
# LẬP TRÌNH THIẾT BỊ DI ĐỘNG

# VIEW - LAYOUT

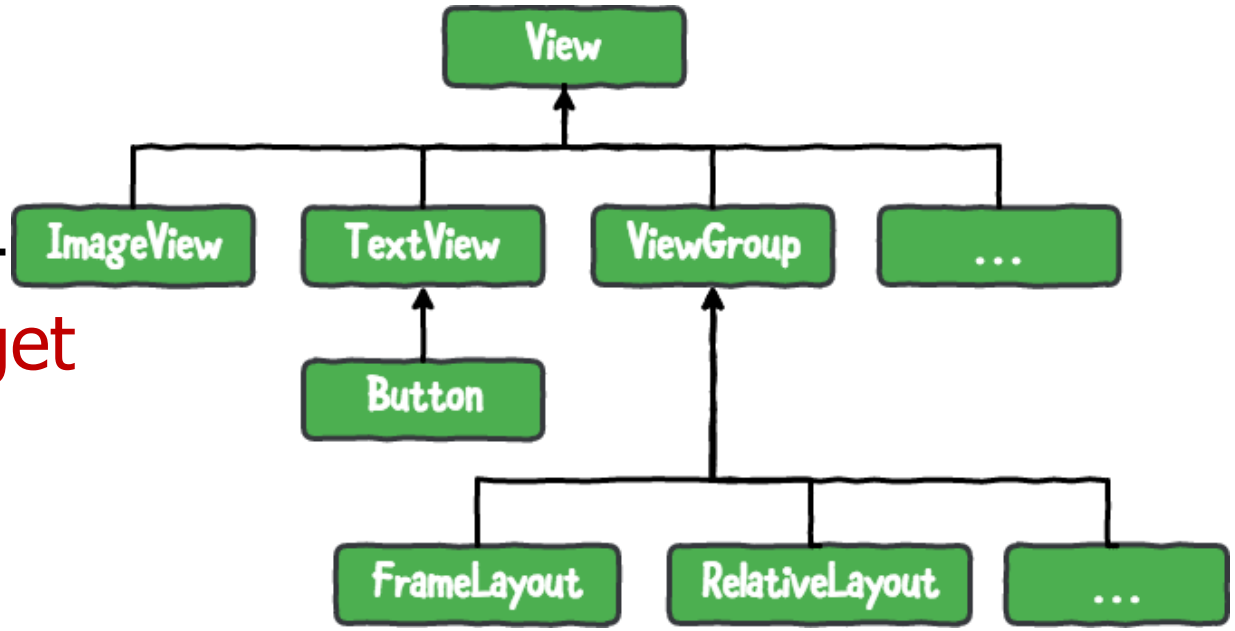
Mai Cường Thọ

# Nội dung

- Khái niệm view & viewgroup
- Làm việc với layout
- Một số layout thông dụng
- Tương tác với các điều khiển
- Một số điều khiển đơn giản

# View - Widget

- View: đối tượng cơ bản để xây dựng các thành phần giao diện
- Hầu hết các thành phần cơ bản của giao diện đều kế thừa từ View:
  - TextView, Button, Spinner, ToggleButton, RadioButton,...
- Nằm trong gói android.widget
  - nên hay gọi là widget
- Custom view:
  - widget tự tạo bằng cách tùy biến view để hoạt động theo cách của riêng mình



