07장. 레이아웃을 이용한 사용자 인터페이스 설계

01. 안드로이드에서 UI 만들기
02. ViewGroup을 이용한 UI 조직화
03. 내장 레이아웃 클래스 활용
04. 뷰 컨테이너 클래스 활용

컴퓨터공학과 변영철 교수

XML을 이용한 레이아웃 작성

- 직접 코드로 작성하기 보다는 XML 파일에서 UI를 작성하면 코드가 깔끔하고 설계 과정이 간단
- 직관적이며 조직화하기에 좋음

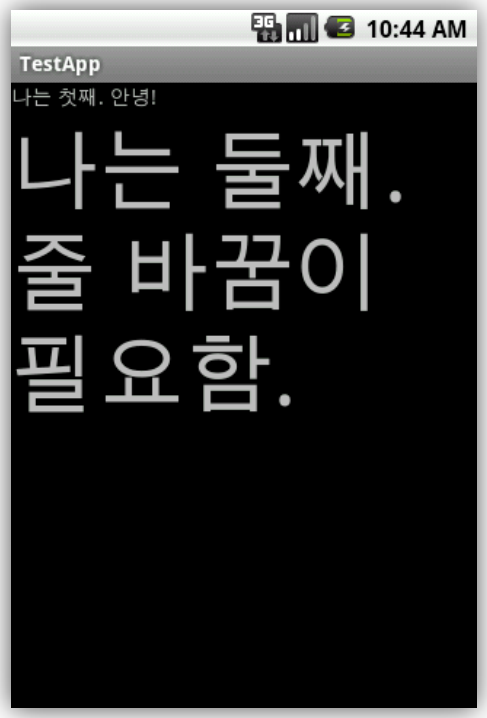
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent”>

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content“
        android:text="@string/hello“ />

</LinearLayout>
```

자동으로 생성되는 XML 레이아웃 리소스

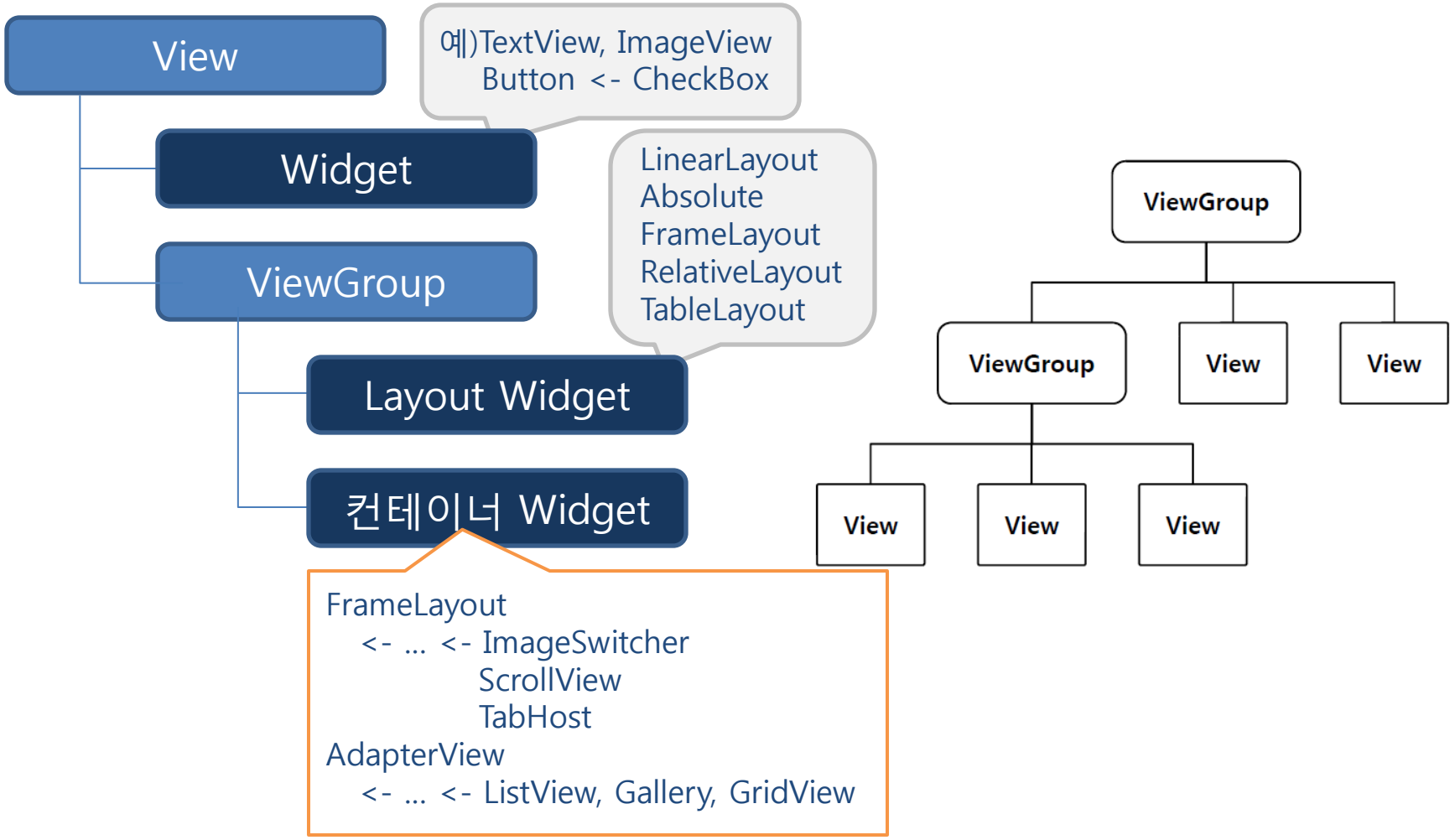
코드에서 레이아웃 작성의 예

- 코드에서 레이아웃을 생성하면 실수를 저지르기 쉽고 코드를 유지보수하기도 어려움 -> 바람직하지 않음

```
public void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);

        TextView text1 = new TextView(this);
        text1.setText("안녕!");

        TextView text2 = new TextView(this);
        text2.setText("나는 둘째. 줄 바꿈이 필요함.");
        text2.setTextSize((float) 60);

        LinearLayout ll = new LinearLayout(this);
        ll.setOrientation(LinearLayout.VERTICAL);

        ll.addView(text1);
        ll.addView(text2);
        setContentView(ll);
}
```



뷰그룹, 레이아웃, 컨테이너



ViewGroup

- ViewGroup
  - 위젯들을 담을 수 있음
  - LinearLayout 같은 레이아웃 위젯들의 부모 클래스
- 두 가지 유형의 ViewGroup 클래스
  - 레이아웃 위젯 클래스
    - ViewGroup 클래스 직계 하위 클래스들(이름이 Layout으로 끝남)
    - LinearLayout, RelativeLayout, AbsoluteLayout 등
  - 뷰 컨테이너 위젯 클래스
    - ViewGroup 클래스의 간접 하위 클래스들
    - 레이아웃 클래스처럼 뷰 객체 컨테이너 역할을 하되, 자기 자신도 보통의 위젯처럼 사용자와 상호작용
    - 클래스 이름에 공통성이 없으며, 각자 제공하는 기능에 따라 명명
    - Gallery, GridView, ImageSwitcher, ScrollView, TabHost, ListView 등

Hierarchy Viewer

- 현재 실행 중인 응용 프로그램 뷰 객체들의 부모-자식 관계를 살펴볼 수 있음
- 특정 뷰 객체를 선택하면 그 객체의 현재 속성을 볼 수 있음
- 뷰 위젯의 표시 관련 문제를 디버깅할 때 유용함: 화면에 제대로 그려지지 않거나 뷰가 통째로 나타나지 않을 경우 뷰 객체 속성 점검
- 다른 사람이 만든 응용의 레이아웃과 디스플레이 방식을 분석할 때 유용하며, 응용의 화면 구성을 거의 완전히 파악할 수 있음

공통 특성(ViewGroup 속성)

특성 이름	적용 대상	설명	값
android:layout_height	부모 뷰, 자식 뷰	뷰의 높이. 자식 뷰의 경우 필수임.	크기(dimension) 값 또는 fill_parent 또는 wrap_content
android:layout_width	부모 뷰, 자식 뷰	뷰의 너비. 자식 뷰의 경우 필수임.	크기 값 또는 fill_parent 또는 wrap_content
android:layout_margin	자식 뷰	뷰의 상하좌우의 여분 공간	크기 값
android:layout_marginTop	자식 뷰	뷰의 위쪽 여분 공간	크기 값
android:layout_marginBottom	자식 뷰	뷰의 아래쪽 여분 공간	크기 값
android:layout_marginRight	자식 뷰	뷰의 오른쪽 여분 공간	크기 값
android:layout_marginLeft	자식 뷰	뷰의 왼쪽 여분 공간	크기 값

AbsoluteLayout

- 자식 뷰들의 x, y 좌표를 정확하게 지정하고자 할 때
- 유연성이 떨어짐 : 화면 크기가 다른 기기에서는 레이아웃이 깨질 수 있음

