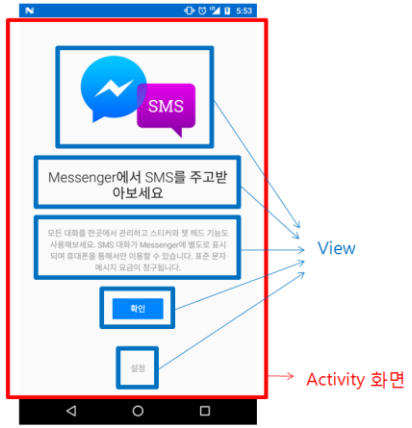
3장. 사용자 인터페이스

3.1 UI 기본 구조

3.1.1. 액티비티-뷰 구조

- 액티비티 자체는 앱의 실행 단위인 컴포넌트
- 액티비티에 버튼, 문자열, 이미지 등을 출력
- 액티비티 내에서 화면 구성을 위한 적절한 뷰 클래스를 출력

- public void setContentView(View view)
- public void setContentView(int layoutResID)



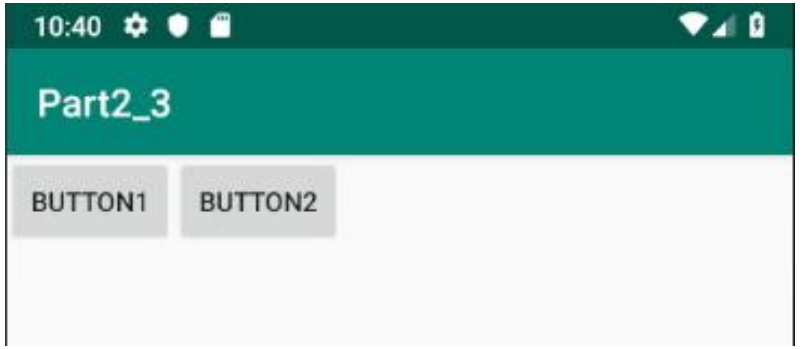
3.1.2. UI 프로그램 작성 방법 – 자바 코드 VS 레이아웃 XML

자바 코드로 화면 구성
레이아웃 XML 파일 자체를 만들지 않고 자바 코드로 모든 뷰를 직접 생성하며, 메서드를 이용하여 뷰 설정을 일일이 지정

```
LinearLayout linear=new LinearLayout(this);
Button bt=new Button(this);
bt.setText("Button 1");
linear.addView(bt);
Button bt2=new Button(this);
bt2.setText("Button 2");
linear.addView(bt2);
setContentView(linear);
```

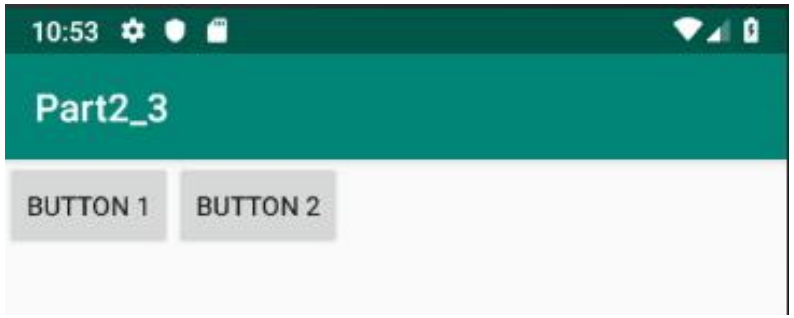
Step by Step 3-1 – 자바 코드로 화면 구성하기

- 모듈 생성
- MainActivity.java 작성
- 앱 실행



Step by Step 3-2 – 레이아웃 XML로 화면 구성하기

- 액티비티 추가
- activity_lab3_2.xml 작성
- 앱 실행



3.1 UI 기본 구조

3.1.3. 뷰의 기초 속성

id 속성

- 뷰의 식별자 속성, 필수 속성은 아니며 필요할 때 추가
- 지정한 id 값은 R.java 파일에 등록
- XML에서 등록한 id 값을 매개변수로 하여 findViewById( ) 함수로 획득

```
<TextView
    android:id="@+id/myText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="hello"/>
```

```
public static final class id {
    ...
    public static final int myText=0x7f0b0059;
    ...
}
```

```
TextView myTextView=findViewById(R.id.myText);
```

