64 실행 결과

문제에서는 총 4 개의 버튼을 요구하며 각 버튼마다 화면의 전환이 이루어져야 한다. 따라서 BaseApp 에서 버튼을 다뤘던 방식을 토대로 4 개의 버튼을 생성해낼 수 있을 것이다. 디자인 틀은 activity_main.xml 을 통해 수정할 수 있다. text 버튼을 눌러 디자인을 문제와 유사하게 구현한다. 버튼의 색은 android:background=“#색상표”를 통해 바꿀 수 있다.



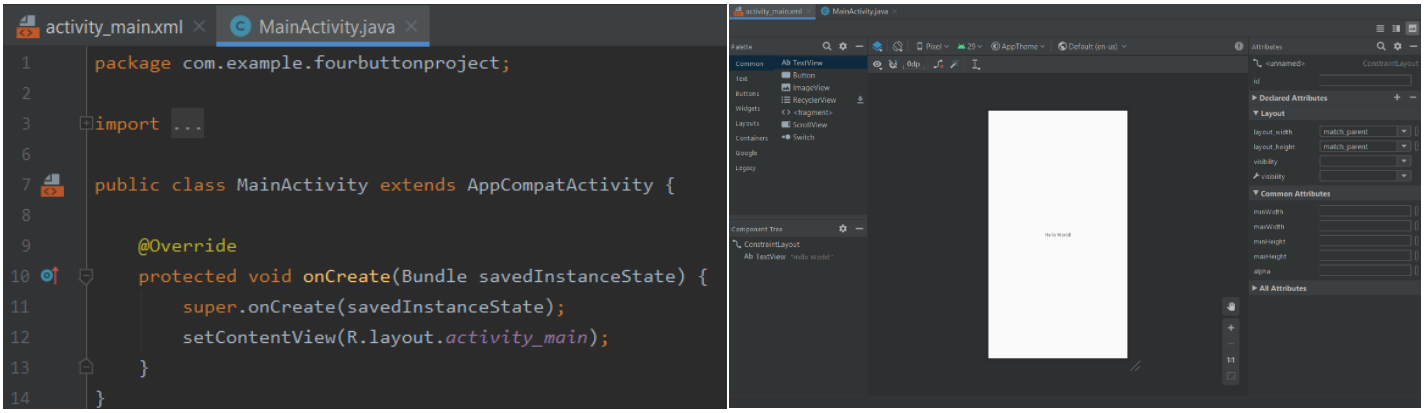
새 프로젝트를 만들기 전에, 템플릿을 고르자. 우선 빈 액티비티 (Empty Activity)로 시작한다.



JAVA 에서는 주로 PascalCase 를 사용하기 때문에 프로젝트 이름은 FourButtonProject 로 해주고,

language 에는 JAVA 를 선택한 뒤 Finish 를 눌러 프로젝트를 시작하자.

UI 구성하기



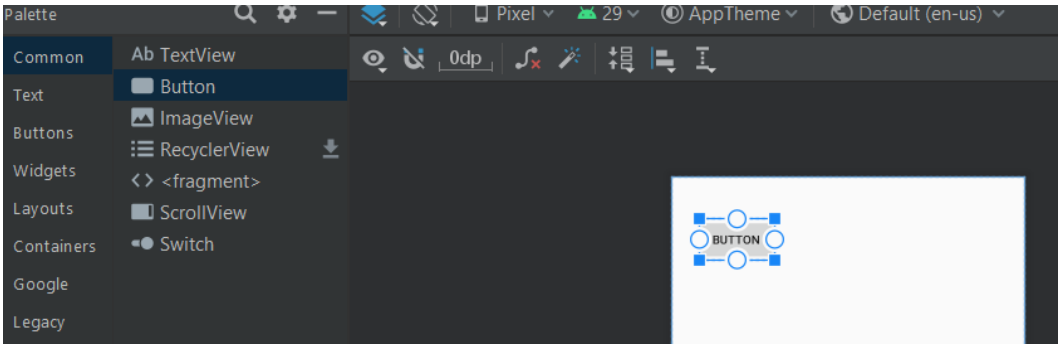
1 프로젝트 시작 시 첫 화면

안드로이드 스튜디오를 실행하고 난 첫 화면의 모습이다.

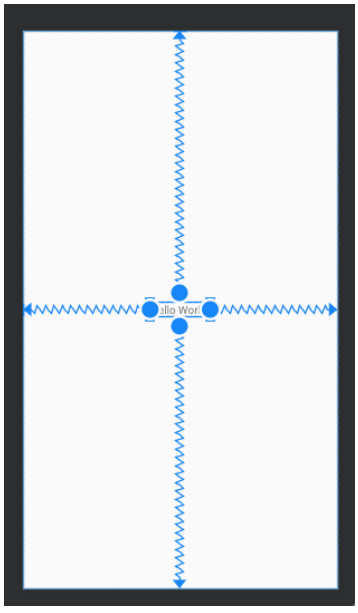
위 화면의 MainActivity.java 와 activity_main.xml 파일 두개를 주로 보게 될 것이다.

Activity_main.xml 에서 마우스를 이용해 UI 를 구성할 수 있다.

자동 생성된 “Hello World” 텍스트를 선택하고 지우자. 마우스로 클릭해 선택한 뒤 키보드의 Del 버튼으로 지울 수 있다.

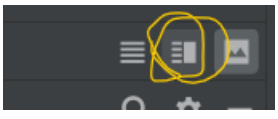


왼쪽 팔레트에서 버튼을 선택한 뒤 화면으로 드래그해 가져오자.