```
<AbsoluteLayout xmlns:android=
        "http://schemas.android.com/apk/res/android"
        android:id="@+id/AbsoluteLayout01"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent">
        <AnalogClock
                android:id="@+id/AnalogClock01"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_x="0px"
                android:layout_y="0px" />
        <DigitalClock
                android:id="@+id/DigitalClock01"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_x="190px"
                android:layout_y="100px" />
</AbsoluteLayout>
```

FrameLayout

- FrameLayout은 여러 자식 뷰들을 겹쳐서(중첩) 그리고자 할 때 사용
- 여러 이미지들을 같은 영역에 표시하고자 할 때 유용

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout android:id="@+id/FrameLayout01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_gravity="center">
        <ImageView android:id="@+id/ImageView01"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/green_rect"
                android:minHeight="200px"
                android:minWidth="200px"></ImageView>
        <ImageView android:id="@+id/ImageView02"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/red_oval"
                android:minHeight="100px"
                android:minWidth="100px"
                android:layout_gravity="center"></ImageView>
</FrameLayout>
```

Layout_gravity :

top, bottom, left, right
center_vertical
fill_vertical
center_horizontal
fill_horizontal
center, fill -> |(or)로 조합

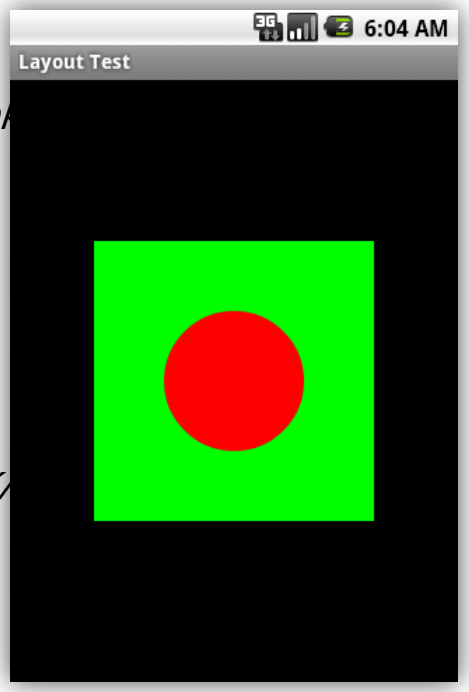
FrameLayout

- green_rect.xml

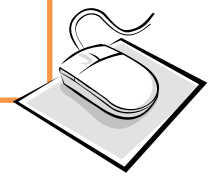
```
<shape xmlns:android="http://schemas.android.com/apk
    android:shape="rectangle">
    <solid android:color="#0f0"/>
</shape>
```

- red_oval.xml

```
<shape mlns:android="http://schemas.android.com/apk/
    android:shape="oval">
    <solid android:color="#f00"/>
</shape>
```



LayoutTest
(FrameLayoutAct)

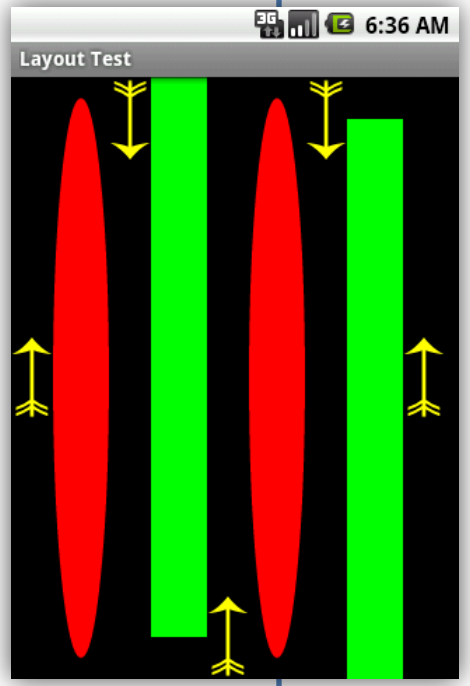


LinearLayout

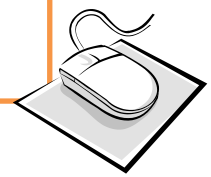
- 위젯(View 객체)들을 한 행 또는 한 열로 차례로 표시
- 입력 양식(form)을 만들 때 편리
- 방향
  - 수평(horizontal), 수직(vertical)
- 필수 속성
  - layout_width, layout_height

LinearLayout

```
<LinearLayout android:orientation="horizontal" android:layout_gravity="center_vertical">
        <ImageView
                android:src="@drawable/arrow" android:layout_gravity="center"></ImageView>
        <ImageView
                android:src="@drawable/red_oval"
                android:minWidth="40px" android:minHeight="400px"
                android:layout_gravity="center_vertical"></ImageView>
        <ImageView
                android:src="@drawable/arrow2"
                android:layout_gravity="top"></ImageView>
        <ImageView
                android:minWidth="40px" android:minHeight="400px"
                android:src="@drawable/green_rect"
                android:layout_gravity="top"></ImageView>
        <ImageView
                android:layout_gravity="bottom" android:src="@drawable/arrow"></ImageView>
        <ImageView
                android:src="@drawable/red_oval"
                android:minWidth="40px" android:minHeight="400px"
                android:layout_gravity="center_vertical"></ImageView>
        <ImageView
                android:src="@drawable/arrow2"></ImageView>
        <ImageView
                android:minWidth="40px" android:minHeight="400px"
                android:src="@drawable/green_rect" android:layout_gravity="bottom"></ImageView>
        <ImageView
                android:src="@drawable/arrow"
                android:layout_gravity="center_vertical"></ImageView>
</LinearLayout>
```



LayoutTest
(LinearLayoutAct)



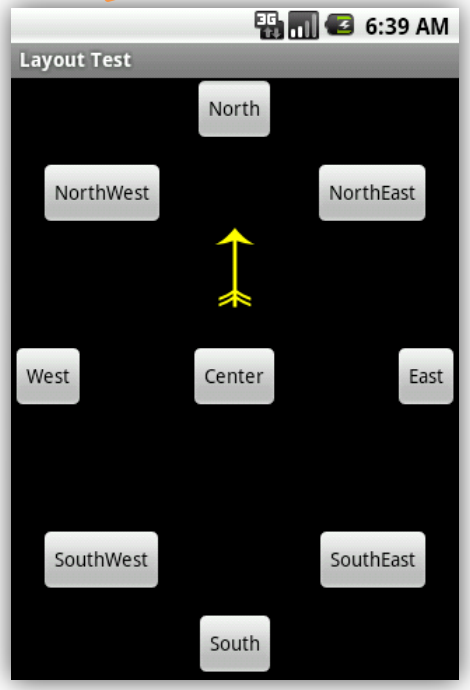
RelativeLayout

- 자식 뷰들을 상대적인 관계에 따라 배치
  - 예) A는 B 아래에, B는 C 왼쪽에
  - 부모 레이아웃 가장자리를 기준으로 설정할 수도 있음
- 주요 속성
  - android:gravity
  - android:layout_centerInParent
  - android:layout_centerHorizontal, _centerVertical
  - android:layout_alignParentTop, _alignParentBottom, _alignParentLeft, _alignParentRight
  - android:layout_alignRight, _alignLeft, _alignTop, _alignBottom
  - android:layout_above, _below
  - android:layout_toLeftOf, _toRightOf

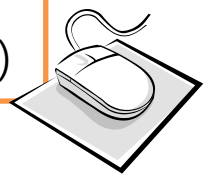
RelativeLayout

```
<RelativeLayout>
    <Button android:text="North"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"></Button>
    <Button android:text="NorthEast"
        android:layout_marginRight="20px"
        android:layout_marginTop="60px"
        android:layout_alignParentRight="true"></Button>
    <Button android:layout_alignParentLeft="true"
        android:text="NorthWest"
        android:layout_marginLeft="20px" android:layout_marginTop="60px"></Button>
    <Button android:text="West"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"></Button>
    <Button android:text="Center"
        android:layout_centerVertical="true"
        android:layout_centerInParent="true"></Button>
    <ImageView android:layout_above="@id/ButtonCenter"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="25px"
        android:src="@drawable/arrow"></ImageView>
    <Button android:text="East"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"></Button>
    <Button android:text="South"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"></Button>
    <Button android:text="SouthEast"
        android:layout_marginRight="20px" android:layout_marginBottom="60px"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"></Button>
    <Button android:text="SouthWest"
        android:layout_marginLeft="20px" android:layout_marginBottom="60px"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"></Button>
</RelativeLayout>
```

Button 객체는 부모에 상대적으로 배치되고, ImageView는 Button과 부모에 상대적으로 배치됨



LayoutTest
(RelativeLayoutAct)