3.1 UI 기본 구조

layout_width, layout_height

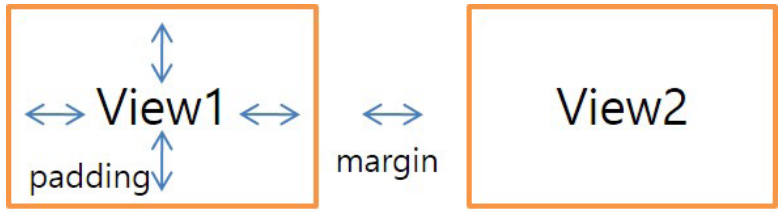
- 뷰의 크기를 지정하기 위한 속성
- match_parent, fill_parent, wrap_content, 100px
- match_parent와 fill_parent는 의미상 동일, 뷰의 크기를 부모 계층의 뷰가 지정한 크기에 꽉 들어차게 자동으로 결정
- wrap_content는 해당 뷰의 내용을 화면에 보이기 위한 적절한 크기를 계산해서 결정

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_lab3_2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#F7FB08">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="hello"
        android:background="#FF0000"
        android:textColor="#FFFFFF"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="world"
        android:background="#0000FF"
        android:textColor="#FFFFFF"/>
</LinearLayout>
```

3.1 UI 기본 구조

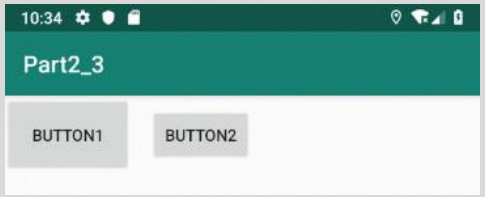
margin, padding 속성

- margin은 뷰와 뷰 사이의 간격을 지정하는 속성이며, padding은 뷰 내부에서 내용과 뷰의 테두리 간 의 간격을 지정하는 속성



단일 방향 margin 속성
: layout_marginLeft, layout_marginRight, layout_marginTop, layout_marginBottom
단일 방향 padding 속성
: paddingLeft, paddingRight, paddingTop, paddingBottom

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button1"
    android:padding="24dp" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button2"
    android:layout_marginLeft="16dp" />
```



