


Advanced UI

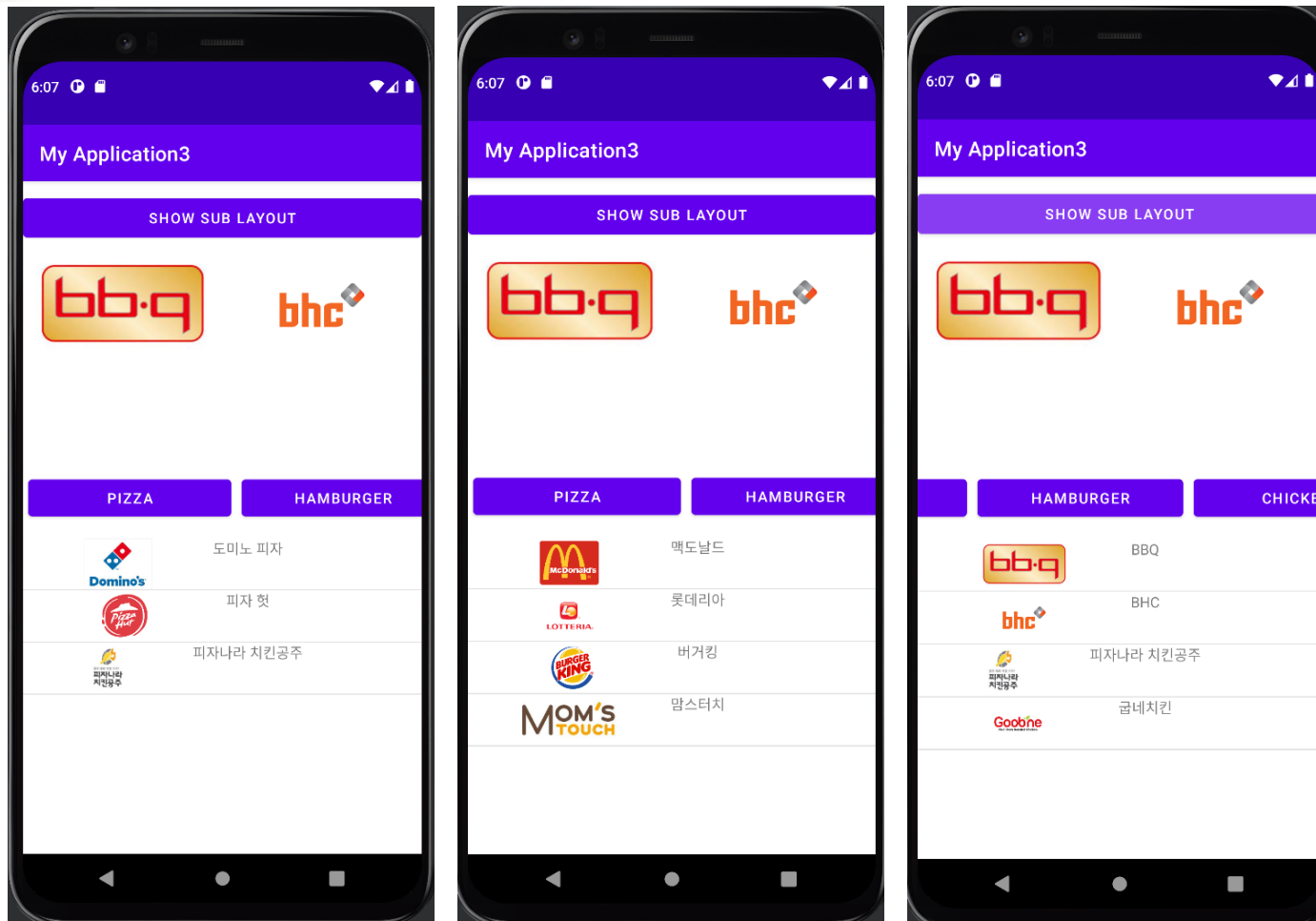
Mobile App Programming



What we learn today?

- **Let's make more complicate, intelligent activity.**
 - Split an **Activity** into multiple **Layouts**.
 - HorizontalScrollView
 - Inflater
 - Create and manage **ListView**.
 - Adapter