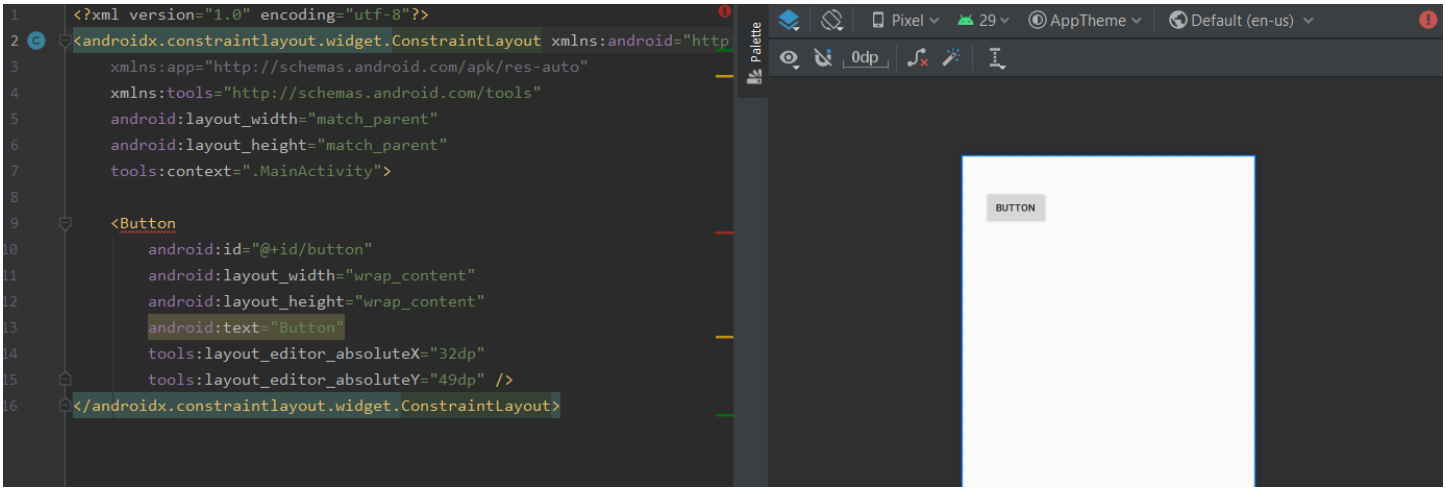


2 Hello World 선택

화면에 버튼이 생긴 것을 볼 수 있다.



마우스로도 할 수 있지만 텍스트 형식의 코드로 버튼을 상세 조정할 것이다. 오른쪽 상단의 세 버튼 중 가운데를 클릭해 코드와 레이아웃을 동시에 볼 수 있다.

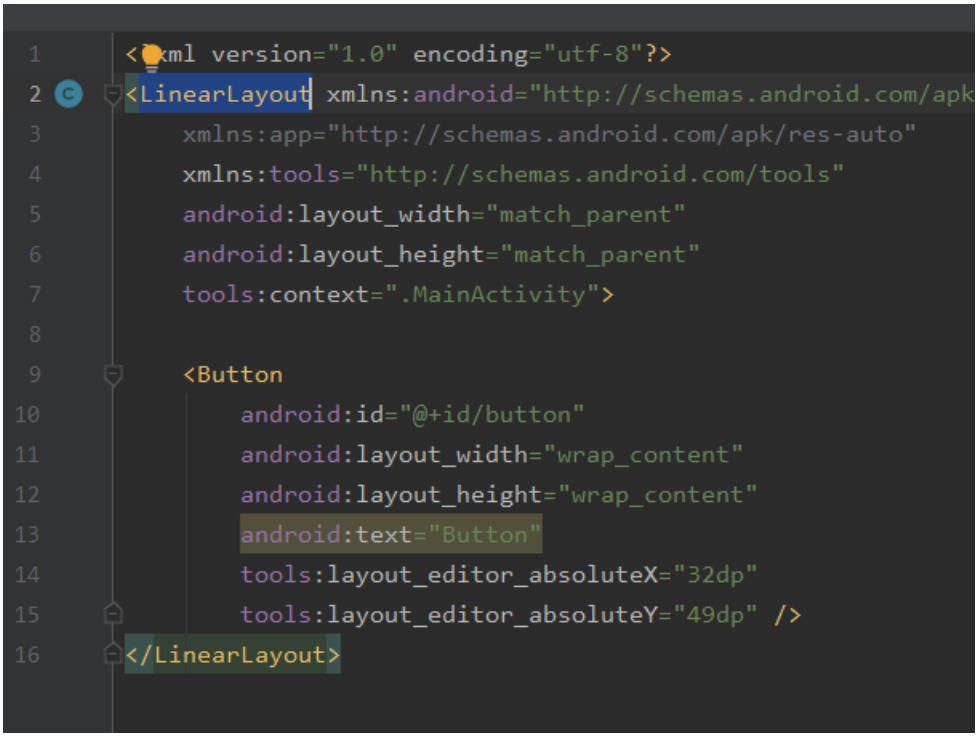


1 마우스로 구성한 UI 가 텍스트로 나타난다

텍스트를 확인해보면 레이아웃이 ConstraintLayout 으로 지정되어 있는 것을 확인할 수 있다.

ConstraintLayout 은 View(하나의 요소)의 위치와 크기를 상대적으로 배치하기 최적화된 레이아웃이다. 이번 프로젝트에서는 LinearLayout 을 사용하자. LinearLayout 은, View 들을 수직, 수평 방향으로 배치하는 레이아웃이다.

