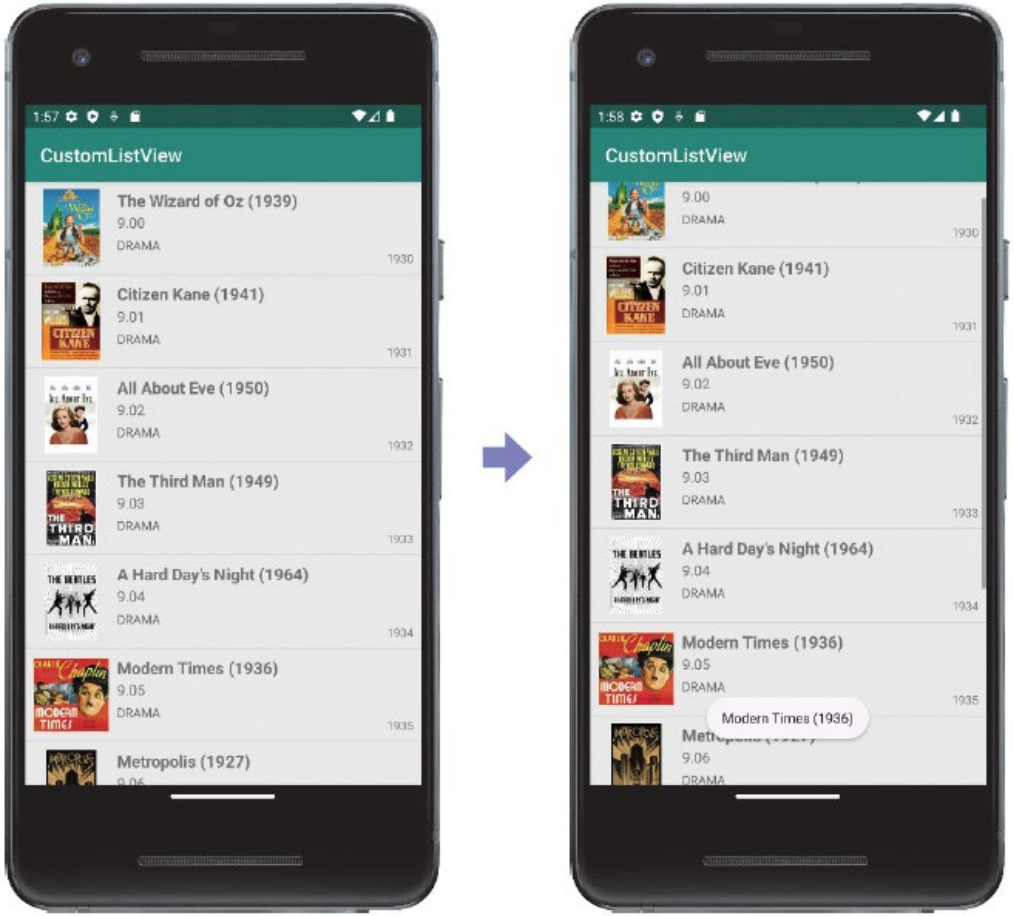
CHAP 8. 어댑터뷰와 프래그먼트

8장의 목표

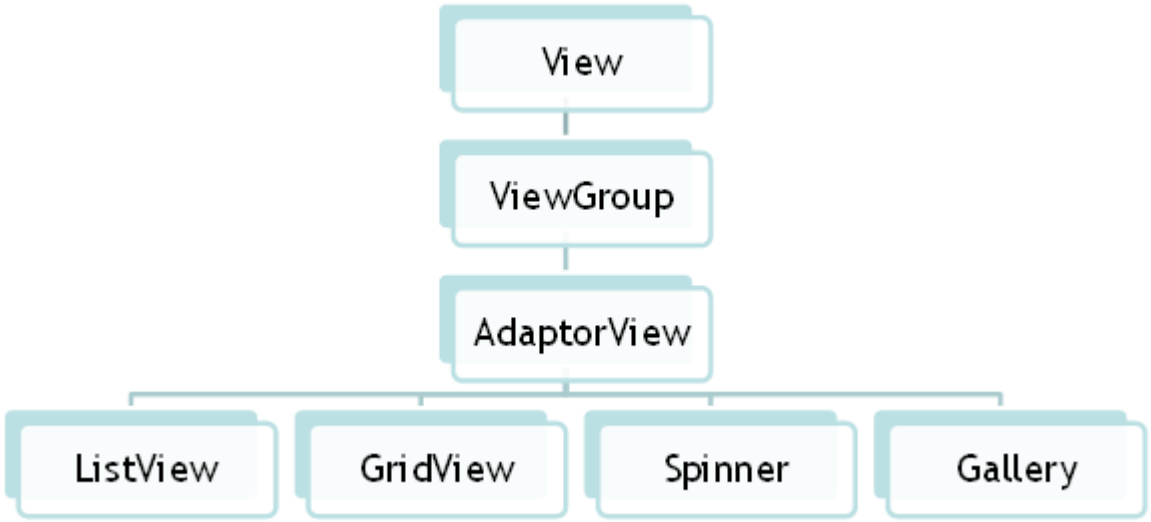

1:57
CustomListView
The Wizard of Oz (1939)
9.00
DRAMA
1930
Citizen Kane (1941)
9.01
DRAMA
1931
All About Eve (1950)
9.02
DRAMA
1932
The Third Man (1949)
9.03
DRAMA
1933
A Hard Day's Night (1964)
9.04
DRAMA
1934
Modern Times (1936)
9.05
DRAMA
1935
Metropolis (1927)
1:58
CustomListView
9.00
DRAMA
1930
Citizen Kane (1941)
9.01
DRAMA
1931
All About Eve (1950)
9.02
DRAMA
1932
The Third Man (1949)
9.03
DRAMA
1933
A Hard Day's Night (1964)
9.04
DRAMA
1934
Modern Times (1936)
9.05
DRAMA
1935
Modern Times (1936)
9.06
DRAMA

어댑터 뷰 클래스

- 화면에 표시된 수많은 항목을 손으로 쓸어넘기면서 본다. 이때 사용되는 위젯이 어댑터 뷰(AdapterView)이다.

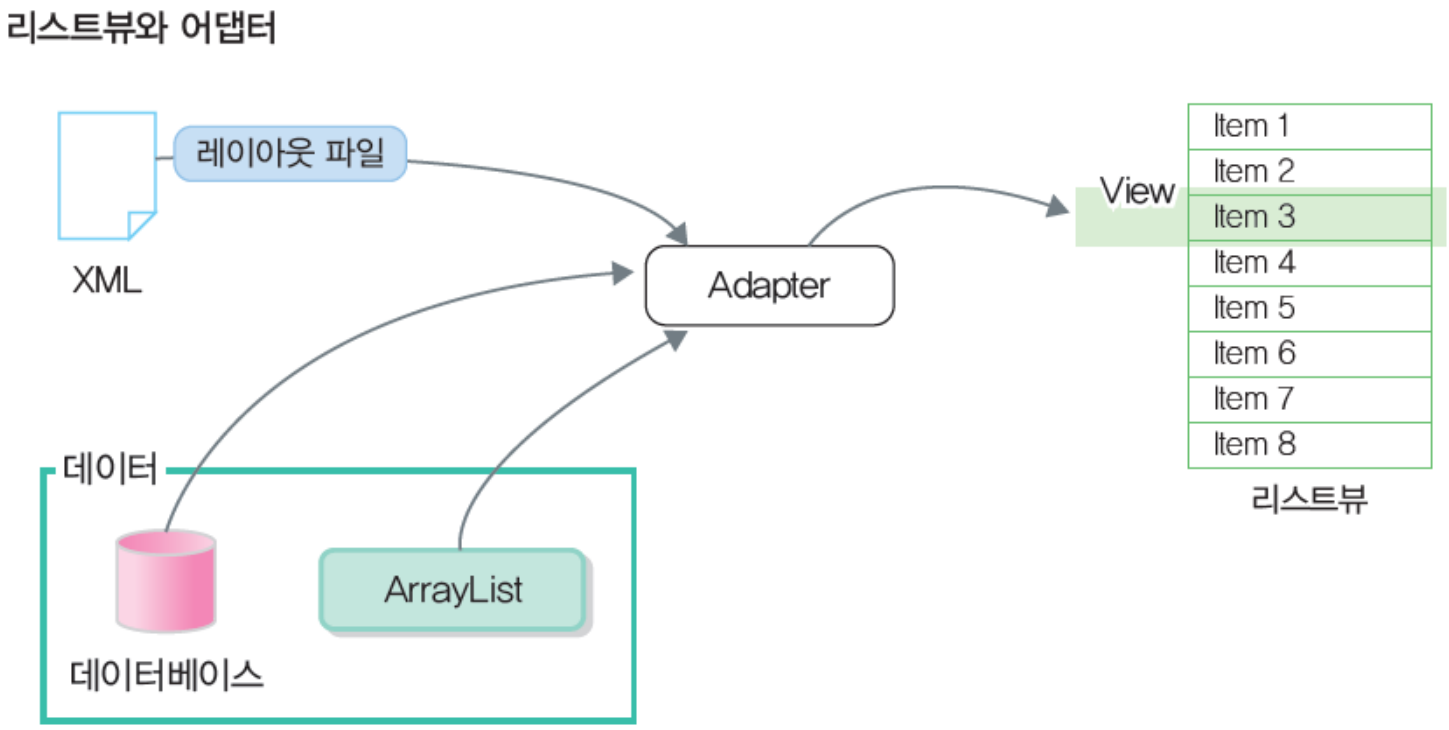
고전적인 어댑터 뷰의 종류

- 리스트 뷰(ListView), 갤러리(Gallery), 스피너(Spinner), 그리드 뷰(GridView)