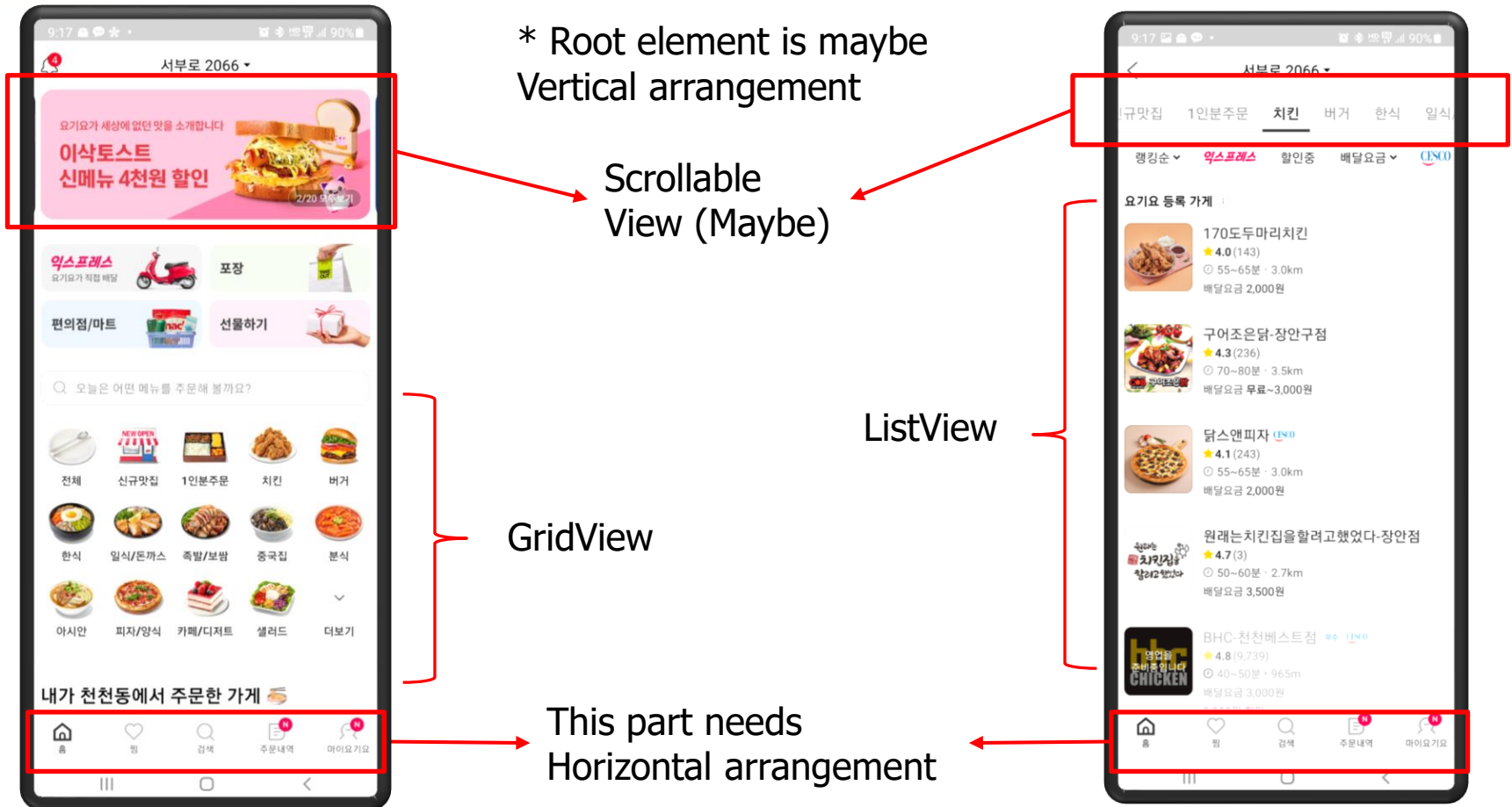
What we make today?





Multiple Layouts

Multiple Layouts





Multiple Layouts

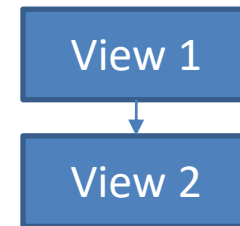
- **Sometimes, we need more than one Layout.**
 - Sometimes, we need **partial arrangement**.
 - Some place needs linear arrangement, or some place needs scrollView.
- **A layout is allowed to contain other Layouts.**
 - All Layouts are also have **constrains** and **attributes** for arrangement.
 - It actually **similar with inserting Views into a Layout**.

Constraint Layout (Default Layout)

- Similar with **relative Layout**
- View(or view group) in constraint Layout need at least 1 constraints (x-axis, y-axis).
- Normally, we can add 4 constraints
 - Top, Bottom, Start, End
 - There are many attributes,
 - `app:layout_constraint{ value(my) }_to{ value(relative) }Of = "{ id of relative }"`
 - Value = Bottom, Top, End, Start



`app:layout_constraintEnd_toStartOf="@id/view2"`



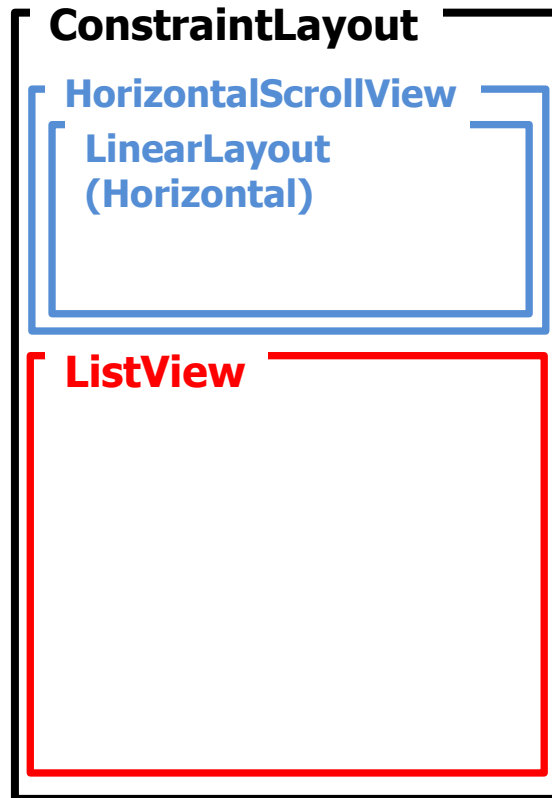
`app:layout_constraintBottom_toTopOf="@id/view2"`

HorizontalScrollView

- **Horizontally aligned** ScrollView.
- **Scrollable** but not **clickable** or **focusable**.
- **It's not Layout!**
 - Only **one View/Layout** can be located in this component.
 - Then, to put multiple Views on it, we should **insert Layout first**.

EX1) Horizontal Scroll View

- Let's insert **HorizontalScrollView** and **LinearLayout** into **ConstraintLayout**.



```
<ConstraintLayout>
  <HorizontalScrollView>
    <LinearLayout>

    ....

  </LinearLayout>
</HorizontalScrollView>

  <ListView />
</ConstraintLayout>
```

