● setContentView(): 엑티비티에서 뷰를 출력하는 메서드. public void setContentView(View view)와 public void setContentView(int layoutResID)가 있다 — ex) setContentView(R.layout.activity_main.xml);

▶ 하나의 액티비티가 화면 전체를 차지한다. 일반적으로 정보를 보여주는 CRUD(목록화면-상세화면-신규작성화면-수정화면) 성격의 앱을 만든다면, 각각의 화면을 위한 액티비티를 4개 작성하고, 각 액티비티가 실행되면서 화면이 교체한다고 생각하면 된다.

▶ 사용자에 의해 하나의 화면을 여러 개의 액티비티로 구성하는 것은 Android 7.0(Nougat)에 추가 된 다중 창 지원(Multi Window Support) 기능이며, 사용자가 스마트폰을 분할 화면 모드(split screen mode)로 전환하면, 한 화면에 두 개의 액티비티가 출력될 수는 있다. 개발 코드에서는 자신이 개발하는 액티비티가 분활 하면 모드를 지원할 것인지에 대한 제어, 분할 화면 모드로 전환되었을 때의 생명주기(lifecycle) 등의 코드를 작성할 수 있다.

● UI 작성 방법엔 액티비티의 자바 코드로 작성하는 방법과 레이아웃 XML에서 작성하는 방법이 있다. 이렇게 준비한 뷰를 setContentView()함수의 매개변수로 넘겨서 화면을 출력한다. 자바 코드로는 UI구성 외에도 사용자 이벤트 처리, 이벤트에 의한 서버 핸들링, 각종 데이터 핸들링 등 많기 때문에, XML로 작성하는 경우가 많다.

→ MainActivity.java

```java
package com.example.practicexml;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.Button;
import android.widget.LinearLayout;

public class MainActivity extends AppCompatActivity {

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

        LinearLayout linear = new LinearLayout(this);
        linear.setOrientation(LinearLayout.HORIZONTAL);

        Button button1 = new Button(this);
        button1.setText("버튼 1");
        linear.addView(button1);

        Button button2 = new Button(this);
        button2.setText("버튼 2");
        linear.addView(button2);

        setContentView(linear);

    }
}
```

▶ .setOrientation(int orientation): LinearLayout.HORIZONTAL / LinearLayout.VERTICAL
▶ LinearLayout/Button 인스턴스 만들 때 매개변수로 context - this를 넣어 주어야 한다.
▶ .addView(View view): Child View를 추가하는 메서드
▶ .setText(CharSequence Text): TextView에 보여지는 텍스트를 설정하는 메서드
▶ setContentView(View view): 뷰를 엑티비티에서 출력하는 메서드

→ activity_main.xml OR 새로운 액티비티 추가: Java 패키지 > New > Activity > Empty Activity. Launcher Acitivity 체크박스를 체크하면, 다른 액티비티와 상관없이 독립적으로 실행하여 테스트가 쉽다. 레이아웃 XML파일에서 작성할 때는 뷰 클래스명을 XML의 태그명으로 작성하고, 필요한 속성을 XML attribute로 지정하여 작성하면 된다.

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

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼1"
        android:id="@+id/버튼1"
        android:layout_marginRight="5dp"
        android:layout_marginLeft="5dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼2"
        android:id="@+id/버튼2"
        android:layout_marginLeft="5dp"/>

</LinearLayout>
```

● 뷰의 기초 속성

▶ id: 뷰의 식별자 속성으로 필수 속성은 아니며 필요할 때 추가하면 된다. 뷰가 화면에 보인다는 것은 내부적으로 TextView 객체가 자동으로 출력되는 것이지만 자바 코드에서 뷰 객체를 획득하여 속성 변경 등의 작업을 수행할 수 없기 때문에 레이아웃 XML에 등록하여 자동으로 생성된 객체를 자바 코드에서 식별할 방법이 필요하고, 이때 이용하는 속성이 id이다. 이렇게 지정한 id값은 R.java파일에 등록된다. 따라서 id값은 자바 명명규칙을 위배할 수 없다(또한 소문자로 시작한다). 이렇게 등록된 id값을 자바 코드에서 findViewById() 메서드의 매개변수로 넘겨서 획득해서 사용할 수 있다.