3.1 UI 기본 구조

clickable 속성
- clickable 속 성은 TextView, ImageView 등에 클릭 이벤트가 발생하게 하는 속성

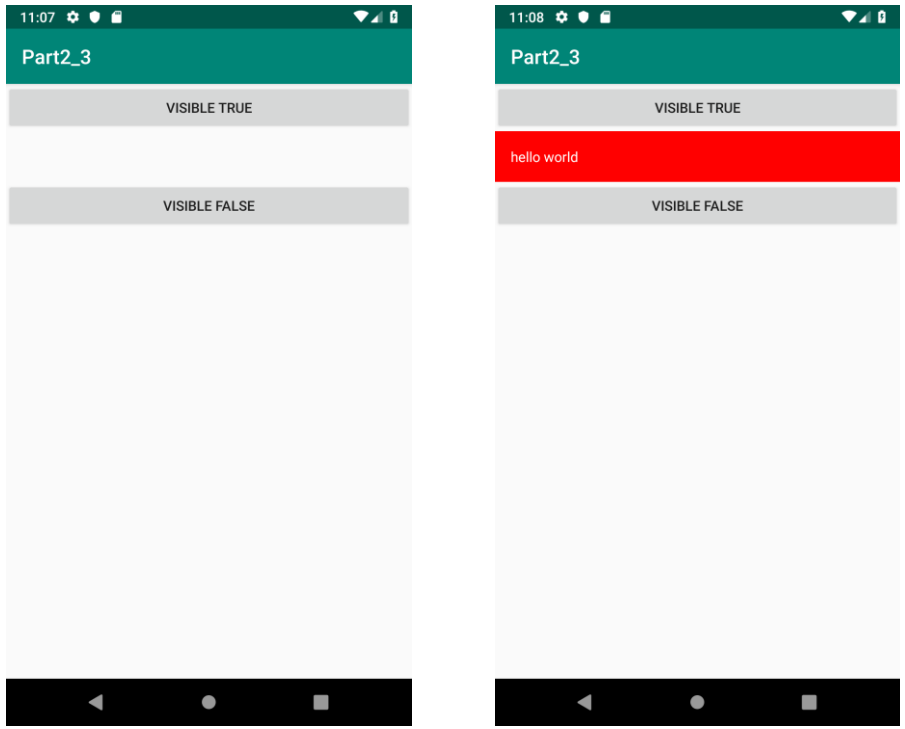
visibility 속성
- 기본값은 “true”이며, “invisible”, “gone” 으로 지정하여 화면에 안 보이게 함


```
public void onClick(View v) {

if(v==trueBtn){
targetBtn.setVisibility(View.VISIBLE);
}else if(v==fasleBtn){
targetBtn.setVisibility(View.INVISIBLE);
}
}
```

Step by Step 3-3 – 뷰 기초 속성 활용

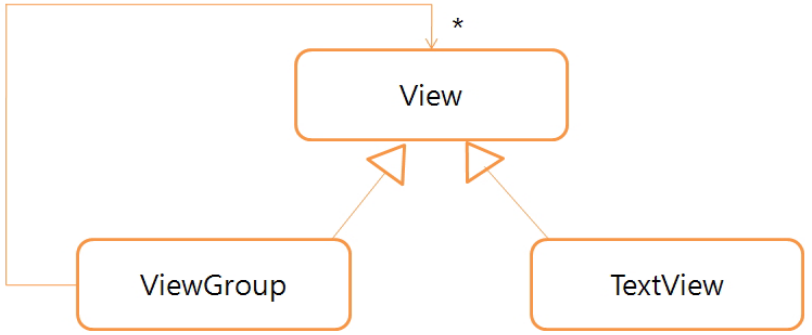
- 액티비티 추가
- activity_lab3_3.xml 작성
- Lab3_3Activity.java 작성
- 앱 실행