# ViewGroup - Layout

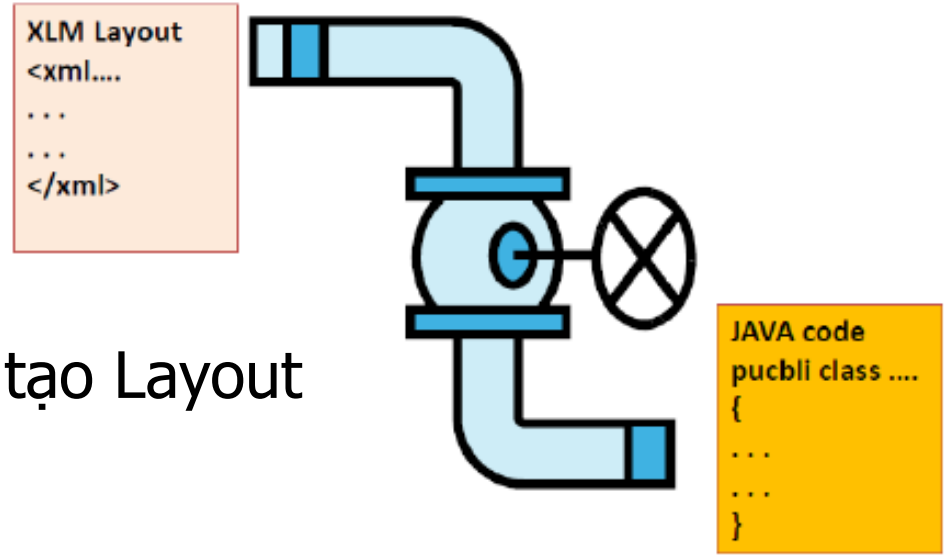
- ViewGroup: view đặc biệt, có thể chứa trong nó các view khác
  - VD1: thông tin về ngày tháng gồm một số text
  - VD2: danh sách các ngày trong tháng gồm các button
- ViewGroup là cửa sổ cha của các view con
- Một view nằm trong ViewGroup cần phải có thông tin về vị trí của nó trong cửa sổ cha
- ViewGroup = các view con + cách bố trí các view con đó bên trong
- ✍ *ViewGroup lồng nhau quá sâu làm chậm ứng dụng*

# Layout

- Layout là ViewGroup đặc biệt
  - Gồm các view con bên trong nó
  - Quy cách bố cục các view con nhất quán
  - Mỗi loại layout có quy tắc bố cục của riêng nó
- Có thể tạo layout theo 2 cách:
  - XML: soạn thông tin ở dạng XML, nạp layout từ XML bằng cách đọc từng dòng XML và tạo các thành phần phù hợp
  - Code: tạo biến layout, tạo từng biến view, đặt view vào trong layout

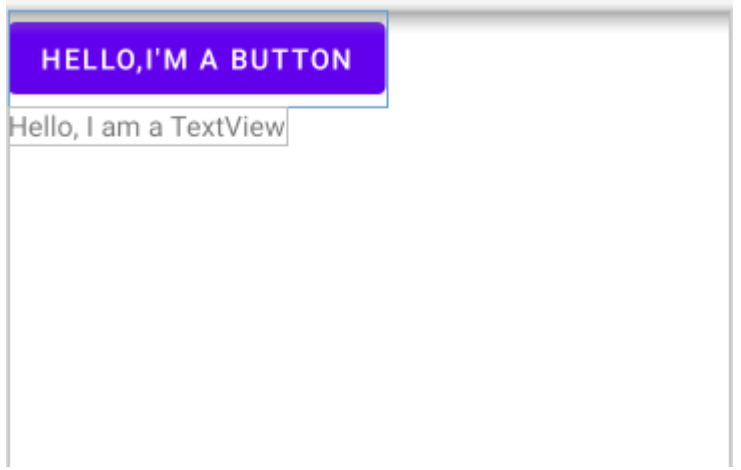
# Layout bằng XML

- Là phương pháp tạo giao diện phổ biến nhất
  - XML có cấu trúc dễ hiểu, phân cấp, giống HTML
  - Tên của thành phần XML tương ứng với lớp java trong code
- Dễ chỉnh sửa trên bằng design hoặc sửa file XML
- Tách rời giữa thiết kế và viết mã
- Thực hiện:
  - thiết kế file layoutXML
  - sau đó dùng bộ nạp LayoutInflater để tạo Layout