EX1) Horizontal Scroll View

```
<HorizontalScrollView
    android:id="@+id/horizontalscrollView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

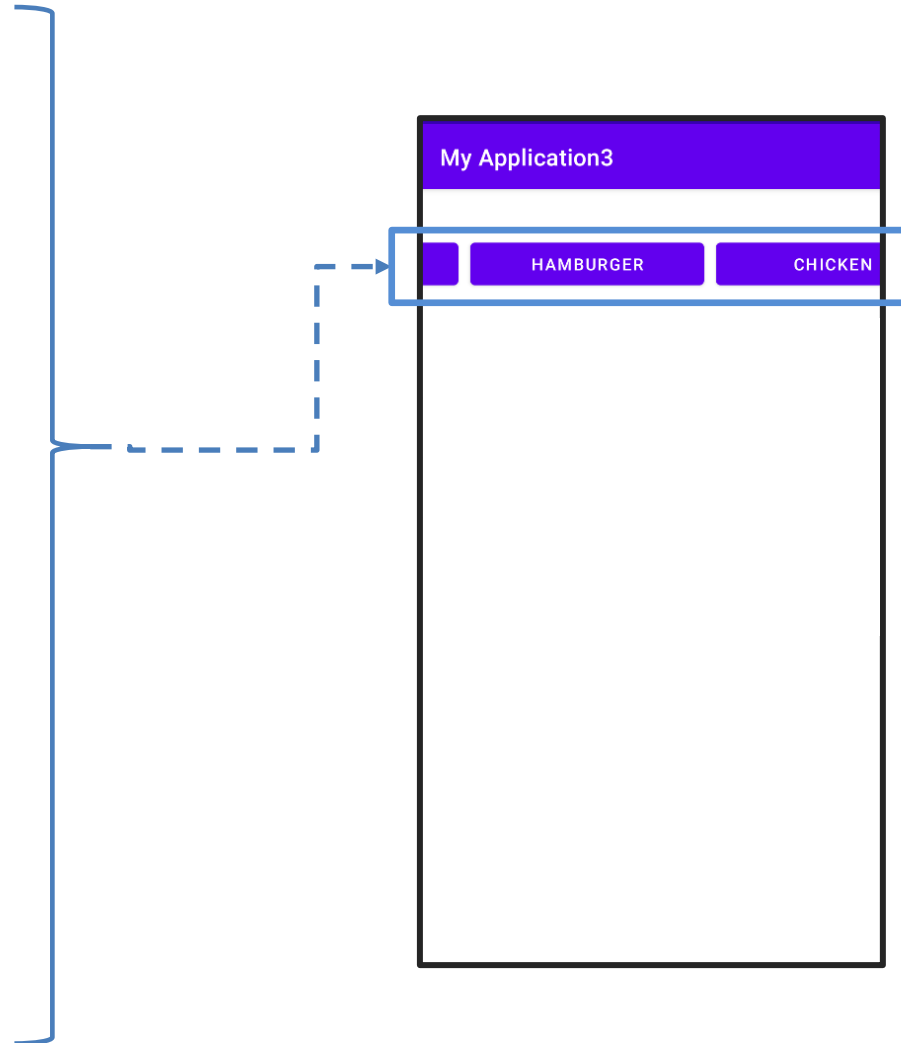
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <Button
            android:id="@+id/button1"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:text="pizza" />

        <Button
            android:id="@+id/button3"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:text="hamburger" />

        <Button
            android:id="@+id/button2"
            android:layout_width="200dp"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:layout_weight="1"
            android:text="chicken" />

    </LinearLayout>
</HorizontalScrollView>
```



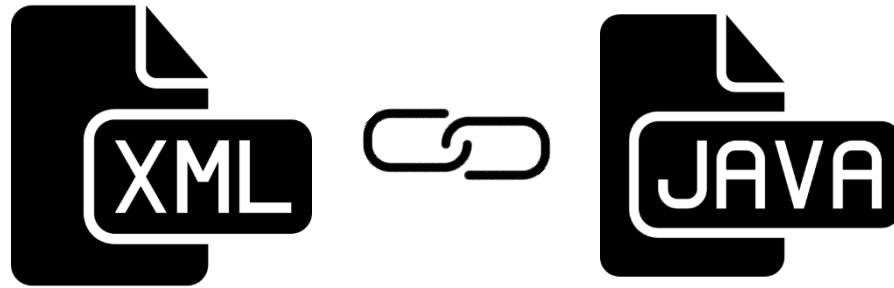


Inflation & LayoutInflater

How can JAVA class use xml resource?

- **JAVA source only can use “objectified” resources.**
- **Somehow xml resources are converted into java objects to enable JAVA access those resources.**
 - Think about “**R.id.someView**”!
 - We already use **objectified resources** when we initialize **View instance**.
 - Then how? who’s going to do those roles?
- **Think about what “setContentView” does.**

What does 'setContentView' do?

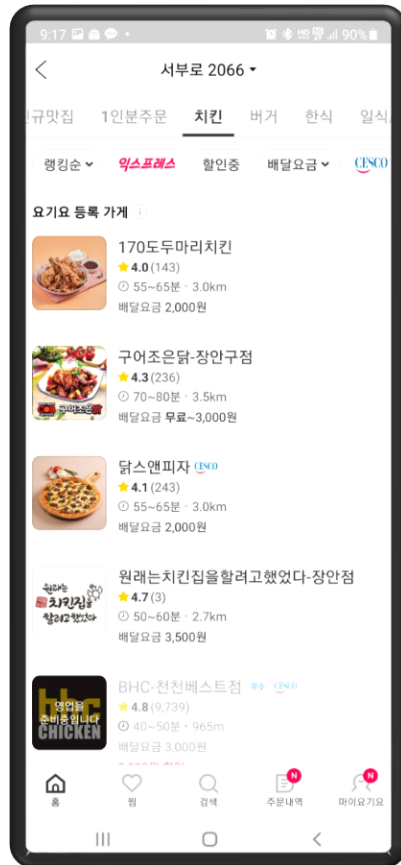


- **Android have some pairs between XML and JAVA.**
 - To use XML contents in the JAVA source codes, we need **objectified Views (or ViewGroup)**.
 - setContentView function **convert contents** defined in XML to **java object**.
- The process that **objectify** XML contents is called **Inflation**.

LayoutInflater

- Convert resources which **declared in XML** to View.
- Commonly used for generating new **View** or **ViewGroup** in JAVA source code, such as **Adapter** or **Activity**.
 - We can design our custom views by using **LayoutInflater** and new .xml files.
 - Or we can use it when we **format our list** or something else.
- **LayoutInflater** is **almost same operation** with “Inflation”

Why we need Inflator?