3.2 뷰 아키텍처

3.2.1. 뷰의 계층구조

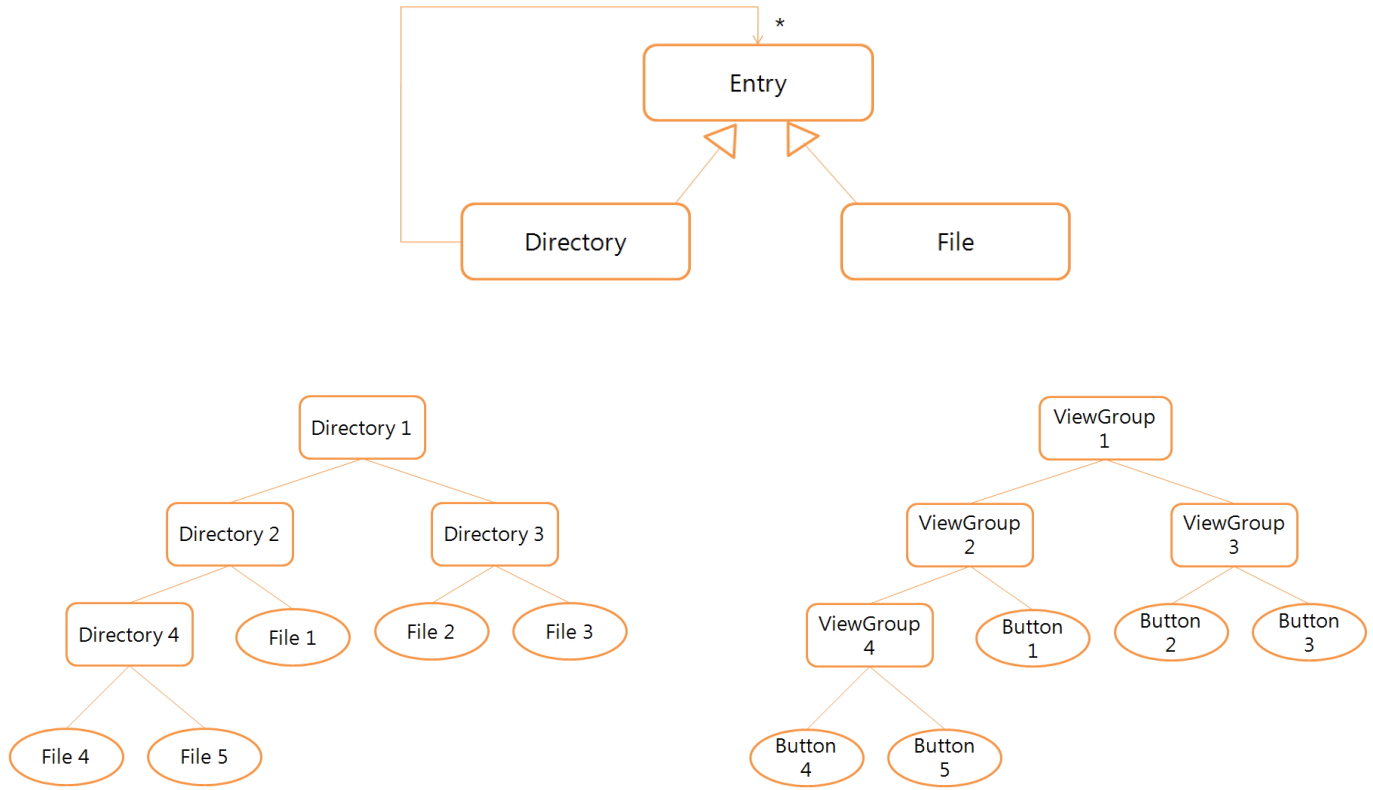
- DOM(Document Object Model)을 따르고 있고, 패턴(Pattern)으로 이야 기하자면 Gof 디자인 패턴의 Composite 패턴이 적용된 구조



- View: 안드로이드 뷰 클래스의 최상위 클래스
- ViewGroup: 다른 뷰 (Button 같은) 여러 개를 뷰그룹에 포함(Add)하여 한꺼번에 제어하기 위한 목적
- TextView: 특정 UI를 출력할 목적으로 제공되는 클래스