# Ví dụ: Layout bằng 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"
             xmlns:tools="http://schemas.android.com/tools"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             android:orientation="vertical"
             tools:context=".MainActivity">
            <Button    android:id="@+id/button"
                       android:layout_width="wrap_content"
                       android:layout_height="wrap_content"
                       android:text="Hello,I'm    a Button"  />
             <TextView
                       android:id="@+id/text"
                       android:layout_width="wrap_content"
                       android:layout_height="wrap_content"
                       android:text="Hello, I am a TextView"/>
</LinearLayout>
```

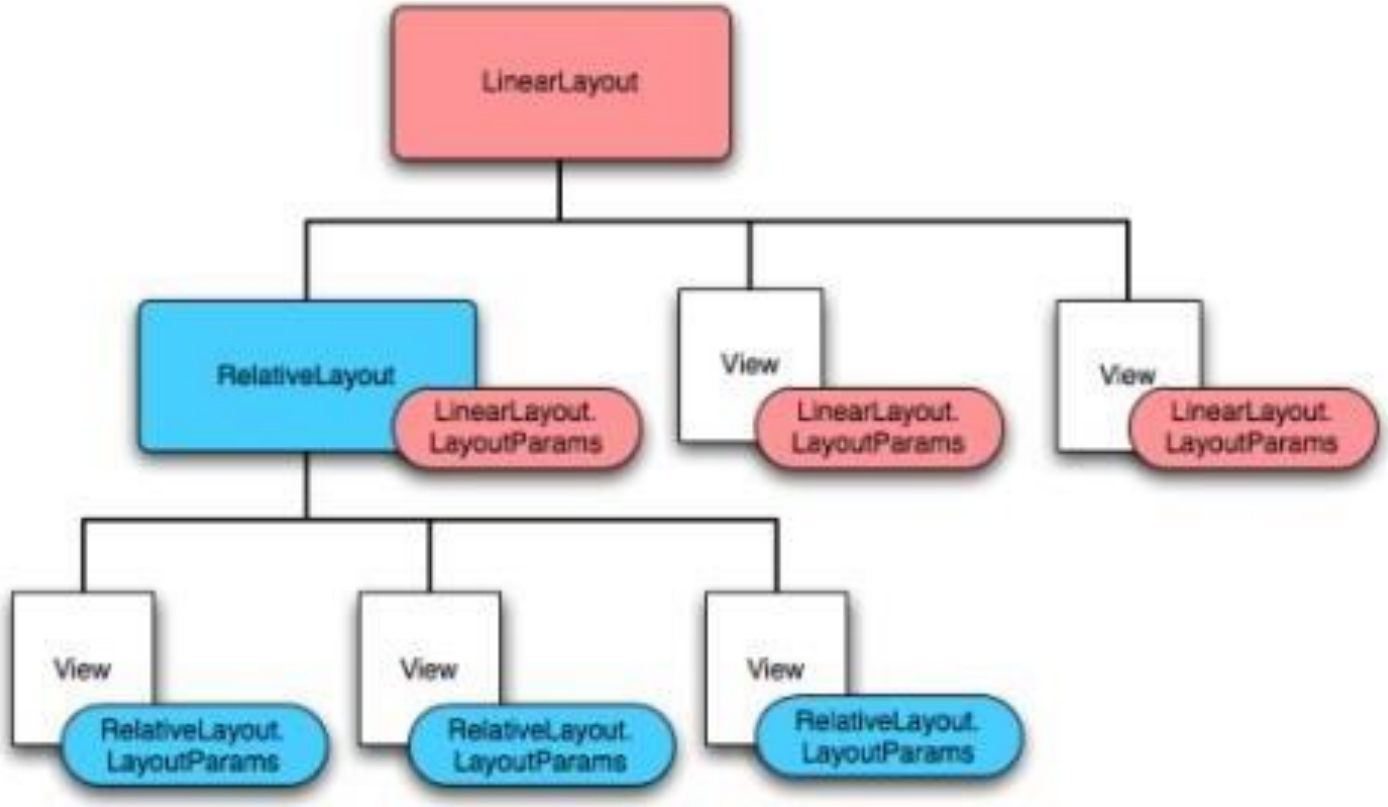


# Ví dụ: Layout bằng code

```
Button myButton = new Button(this);
myButton.setText("Press me");
myButton.setBackgroundColor(Color.YELLOW);