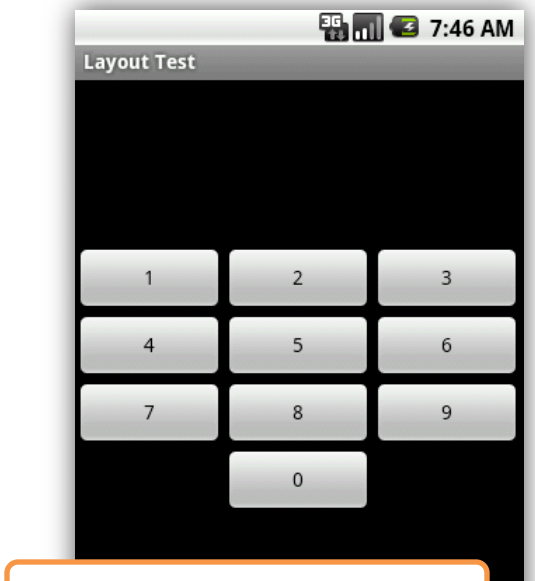
TableLayout

- 자식 뷰들을 표 형태로 배치
- TableRow 레이아웃(기본적으로 수평 방향의 LinearLayout)이 수직으로 여러 개 나열
- TableRow의 각 항목은 뷰 또는 뷰를 담는 레이아웃
- 여러 열들을 하나로 합치는 것도 가능
- 각 열의 너비는 그 열에서 가장 큰 뷰 객체에 맞게 조정
- TableLayout과 자식 뷰에 적용할 수 있는 특성들은 android.widget.TableLayout.LayoutParams 참조
- TableRow와 자식 뷰에 적용할 수 있는 특성들은 android.widget.TableRow.LayoutParams 참조

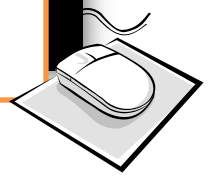
TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    android:id="@+id/TableLayout01"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="*" android:gravity="center_vertical">
    <TableRow android:id="@+id/TableRow01"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent">
        <Button android:id="@+id/Button01" android:text="1"></Button>
        <Button android:id="@+id/Button02" android:text="2"></Button>
        <Button android:id="@+id/Button03" android:text="3"></Button>
    </TableRow>
    <TableRow android:id="@+id/TableRow02"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent">
        <Button android:id="@+id/Button04" android:text="4"></Button>
        <Button android:id="@+id/Button05" android:text="5"></Button>
        <Button android:id="@+id/Button06" android:text="6"></Button>
    </TableRow>
    <TableRow android:id="@+id/TableRow02"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent">
        <Button android:id="@+id/Button07" android:text="7"></Button>
        <Button android:id="@+id/Button08" android:text="8"></Button>
        <Button android:id="@+id/Button09" android:text="9"></Button>
    </TableRow>
    <TableRow android:id="@+id/TableRow02"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent">
        <Button android:id="@+id/Button00" android:text="0" android:layout_column="1"></Button>
    </TableRow>
</TableLayout>
```

TableLayout의 모든 열은 화면 너비에 맞게 늘어나도록 설정됨. 첫 TableRow는 세 개의 열 각각에 Button 객체가 있음. 네 째 TableRow는 Button 하나를 두 번째 열에 명시적으로 배치함.

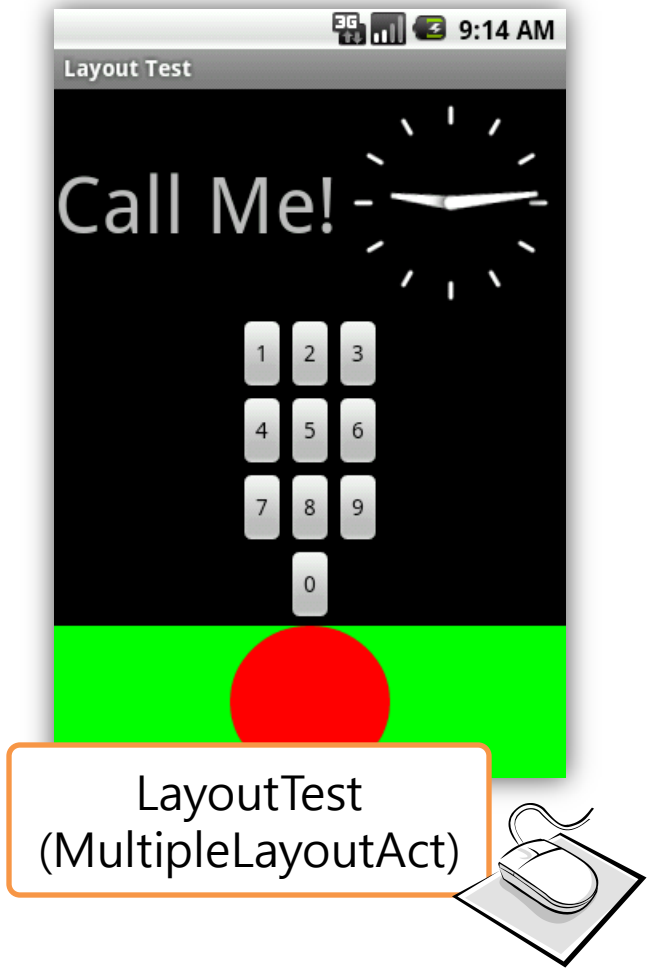


LayoutTest
(TableLayoutAct)



한 화면에 여러 레이아웃 사용하기

- 여러 뷰 객체를 담은 레이아웃을 다른 레이아웃에 담는 것도 가능
- 여러 레이아웃 뷰들을 조합하여 화면을 구성한 예
- LinearLayout, TableLayout, FrameLayout



뷰 컨테이너 위젯

- 레이아웃 한계점
  - 이 화면에 뷰 객체들을 배치하는 데 유용하긴 하지만, 사용자와 상호작용이 없음
- 뷰 컨테이너 특징
  - 다른 뷰 객체들을 담을 수 있을 뿐만 아니라 사용자가 컨테이너에 담긴 뷰 객체들을 표준적인 방식으로 탐색할 수 있게 하는 상호작용 기능도 제공
- 뷰 컨테이너 유형
  - 아답터 뷰(AdapterView)로부터 파생된 자료 주도적 뷰 컨테이너 : GridView, ListView, Gallery 등
  - ViewFlipper, ImageSwitcher, TextSwitcher 등의 전환식 컨테이너
  - 탭 방식의 TabHost와 TabHost를 통한 탭 기능
  - 대화상자들

아답터 뷰

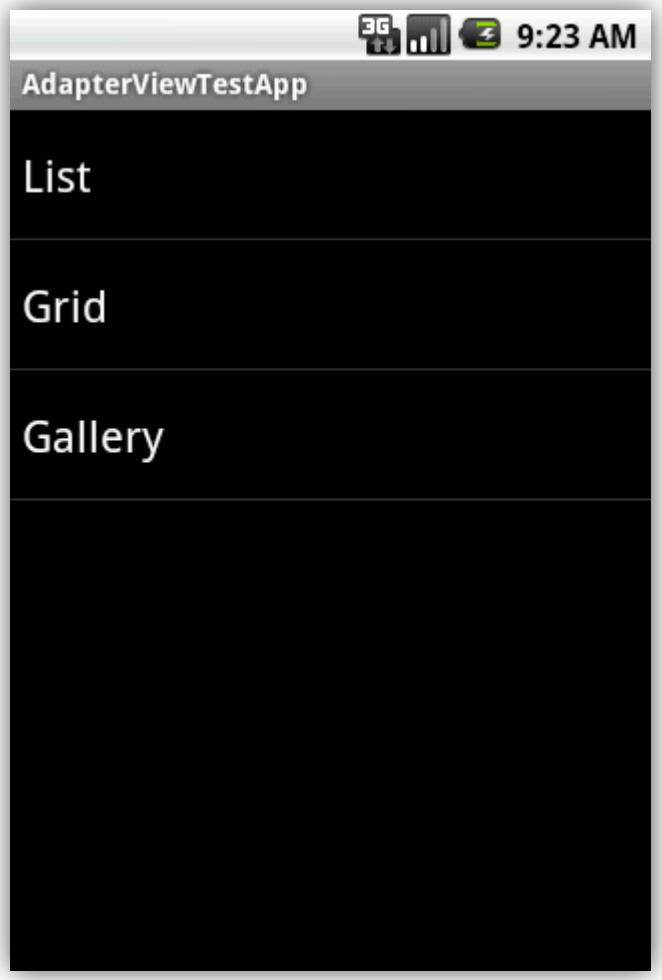
- ListView
  - 뷰 객체들을 수직 방향의 목록 형태로 표시, 수직 스크롤
  - 목록의 각 항목은 자료를 담은 뷰
  - 항목을 선택해서 작업 수행
- GridView
  - 고정된 수의 열로 이루어진 격자(grid) 에 뷰 객체들을 배치
  - 격자의 각 칸에 이미지 아이콘이 배치되는 경우가 많음
  - 항목을 선택해서 작업 수행
- GalleryView
  - 뷰 객체들을 수평 방향의 목록 형태로 표시
  - 이미지들을 표시하는 데 흔히 쓰임이며 수평 스크롤 지원
  - 한 항목을 선택해서 일정한 작업 수행

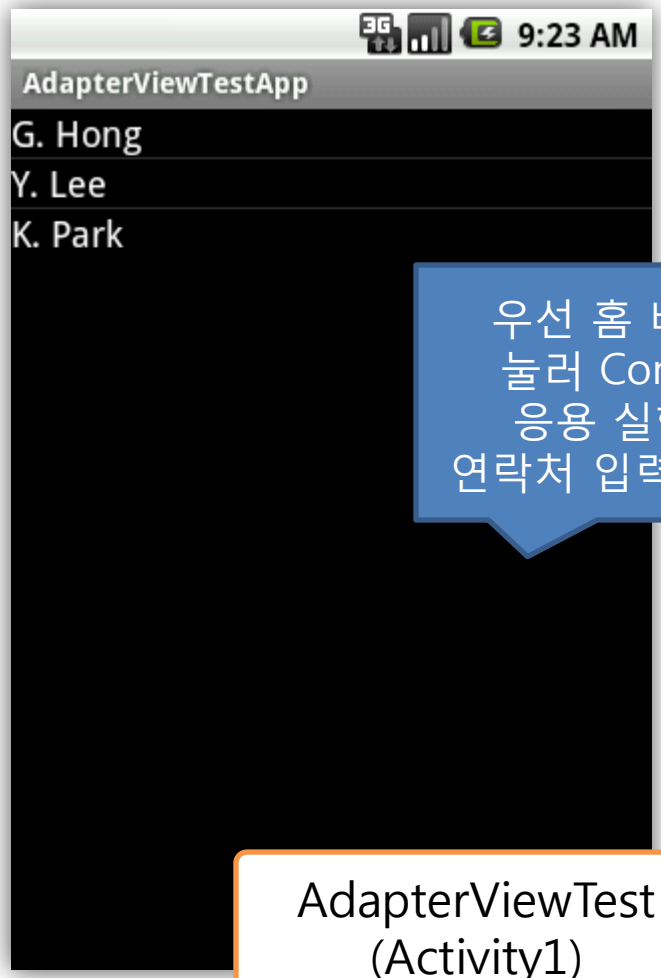
아답터

Known Indirect Subclasses of ListAdapter
ArrayAdapter<T>, BaseAdapter, CursorAdapter, HeaderViewListAdapter, ResourceCursorAdapter, SimpleAdapter, SimpleCursorAdapter, WrapperListAdapter 등

- 아답터
  - 데이터 소스(data source)에서 자료를 가져온 후 지정한 레이아웃으로 자식 뷰들을 만들고 뷰 컨테이너를 채움
  - 가장 흔히 쓰이는 아답터 : ArrayAdapter, CursorAdapter
  - 아답터를 생성할 때에는 레이아웃(식별자)을 지정함
  - 레이아웃은 컨테이너 각 행을 채울 때 하나의 틀로 쓰임
- ArrayAdapter
  - 데이터 소스(표시할 자료)가 배열에 저장되어 있는 경우
- CursorAdapter
  - 데이터 소스가 데이터베이스인 경우
  - 데이터베이스 질의 결과 데이터는 커서로 접근 가능

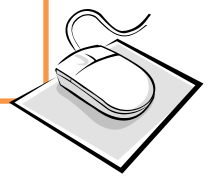
ListView와 CursorAdapter





우선 홈 버튼을 눌러 Contacts 응용 실행 후 연락처 입력해야 함

AdapterViewTest (Activity1)



ListView와 CursorAdapter

- CursorAdapter는 커서에 있는 값들(하나 이상의 열)을 특정 레이아웃으로 뷰 컨터이너를 채우는 아답터
- /res/layout/list.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/list_view" />
</LinearLayout>
```