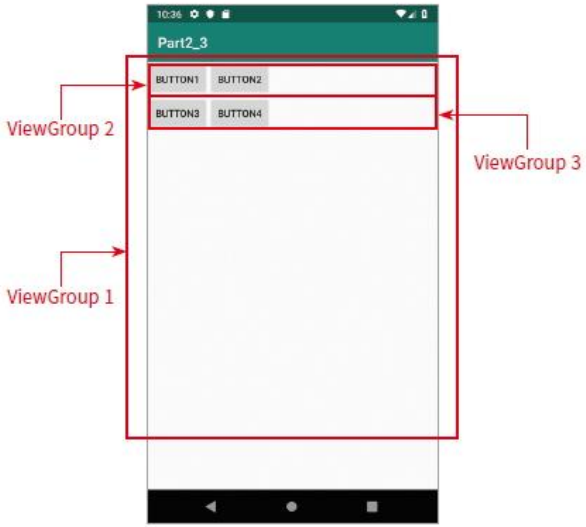
3.2 뷰 아키텍처

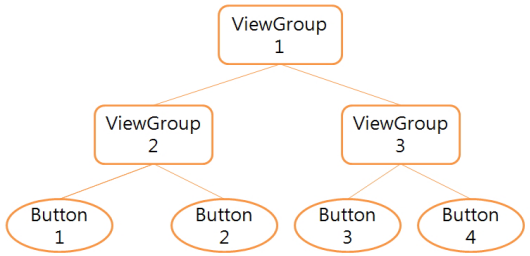
윈도우 탐색 창의 파일과 디렉터리의 계층구조



3.2 뷰 아키텍처

뷰 클래스의 객체들을 계층구조로 묶어서 사용





3.2.2. 뷰 계층구조 구현

- 레이아웃 XML로 계층구조 구현

```
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="horizontal">
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Button 1"/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Button 2"/>
</LinearLayout>
```

3.2 뷰 아키텍처

- 자바 코드로 계층구조 구현

```
LinearLayout linearLayout=new LinearLayout(this);
Button bt1=new Button(this);
linearLayout.addView(bt1);
Button bt2=new Button(this);
linearLayout.addView(bt2);
```

3.3 기초 뷰 활용

3.3.1. TextView

- 대입된 문자열을 화면에 출력
- android:text 화면에 출력할 문자열을 지정
- android:typeface 화면에 출력할 문자열의 폰트를 지정

```
TextView textView=findViewById(R.id.text_font);
Typeface typeface=Typeface.createFromAsset(getAssets(), "xmas.ttf");
textView.setTypeface(typeface);
```

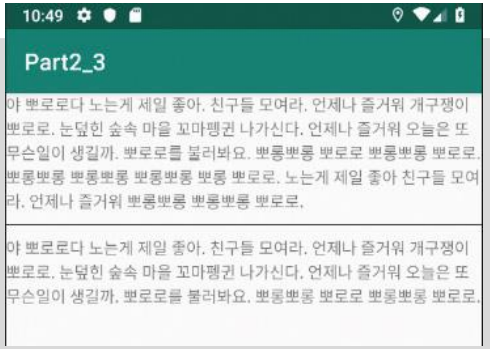
- android:textStyle
- android:textColor
- android:textSize
- android:autoLink 문자열 내에 autoLink 값에 해당하는 URL 문자열이 자동 링크 형태로 출력

```
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="가나다라 http://www.google.com 마바사 a@a.com 아자차카타 1234-5678"
android:autoLink="web|email|phone"/>
```

3.3 기초 뷰 활용

- android:maxLines 특정 줄만큼만 출력

```
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/long_text"/>
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/long_text"
android:maxLines="3"/>
```



- ellipsize 줄임 표시(...), 속성값으로 end, start, middle 등을 지정

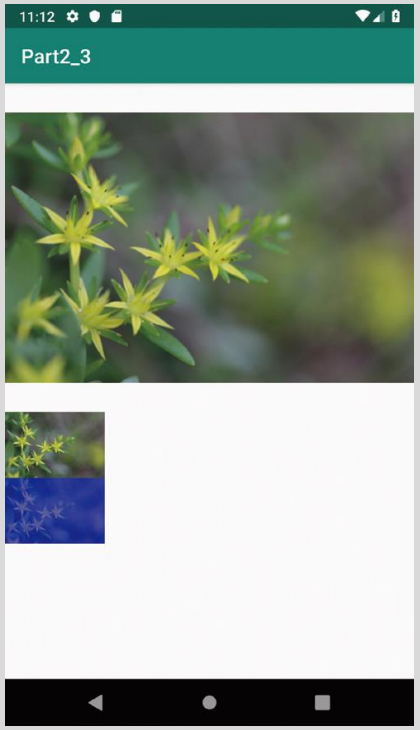
```
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/long_text"
android:ellipsize="end"
android:maxLines="3"
/>
```