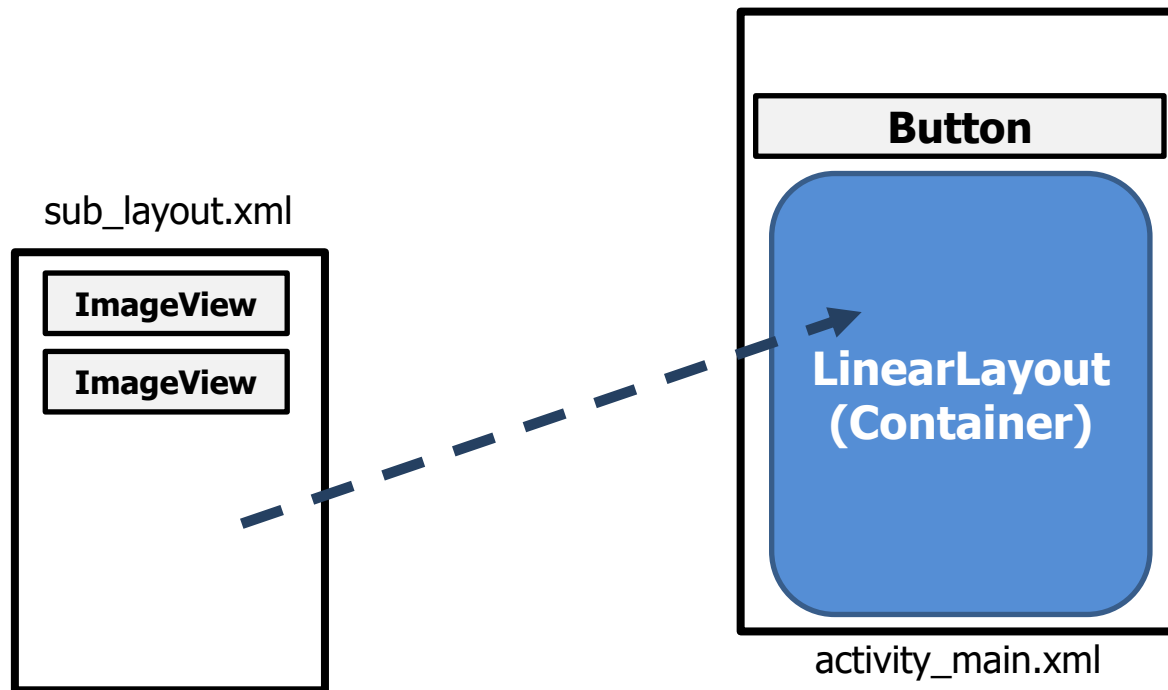
If you click category, below list view will be changed properly.

Number of Restaurants will change according to your location.

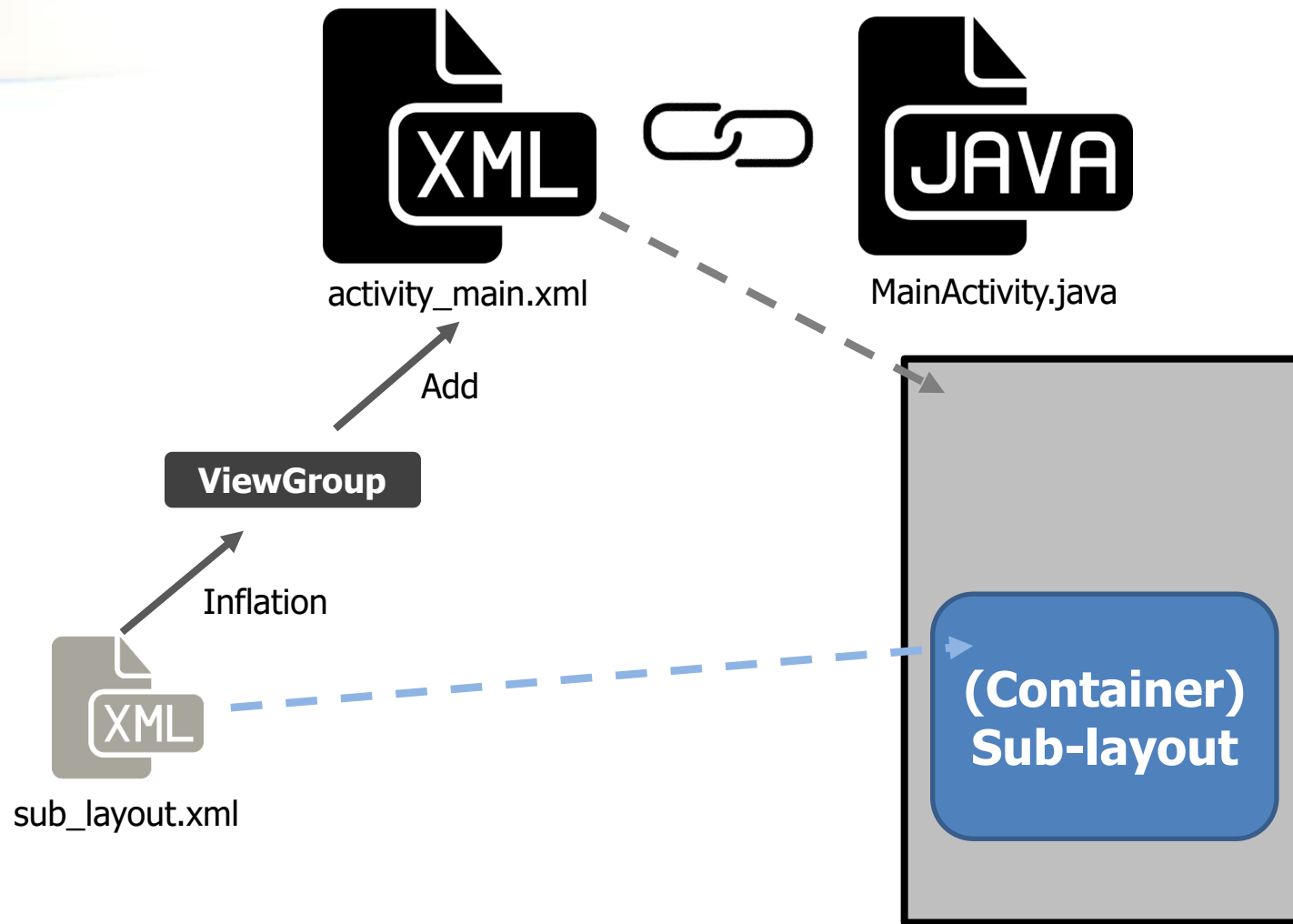
We need to change layout while runtime!