- onCreate 메소드

```
setContentView(R.layout.list);
```

ListView와 CursorAdapter

- 연락처(Contacts) DB를 다루는 Cursor와 아답터 생성

```
Cursor cursor = getContentResolver().query(People.CONTENT_URI, null, null, null, null);
startManagingCursor(cursor);

SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
      android.R.layout.simple_gallery_item, cursor,
      new String[]{People.NAME}, new int[]{android.R.id.text1});
```

- startManagingCursor 메소드
  - 액티비티가 자신의 생명주기에 맞추어 커서 생명주기를 관리할 수 있도록 설정함. 가령 액티비티가 stop되면 자동적으로 커서의 deactivate가 호출됨.
- 퍼미션 설정(AndroidManifest.xml)

```
<uses-permission
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="android.permission.READ_CONTACTS"></uses-permission>
```

ListView와 CursorAdapter

- CursorAdapter가 제대로 작동하려면 Cursor가 가리키는 질의 결과에 반드시 _id라는 이름의 필드가 존재해야 하며, 이는 연락처 DB에 이미 있음
- SimpleCursorAdapter는 네째 인수(People.Name)로 지정된 열의 자료를 마지막 인수로 지정된 위젯(android.R.id.*text1*)에 연결함

```
ListView lview = (ListView)findViewById(R.id.list_view);
lview.setAdapter(adapter);
```

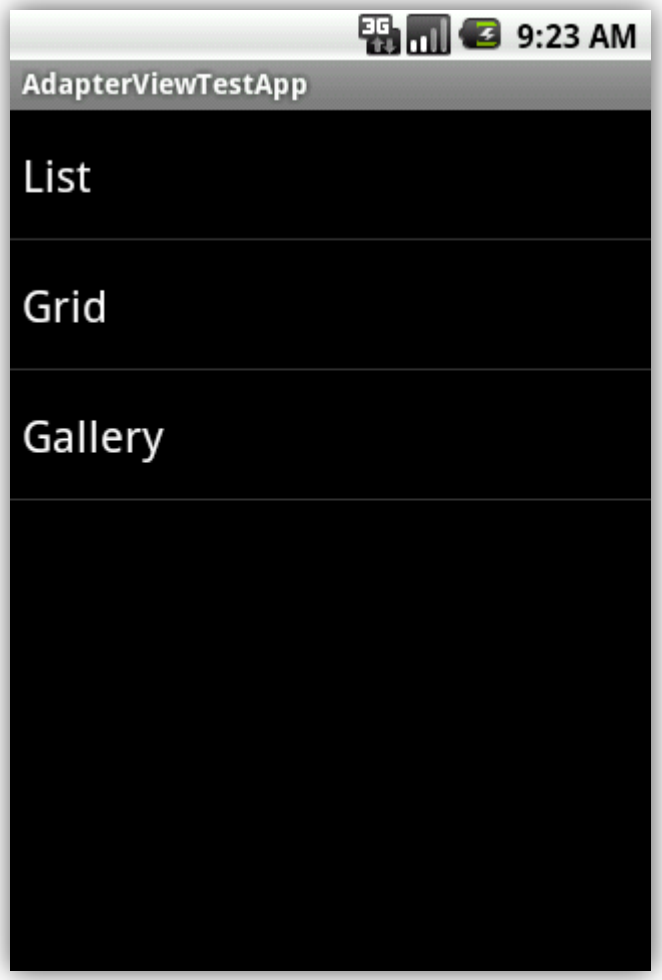
- 이후 SimpleCursorAdapter 객체를 뷰 컨테이너에 적용하면 데이터베이스 각 행의 열로부터 생성된 위젯들이 컨테이너에 채워짐

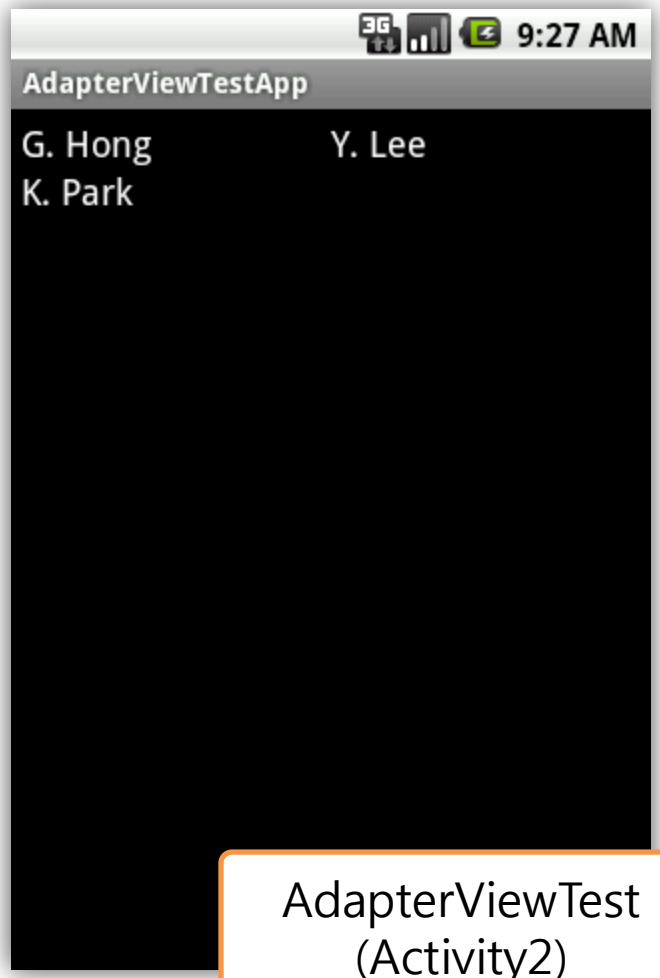
ListView와 CursorAdapter

- 항목을 클릭(ItemClick) 할 경우 실행하는 함수 등록

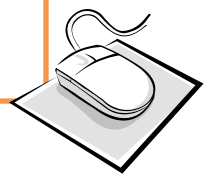
```
lview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        public void onItemClick( AdapterView<?> parent, View view, int position, long id) {
            Toast.makeText(Activity1.this, "Clicked _id="+id, Toast.LENGTH_SHORT).show();
        }
});
```