가장 단순하고 직관적이기 때문에 사용했다.



ConstraintLayout 에서 LinearLayout 으로 바꿔줬다. 위 텍스트만 바꿔도 아래는 자동으로 바뀐다.

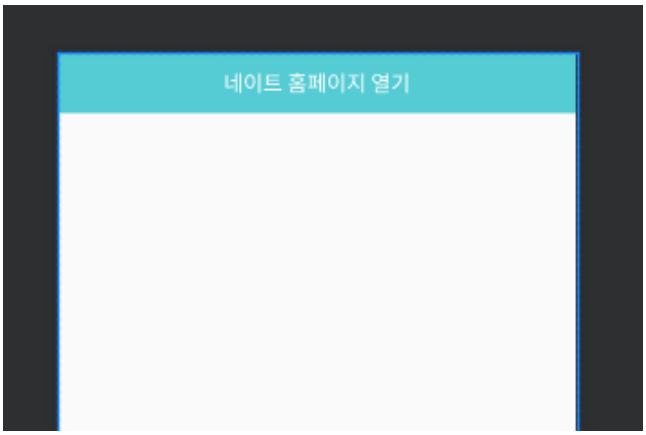
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.a
    xmlns:app="http://schemas.android.com/apk
    xmlns:tools="http://schemas.android.com/t
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
```

위에서 설명했듯, LinearLayout 은 View 들을 한 방향으로 배치하는 레이아웃이다. 이번 프로젝트에서 4 개의 버튼을 수직방향으로 배치하기 위해, LinearLayout 의 orientation 속성을 vertical 로 지정해주자.

android:orientation=”vertical” 문장을 추가해주면 된다.

2 LinearLayout 방향 정하기

```
<Button
    android:id="@+id/btn_page"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#55ccd4"
    android:textColor="#fff"
    android:textSize="17dp"
    android:text="네이트 홈페이지 열기"
    />
```



마우스 드래그를 이용해 추가해줬던 버튼을 수정해보자.

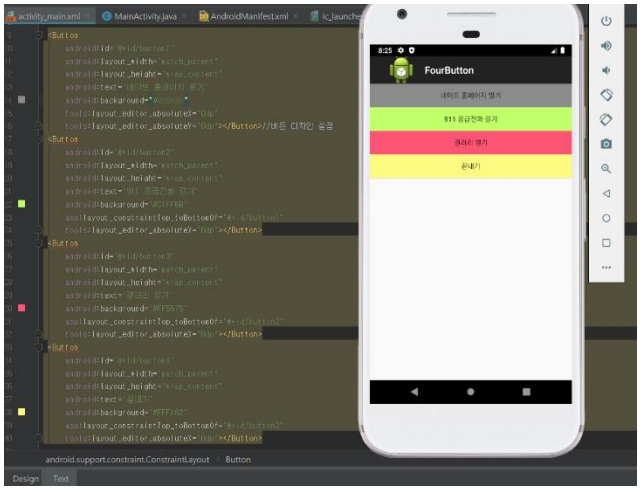
android:layout_width=”match_parent” 문장으로 버튼의 가로를 부모 View 와 맞춰줬다.

```
    />
<Button
    android:id="@+id/btn_call"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#c7dac7"
    android:textColor="#fff"
    android:textSize="17dp"
    android:text="911 응급전화 열기" />

<Button
    android:id="@+id/btn_gallery"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#ffb68d"
    android:textColor="#fff"
    android:textSize="17dp"
    android:text="갤러리 열기" />

<Button
    android:id="@+id/btn_exit"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

이때, 부모 View 가 없기 때문에 화면의 가로 길이와 맞춰진다.

배경색과 글자색을 원하는 대로 수정하고, 버튼을 복사해 4 개로 만들어주자.

4 개 버튼의 텍스트와 배경색을 바꿔줬다. 이때 아이디도 같이 바꿔주자.



아이디는 네 버튼 모두 다르게 지정해야 한다. Java 코드에서 버튼을 가져올 때 아이디로 구분해 View 를 가져오기 때문이다. 이번 프로젝트에서 아이디는 거의 사용하지 않지만 꼭 써주자.

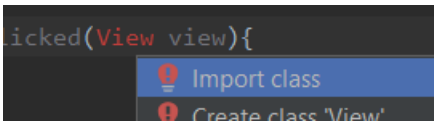
Run App 을 통해서 가상에뮬레이터를 실행시키면 이와 같은 결과를 볼 수 있다. 그러나 이 상태에서는 버튼을 눌러도 아무런 변화가 없으므로 MainActivity.java 에 추가로 동작하고자 하는 소스 코드를 작성해 주어야 한다.

웹사이트 열기

함수를 만들고 onClick 속성으로 함수를 실행시키는 방법

```
import ...

public class MainActivity extends AppCompatActivity {

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

    public void onPageBtnClicked(View view){

    }
}
```

자동 생성된 onCreate 함수 아래 onPageBtnClicked 함수를 만들어줬다. 접근자는 public 으로 설정해줬다.



View 가 import 되지 않아 “View” 기호를 확인할 수 없다는 오류가 난다. 커서를 View 의 위치로 옮겨주고 Alt+Enter 를 치면 자동으로 import 할 수 있다.