EX2) Sub Layout

- Let's make a new view with **LayoutInflater** and **XML**.
 - **activity_main.xml** declare your MainActivity's UI.
 - **sub_layout.xml** declare your own View.



EX2) Sub Layout - LayoutInflater



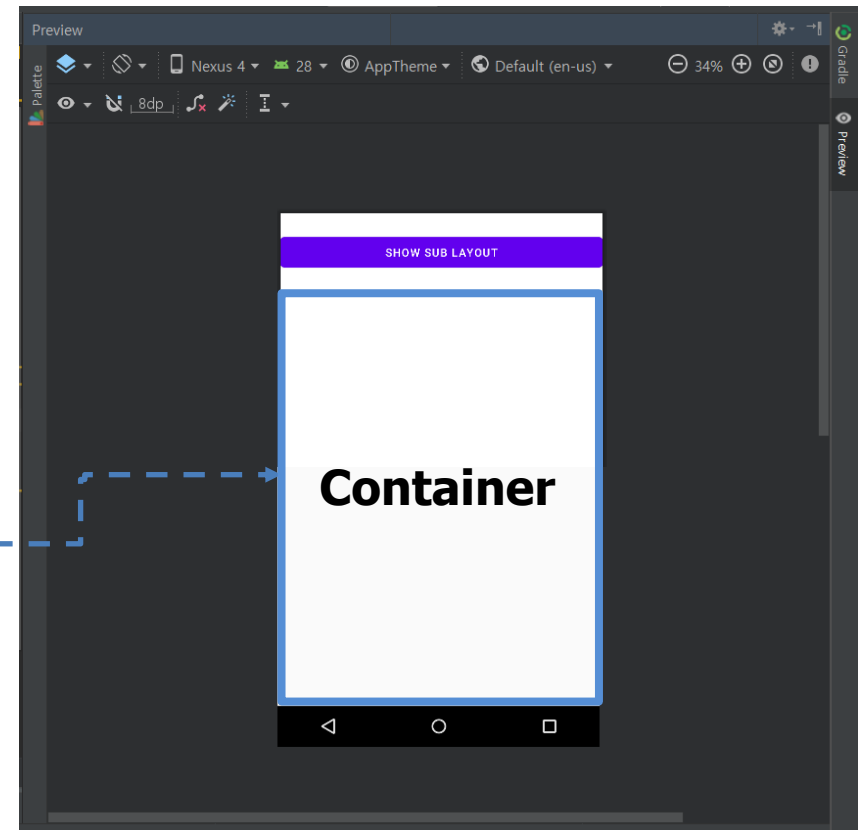
EX2) Sub Layout – activity_main.xml

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

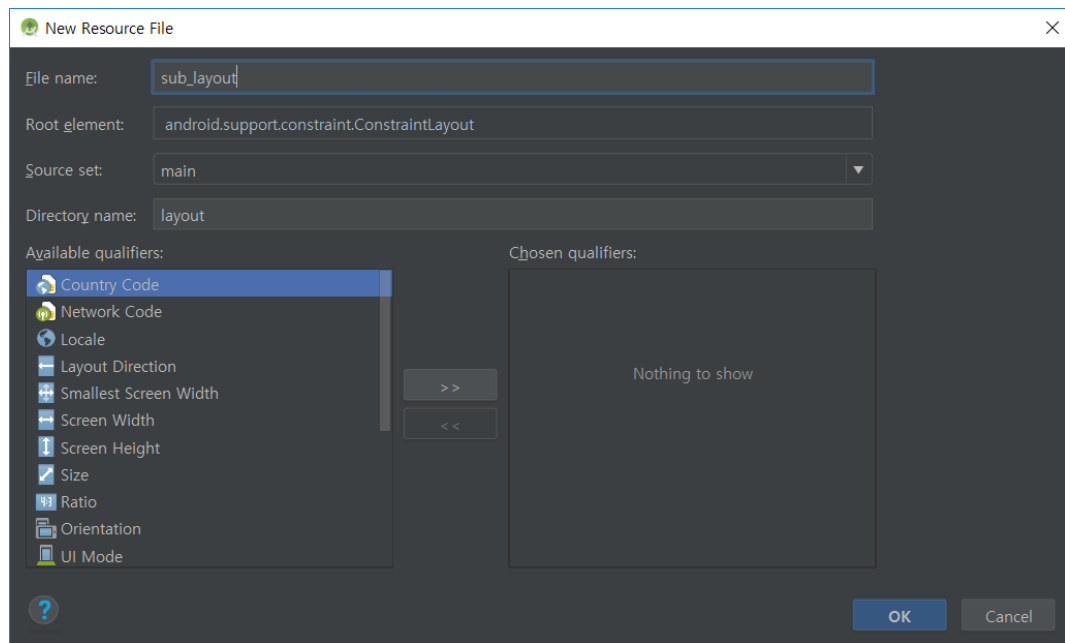
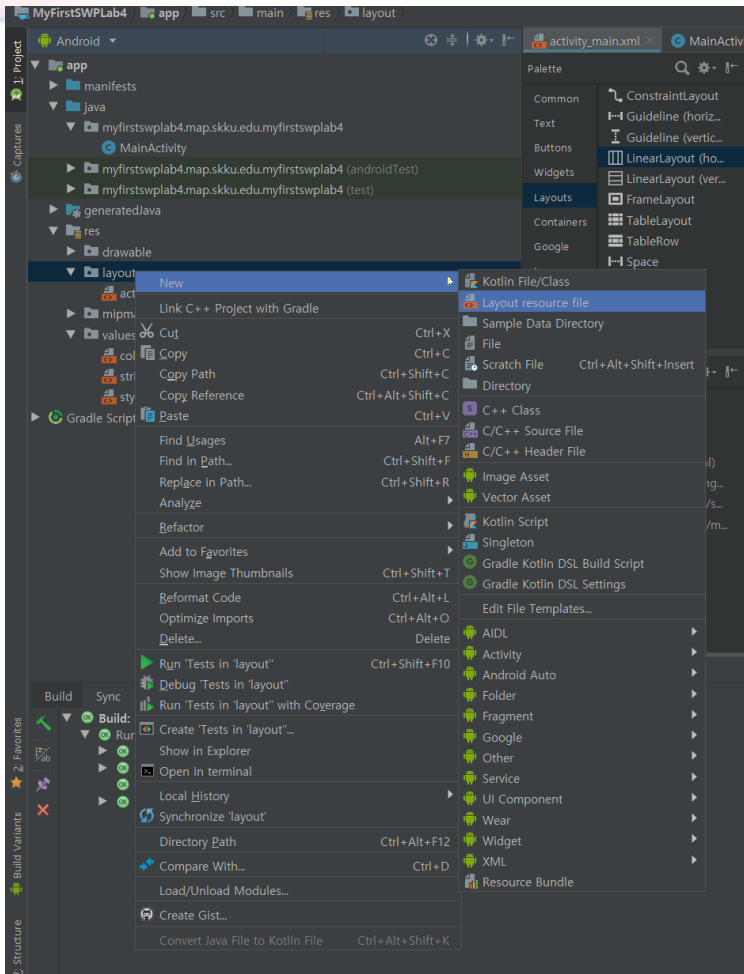
    <Button
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/linearlayout2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:id="@+id/button4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="SHOW SUB LAYOUT" />