GridView와 CursorAdapter





AdapterViewTest
(Activity2)



GridView와 CursorAdapter

- /res/layout/grid.xml

```
<GridView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/rid_view"
    android:numColumns="2">
</GridView>
```

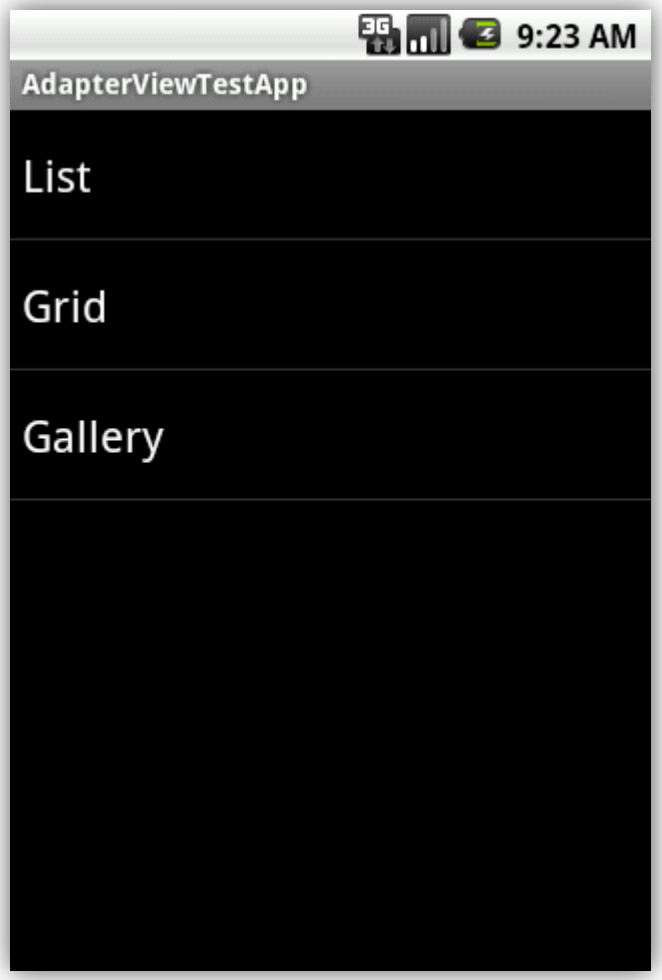
- onCreate 메소드

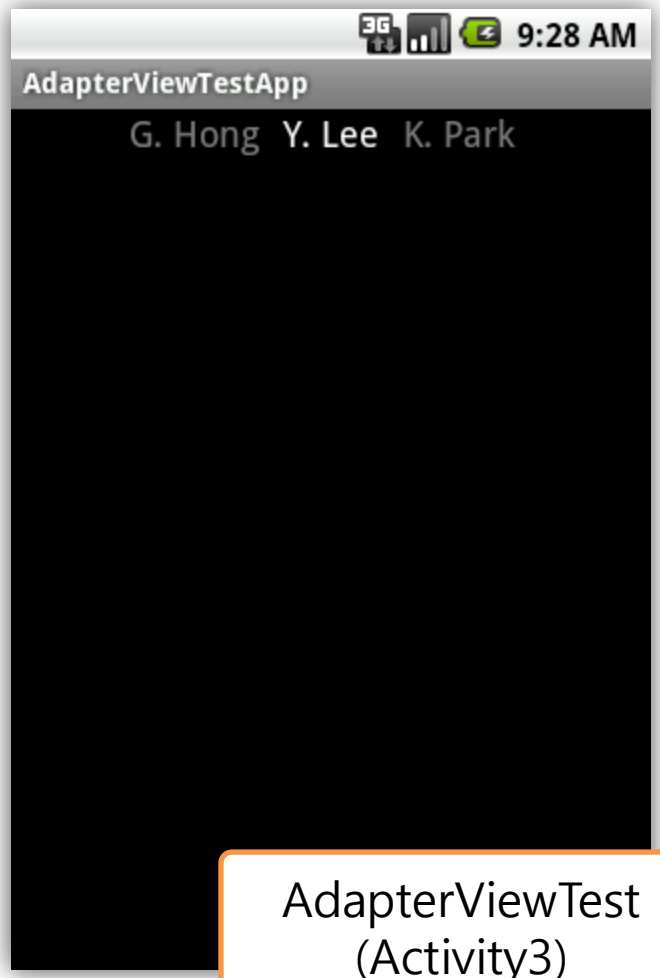
```
setContentView(R.layout.grid);
```

GridView와 CursorAdapter

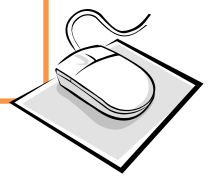
```
Cursor cursor =
    getContentResolver().query(People.CONTENT_URI, null, null, null, null);

startManagingCursor(cursor);

SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
    android.R.layout.simple_gallery_item,
    cursor,
    new String[]{People.NAME},
    new int[]{android.R.id.text1});

GridView gview= (GridView)findViewById(R.id.grid_view);
gview.setAdapter(adapter);

gview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    public void onItemClick(
        AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(Activity2.this, "Clicked _id="+id, Toast.LENGTH_SHORT).show();
    }
});
```

Gallery와 CursorAdapter





AdapterViewTest
(Activity3)



Gallery와 CursorAdapter

- Gallery
  - 뷰 객체들을 수평 방향의 목록 형태로 표시. 수평 스크롤 지원
  - 이미지들을 표시하는 데 흔히 쓰임
  - 사용자는 한 항목을 선택해서 일정한 작업을 수행
- /res/layout/grid.xml

```
<Gallery
    android:id="@+id/gallery_view"
    android:layout_height="wrap_content"
    xmlns:android=http://schemas.android.com/apk/res/android
    android:layout_width="fill_parent"
    android:spacing="12px"
    android:unselectedAlpha=".50">
</Gallery>
```

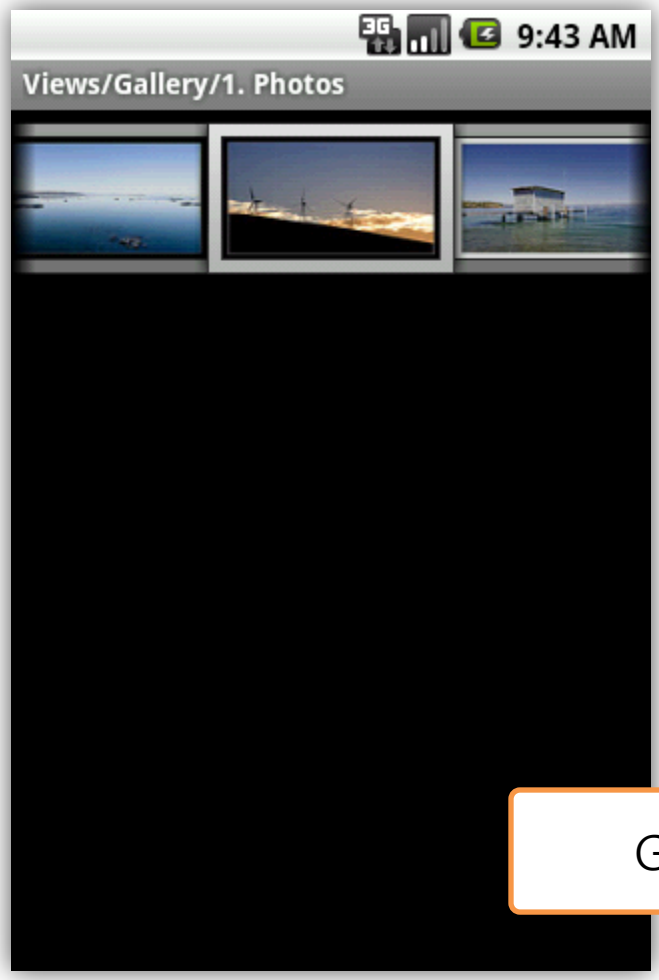
- onCreate 메소드

```
setContentView(R.layout.gallery);
```

