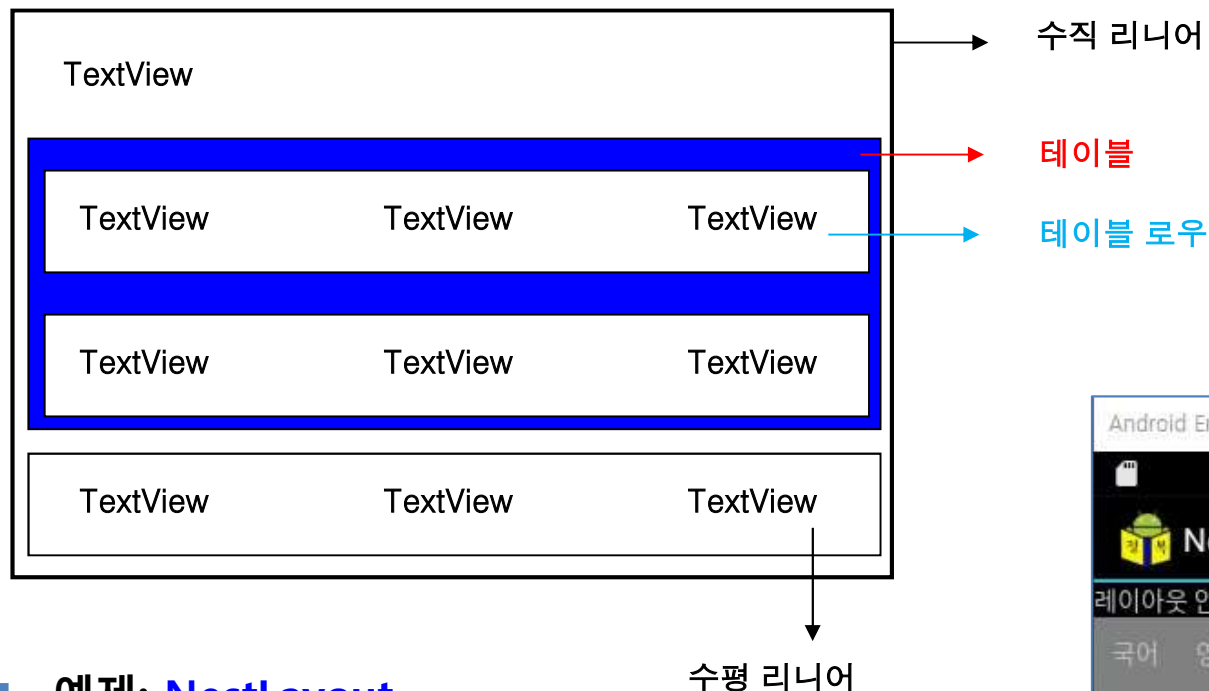


레이아웃 관리

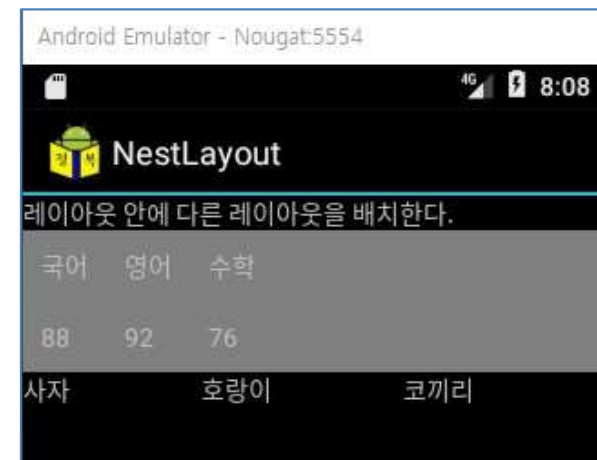
성신여자대학교 컴퓨터공학과
우종정 교수

레이아웃 중첩

- 레이아웃 자체는 단순하지만 조합하면 다양하게 응용
- 레이아웃은 기능적으로는 뷰의 부모이면서 문법적으로 뷰의 자식 클래스이다. 따라서 뷰끼리 중첩 가능하다.