    <LinearLayout
        app:layout_constraintTop_toBottomOf="@id/button4"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:id="@+id/linearlayout2"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:orientation="horizontal"></LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```



EX2) Sub Layout – Create sub_layout.xml



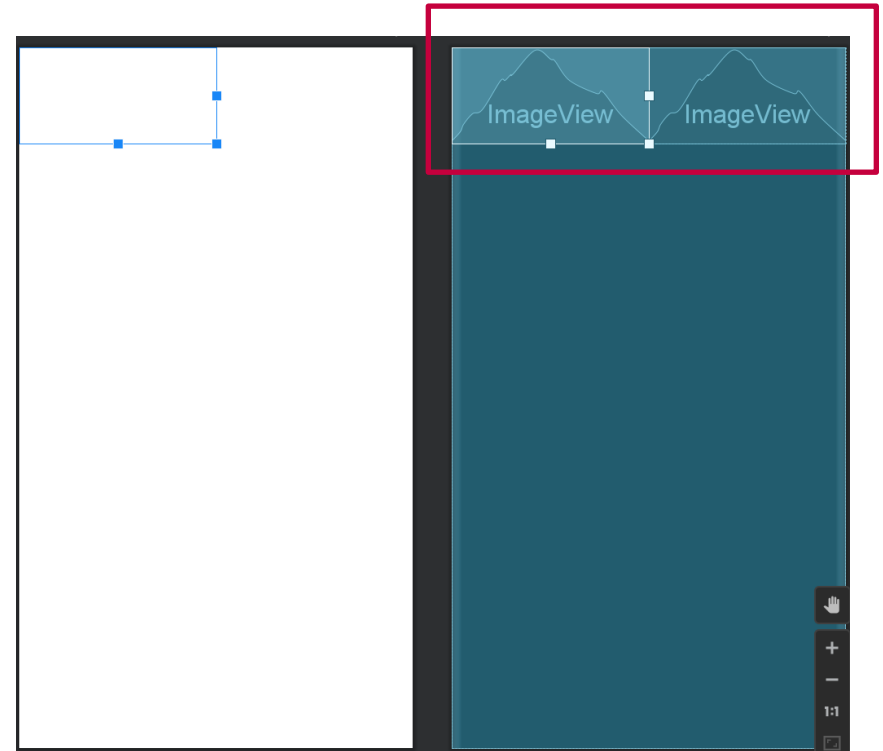
EX2) Sub Layout – sub_layout.xml

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

    android:layout_width="match_parent"
    android:orientation="horizontal"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="200dp"
        android:layout_height="100dp" />

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="200dp"
        android:layout_height="100dp" />
</LinearLayout>
```



EX2) Sub Layout – MainActivity.java

```
package com.example.week4;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

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

        LinearLayout linearLayout = findViewById(R.id.linearlayout2);
        Button btn = findViewById(R.id.button4);
```

Convert container's view to sub_layout

```
        btn.setOnClickListener(view -> {
            LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            inflater.inflate(R.layout.sub_layout, linearLayout, true);

            ImageView img1 = findViewById(R.id.imageView);
            ImageView img2 = findViewById(R.id.imageView2);