3.3 기초 뷰 활용

3.3.2. ImageView

- 화면에 이미지를 출력하고자 할 때 사용하는 뷰
- src
- maxWidth, maxHeight 화면에 출력할 이미지의 최대 크기를 지정
- adjustViewBounds 이미지의 크기를 변경할 때 가로세로 비율을 유지
- tint 이미지 위에 다른 색상을 입힐 때 사용하는 속성

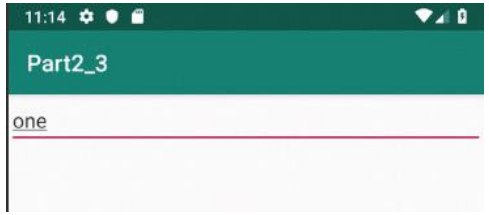
```
<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/sample"/>
<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/sample"
android:maxWidth="100dp"
android:maxHeight="100dp"
android:adjustViewBounds="true"/>
<ImageView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/sample"
android:maxWidth="100dp"
android:maxHeight="100dp"
android:tint="#800000ff"
android:adjustViewBounds="true"/>
```

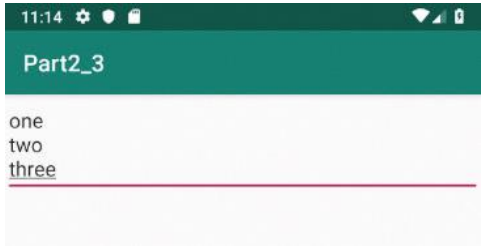


3.3 기초 뷰 활용

3.3.3. EditText

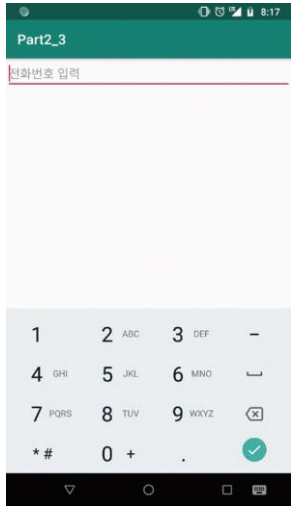
- 사용자에게 데이터를 입력받을 때 사용하는 뷰
- 키보드의 엔터 키를 눌러 개행하면, 입력 창이 여러 줄로 자동으로 늘어남.





- lines 처음 화면에 보일 때부터 특정 줄만큼 보이게 할 때 사용하는 속성
- maxLines 처음 화면에 보일 때는 한 줄 입력 크 기로 보이다가 최대 늘어나는 라인수 지정
- inputType 키보드의 모드를 제어, 한 줄 혹은 여러 줄 입력을 강제, phone, number, textEmailAddress 등을 지정
- gravity 글의 위치를 지정

```
<EditText
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:inputType="phone"
android:hint="전화번호 입력"/>
```