RelativeLayout myLayout = new RelativeLayout(this);
RelativeLayout.LayoutParams buttonParams =new RelativeLayout.LayoutParams(
                                RelativeLayout.LayoutParams.WRAP_CONTENT,
                                RelativeLayout.LayoutParams.WRAP_CONTENT );
buttonParams.addRule(RelativeLayout.CENTER_HORIZONTAL);
buttonParams.addRule(RelativeLayout.CENTER_VERTICAL);
myLayout.addView(myButton, buttonParams);
setContentView(myLayout);
```

# Layout Parameter (tham số)

- Quy định cách đặt để của view trong layout
- Mỗi view cần đính kèm LayoutParams khi đặt vào  trong Layout



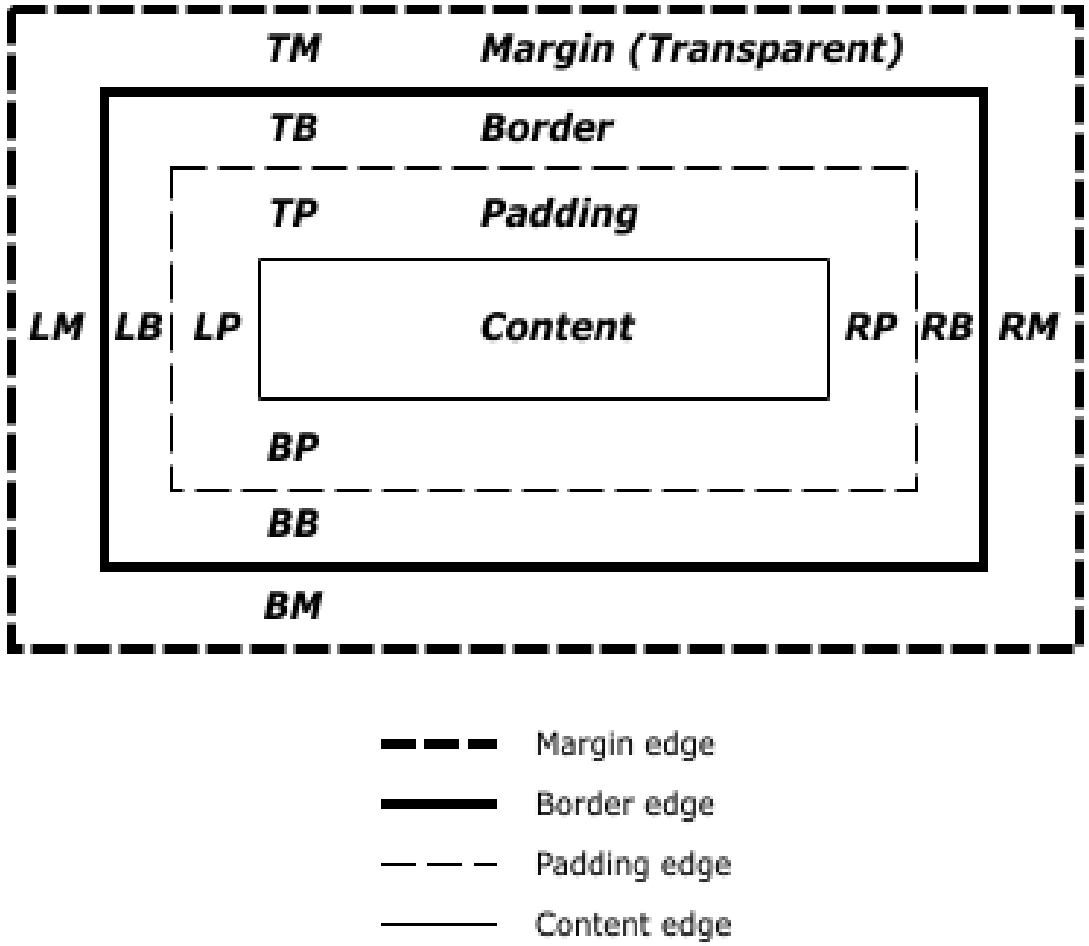
# Tham số của layout và view

Bản thân layout và view cũng có các tham số của nó khi được đặt vào view cha

- Vị trí (position): cặp tọa độ Left/Top
- Kích thước (size): cặp giá trị Width/Height
- Lề (margin): tham số trong LayoutParams (kiểu MarginLayoutParams), quy định khoảng cách của view với các thành phần xung quanh
- Đệm (padding): vùng trống từ nội dung của view ra các viền, sử dụng phương thức setPadding(int,int,int,int) để điều chỉnh, đơn vị đo thường là dp

# Tham số của layout và view..

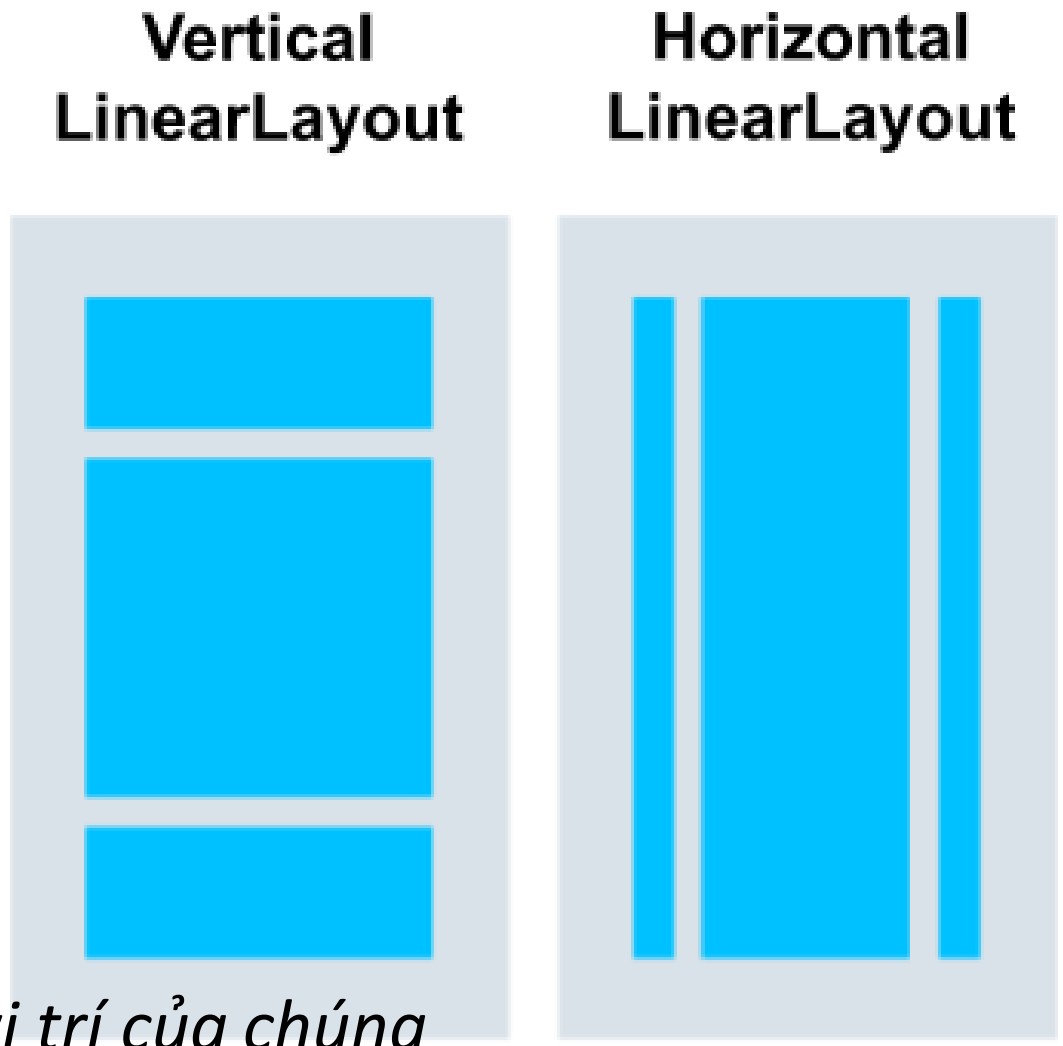
- Kích thước của view không bao gồm độ dày của margin