            img1.setImageResource(R.drawable.bbq);
            img2.setImageResource(R.drawable.bhc);
        });
    }
}
```






ListView (Custom)

ListView

- A ViewGroup which **groups several items** and display them **in vertical scrollable list**.
- The list items are automatically inserted to the **ListView using Adapter**.
 - Adapter pull contents from a source (e.g. DB, ArrayList etc.)
- To make ListView, you have to make
 - 1) Custom Adapter -> .java file
 - 2) Custom List Layout -> .xml file
 - 3) Set custom adapter to listview

EX3) ListView (Custom)

PIZZA		HAMBURGER	
	도미노 피자		
	피자 헛		
	피자나라 치킨공주		

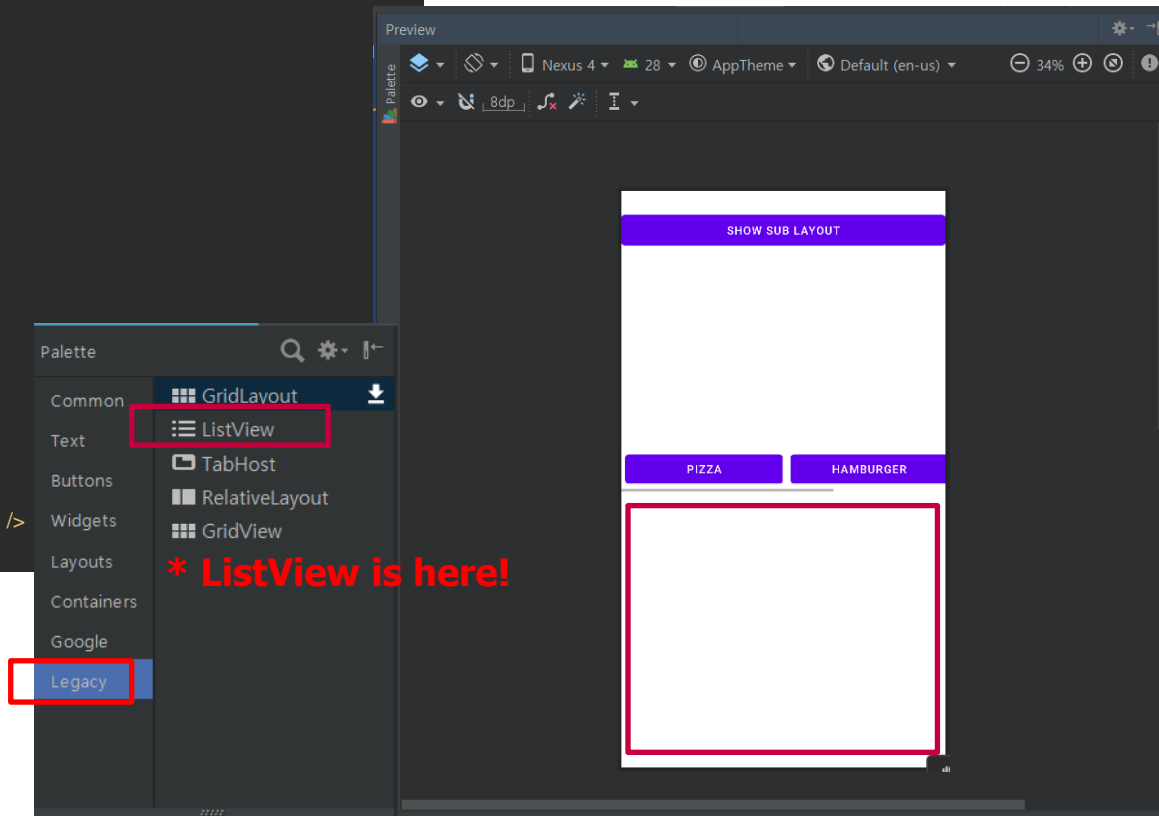
PIZZA		HAMBURGER	
	맥도날드		
	롯데리아		
	버거킹		
	맘스터치		

EX3) ListView (Custom) – activity_main.xml

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

    ....
    <HorizontalScrollView
        ....
        <LinearLayout>
            ....
            <LinearLayout>
                </HorizontalScrollView>

    <ListView
        android:id="@+id/listview1"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/horizontalscrollview1" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



EX3) ListView (Custom) – ListViewAdapter.java

1) Create New Class "Restaurant"

```
class Restaurant{
    public int id;
    public String name;
    public Restaurant(String name, int id){
        this.name = name;
        this.id = id;
    }
}
```

2) Extend "BaseAdapter"

```
package com.example.week4;

import java.util.ArrayList;

public class ListViewAdapter extends BaseAdapter {
    private ArrayList<Restaurant> items;
    private Context mContext;
```

3) Make ArrayList to save items

```
    ListViewAdapter (ArrayList<Restaurant> items, Context mContext){
        this.mContext = mContext;
        this.items = items;
    }
```

4) Define 3 override functions

```
    @Override
    public int getCount() {
        return items.size();
    }
```

```
    @Override
    public Object getItem(int i) {
        return items.get(i);
    }
```

```
    @Override
    public long getItemId(int i) {
        return i;
    }
```

5) Define "getView" function <IMPORTANT!!>

```
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        if(view == null) {
            LayoutInflater inflater = (LayoutInflater) mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            view = inflater.inflate(R.layout.listview, viewGroup, false);
        }
        ImageView img = view.findViewById(R.id.imageView3);
        TextView text = view.findViewById(R.id.textView);

        text.setText(items.get(i).name);
        img.setImageResource(items.get(i).id);

        return view;
    }
```

→ This part change viewGroup to listview_layout.xml

→ This part generate each list element (image, string)

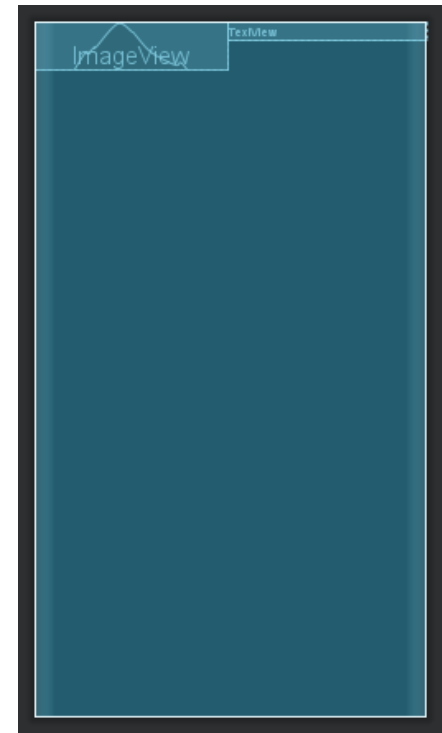
EX3) ListView (Custom) – Create listview_layout.xml

**Just ImageView and Textview
in horizontal Linear Layout**

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

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="50dp"
        android:layout_height="50dp"/>

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="TextView" />
</LinearLayout>
```