첫번째 버튼을 클릭하면 이 함수를 실행하기 위해, 다시 activity_main.xml 로 돌아가 버튼에 onClick 속성을 추가해주자.

```
<Button
    android:id="@+id/btn_page"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#55ccd4"
    android:textColor="#fff"
    android:textSize="17dp"
    android:text="네이트 홈페이지 열기"
    android:onClick="onPageBtnClicked"
    />
```

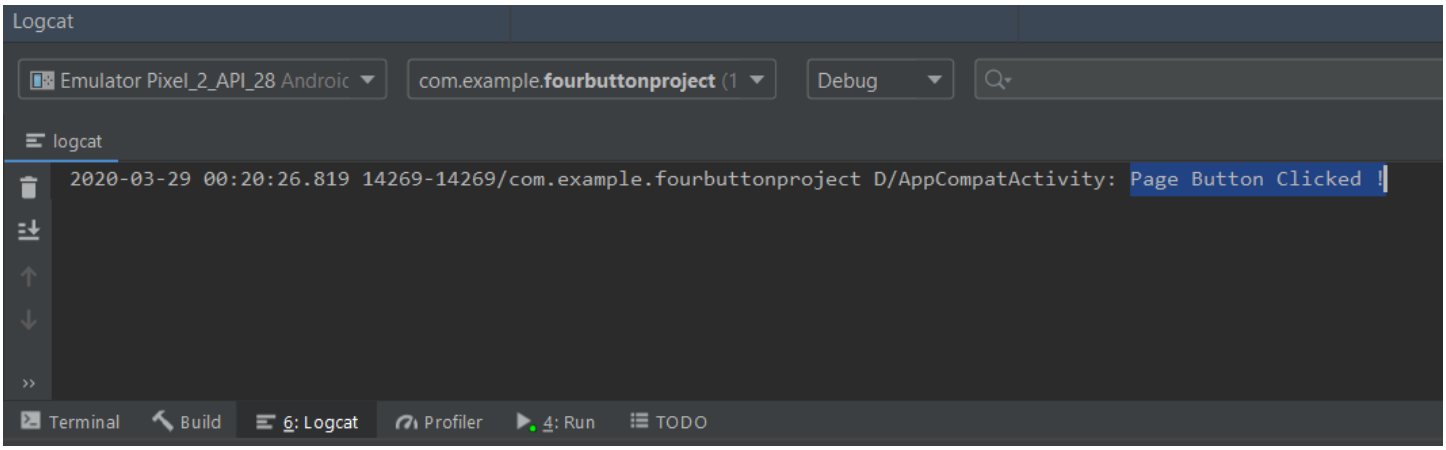
페이지를 여는 버튼에 onClick 속성을 추가하고 함수 이름을 입력하면, 버튼을 클릭했을 때 그 함수가 동작한다.

함수가 잘 동작하는지 테스트를 해보자.

```
public void onPageBtnClicked(View view){
    Log.d( tag: "AppCompatActivity", msg: "Page Button Clicked !");
}
```

Log.d("tag", "메시지")로 로그를 출력할 수 있다. * 로그 메서드에는 d 외에 v, I, w, e, a 등이 있다.

위와 같이 적고 저장한 뒤 AVD, 혹은 연결된 안드로이드에서 네이트 페이지 열기 버튼을 클릭하면



Logcat 에 정해준 메시지가 출력되는 것을 볼 수 있다. onClick 이 잘 동작하는 것을 확인했으니, 웹페이지를 열어보자.

```
public void onPageBtnClicked(View view){
    Intent intent = new Intent(Intent.ACTION_VIEW);
}
```

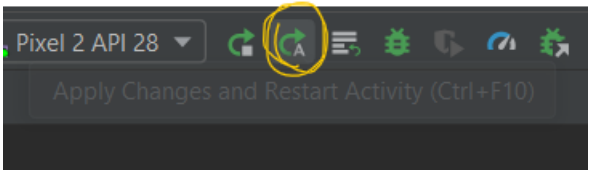
현재 액티비티 외 다른 액티비티를 열기 위해선, Intent(참고: https://developer.android.com/guide/components/intents-filters?hl=ko)가 필요하다. 웹페이지를 보여주기 위해 ACTION_VIEW 를 작업으로 지정해주자.

```
public void onPageBtnClicked(View view){
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("https://m.nate.com"));
    startActivity(intent);
}
```

웹페이지의 주소를 전달해주고, intent 를 startActivity 로 전달하면 끝이다.

저장하고 AVD 나 연결된 안드로이드에서 코드를 실행해보자.

이때, 우측상단의 버튼으로 변경사항을 적용하고 앱을 다시 시작할 수 있다.



클릭 리스너로 클릭에 반응하기

버튼을 사용할 수 있는 Button 변수를 선언한다. 또한 findViewById()메소드를 통해서 activity_main.xml 에서 만든 객체에 접근할 수 있다. 클릭 리스너(OnClickListener())는 버튼을 눌렀을 때의 동작을 수행한다. 한마디로 인터페이스라고 할 수 있는데 이곳에 동작할 코드를 넣어주면 된다.

통화 연결하기

```
public void onCallBtnClicked(View view){
    Intent intent = new Intent();
    startActivity(intent);
}
```

```
<Button
    android:id="@+id/btn_call"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#c7dac7"
    android:textColor="#fff"
    android:textSize="17dp"
    android:text="911 응급전화 열기"
    android:onClick="onCallBtnClicked"
    />
```

위와 마찬가지로, onCallBtnClicked 라는 함수를 만들어 주고,

activity_main.xml 에서 onClick 속성에 추가해주자.