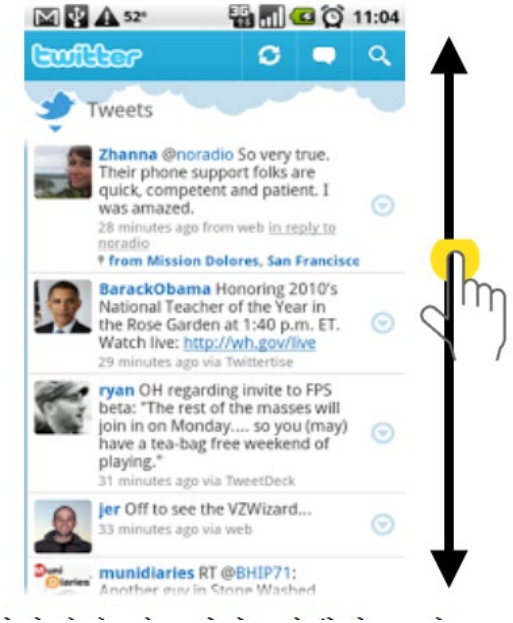
어댑터 뷰

- **어댑터 뷰(AdapterView)**는 배열이나 파일, 데이터베이스에 저장된 데이터를 화면에 표시할 때 유용한 뷰



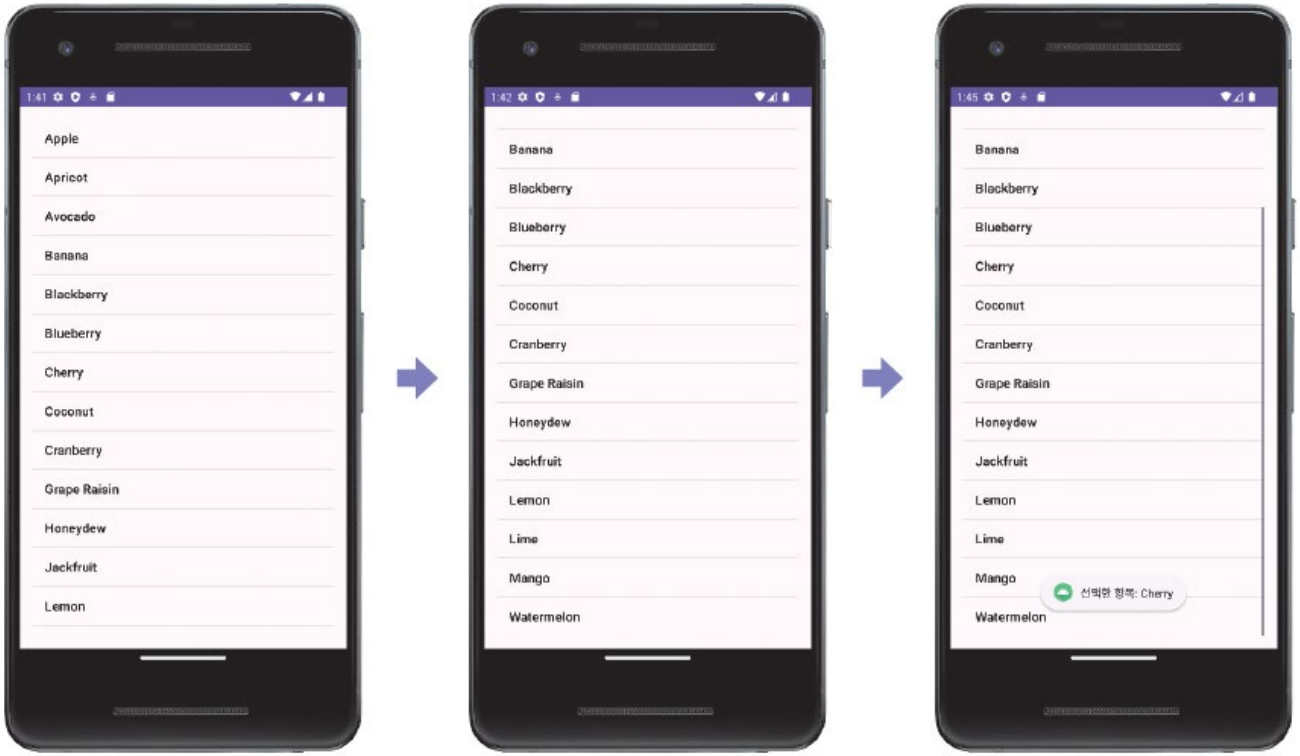
리스트 뷰

- **리스트 뷰(ListView)**는 항목들을 수직으로 보여주는 어댑터 뷰로서 상하로 스크롤이 가능



예제: 리스트 뷰

- 리스트 뷰를 가진 액티비티를 간단하게 작성하고 여기에 어댑터를 연결하여서 데이터를 표시하는 예제를 살펴보자.



예제: 리스트 뷰

activity_main.xml

```
<RelativeLayout>
    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

예제: 리스트 뷰

MainActivity.java

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // 데이터 배열 생성
        String[] data = {"Apple", "Apricot", "Avocado", "Banana", "Blackberry",
                "Blueberry", "Cherry", "Coconut", "Cranberry",
                "Grape Raisin", "Honeydew", "Jackfruit", "Lemon", "Lime",
                "Mango", "Watermelon"};

        // ArrayAdapter를 사용하여 데이터를 ListView에 연결
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_
        item_1, data);
        ListView listView = findViewById(R.id.listView);
        listView.setAdapter(adapter);
```

배열에 저장된 데이터

어댑터 생성

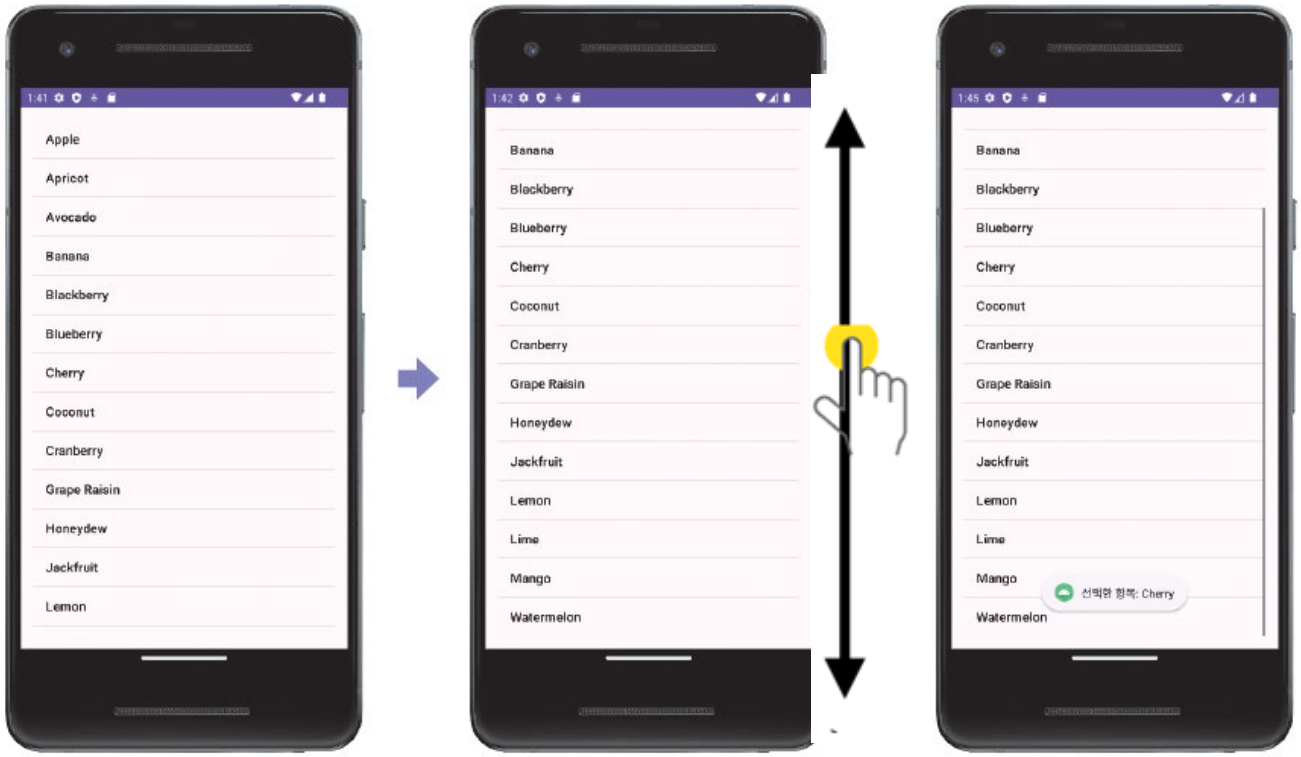
리스트 뷰에 어댑터 설정

예제: 리스트 뷰

```
        // ListView에 항목 클릭 리스너 추가
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

                // 클릭한 항목의 텍스트를 가져와서 토스트 메시지로 표시
                String selectedItem = data[position];
                Toast.makeText(getApplicationContext(), "선택한 항목: " + selectedItem, Toast.
                LENGTH_SHORT).show();
            }
        });
    }
}
```

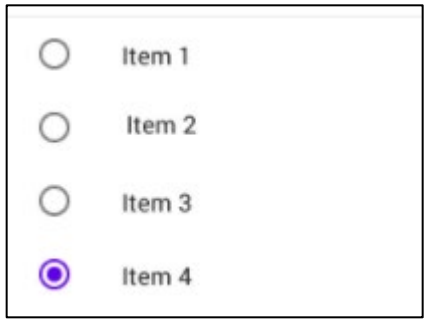
사용자가 리스트 뷰의 항목을 클릭하게 되면 onListItemClick()이 호출되고 화면 하단에 토스트 메시지가 표시된다. ListActivity가 가지고 있는 메소드이다.