EX3) ListView (Custom) – MainActivity.java

```
package com.example.myapplication3;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private ListView listview;

    private ListViewAdapter listviewadapter;
    private ArrayList<Restaurant> items;

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

        .....

        listview = findViewById(R.id.listview1);
        items = new ArrayList<Restaurant>();
        items.add(new Restaurant("domino pizza", R.drawable.domino));
        items.add(new Restaurant("pizza hut", R.drawable.pizzahut));
        items.add(new Restaurant("pizzanarachickengongju", R.drawable.pizzanarachickengongju));
        listviewAdapter = new ListViewAdapter(items, getApplicationContext());

        listview.setAdapter(listviewAdapter);
    }
}
```

1) Define private variables

- **ListView**
- **ListViewAdapter**
- **ArrayList**

2) Connect listview to variable

3) Init ArrayList and add some items

4) Generate new ListViewAdapter

5) Set Adapter to listview

Tip) If you call setAdapter with another adapter one more, list view will change based on new adapter.

[Lab – Practice #3]

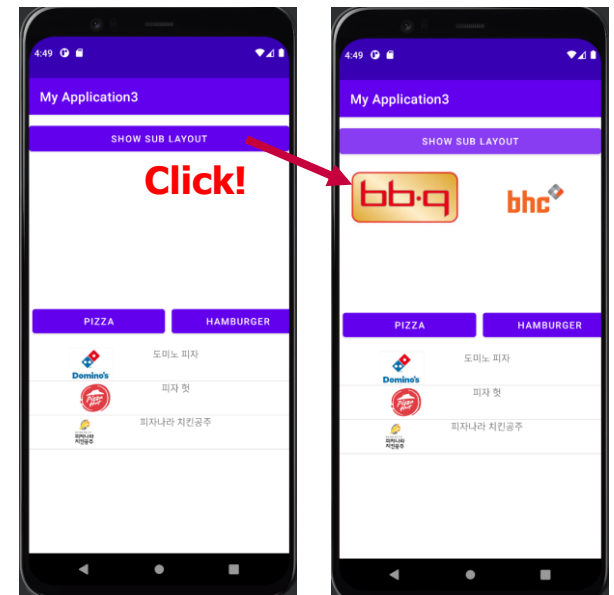
- **Let's make an application satisfying the belows.**

- 3 Contents which we learned today.

- **Sub Layout**
- Horizontal Scroll View
- List View

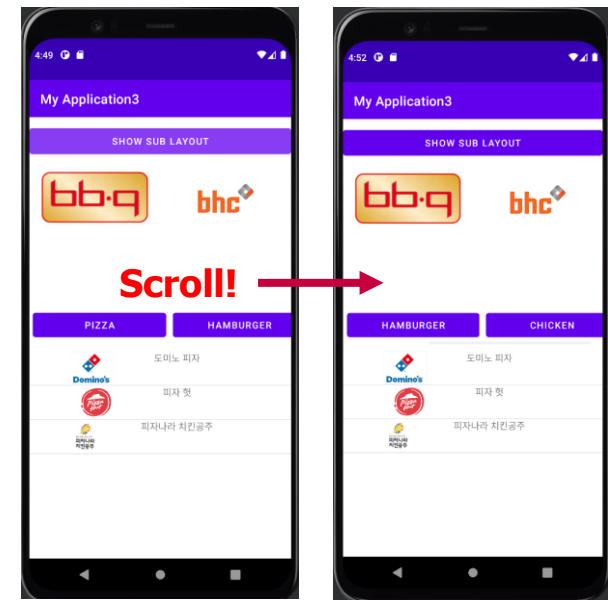
- Sub Layout

- Make 1 button "SHOW SUB LAYOUT".
- Make 1 LinearLayout (will be replaced)
- Make sublayout.xml file
- There are 2 image views in sublayout.xml
- If you click the button, LinearLayout will be changed to sublayout.xml, with 2 logos (bbq, bhc)



[Lab – Practice #3]

- **Let's make an application satisfying the belows.**
 - 3 Contents which we learned today.
 - Sub Layout
 - **Horizontal Scroll View**
 - List View
 - Horizontal Scroll View
 - Make 1 Horizontal Scroll View.
 - Put 3 button "pizza", "hamburger", "chicken".
 - Call *setOnClickListener()* functions to each button.



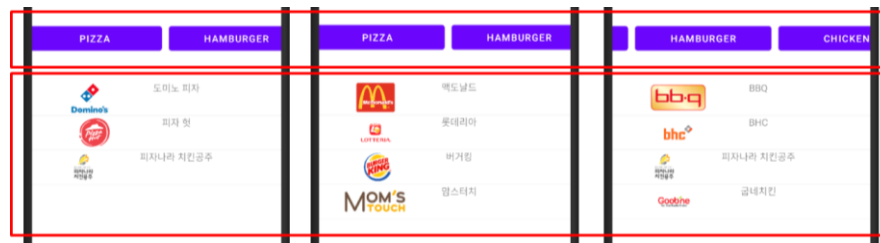
[Lab – Practice #3]

- **Let's make an application satisfying the belows.**

- 3 Contents which we learned today.

- Sub Layout
- Horizontal Scroll View
- **List View**

- List View



- **Make 1 ListView in activity_main.xml**

- Make 3 ListViewAdapter and ListArray.
- When you click the button in HorizontalScrollView, change below listview.
- Chicken – bbq, bhc, 피자나라 치킨공주, 굿네치킨
- Pizza – 도미노 피자, 피자헛, 피자나라 치킨공주
- Hamburger – 맥도날드, 롯데리아, 버거킹, 맘스터치
- Each element composed to its name(textview) and logo image(imageview) -> **Use logo.zip**

[Lab – Practice #3]

- **Submit to ICAMPUS**
 - Extract your project to studentID.zip file.
 - (File -> Export -> Extract to Zip)
 - Design Issue
 - **If we can check each function works well, design is not important in this assignment. Just make similar to right pictures.**