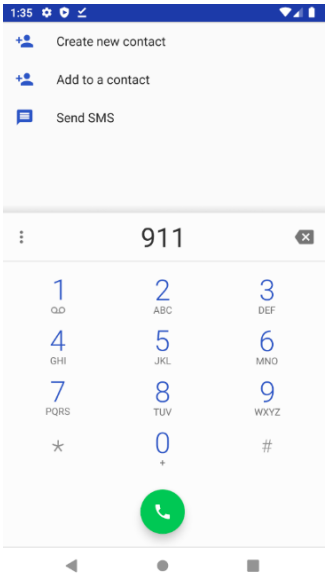
이번에도 다른 앱을 열어야 하기 때문에 미리 intent 를

만들어줬다.

다이얼을 열어 보여주는 것이기 때문에, ACTION_VIEW 작업을 사용할 것 같지만, ACTION_DIAL 작업을 사용한다.

```
public void onCallBtnClicked(View view){
    Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:911"));
    startActivity(intent);
}
```

바로 AVD 에서 테스트해보자.



테스트 성공!

갤러리를 열어보자

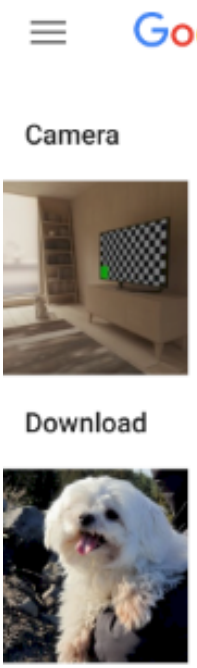
```
public void onGalleryBtnClicked(View view){
    Intent intent = new Intent(Intent.ACTION_VIEW);
    startActivity(intent);
}
```

마찬가지로, onGalleryBtnClicked 를 만들어줬다. 갤러리를 열어 보여주는 작업을 하기 때문에 ACTION_VIEW 를 지정해줬다. Activity_main.xml 에서 세번째 버튼에 onClick 속성을 넣어주는 걸 잊지 말자.

```
public void onGalleryBtnClicked(View view){
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("content://media/internal/images/media"));
    startActivity(intent);
}
```

이제 갤러리 위치만 적어주면 된다. Uri.parse(“content://media/internal/images/media”)로 갤러리를 열 수 있다.

위와 같이 적어주고 테스트해보자.

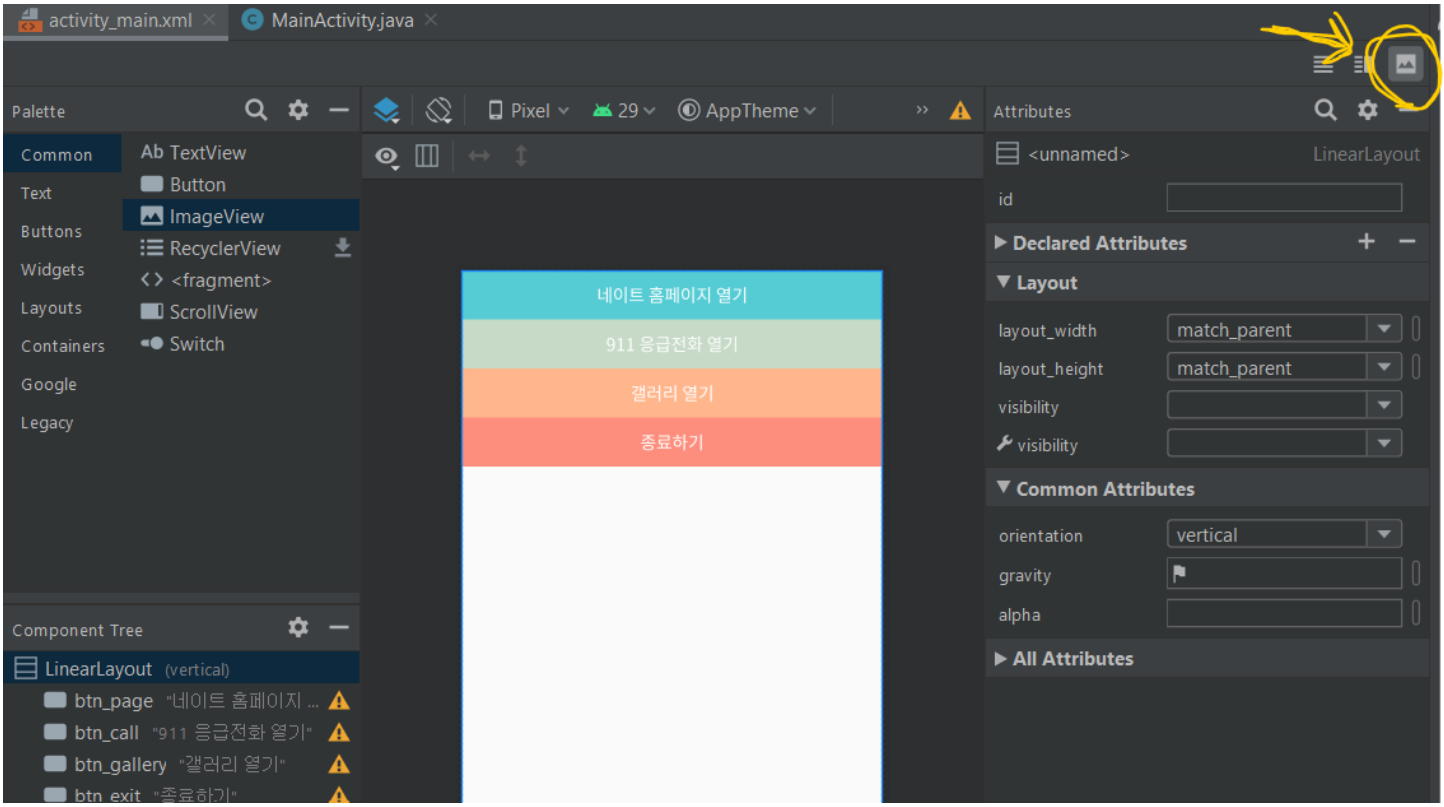


테스트 성공! Google Photos 앱이 열린다.