실행결과


Apple
Apricot
Avocado
Banana
Blackberry
Blueberry
Cherry
Coconut
Cranberry
Grape Raisin
Honeydew
Jackfruit
Lemon
Banana
Blackberry
Blueberry
Cherry
Coconut
Cranberry
Grape Raisin
Honeydew
Jackfruit
Lemon
Lime
Mango
Watermelon
Banana
Blackberry
Blueberry
Cherry
Coconut
Cranberry
Grape Raisin
Honeydew
Jackfruit
Lemon
Lime
Mango
Watermelon
선택한 항목: Cherry

리스트 뷰의 표준 레이아웃

레이아웃 ID	설명
simple_list_item_1	하나의 텍스트 뷰 사용
simple_list_item_2	두 개의 텍스트 뷰 사용
simple_list_item_checked	항목당 체크 표시
simple_list_item_single_choice	한 개의 항목만 선택
simple_list_item_multiple_choice	여러 개의 항목 선택 가능

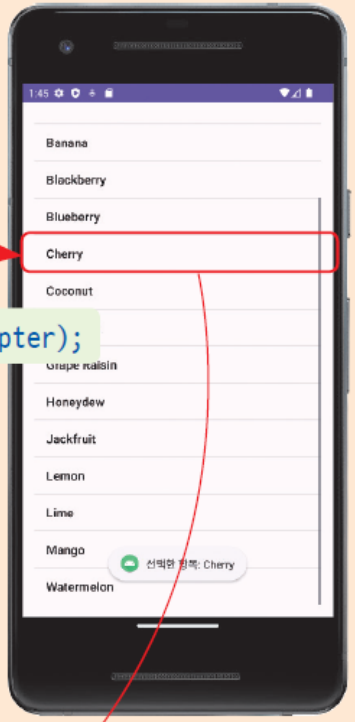


리스트 뷰와 arrayAdapter

```
String[] values = { "Apple", "Apricot",
"Avocado", "Banana", "Blackberry",
"Blueberry", "Cherry", "Coconut",
"Cranberry", "Grape Raisin",
"Honeydew", "Jackfruit", "Lemon",
"Lime", "Mango", "Watermelon" };
```

```
adapter = new ArrayAdapter<String>(this,
  android.R.layout.simple_list_item_1, values);
```

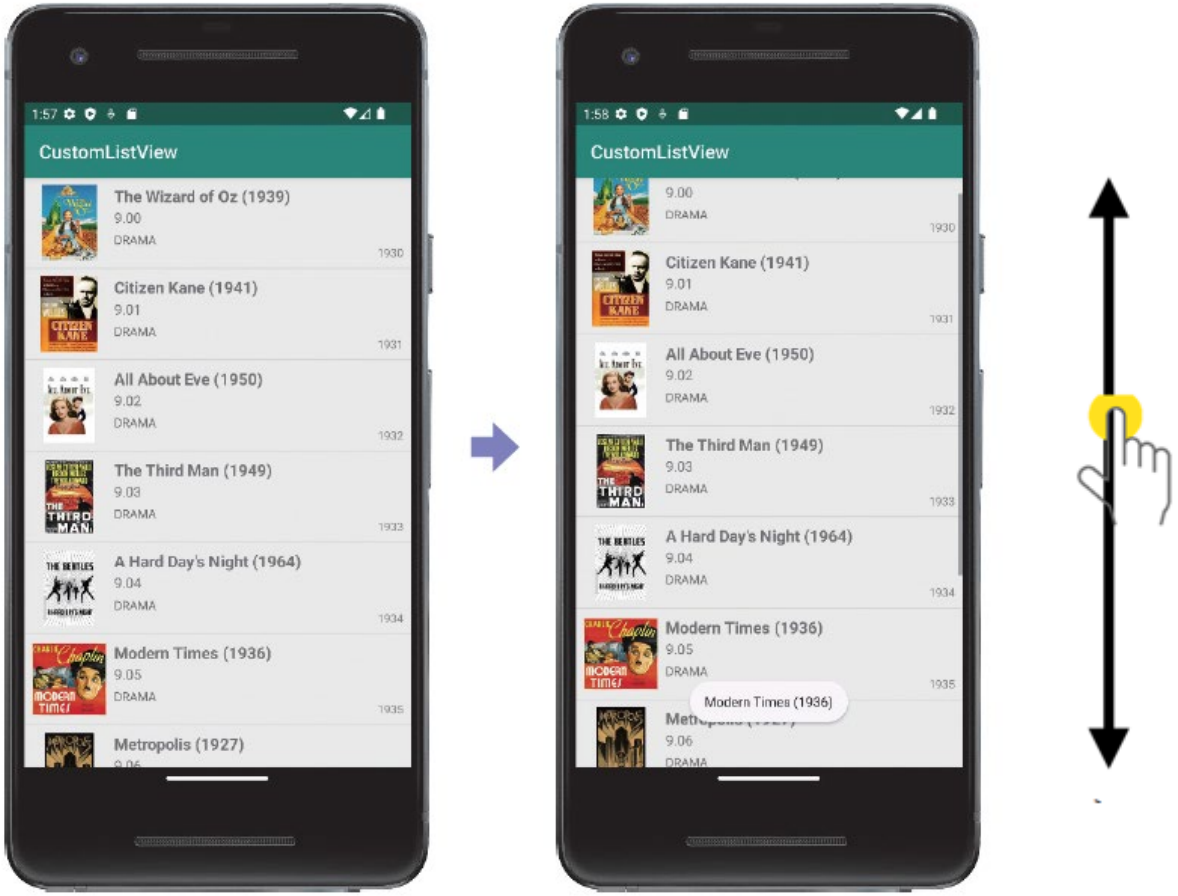
```
setAdapter(adapter);
```



리스트 뷰에서 사용자가 특정한 항목을 선택하면 이벤트가 발생한다. 이벤트가 발생하면 onListItemClick()이 호출된다.

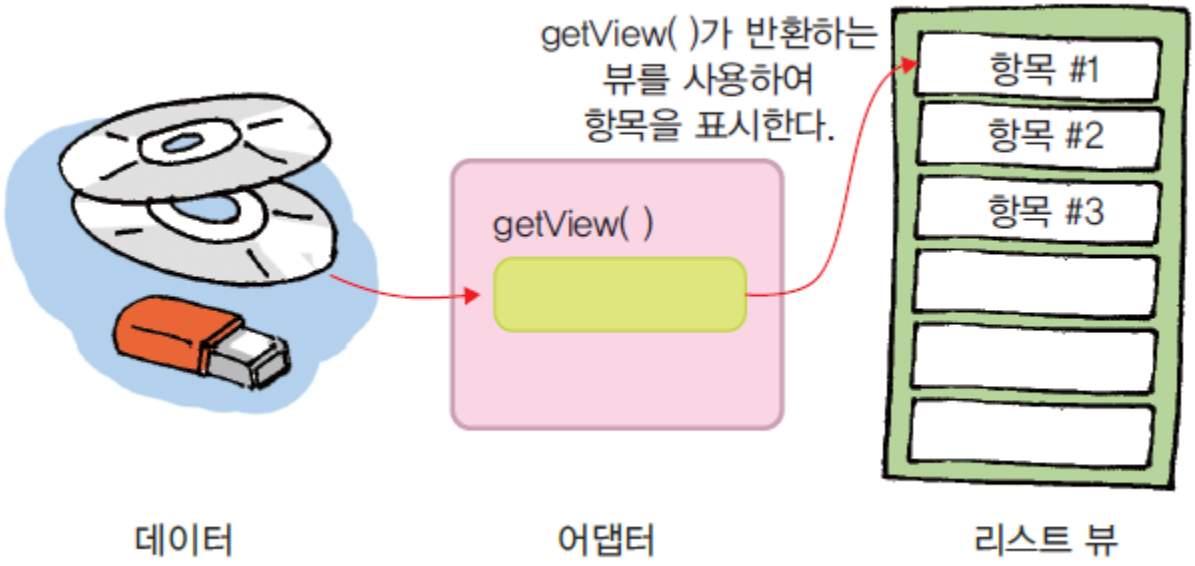
```
protected void onItemClick(ListView l, View v, int position, long id) {
    String selectedItem = data[position];
        Toast.makeText(getApplicationContext(), "선택한 항목: " + selectedItem,
        Toast.LENGTH_SHORT).show();
}
```

예제: 영화 리스트 만들기

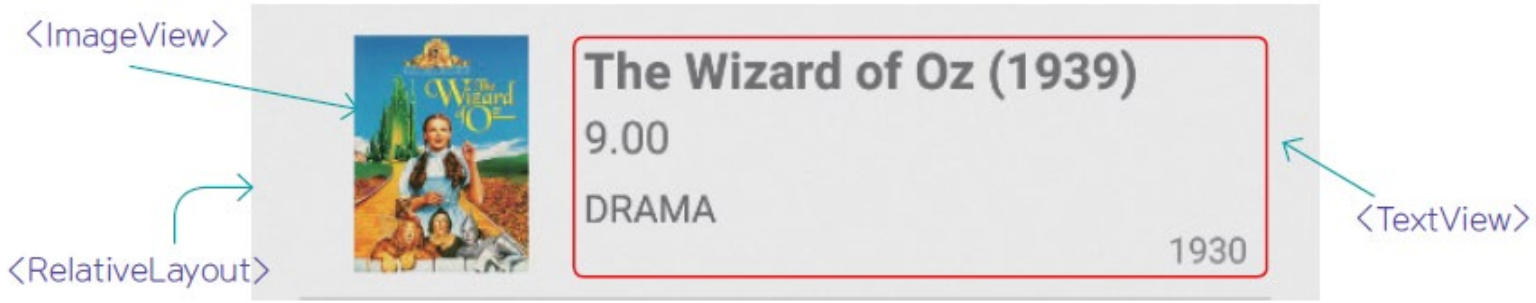

1:57
CustomListView
The Wizard of Oz (1939)
9.00
DRAMA
1930
Citizen Kane (1941)
9.01
DRAMA
1931
All About Eve (1950)
9.02
DRAMA
1932
The Third Man (1949)
9.03
DRAMA
1933
A Hard Day's Night (1964)
9.04
DRAMA
1934
Modern Times (1936)
9.05
DRAMA
1935
Metropolis (1927)
1:58
CustomListView
9.00
DRAMA
1930
Citizen Kane (1941)
9.01
DRAMA
1931
All About Eve (1950)
9.02
DRAMA
1932
The Third Man (1949)
9.03
DRAMA
1933
A Hard Day's Night (1964)
9.04
DRAMA
1934
Modern Times (1936)
9.05
DRAMA
1935
Modern Times (1936)
9.06
DRAMA

예제: 리스튜 뷰

- 이런 경우에는 개발자가 직접 어댑터 클래스를 작성하여 사용하는 것이 좋다. 리스트 뷰는 항목을 표시할 때, 어댑터 클래스의 getView( )를 호출하여서 반환되는 뷰를 이용하여서 항목을 표시한다. 따라서 getView( )를 재정의하여서 우리가 필요한 동작을 수행하도록 하면 된다.