- 예제: [NestLayout](#)

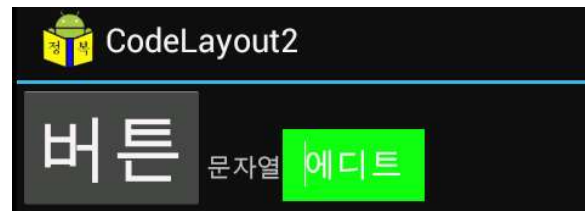
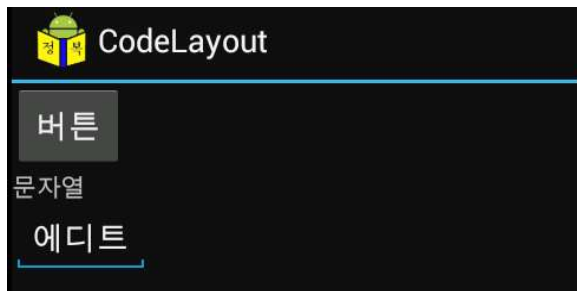


레이아웃 속성 변경

- 디자인 타임에 결정할 수 없는 속성은 동적으로 조정
- **LinearLayout의 메서드 예**
 - orientation : getOrientation, setOrientation
 - 속성값은 대문자 상수로 정의되어 있다. 예를 들면 HORIZONTAL, VERTICAL
- **TextView의 속성 변경 메서드**
 - void setGravity(int gravity);
 - void setText (CharSequence text)
 - void setTextColor (int color)
 - void setTextSize (float size)
- 여러 타입의 인수에 대해 오버로딩되어 있다.
- XML 리소스에 지정하지 않은 속성이라도 실행 중에 지정할 수 있다.

레이아웃 속성 변경

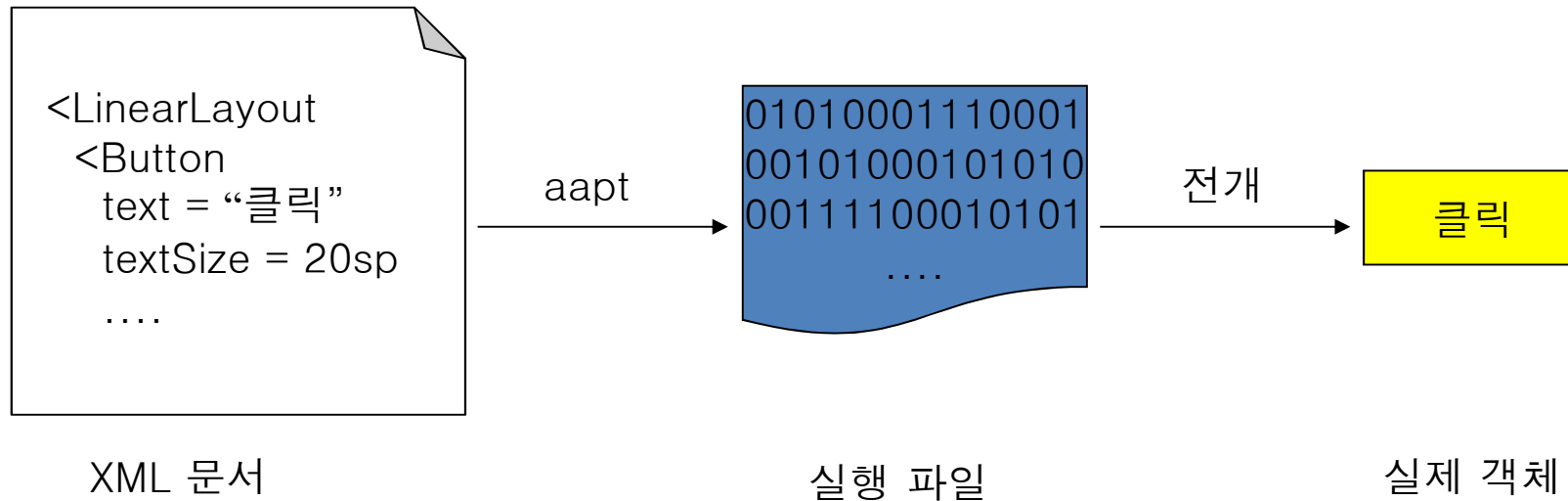
- 코드에서 위젯을 참조하려면 id가 지정되어 있어야 한다.
- 위젯 객체를 찾기 위한 메서드
 - `public final T Activity.findViewById (int id)`
- 객체를 찾은 후 모든 메서드를 호출할 수 있다.
- 코드의 뷰와 XML의 뷰 엘리먼트 차이점 :
 - 코드에서 크기는 픽셀 단위, XML에서 크기는 다양한 단위
 - 코드의 색상 알파값은 투명(0x00), XML의 색상 알파값은 불투명(0xff)



- 예제: [CodeLayout2, layout](#)

레이아웃 전개

- XML 리소스는 aapt에 의해 컴파일되어 이진 형태로 실행 파일에 내장된다.