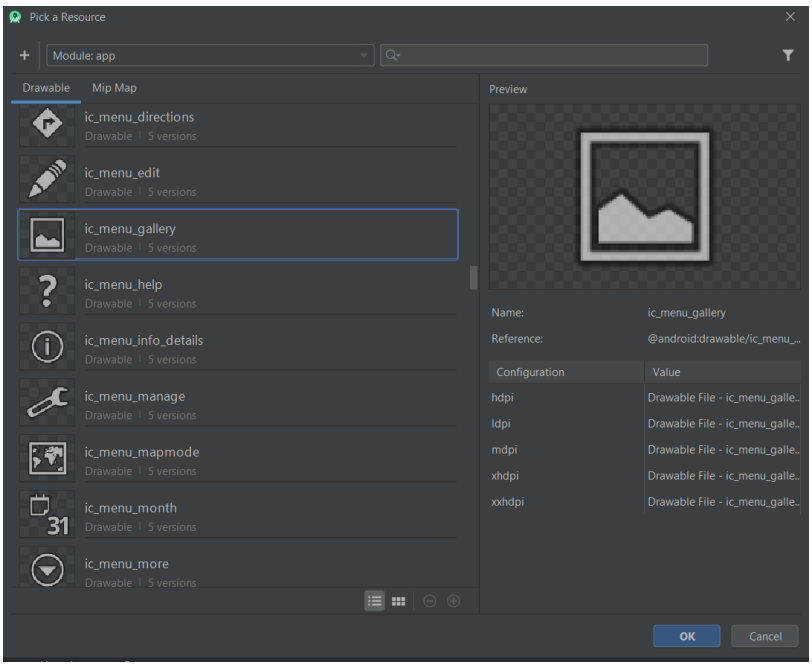
갤러리에서 이미지를 선택해 가져오자

갤러리를 열어보는데 성공했다. 그럼 갤러리에서 사진을 선택해 가져오려면 어떻게 해야 할까?

이미지를 가져오기 전에, ImageView 를 만들어두자.