```java
TextView Text1 = findViewById(R.id.text1);
```

▶ layout_width, layout_height: 뷰의 필수 속성으로 매개변수로 들어 갈 수 있는 것은 match_parent, fill_parent ,wrap_content, 또는 숫자dp/px이다. 스마트폰의 크기가 다양하기 때문에 직접 수치로 크기를 지정하면 이에 대응하는 화면을 개발할 수 없기 때문에, wrap_content, match_parent 등이— 단어로 크기를 지정하여 내부적으로 스마트폰 크기에 대응하는 적절한 뷰 크기를 동적으로 계산하게 하는 방법을 사용한다.

▶ margin, padding: margin은 뷰와 뷰 사이의 간격을 지정하는 속성이며, padding은 뷰 내부에서 내용과 뷰의 테두리 간의 간격을 지정하는 속성이다. margin, padding 속성을 이용하면 네 방향 간격 모두 같은 크기로 설정되며, 단일 방향 속성으로는 layout_marginleft, layout_marginRight, layout_marginTop, layout_marginBotton / paddingLeft, paddingRight, paddingTop, paddingBottom이 있다.

▶ clickable: 모든 뷰는 기본적으로 사용자의 클릭 이벤트나 롱클릭 이벤트에 반응한다. 대표적인 사용자 이벤트 뷰인 버튼은 clickable 속성을 지정하지 않아도 기본으로 클릭이나 롱클릭 이벤트를 처리할 수 있다. 하지만 문자열을 출력하는 TextView나 이미지를 출력하는 ImageView는 clickable 속성을 명시적으로 지정(true)하지 않으면 이벤트에 반응하지 않는다. 결국 clickable 속성은 TextView, ImageView 등의 클릭 이벤트가 발생하게 하는 속성이다.

▶ visibility: 화면에 출력할지 지정하는 속성. 기본 값은 true이다. invisible은 크기를 확보하면서 화면에 출력되지 않는 것이고(빈 공간으로 나타남), gone은 크기를 확보하지 못하면서 화면에 출력되지 않는 것이다(있었던 곳이 빈

공간조차 되지 않음). visibility는 초기에 특정 뷰를 화면에 출력하지 않으려는 의도로 사용하고, 특정 순간에 동적으로 화면을 출력할 수 있다. 이때 코드에서는 setVisibility()메서드를 사용하여 뷰의 표시 상태를 조정한다.

→ .setVisibility(int visibility): View.VISIBLE, View.INVISIBLE, View.GONE

1) 자바 파일에서 xml 파일의 버든 id를 참조해서 기능을 추가

```
package com.example.practicexml;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity2 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        Button button1 = findViewById(R.id.버튼1);
        Button button2 = findViewById(R.id.버튼2);

        button2.setOnClickListener(new View.OnClickListener() {
            int num = 0;
            @Override
            public void onClick(View v) {
                if (num == 0) {
                    button1.setVisibility(View.INVISIBLE);
                    num = 1;
                } else if (num == 1) {
                    button1.setVisibility(View.VISIBLE);
                    num = 0;
                }
            }});
        }

    }
```

2. xml에서 직접 버든에 onClick 속성을 부여해주는 방법

```java
package com.example.practicexml;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity2 extends AppCompatActivity {

    Button button1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        button1 = findViewById(R.id.버튼1);

    }

    int num = 0;

    public void clickButton2(View v) {
        if (num == 0) {
            button1.setVisibility(View.INVISIBLE);
            num = 1;
        } else if (num == 1) {
            button1.setVisibility(View.VISIBLE);
            num = 0;
        }
    }

}
```

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

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼1"
        android:id="@+id/버튼1"
        android:layout_marginRight="5dp"
        android:layout_marginLeft="5dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="버튼2"
        android:id="@+id/버튼2"
        android:layout_marginLeft="5dp"
        android:onClick="clickButton2" />

</LinearLayout>
```

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

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="VISIBLE TRUE"
        android:gravity="center"
        android:id="@+id/button1"
        android:onClick="clickButton" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="hello world"
        android:background="#FFCCFF"
        android:id="@+id/text"
        android:visibility="invisible"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="VISIBLE FALSE"
        android:gravity="center"
        android:id="@+id/button2"
        android:onClick="clickButton" />

</LinearLayout>
```

```java
package com.example.visibilitypractice;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button button1;
    Button button2;
    TextView text;

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

        button1 = findViewById(R.id.button1);
        button2 = findViewById(R.id.button2);
        text = findViewById(R.id.text);
    }

    public void clickButton(View v) {
        if (v == button1) {
            text.setVisibility(View.VISIBLE);
        } else if (v == button2) {
            text.setVisibility(View.INVISIBLE);
        }
    }

}
```

Resources: 깡쌤의 안드로이드 프로그래밍