- **전개(inflation)** : XML 문서의 레이아웃 혹은 뷰로부터 뷰 객체를 생성하는 동작
- **setContentView** : XML을 전개하거나 혹은 XML로 정의된 레이아웃을 전개하여 액티비티를 채운다.
- 예제 : [Inflation](#), [layout](#)

레이아웃 전개

■ ViewGroup의 `addView()` 메서드는 뷰를 부모 레이아웃과 연결

- `void addView (View child)`
- `void addView (View child, int index)`
- `void addView (View child, ViewGroup.LayoutParams params)`

■ 뷰를 액티비티에 등록(뷰 내용을 액티비티에 채우기)

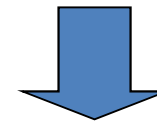
- `public void setContentView (int layoutResID)`
- `public void setContentView (View view [,
ViewGroup.LayoutParams params])`

■ 예제 : [Inflation2](#)

레이아웃 전개

- 리소스 ID로부터 전개하는 부분만 따로 수행 가능
- `inflate()` 메서드 이용하여 리소스를 뷰 객체로 전개
 - 레이아웃 전개자의 멤버 메서드 `inflate(int resource, ViewGroup root)`
 - View의 정적 메서드 `public static View inflate(Context context, int resource, ViewGroup root)`
- 레이아웃 전개자 획득
 - `(LayoutInflater)getSystemService(Context.LAYOUT_INFLATER_SERVICE)`
 - LayoutInflater의 정적 메서드 `static LayoutInflater from(Context)`
- 예제: [Inflation3](#), [Inflation4 mytext](#)

```
TextView text = new TextView(this);  
text.setText("TextView");  
text.setGravity(Gravity.CENTER);  
text.setTextColor(Color.RED);  
text.setTextSize(20);
```



```
TextView text = (TextView)View.inflate(this, R.layout.mytext, null);
```

레이아웃 전개

- 전개의 이유 : 특정 뷰 그룹만 따로 전개하기 위해
- 예제 : 다음과 같이 동적으로 뷰 그룹을 생성하여 배치하려면
 - [Inflation5](#), [layout](#), [newmessage](#)



레이아웃 파라미터

■ 소개

- 이름이 layout_로 시작되는 속성
- 일반 속성에 비해 ① 소속, ② 적용 시점, ③ 변경 방법이 다르다.
- 레이아웃 파라미터는 뷰 자체의 성질이 아니라 뷰 외부와의 관계를 지정하기 때문에 위젯에 종속적인 일반 속성과 달리 레이아웃 파라미터는 소속 레이아웃에 종속적
- 소속

<TextView
일반 속성
text="..."

<ImageView
일반 속성
src="..."

<LinearLayout
<TextView
레이아웃 파라미터
layout_weight="..."

<RelativeLayout
<ImageView
레이아웃 파라미터
layout_above="..."

소속된 위젯에 따라 달라짐

소속된 레이아웃에 따라 달라짐

- 적용 시점: 일반 속성은 위젯 생성 직후 적용, Layout 속성은 뷰가 레이아웃에 배치될 때 적용
- 변경 방법: 일반 속성은 위젯에 직접 적용하지만, Layout 속성은 레이아웃 파라미터 클래스의 객체를 인수로 적용