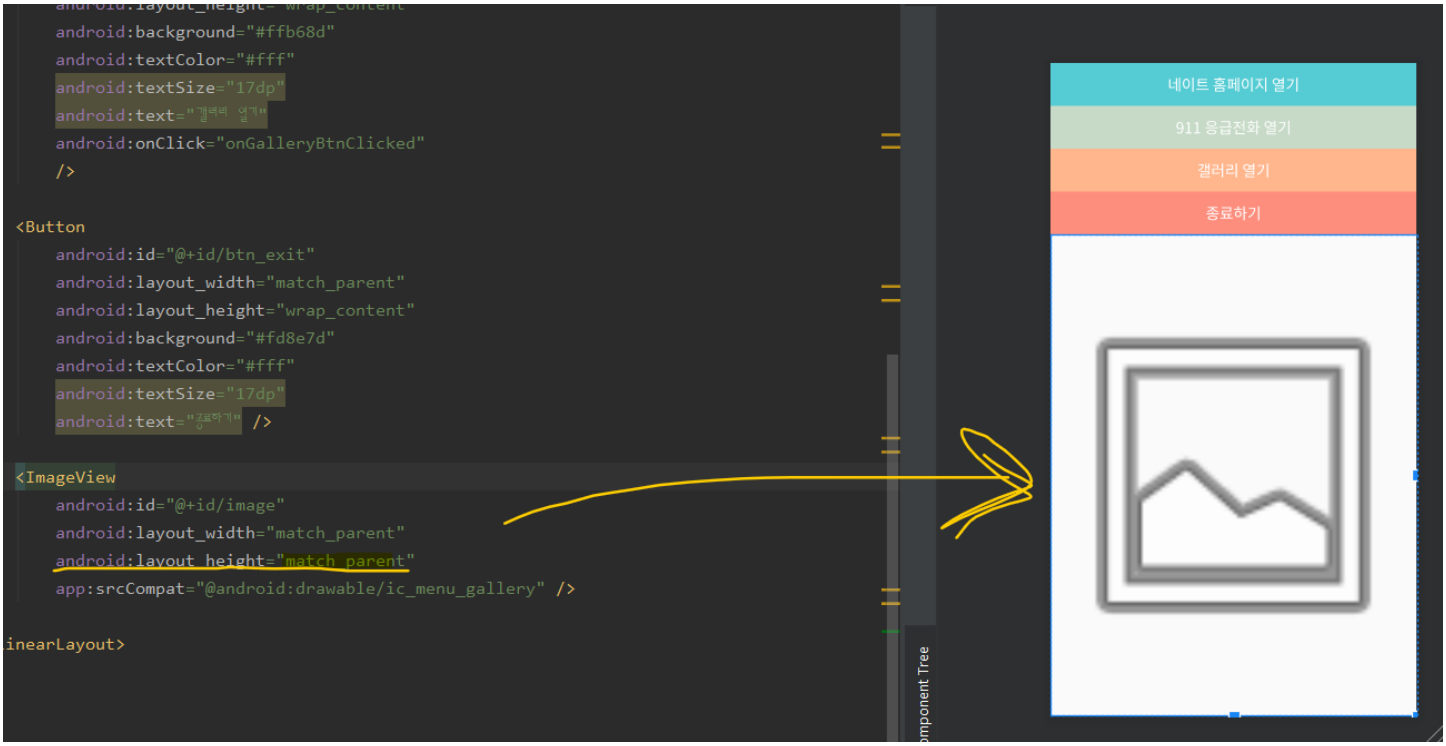


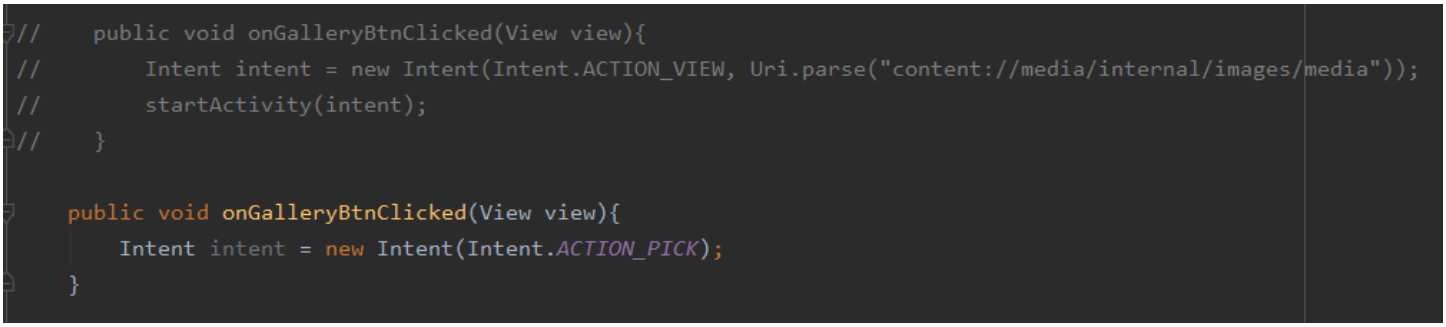
화면 구성 페이지로 돌아와서, 왼쪽 팔레트의 ImageView 를 드래그해 화면에 두자.



그리고 나온 팝업창에서 원하는 이미지 아무거나 선택해 OK 를 누르자.

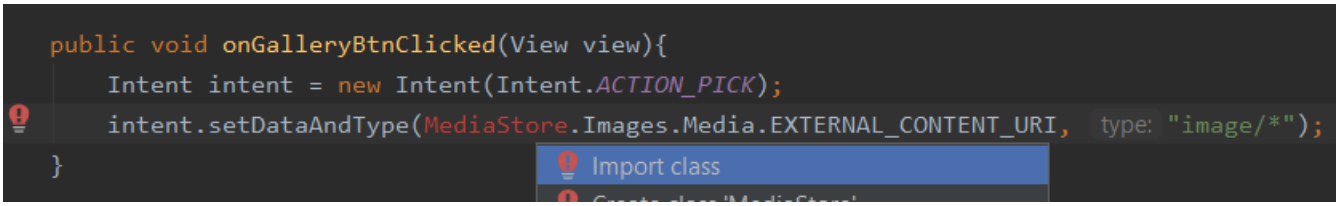
높이와 너비를 math_parent 로 꽉 차게 만들어 주고 MainActivity.java 로 돌아오자.





전에 썼던 코드를 주석 처리하고, 같은 이름의 함수를 만들어 줬다.

이때, 작업은 ACTION_VIEW 가 아닌 ACTION_PICK 을 사용한다. 이름대로 하나를 선택하는 작업이다.



이제 어디서 무엇을 가져올지 지정하도록 하자. 안드로이드 시스템에서 제공하는 데이터 DB 인 MediaStore 에서 가져올 것이고, 타입은 이미지로 정해줬다. MediaStore 에 커서를 두고 Alt+Enter 로 클래스를 import 할 수 있다.

```
private final int GET_GALLERY_IMAGE = 200;

public void onGalleryBtnClicked(View view){
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setDataAndType(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,  type: "image/*");
    startActivityForResult(intent, GET_GALLERY_IMAGE);
}
```

결과 값을 가져오기 위해 startActivity 가 아닌 startActivityForResult 를 사용했다.

이때, 실행하는 액티비티가 여러 개일 경우 결과값을 구분하기 위해 0 이상의 Integer 를 같이 건네 줘야 한다.

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

    }
```

이제 액티비티에서 결과값을 가져오는 함수를 만들어 주자. onActivityResult 만 쳐도 자동완성으로 만들 수 있다.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if (requestCode == GET_GALLERY_IMAGE && resultCode == RESULT_OK && data != null && data.getData() != null){

    }
}
```

결과값을 구분하기 위해 같이 준 정수 값이 일치하는지, 데이터가 존재하는지 여부를 검사하고, 받아온 데이터를 이미지 View 에 적용시켜줄 것이다.

```
if (requestCode == GET_GALLERY_IMAGE && res
    ImageView image;
    image = findViewById(R.id.image);
}
```

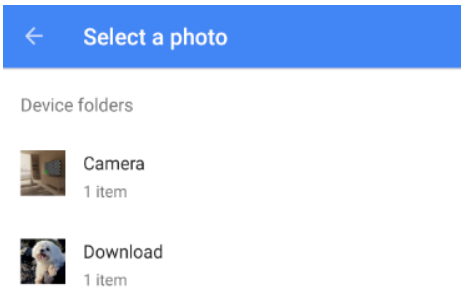
ImageView 의 아이디로 ImageView 를 가져왔다.