뷰의 레이아웃 설계


<ImageView>
<RelativeLayout>
The Wizard of Oz (1939)
9.00
DRAMA
1930
<TextView>

레이아웃 파일

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <ListView
        android:id="@+id/list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
    </ListView>

</RelativeLayout>
```

레이아웃에 리스트 뷰를 배치한다.

리스트의 항목을 나타내는 뷰 설계

listitem.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#eeeeee"
    android:padding="8dp" >

    <ImageView
        android:id="@+id/image"
        android:layout_width="80dp"
        android:layout_height="80dp"
```

영화 포스터 이미지

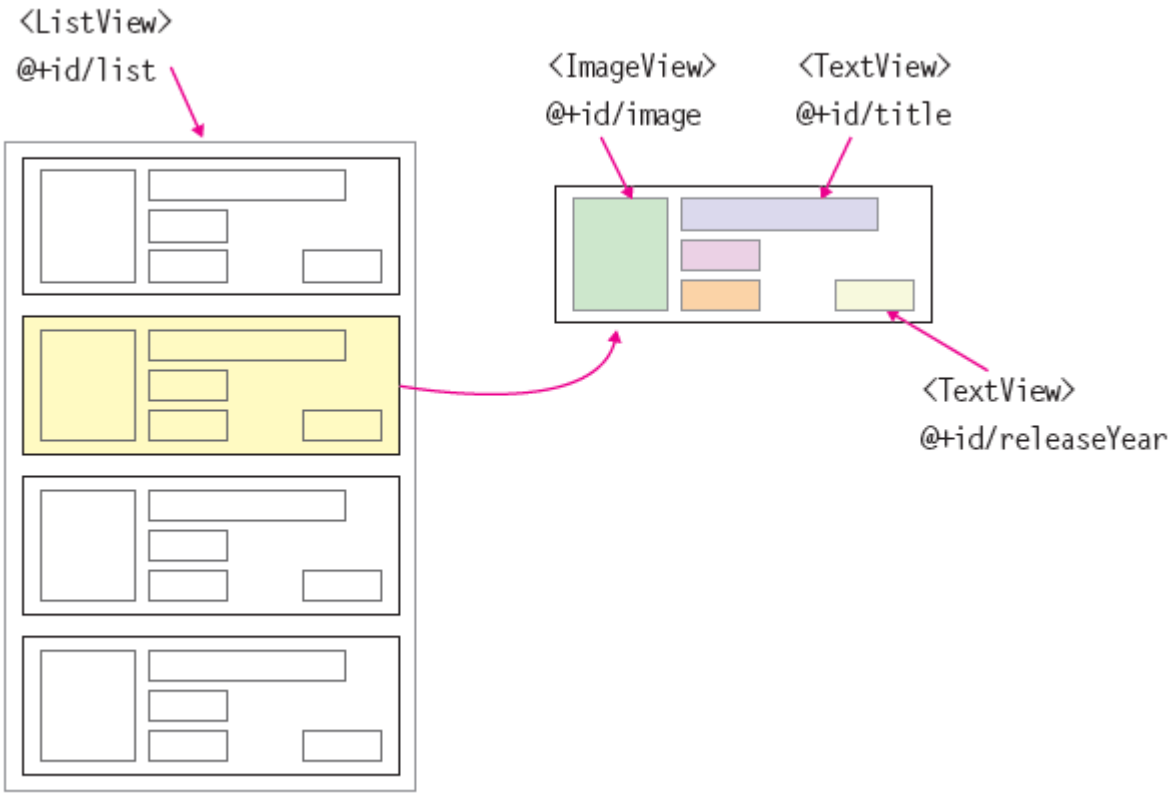
리스트의 항목을 나타내는 뷰 설계

```
        android:layout_alignParentLeft="true"
        android:layout_marginRight="8dp" />

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/image"
        android:layout_toRightOf="@+id/image"
        android:textSize="17dp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/rating"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/title"
        android:layout_marginTop="1dip"
        android:layout_toRightOf="@+id/image"
        android:textSize="15dip" />
```

리스트의 항목을 나타내는 뷰 설계

```
<TextView
    android:id="@+id/genre"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/rating"
    android:layout_marginTop="5dp"
    android:layout_toRightOf="@+id/image"
    android:textColor="#666666"
    android:textSize="13dip" />

<TextView
    android:id="@+id/releaseYear"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:textColor="#888888"
    android:textSize="12dip" />

</RelativeLayout>
```

뷰의 id 부여


<ListView>
@+id/list
<ImageView>
@+id/image
<TextView>
@+id/title
<TextView>
@+id/releaseYear

자바 소스

MainActivity.java

```
package kr.co.company.customlistview;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class MainActivity extends AppCompatActivity {
    ListView list;
    String[] titles = {
            "The Wizard of Oz (1939)",
            "Citizen Kane (1941)",
            "All About Eve (1950)",
            "The Third Man (1949)",
            "A Hard Day's Night (1964)",
            "Modern Times (1936)",
            "Metropolis (1927)",
            "Metropolis (1927)",
            "Metropolis (1927)",
            "Metropolis (1927)"
    } ;
```

자바

```
Integer[] images = {
        R.drawable.movie1,
        R.drawable.movie2,
        R.drawable.movie3,
        R.drawable.movie4,
        R.drawable.movie5,
        R.drawable.movie6,
        R.drawable.movie7.
        R.drawable.movie7,
        R.drawable.movie7,
        R.drawable.movie7
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    CustomList adapter = new CustomList(MainActivity.this);
    list=(ListView)findViewById(R.id.list);
    list.setAdapter(adapter);
    list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view,
                                int position, long id) {
            Toast.makeText(getBaseContext(), titles[+position],
                                Toast.LENGTH_SHORT).show();
        }
    });
}
```

우리가 만든 커스텀 어댑터를 생성한다.

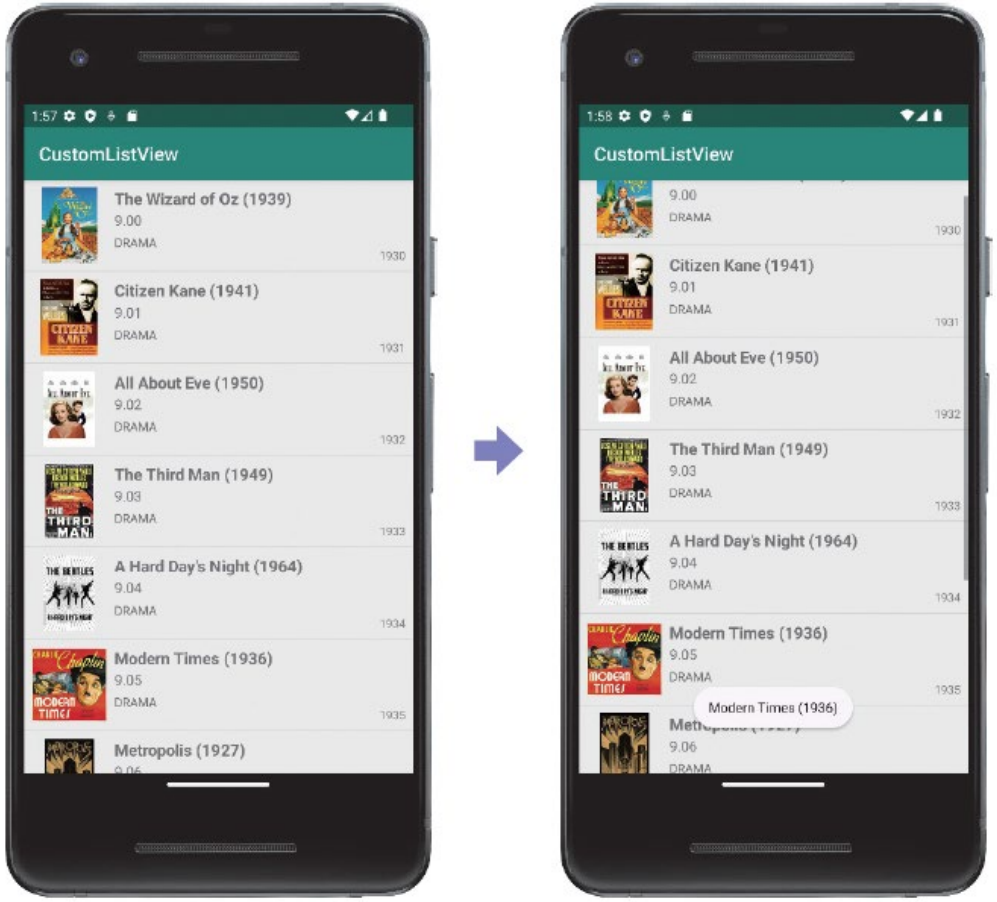
자바 소스

```
public class CustomList extends ArrayAdapter<String> {
    private final Activity context;
    public CustomList(Activity context ) {
        super(context, R.layout.listitem, titles);
        this.context = context;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent) {
        LayoutInflater inflater = context.getLayoutInflater();
        View rowView= inflater.inflate(R.layout.listitem, null, true);
        ImageView imageView = (ImageView) rowView.findViewById(R.id.image);
        TextView title = (TextView) rowView.findViewById(R.id.title);
        TextView rating = (TextView) rowView.findViewById(R.id.rating);
        TextView genre = (TextView) rowView.findViewById(R.id.genre);
        TextView year = (TextView) rowView.findViewById(R.id.releaseYear);

        title.setText(titles[position]);
        imageView.setImageResource(images[position]);
        rating.setText("9.0"+position);
        genre.setText("DRAMA");
        year.setText(1930+position+"");
        return rowView;
    }
}
}
```

내부 클래스로 정의한다.

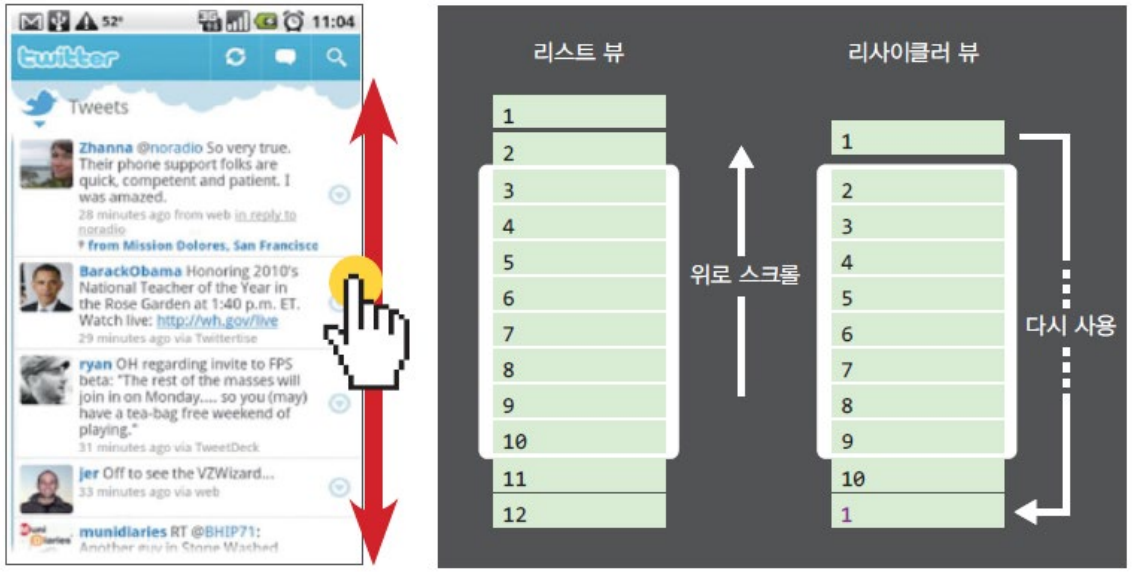
레이아웃을 팽창시켜서 뷰를 생성한다.