- Trong android không có khái niệm border
- Muốn một view có border, lập trình viên sử dụng thủ thuật thiết lập border thông qua background



# Một số Layout thông dụng

# LinearLayout

- Các view bên trong nó được xếp liên tiếp thành một hàng hoặc một cột
- Qui định bởi thuộc tính android:orientation:
  - "vertical": các view bên trong được sắp xếp theo chiều dọc
  - "horizontal": các view bên tron sắp xếp theo chiều ngang




*LinearLayout không thay đổi kích thước các view con, chỉ điều chỉnh vị trí của chúng*

# Demo LinearLayout



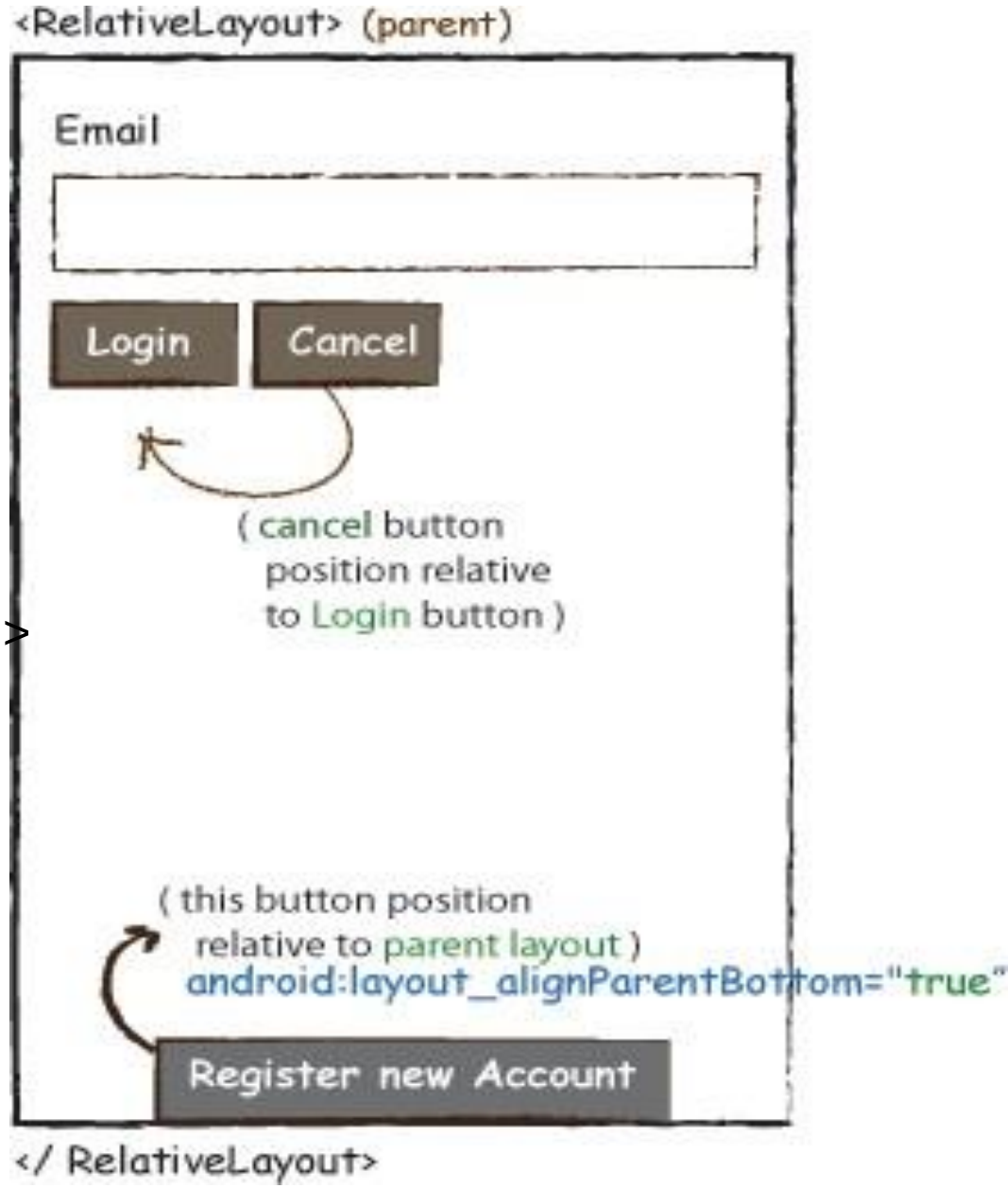
```
android:gravity="right"
```