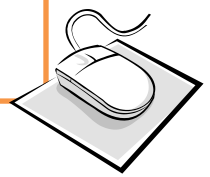
Gallery와 CursorAdapter

```
Cursor cursor =
    getContentResolver().query(People.CONTENT_URI, null, null, null, null);

startManagingCursor(cursor);

SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
android.R.layout.simple_gallery_item, // R.layout.scratch_layout
    cursor,
    new String[]{People.NAME},
    new int[]{android.R.id.text1}); //R.id.scratch_text1

Gallery av = (Gallery)findViewById(R.id.gallery_view);
av.setAdapter((SpinnerAdapter) adapter);

galley.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    public void onItemClick(
        AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(Activity3.this, "Clicked _id="+id, Toast.LENGTH_SHORT).show();
    }
});
```

Gallery와 CursorAdapter



GalleryTest



GridView와 ArrayAdapter

- GridView
  - 뷰 객체들을 고정된 개수의 열로 이루어진 격자 형태로 배치
  - 격자의 각 칸에 이미지 아이콘이 배치되는 경우가 많다. 사용자는 한 항목을 선택해서 일정한 작업을 수행
- ArrayAdapter
  - 배열과 배열에 있는 각 자료로 생성할 위젯을 정의하는 레이아웃을 주면 이를 이용하여 뷰 컨테이너를 적절히 채움

GridView와 ArrayAdapter

/res/layout/grid.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TextView android:id="@+id/items"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Grid View"
                android:textSize="36px" />

        <GridView android:id="@+id/text_grid"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:numColumns="auto_fit"
                android:padding="5dp"/>
</LinearLayout>
```

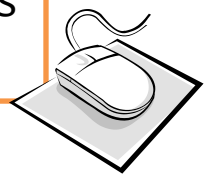
/res/layout/bigtext.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:textSize="84dp"
        android:gravity="center"/>
```

GridView와 ArrayAdapter

```
private static final String[] numbers
    = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C"};

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.grid);

    GridView gview = (GridView)findViewById(R.id.text_grid);
    gview.setAdapter(new ArrayAdapter<String>(this,
        R.layout.bigtext, numbers));

    gview.setOnItemClickListener(
        new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View
            view, int position, long id)
            {
                TextView text = (TextView) view;
                String num = (String) text.getText();
                num = Integer.toString((Integer.parseInt(num) + 1));
                text.setText(num);
            }
        }
    );
}
```

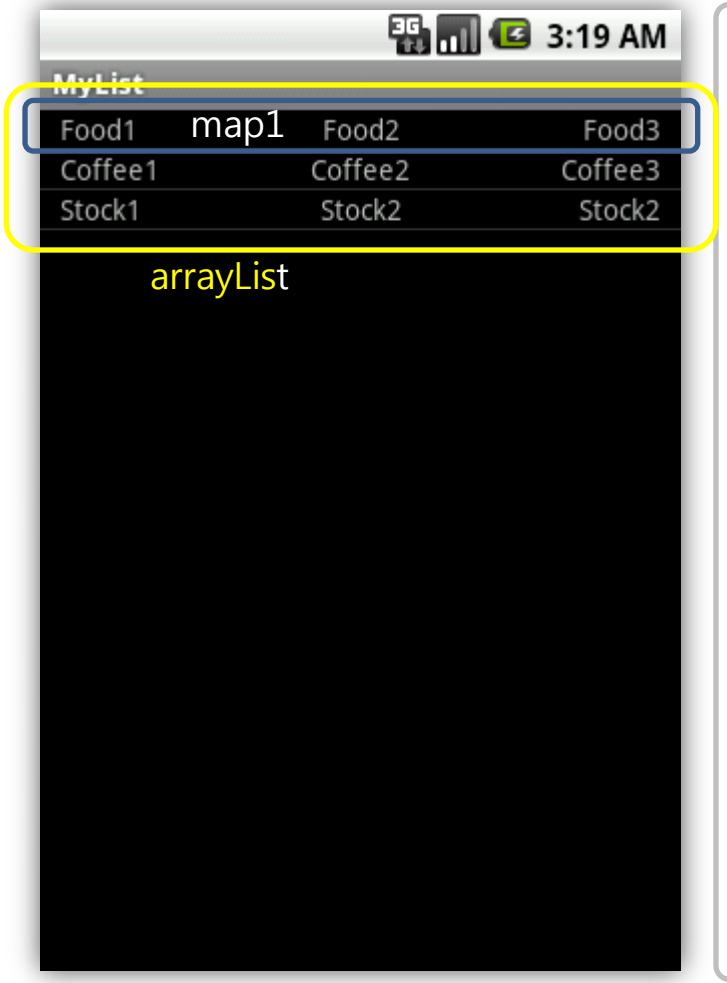
LayoutContainerExamples
(GridLayout)



ListView와 SimpleAdapter

- ListView
  - 뷰 객체들을 수직 방향의 목록 형태로 보여줌.
  - 수직 스크롤 지원
  - 일반적으로 목록의 각 항목은 자료를 담은 뷰
  - 사용자는 목록의 한 항목을 선택해서 일정한 작업을 수행
- SimpleAdapter
  - 정적 데이터를 XML의 뷰로 매핑하는 아답터

ListView와 SimpleAdapter



01. ListView 위젯은 (아답터가 데이터를 채워주는) 아답터 뷰 위젯

02. 아답터 객체(adapter)를 생성한 후 리스트 뷰 객체(listview)에게 setAdapter하라고만 하면 됨

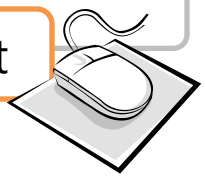
```
listview.setAdapter(adapter);
```

03.아답터 객체를 만듦. 이를 위해 다음 정보가 필요

*하나의 레코드 데이터를 갖는 HashMap 객체
*여러 HashMap 객체를 담고 있는 ArrayList 객체(arrayList)
*실제로 데이터를 표시를 위한 xml 레이아웃 파일 (R.layout.item) -> TextView를 가지고 있음
*각 필드 데이터를 찾는데 사용할 키
*찾은 필드 데이터를 넣을 TextView ID

```
SimpleAdapter adapter = new SimpleAdapter(
   this, arrayList, R.layout.item,
   new String[] {COL1, COL2, COL3},
   new int[]{R.id.text_1, R.id.text_2, R.id.text_3});
```

ListViewTest



이벤트 처리

- AdapterView 파생 컨테이너는 단순히 자료 항목들의 표시만이 아니라 사용자가 그 중 한 항목을 선택할 수 있음
- 지금까지 이야기한 ListView와 GridView, Gallery는 그러한 사용자 선택 상호작용을 지원
- 항목을 클릭하면 ItemClick 이벤트가 발생하고 따라서 이를 처리할 수 있는 OnItemClickListener를 등록하여 onItemClick 메소드를 정의

```
av.setOnItemClickListener( new AdapterView.OnItemClickListener() {
    public void onItemClick( AdapterView<?> parent, View view,
        int position, long id) {
            Toast.makeText(Scratch.this, "Clicked _id="+id,
            Toast.LENGTH_SHORT).show();
        }
    }
});
```

ListActivity 위젯

- Activity + ListView 대신 ListActivity를 이용하면 코드가 간편해짐
- 첫째로, ListActivity를 파생한 활동 클래스에서는 항목 관련 사건들을 처리할 때 그냥 활동 클래스 안에서 콜백 메서드를 구현
- 둘째로, 적응자를 적용할 때 findViewById를 거치지 않고 바로 setListAdapter 메서드를 호출
- 목록에 헤더(header)와 푸터(footer)를 표시하려면, ListView.FixedViewInfo 클래스와 ListView.addHeaderView, ListView.addFooterView를 이용

ListActivity를 이용한 메뉴

01. 메뉴 문자열(키)과 그 메뉴를 눌렀을 때 실행할 인텐트(값) 쌍을 여러 개 담은 맵 생성
02. 메뉴 항목들을(keys) 특정 레이아웃으로(TextView 리소스 ID) 뷰 컨테이너 위젯에 채우는 아답터 객체 생성
03. 뷰 컨테이너 위젯에 설정
04. 항목을 선택(ListItemClick)할 경우 실행되는 리스너 함수(onListItemClick) 재정의