예제: 실행 결과


1:57
CustomListView
The Wizard of Oz (1939)
9.00
DRAMA
1930
Citizen Kane (1941)
9.01
DRAMA
1931
All About Eve (1950)
9.02
DRAMA
1932
The Third Man (1949)
9.03
DRAMA
1933
A Hard Day's Night (1964)
9.04
DRAMA
1934
Modern Times (1936)
9.05
DRAMA
1935
Metropolis (1927)
1:58
CustomListView
9.00
DRAMA
1930
Citizen Kane (1941)
9.01
DRAMA
1931
All About Eve (1950)
9.02
DRAMA
1932
The Third Man (1949)
9.03
DRAMA
1933
A Hard Day's Night (1964)
9.04
DRAMA
1934
Modern Times (1936)
9.05
DRAMA
1935
Modern Times (1936)
9.06
DRAMA

리사이클러 뷰

- 리사이클러 뷰(RecyclerView)는 리스트 뷰와 아주 유사하지만 뷰들을 재활용하기 때문에 빠르게 실행된다. 최근에 사용이 권장되고 있다.

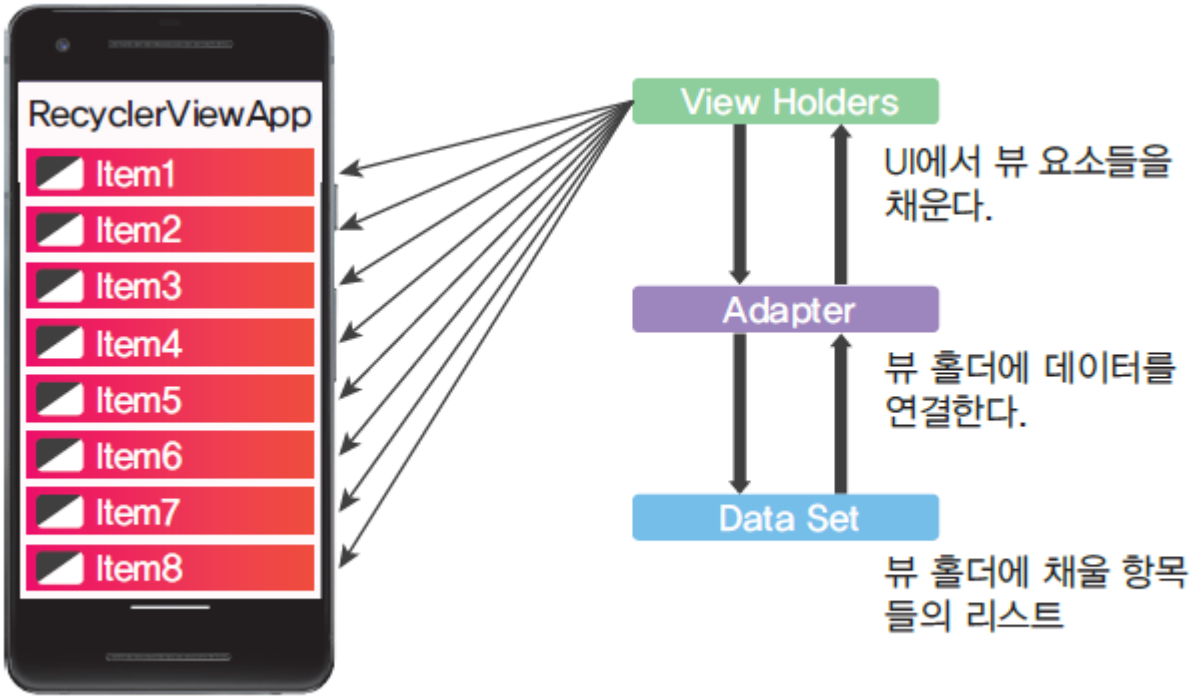


ListView의 문제점

- 리스트뷰(ListView)의 경우, 리스트 항목이 변경될 때마다, 매번 항목을 보여주는 뷰를 새로 생성해야 한다는 문제가 있다.
- 만약 데이터 집합의 크기가 큰 경우에는 성능 저하가 일어날 수 있다.
- RecyclerView 는 ListView의 개선판이다.
- RecyclerView 는 항목을 표시하기 위해 생성한 뷰를 재활용(recycle)한다.
- 뷰홀더(ViewHolder) 패턴을 사용한다.

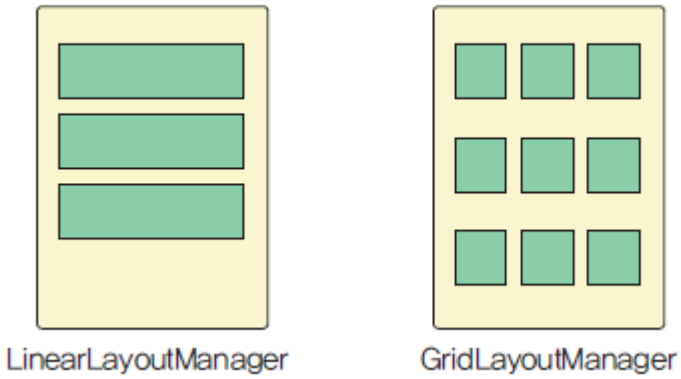
주요 클래스

- 3개의 중요한 클래스가 있다.



리사이클러 뷰

- 2가지의 배치 관리자 가능



ViewHolder 클래스와 Adapter 클래스

- ViewHolder 클래스
  - 리스트 항목의 레이아웃을 포함하는 View의 래퍼 클래스
  - RecyclerView.ViewHolder를 상속받아서 정의한다.

- Adapter 클래스
  - RecyclerView.Adapter을 상속받으며 다음과 같은 3개의 메소드를 재정의해야 한다.
  - *getItemCount() : 아이템의 개수를 반환한다.*
  - *onBindViewHolder() : 뷰에 데이터를 연결한다(바인딩).*
  - *onCreateViewHoder() : 항목을 보여주는 뷰가 들어 있는 ViewHoder을 반환한다.*

RecyclerView 구현 단계

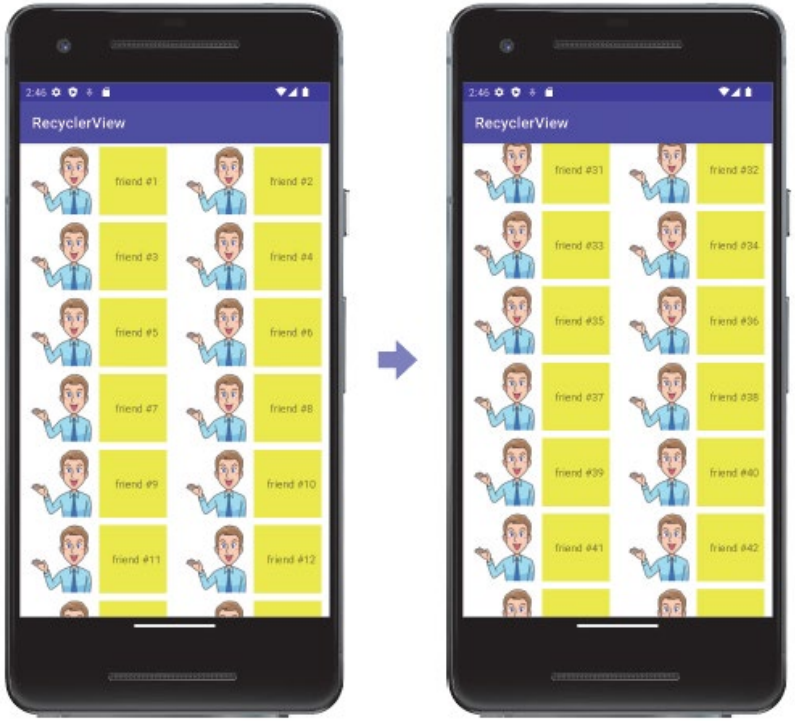
① LinearLayout과 GridLayout 중 하나를 결정한다. RecyclerView 라이브러리의 표준 레이아웃 관리자 중 하나를 사용할 수 있다.

② 리스트에 있는 각 항목의 모양과 동작 방식을 설계한다. 이 설계에 따라 ViewHolder 클래스를 상속받아서 작성한다. ViewHolder는 View 클래스의 랩퍼라고 생각하면 된다.

③ 데이터와 ViewHolder를 연결하는 Adapter를 정의한다

예제: 리사이클러 뷰 사용하기

- 리사이클러 뷰를 가진 액티비티를 간단하게 작성하고 여기에 어댑터를 연결하여서 친구들의 데이터를 그리드(격자) 형태로 표시하는 예제를 살펴보자.



예제: 리사이클러 뷰 사용하기

activity_main.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">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

전체 화면의 레이아웃

리사이클러 뷰

예제: 리사이클러 뷰 사용하기

item.xml

```
<?xml version="1.0" encoding="utf-8"?>
LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="200dp"
    android:layout_height="50dp"
    android:orientation="horizontal"
    android:padding="5dp">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="50dp"
        android:layout_height="match_parent"
        app:srcCompat="@drawable/friend" />

    <TextView
        android:id="@+id/info_text"
        android:layout_width="80dp"
        android:layout_height="match_parent"
        android:background="#FFFF00"
        android:gravity="center" />

</LinearLayout>
```

리스트의 개별 항목을 표시하는데 사용되는 레이아웃

하나의 항목은 이미지와 텍스트 뷰로 표시된다.