3.3 기초 뷰 활용

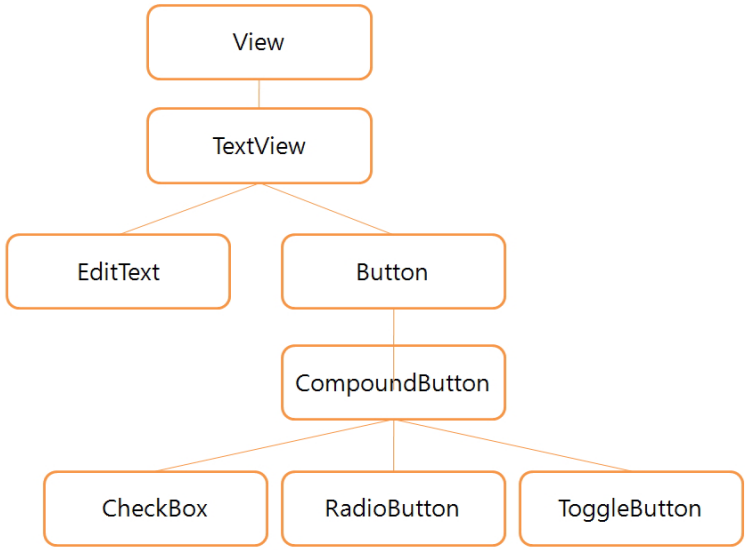
속성값	설명
none	inputType이 지정 안 된 상태. 모든 문자 입력 가능하며 줄 바꿈 가능
text	none 속성과 동일. 단, 줄 바꿈만 불가능
textCapCharacters	키보드가 자동 대문자 입력 모드
textCapWords	각 단어의 첫 글자 입력 시 키보드가 자동 대문자 입력 모드
textCapSentences	각 문자의 첫 글자 입력 시 키보드가 자동 대문자 입력 모드
textMultiLine	여러 줄 입력 가능
textNoSuggestions	단어 입력 시 키보드의 추천 단어 보여주기 비활성
textUri	키보드의 URL 입력 모드
textEmailAddress	키보드의 이메일 주소 입력 모드
textPassword	비밀번호 입력용으로 입력된 문자가 점으로 표시. 키보드는 영문자와 숫자, 특수 키만 표시
textVisiblePassword	textPassword와 동일. 단, 입력된 문자열 표시
number	키보드의 숫자 입력 모드
numberSigned	number와 동일. 단, 부호 키인 – 입력 가능
numberDecimal	number와 동일. 단, 소수점 입력 가능
numberPassword	숫자 키만 입력 가능. 단, 입력된 문자는 점으로 표시
phone	전화번호 입력 모드
datetime	날짜와 시간을 입력하기 위한 /, : 키 제공
date	날짜를 입력하기 위한 / 키 제공
time	시간을 입력하기 위한 : 키 제공

3.3 기초 뷰 활용

3.3.4. Button

- Button 하위 클래스로 CheckBox, RadioButton, ToggleButton

```
<Button
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="click me"/>
```



3.3 기초 뷰 활용

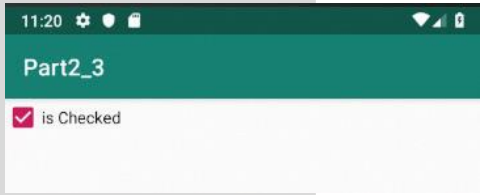
3.3.5. Checkbox, RadioButton

Checkbox

```
<CheckBox
android:id="@+id/checkbox"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="is unChecked"/>
```

- isChecked( ): 해당 Checkbox가 체크된 상태인지를 반환
- setChecked( ): 체크 상태를 바꾸기 위한 함수
- toggle( ): 반대로 상태 변경

```
checkBox=findViewById(R.id.checkbox);
checkBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(isChecked){
            checkBox.setText("is Checked");
        }else {
            checkBox.setText("is unChecked");
        }
    }
});
```



3.3 기초 뷰 활용

RadioButton

- RadioGroup으로 묶인 RadioButton 중 하나만 체크

```
<RadioGroup
android:id="@+id/radioGroud"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
<RadioButton
android:id="@+id/radio1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="radio 1"/>
<RadioButton
android:id="@+id/radio2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="radio 2"/>
</RadioGroup>
```

- check( ): 해당 RadioButton이 체크
- clearCheck( ): 체크 상태를 해제
- getCheckedRadioButtonId( ): 체크된 RadioButton의 id 값을 획득

Step by Step 3-4 – TextView 활용

- 액티비티 추가
- assets 폴더 생성
- ttf 파일 준비
- Strings.xml 추가
- 이미지 리소스 복사
- activity_lab3_4.xml 작성
- Lab3_4Activity.java 작성
- Lab3_4Activity.java 실행