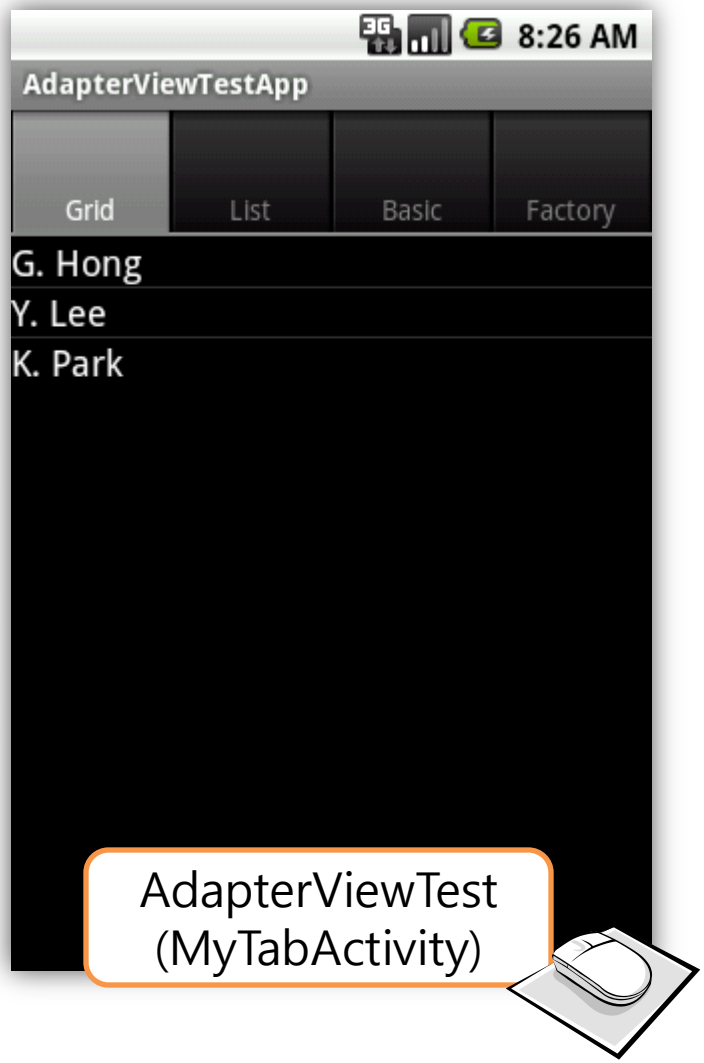
```
public class MainActivity extends ListActivity
{
    private Map<String, Object> map = new HashMap<String, Object>();

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

        map.put("List", new Intent(this, Activity1.class));
        map.put("Grid", new Intent(this, Activity2.class));
        map.put("Gallery", new Intent(this, Activity3.class));

        // 해쉬 맵에 있는 키값(메뉴 항목)들을 배열로 추출
        String[] keys = map.keySet().toArray(new String[map.keySet().size()]);

        // 배열에 있는 요소들을, 지정한 TextView 리소스 ID 형태로, 뷰 컨테이너 위젯에 채우는, 아답터 객체 생성
        ArrayAdapter<String> adapter =
            new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, keys);

        // 액티비티에 아답터 설정
        setListAdapter(adapter);
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
                String key = (String) l.getItemAtPosition(position);
                startActivity((Intent) map.get(key));
    }
}
```

findViewById를 거치지 않고 바로 호출

이벤트를 처리할 때 그냥 Activity 안에서 리스너 메서드 구현

TabActivity

- TabActivity와 TabHost를 이용
- TabHost는 여러 개의 TabSpec 객체들로 구성됨
- 각 TabSpec은 탭 제목과 탭 내용 등의 탭 정보를 담음
- 탭은 어떤 뷰를 담는 컨테이너
- 탭의 내용은 미리 정의된 뷰일 수도 있고 Intent 객체를 통해서 실행되는 Activity일 수도 있으며, 팩토리(TabContentFactory 구현)가 생성한 뷰일 수도 있음



04. 뷰 컨테이너 클래스 활용(25)

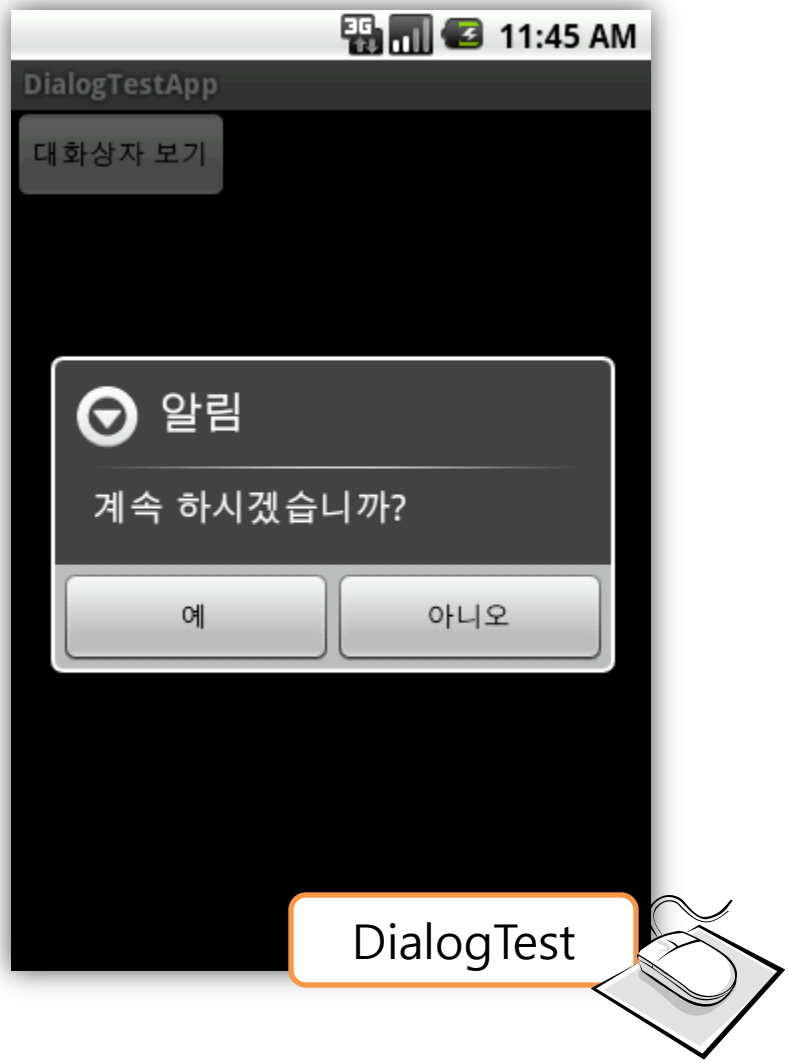
```
public class MyTabActivity extends TabActivity implements android.widget.TabHost.TabContentFactory {
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TabHost tabHost = getTabHost();

        LayoutInflater.from(this).inflate(
            R.layout.example_layout, tabHost.getTabContentView(), true);

        TabSpec tspec = tabHost.newTabSpec("tab1");
        tspec.setIndicator("Grid");
        tspec.setContent(new Intent(this, Activity1.class));
        tabHost.addTab(tspec);

        tabHost.addTab(tabHost.newTabSpec("tab2").setIndicator("List").setContent(new Intent(this, Activity2.class)));
        tabHost.addTab(tabHost.newTabSpec("tab3").setIndicator("Basic").setContent(R.id.example_layout));
        tabHost.addTab(tabHost.newTabSpec("tab4").setIndicator("Factory").setContent(this));
    }

    public View createTabContent(String tag) {
        if (tag.compareTo("tab4") == 0) {
            TextView tv = new TextView(this);
            Date now = new Date();
            tv.setText("I'm from a factory. Created: " + now.toString());
            tv.setTextSize((float) 24);
            return (tv);
        } else {
            return null;
        }
    }
}
```

TabActivity

- LayoutInflater는 XML 레이아웃 자원 파일에 정의된 뷰들로부터 실제 뷰 객체들을 생성
- 보통은 setContentView에 의해 그런 일이 일어나나, 이 예제에서는 setContentView()를 사용하지 않음
- 셋째 탭을 만들 때 뷰들을 식별자로 참조해야 하는데, 그러려면 이처럼 LayoutInflater를 직접 사용해야 함
- 처음 두 addTab() 호출은 Activity를 띄우는 Intent 객체를 생성해서 탭의 내용으로 지정
- 셋째 addTab() 호출에서는 레이아웃 식별자를 지정. 이 레이아웃 식별자는 이전에 LayoutInflater를 적용했을 때 지정한 자원 파일의 한 레이아웃이어야 함