예제: 리사이클러 뷰 사용하기

```
public class MainActivity extends AppCompatActivity implements MyAdapter.ItemClickListener {

    MyAdapter adapter;

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

        String[] data = new String[100];
        for (int i = 1; i <= 100; i++) {
            data[i - 1] = "friend #"+i;
        }
        RecyclerView recyclerView = findViewById(R.id.rview);
        int columns = 3;
        recyclerView.setLayoutManager(new GridLayoutManager(this, columns));
        adapter = new MyAdapter(this, data);
        adapter.setClickListener(this);
        recyclerView.setAdapter(adapter);
    }

    @Override
    public void onItemClick(View view, int position) {
        Log.i("TAG", "item: " + adapter.getItem(position) + "number: " + position);    }
}
```

예제: 리사이클러 뷰 사용하기

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {

    private String[] mData;
    private LayoutInflater mInflater;
    private ItemClickListener mClickListener;

    MyAdapter(Context context, String[] data) {
        this.mInflater = LayoutInflater.from(context);
        this.mData = data;
    }

    @Override
    @NonNull
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = mInflater.inflate(R.layout.item, parent, false);
        return new ViewHolder(view);
    }
```

RecyclerView.Adapter을 상속받으며 3개의 메소드를 재정의해야 한다.

새로운 ViewHolder 객체를 하나 만들어서 반환한다.

예제: 리사이클러 뷰 사용하기

```
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    holder.myTextView.setText(mData[position]);
}

@Override
public int getItemCount() {
    return mData.length;
}

public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
    TextView myTextView;

    ViewHolder(View itemView) {
        super(itemView);
        myTextView = itemView.findViewById(R.id.info_text);
        itemView.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        if (mClickListener != null) mClickListener.onItemClick(view, getAdapterPosition());
    }
}
```

뷰홀더 안의 텍스트 뷰의 내용을 바꾼다(연결한다).

항목의 전체 개수를 반환한다.

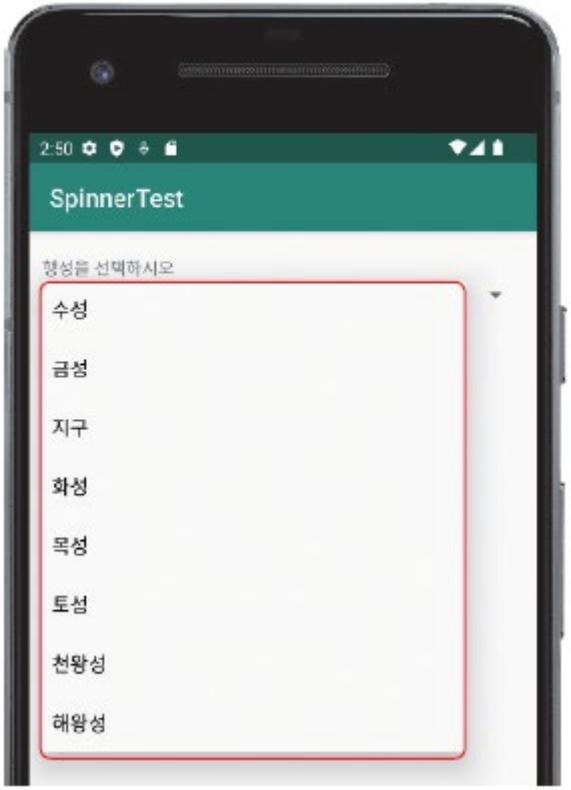
ViewHolder 클래스이다. Adaptor 클래스 안에 내부 클래스로 정의되었다.

예제: 리사이클러 뷰 사용하기

```
    String getItem(int id) {
        return mData[id];
    }

    void setClickListener(ItemClickListener itemClickListener) {
        this.mClickListener = itemClickListener;
    }

    public interface ItemClickListener {
        void onItemClick(View view, int position);
    }
}
```

스피너

- 스피너(**Spinner**)는 항목을 선택하기 위한 드롭 다운 리스트



예제: 행성들을 스피너로 표시하기

- 예제로 태양계 행성들을 표시하는 간단한 스피너 위젯을 생성하여 보자.

예제: 스피너

strings.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">SpinnerActivity</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="planet_prompt">행성을 선택하시오</string>

    <string-array name="planets_array">
        <item>수성</item>
        <item>금성</item>
        <item>지구</item>
        <item>화성</item>
        <item>목성</item>
        <item>토성</item>
        <item>천왕성</item>
        <item>해왕성</item>
    </string-array>

</resources>
```

planets_array 배열을 리소스로 정의

예제: 스피너

MainActivity.java

코드 작성

```
package kr.co.company.spinnertest;
// 소스만 입력하고 Ctrl-Shift-O를 눌러서 import 문장을 자동으로 생성한다.

public class MainActivity extends AppCompatActivity {

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

        Spinner spinner = (Spinner) findViewById(R.id.spinner);
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
                this, R.array.planets_array, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

리소스로부터 ArrayAdapter를 생성하는 메소드이다.

setDropDownViewResource(int)는 사용자가 클릭하여서 항목이 오픈될 때에 항목을 표시하는 뷰의 모양을 정의한다. android.R.layout.simple_spinner_dropdown_item은 표준적인 뷰를 가리킨다.

예제: 스피너

```
        spinner.setAdapter(adapter); // 스피너와 어댑터를 연결한다.
        spinner.setOnItemSelectedListener(new OnItemSelectedListener() {
              public void onItemSelected(AdapterView<?> parent, View view,
                   int pos, long id) {
                Toast.makeText(parent.getContext(),
                       "선택된 행성은 " + parent.getItemAtPosition(pos).toString(),
                       Toast.LENGTH_SHORT).show();
                }

                public void onNothingSelected(AdapterView<?> arg0) {
                }
        });
    }
}
```

AdapterView.OnItemSelectedListener를 구현하는 무명 클래스를 생성한다. 이것은 스피너에서 항목이 선택되었을 때 애플리케이션에 알려주는 콜백 메소드를 제공한다.

AdapterView.OnItemSelectedListener는 onItemSelected()와 onNothingSelected() 메소드를 필요로 한다. 전자는 어댑터 뷰의 항목이 선택되는 경우에 호출된다. 이 경우 토스트 메시지가 선택된 텍스트를 표시한다. 후자는 어댑터 뷰에서 선택이 사라지는 경우에 호출된다. 이 경우에는 아무것도 표시되지 않는다.

예제: 스피너


```
public void onItemSelected(AdapterView<?>
 parent,
  View view, int pos, long id)
{
  parent.getItemAtPosition(pos);
}
```

프래그먼트

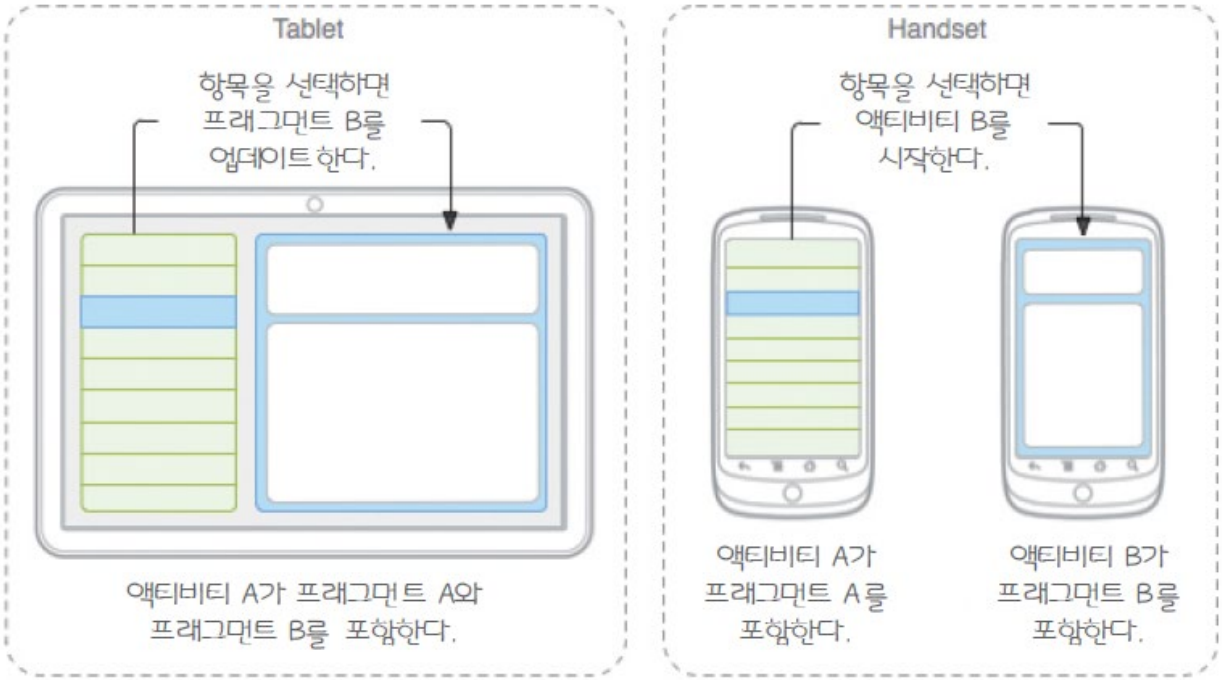
- 프래그먼트(fragment)는 안드로이드 버전 3.0부터 추가된 기능이다. 프래그먼트는 액티비티 안에 위치할 수 있는 사용자 인터페이스의 하나의 조각이라 할 수 있다. 즉 액티비티의 사용자 인터페이스를 여러 개의 조각으로 나눌 수 있고 이 조각을 프래그먼트라고 한다.

- 프래그먼트는 태블릿과 같은 넓은 화면을 가지는 모바일 장치를 위한 메커니즘이 다. 태블릿이 등장하면서 큰 화면을 액티비티 혼자서 전부 사용하기는 힘들어졌다.

- 액티비티는 일반적으로 하나의 레이아웃만을 가지기 때문에 동적으로 레이아웃을 변경하기는 상당히 힘들다.

프래그먼트

- 예를 들어, 스마트폰과 같은 화면이 작은 장치에서는, 화면에서 하나의 프래그먼트만 보여주는 것이 적절하다. 반대로 큰 화면을 가지는 태블릿에서는, 보다 많은 정보를 사용자에게 보여주기 위해서 화면 안에 프래그먼트들을 여러 개를 배치할 수 있다.

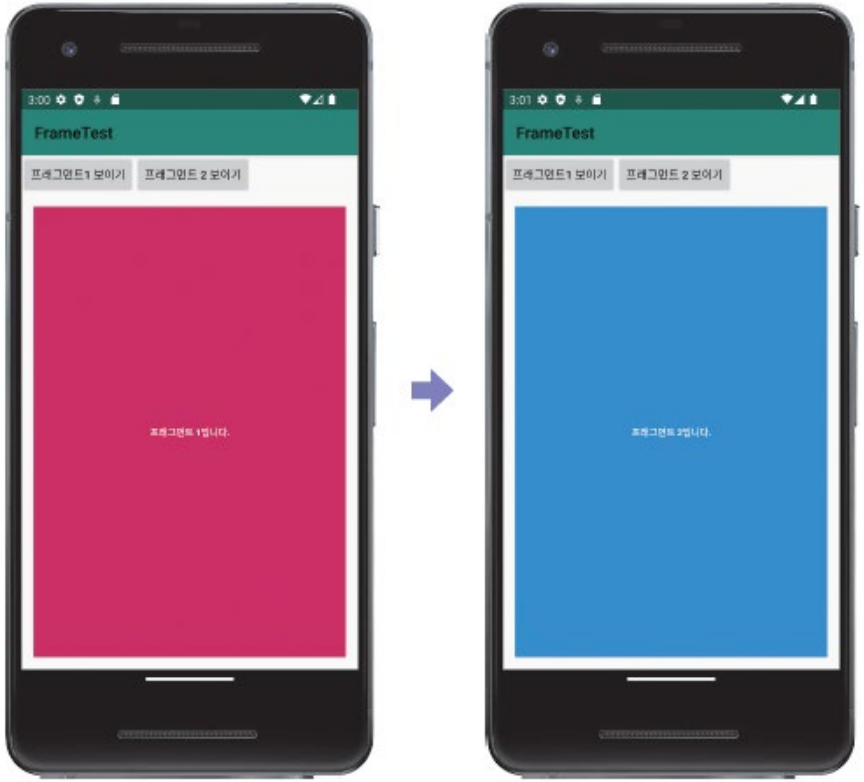


프래그먼트의 특징

- 프래그먼트는 자신만의 레이아웃을 가지고 있다.
- 프래그먼트는 자신만의 수명주기 콜백 메소드를 가질 수 있다.
- 액티비티가 실행될 때 프래그먼트를 액티비티에 추가하거나 삭제할 수 있다.
- 하나의 액티비티 안에 여러 개의 프래그먼트를 결합하여서 multi-pane UI를 구축할 수 있다.
- 하나의 프래그먼트를 여러 액티비티에서 사용할 수 있다.
- 프래그먼트의 수명주기는 호스트 액티비티의 수명주기와 연관되어 있다. 예를 들어서 액티비티가 정지되면 그 안의 모든 프래그먼트도 정지된다.

예제: 프래그먼트 사용하기

- 2개의 프래그먼트를 만들어서 버튼을 누를 때마다 Fragment1과 Fragment2가 교체되도록 하자.



예제: 프래그먼트 #1

Fragment1.java