레이아웃 파라미터

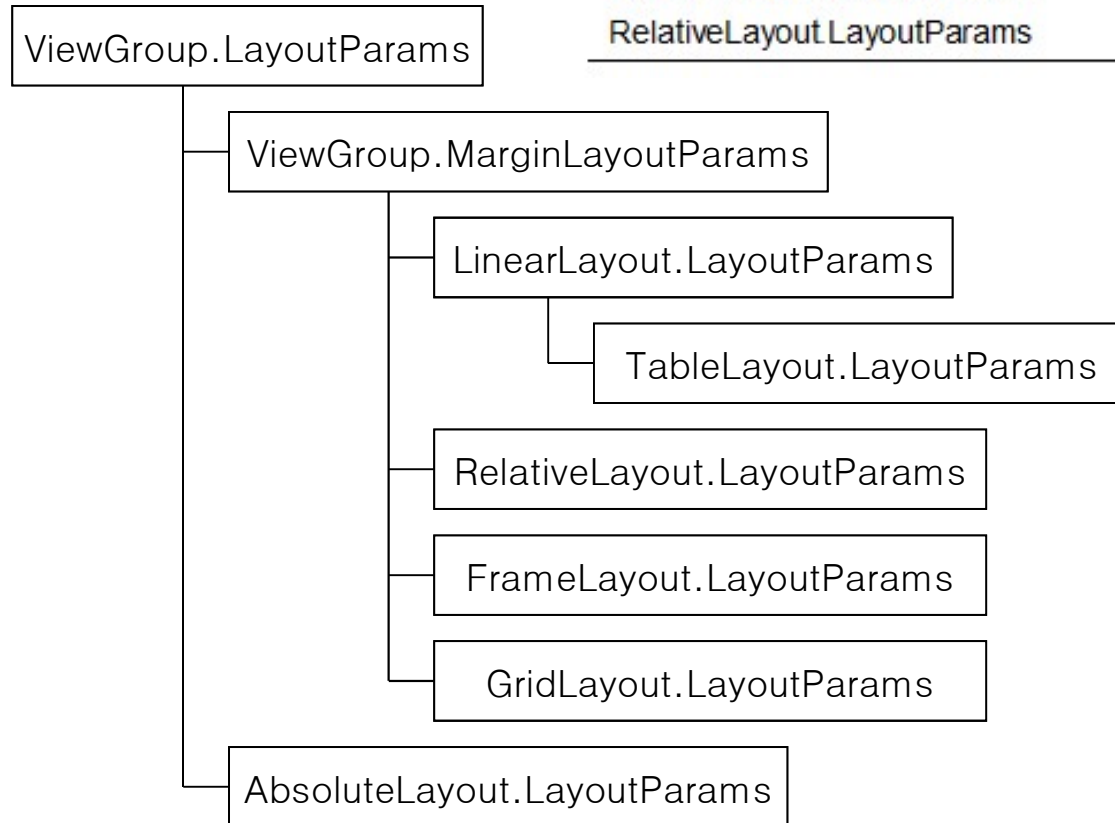
■ 소개

- 일반 속성의 경우 오류. 소속이 다른 레이아웃 파라미터는 무시 => 최근 안드로이드 스튜디오에서?
 - <LinearLayout>
 - <TextView
 - android:layout_alignParentRight="true" ✕
- 레이아웃 파라미터 변경을 위한 메서드는 위젯에 정의되어 있지 않음.
 - TextView.setLayoutWidth() ✕
 - TextView.setLayoutWeight() ✕
 - TextView.setLayoutX()
- 부모에 추가될 때 사용되므로 addView 혹은 setContentView의 인수로 전달. 생략 시 디폴트가 적용.
 - void addView (View child [, ViewGroup.LayoutParams params])
 - void setContentView (View view [, ViewGroup.LayoutParams params])

레이아웃 파라미터

■ 종류 및 계층 구조

| 레이아웃 | 파라미터 |
|------------------------------|---------------------------------------|
| ViewGroup.LayoutParams | layout_width, layout_height |
| ViewGroup.MarginLayoutParams | layout_marginLeft, layout_marginRight |
| LinearLayout.LayoutParams | layout_gravity, layout_weight |
| AbsoluteLayout.LayoutParams | layout_x, layout_y |
| RelativeLayout.LayoutParams | layout_above, layout_alignParentRight |



레이아웃 파라미터

■ 예제

- 리니어에 TextView를 wrap_content, wrap_content으로 중앙에 배치
- [LayoutParameter](#), [layout](#)
- 코드로 배치하되 레이아웃 파라미터를 생략하면 디폴트가 적용
- 디폴트 높이: wrap_content
- 디폴트 넓이: match_parent
- [LayoutParameter2](#)
- 레이아웃 파라미터를 지정
- [LayoutParameter3](#)