Dialog

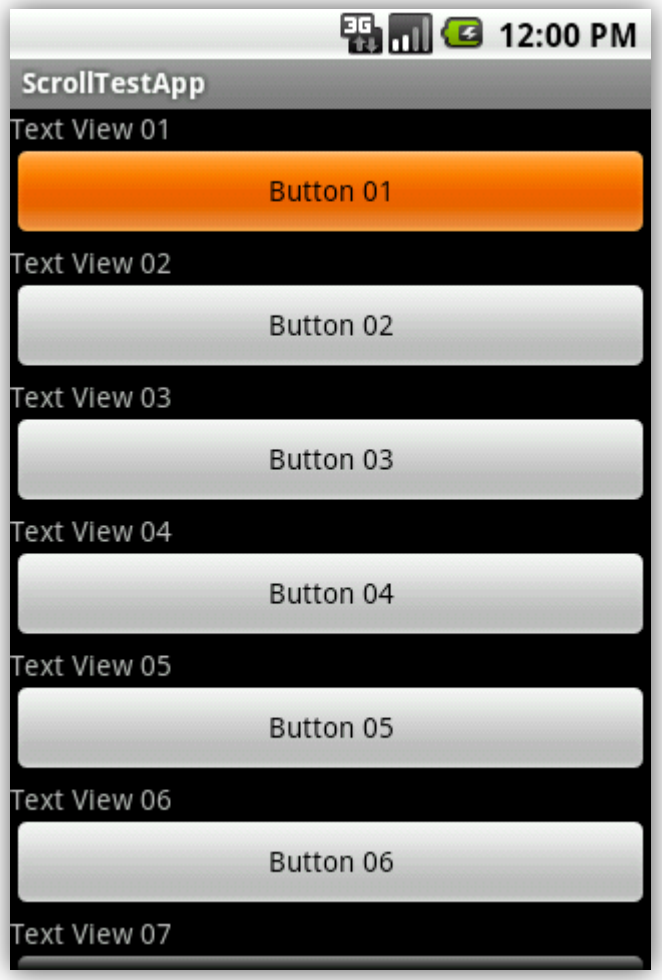
- 대화상자는 Activity와 마찬가지로 레이아우슬 담을 수 있으며, 팝업 창 형태라는 점이 다름
- 대화상자는 Dialog 클래스에서 파생되며, 옆의 대화상자는 AlertDialog 대화상자 모습

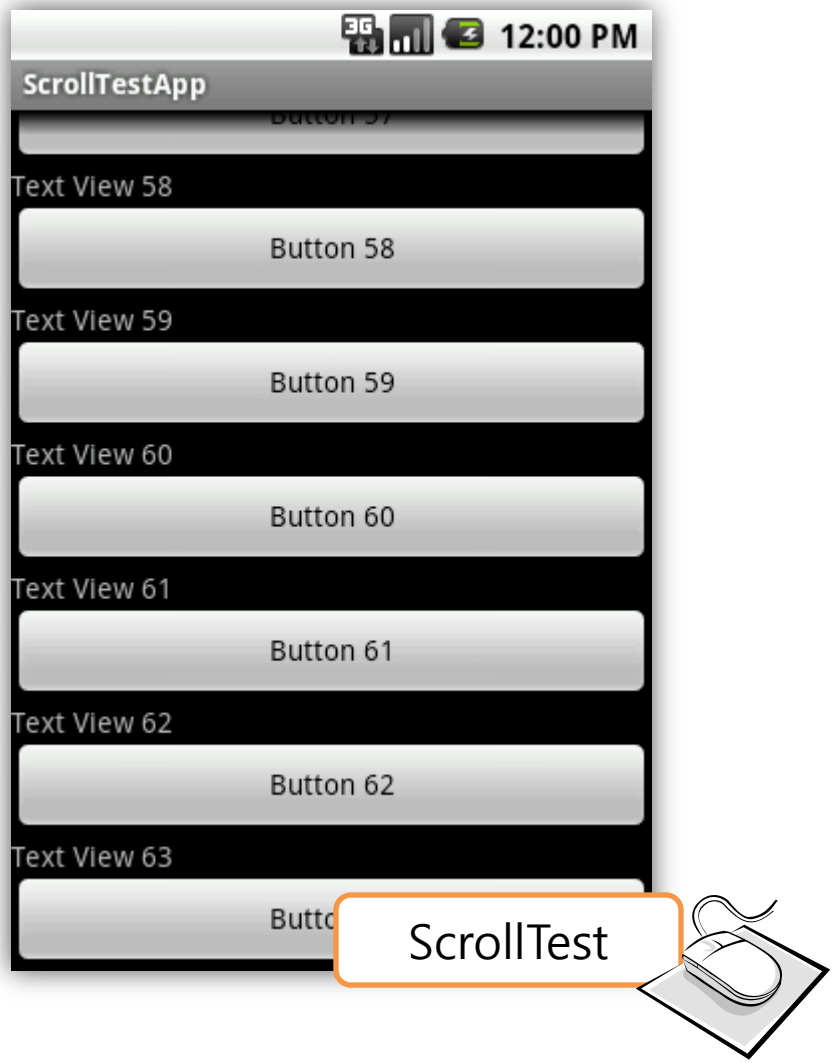


Dialog

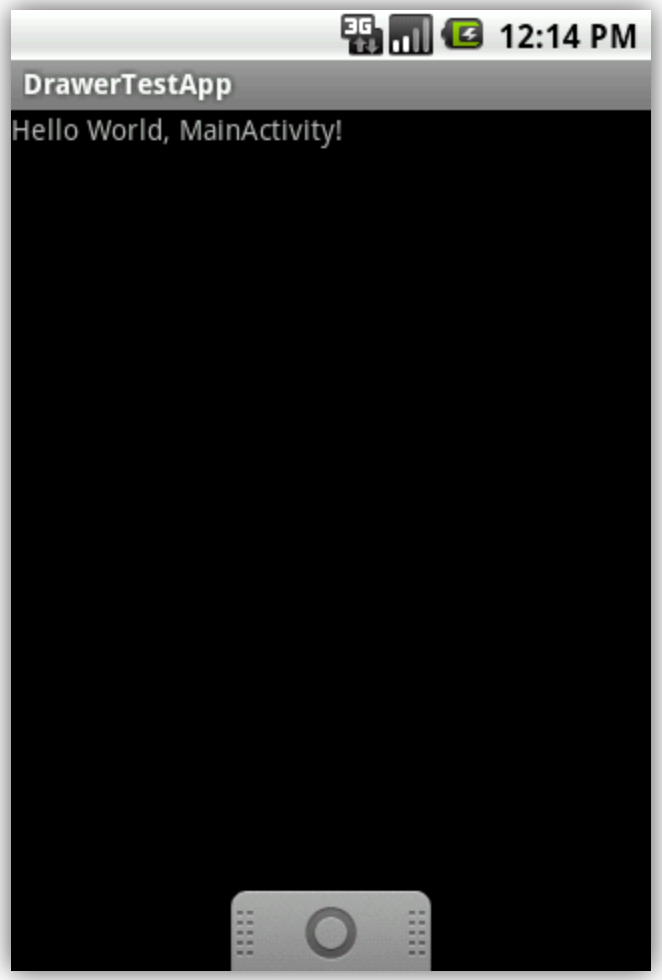
```
public class MainActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button b = (Button)findViewById(R.id.show_btn);

        b.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                new AlertDialog.Builder(MainActivity.this)
                .setMessage("계속 하시겠습니까?")
                .setPositiveButton("확인", null)
                .show();
            }
        });
    }
}
```

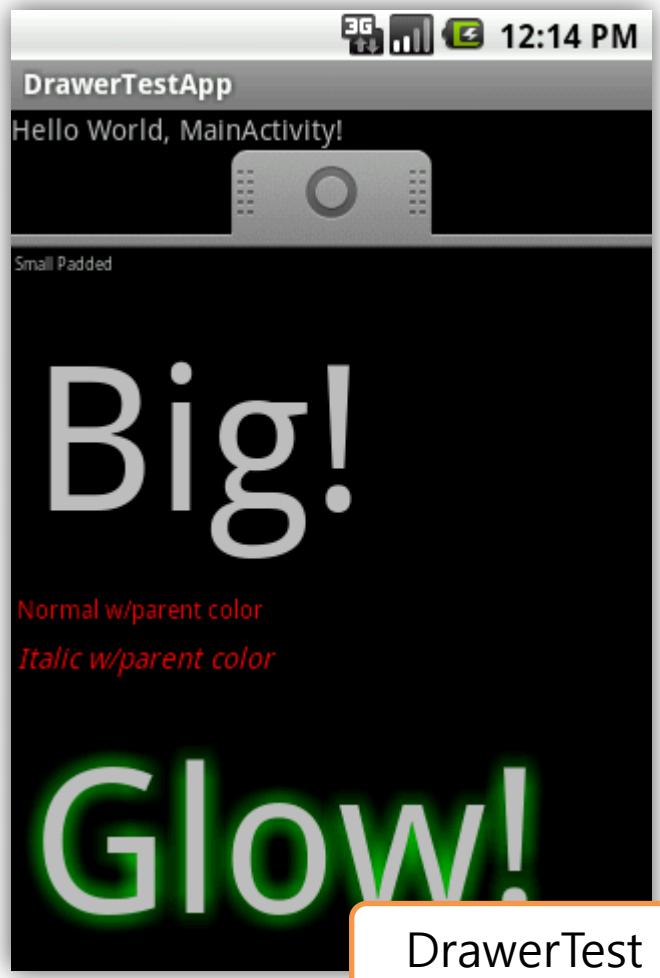
ScrollView


12:00 PM
ScrollTestApp
Text View 01
Button 01
Text View 02
Button 02
Text View 03
Button 03
Text View 04
Button 04
Text View 05
Button 05
Text View 06
Button 06
Text View 07


12:00 PM
ScrollTestApp
Text View 58
Button 58
Text View 59
Button 59
Text View 60
Button 60
Text View 61
Button 61
Text View 62
Button 62
Text View 63
ScrollTest

SlidingDrawer





DrawerTest