```
package kr.co.company.frametest;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class Fragment1 extends Fragment {

    public Fragment1() {    }

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

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment1, container, false);
    }
}
```

예제: 프래그먼트 #1

res/layout/fragment1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Fragment1" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/colorAccent"
        android:gravity="center"
        android:text="프래그먼트 1입니다."
        android:textColor="#fff"
        android:textSize="30pt"
        android:textStyle="bold" />

</FrameLayout>
```

예제: 프래그먼트 #2

Fragment2.java

```
package kr.co.company.fragmentest;
// 소스만 입력하고 Alt+Enter를 눌러서 import 문장을 자동으로 생성한다.

public class Fragment2 extends Fragment {

    public Fragment2() {    }

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

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment2, container, false);
   }
}
```

예제: 프래그먼트 #2

res/layout/Fragment2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Fragment2" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#2196F3"
        android:gravity="center"
        android:text="프래그먼트 2입니다."
        android:textColor="#fff"
        android:textSize="30pt"
        android:textStyle="bold"/>

</FrameLayout>
```

예제: 메인 액티비티

res/layout-large/activity_main.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">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/button1"
            android:onClick="setFrag1"
            android:text="프래그먼트1 보이기" />

        <Button
```

예제: 메인 액티비티

```
            android:id="@+id/button2"
            android:onClick="setFrag2"
            android:text="프래그먼트 2 보이기" />

    </LinearLayout>
    <FrameLayout
        android:id="@+id/frame_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="15dp">

    </FrameLayout>
</LinearLayout>
```

프레임 레이아웃, 여기에 프래그먼트가 표시된다.

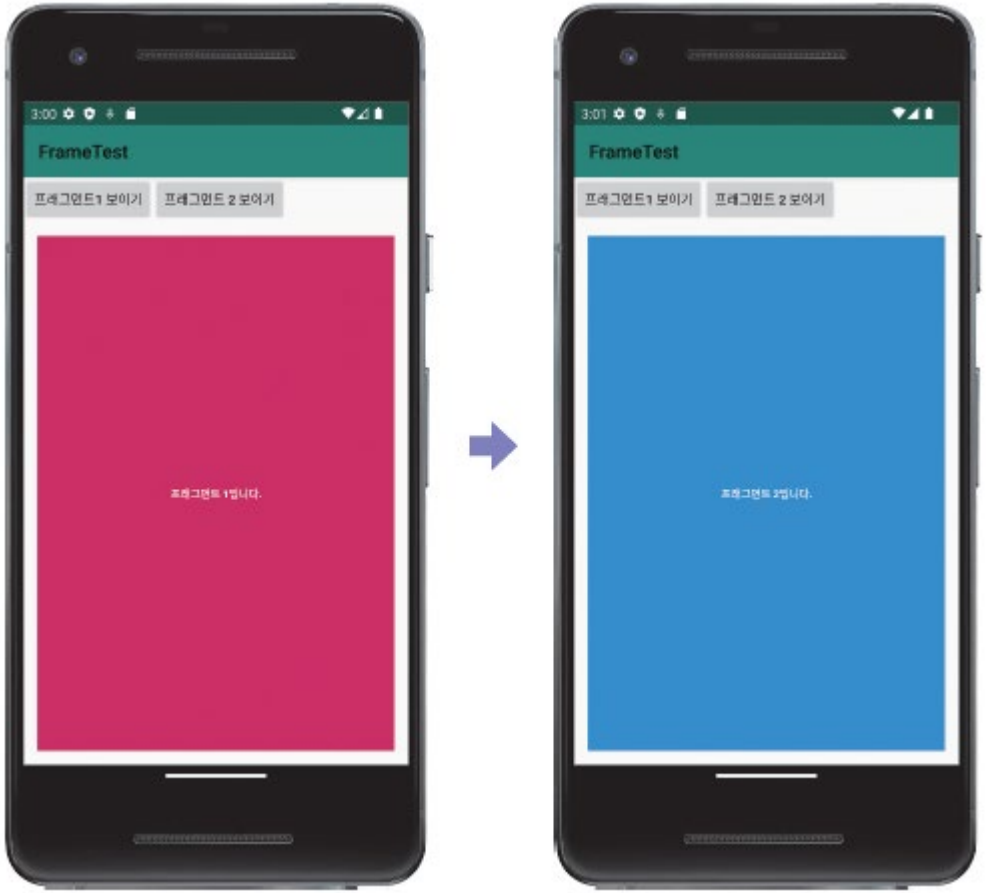
예제: 메인 액티비티

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void setFrag1(View v) {
        FragmentManager manager = getSupportFragmentManager();
        FragmentTransaction ft = manager.beginTransaction();

        ft.replace(R.id.frame_container, new Fragment1(), "one");
        ft.commitAllowingStateLoss();
    }
    public void setFrag2(View v) {
        FragmentManager manager = getSupportFragmentManager();
        FragmentTransaction ft = manager.beginTransaction();