이제 이 ImageView 에 가져온 데이터를 적용해 주자.

```
if (requestCode == GET_GALLERY_IMAGE && re
    ImageView image;
    image = findViewById(R.id.image);
    image.setImageURI(data.getData());
}
```

image.setImageURI(data.getData()); 문장을 추가해 아이디를 통해 가져온 ImageView 의 이미지를 갤러리에서 선택한 이미지로 바꿔준다.

이제 테스트를 해보자.



갤러리 열기 버튼을 누르면, 선택한 이미지로 ImageView 가 바뀌는 것을 볼 수 있다.

앱 종료하기

마지막으로, 종료 버튼을 만들어보자.

이번에는 intent 를 사용하지 않는다.

onExitBtnClicked 함수를 만들고 마지막 버튼에 onClick 속성을 추가해주자.

```
public void onExitBtnClicked(View view){

}
```

종료 버튼을 누르면 종료하시겠습니까?하고 물어보고, “네”라고 대답할 시 앱을 종료할 것이다.

버튼을 누르면 먼저 종료하시겠습니까?를 물어보게 하자.

```
public void onExitBtnClicked(View view){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActivity.this);
    builder.setMessage("종료하시겠습니까?");
}
```

팝업창을 AlertDialog 라 한다. AlertDialog 를 현재 액티비티에 만들어주고, “종료하시겠습니까?” 메시지를 추가하자.

```
ublic void onExitBtnClicked(View view){
  AlertDialog.Builder builder = new AlertD
  builder.setMessage("종료하시겠습니까?");
  builder.setCancelable(false)
          .setPositiveButton()
          .setNegativeButton()
```

setCancelable 을 false 로 해 취소를 못하게 하고,

PositiveButton 을 눌렀을 때와 NagativeButton 을 눌렀을 때를 다르게 설정할 것이다.

```
public void onExitBtnClicked(View view){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActivity.this);
    builder.setMessage("종료하시겠습니까?");
    builder.setCancelable(false)
            .setPositiveButton( text: "네", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {

                }
            })
            .setNegativeButton( textId: "아니오")
}
```

PositiveButton 을 “네”로 설정하고, “네” 버튼을 눌렀을 때의 행동을 정해줘야 한다.

“네”, new Dialog 까지 입력하면 자동 완성되는 함수에 행동을 입력해주기만 하면 된다.

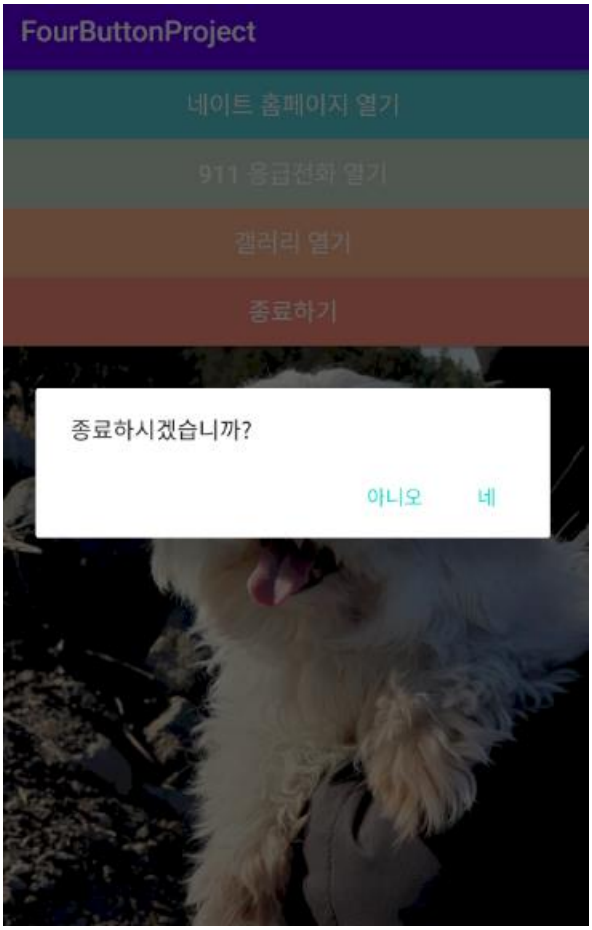
종료 함수 finish()를 적어주자.

```
blic void onExitBtnClicked(View view){
  AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActivity.t
  builder.setMessage("종료하시겠습니까?");
  builder.setCancelable(false)
          .setPositiveButton( text: "네", new DialogInterface.OnClickListener() {
              @Override
              public void onClick(DialogInterface dialog, int which) {
                  finish();
              }
          })
          .setNegativeButton( text: "아니오", new DialogInterface.OnClickListener()
              @Override
              public void onClick(DialogInterface dialog, int which) {
                  dialog.cancel();
              }
          });
```

마찬가지로, “아니오” 버튼에는 팝업 메시지를 취소하도록 해주었다.

```
public void onExitBtnClicked(View view){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: MainActivity.this);
    builder.setMessage("종료하시겠습니까?");
    builder.setCancelable(false)
            .setPositiveButton( text: "네", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    finish();
                }
            })
            .setNegativeButton( text: "아니오", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.cancel();
                }
            });
    AlertDialog alert = builder.create();
    alert.show();
}
```

이제 마지막 두 문장을 추가한 뒤 앱을 실행해보자.



“아니오”를 클릭하면 팝업메시지가 사라지고,

“네”를 클릭하면 앱이 종료되는 것을 확인할 수 있다.