        ft.replace(R.id.frame_container, new Fragment2(), "two");
        ft.commitAllowingStateLoss();
    }
}
```

프래그먼트 매니저를 불러
프래그먼트를 교체한다.

예제: 프래그먼트 사용하기

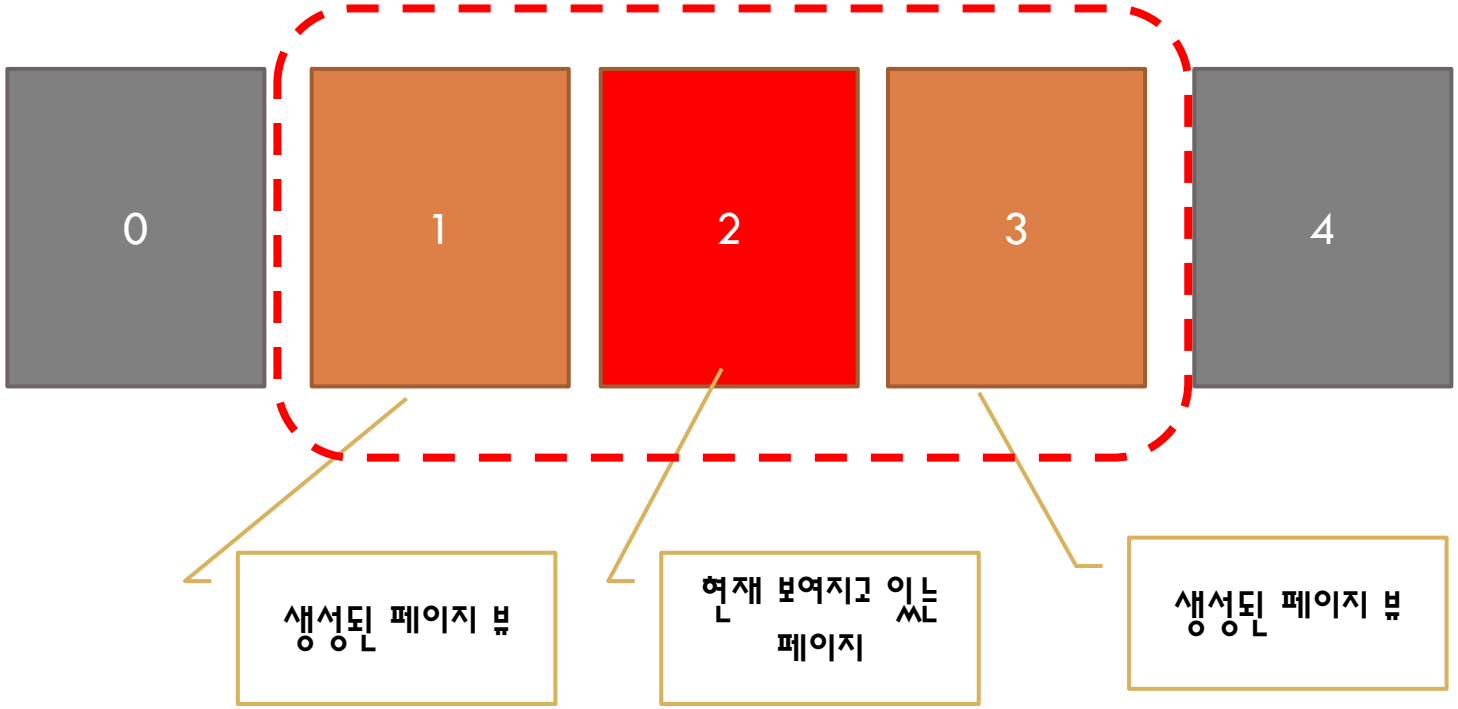


뷰 페이저

- 뷰 페이저(ViewPager)는 손가락으로 화면을 왼쪽이나 오른쪽으로 밀어서(이것을 스와이프라고 한다.) 다른 페이지로 이동할 수 있는 기능이다. 뷰 페이저는 내부적으로 프래그먼트 기능을 사용한다.

뷰페이저

- 내부적으로 3개의 페이지만 생성시켜서 사용한다.

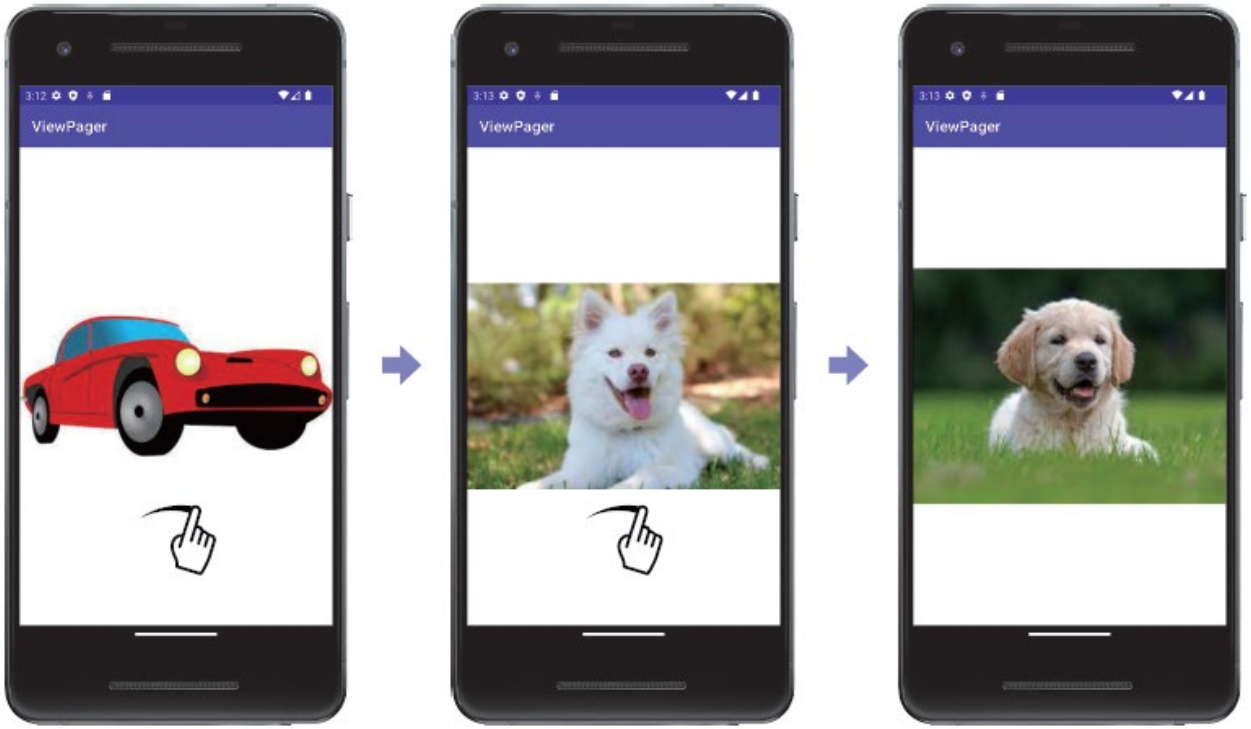


PagerAdaptor 클래스

메소드	설명
instantiateItem(ViewGroup container, int position)	position에 해당하는 페이지 생성한다.
destroyItem(ViewGroup container, int position, Object object)	position 위치의 페이지 제거한다.
getCount()	사용 가능한 뷰의 갯수를 반환한다.
isViewFromObject(View view, Object object)	페이지뷰가 특정 키 객체와 연관되는지 여부를 반환한다.

예제: 뷰페이저 사용하기

- 화면에서 이미지 슬라이드를 보여줄 때 사용할 수 있다. 우리는 이미지를 스와이프로 교체하면서 볼 수 있는 앱을 작성해보자.



예제: 뷰페이저 사용하기

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</RelativeLayout>
```

예제: 뷰페이저 사용하기

item.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">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</LinearLayout>
```

예제: 뷰페이저 사용하기

```
public class MyPagerAdapter extends PagerAdapter {

    Context context;
    int[] images;

    LayoutInflater layoutInflater;

    public MyPagerAdapter(Context context, int[] images) {
        this.context = context;
        this.images = images;
        layoutInflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERV
    }

    @Override
    public int getCount() {
        return images.length;
    }

    @Override
    public boolean isViewFromObject(@NonNull View view, @NonNull Obje
        return view == ((LinearLayout) object);
    }
```

ViewPager와 데이터를 연결하는 어댑터 클래스이다.

전체 페이지 개수를 반환한다.

특정 페이지 뷰가 특정 키 객체와 연관되는지 여부를 체크하는 메소드, instantiateItem()이 반환하는 값과 관련이 있다.

예제: 뷰페이저 사용하기

position에 해당되는 페이지 뷰를 생성하여 반환한다.
여기서 페이지 식별을 위한 Object 객체를 반환한다.

```
    @Override
    public Object instantiateItem(@NonNull ViewGroup container, final int position) {
        View itemView = layoutInflater.inflate(R.layout.item, container, false);
        ImageView imageView = (ImageView) itemView.findViewById(R.id.imageView);
        imageView.setImageResource(images[position]);
        Objects.requireNonNull(container).addView(itemView);
        return itemView;
    }

    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        container.removeView((LinearLayout) object);
    }
}
```

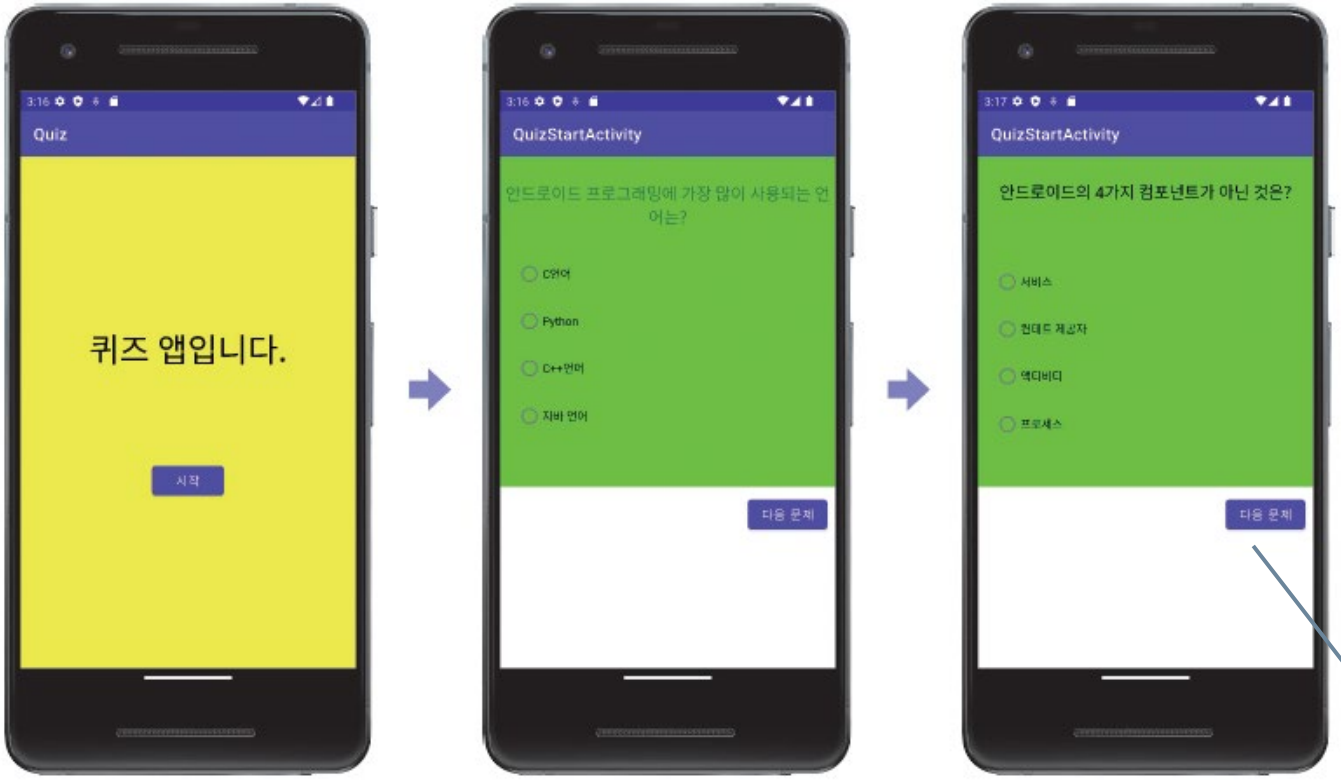
예제: 뷰페이저 사용하기

```
public class MainActivity extends AppCompatActivity {

    ViewPager viewPager;

    int[] images = {R.drawable.image1, R.drawable.image2, R.drawable.image3};
    MyPagerAdapter myPagerAdapter;

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

        viewPager = (ViewPager)findViewById(R.id.viewPager);
        myPagerAdapter = new MyPagerAdapter(MainActivity.this, images);
        viewPager.setAdapter(myPagerAdapter);
    }
}
```

뷰페이저와 데이터를 연결한다.

Coding Challenge:
프래그먼트를 이용한 퀴즈 앱 제작

- 프래그먼트를 이용하여 다음과 같은 퀴즈 앱을 작성해보자.



Q & A