```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Weight 2"
    android:background="#761212"
    android:layout_margin="5dp"
    android:id="@+id/button"
    android:layout_weight="2" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#761212"
    android:layout_margin="5dp"
    android:layout_weight="1"
    android:text="Weight 1" />
</LinearLayout>
```

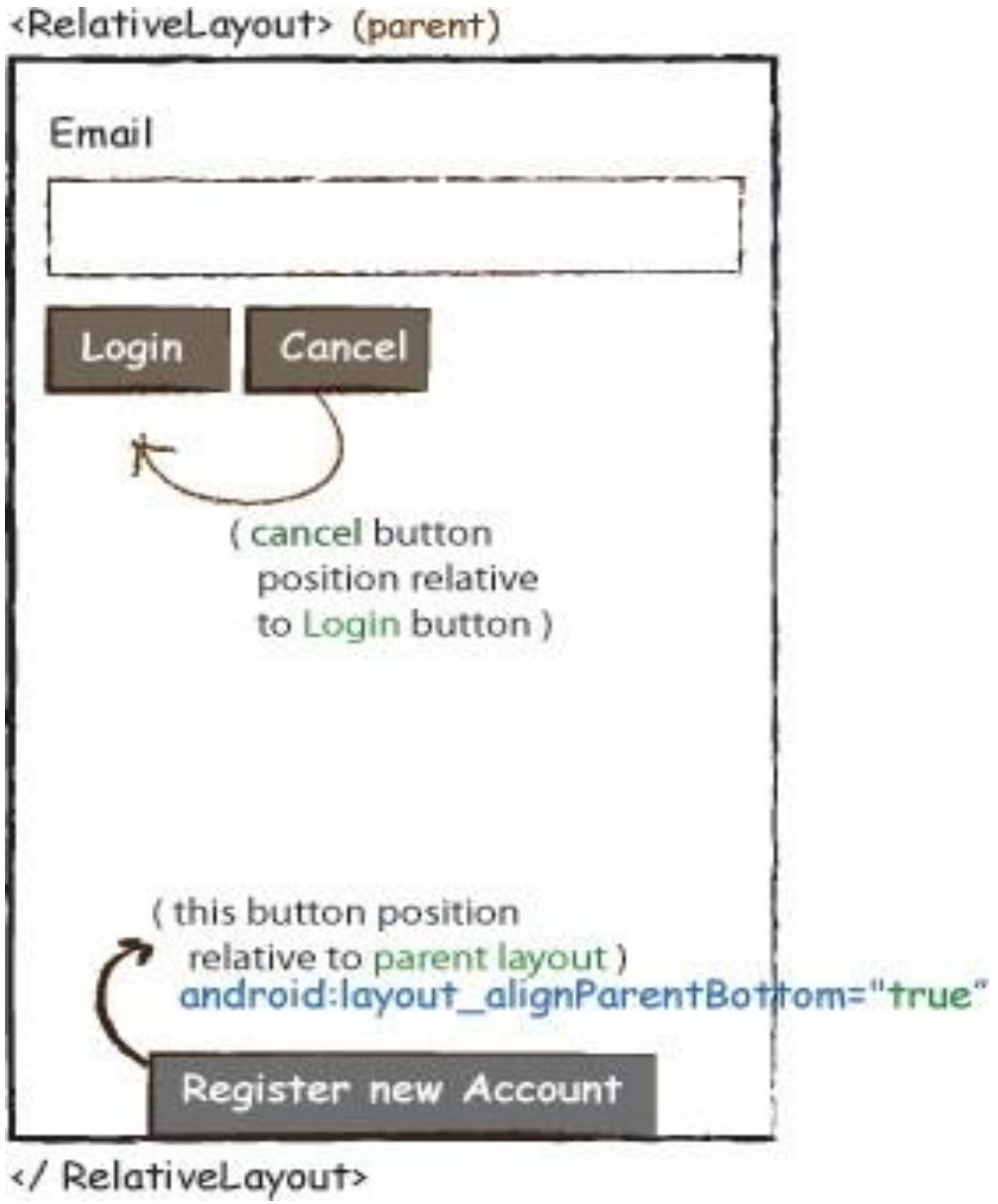
# RelativeLayout

- Là loại layout phổ biến nhất trong thiết kế giao diện
- Các view con trong layout xác định vị trí và kích thước dựa trên quan hệ với view cha hoặc các view con khác
- Dùng trong trường hợp đặt trọng tâm vào mối quan hệ giữa các thành phần
- Ý tưởng của RelativeLayout được phát triển và nâng cấp thành **ConstraintLayout**, hiện là loại layout mặc định khi thiết kế giao diện

```
<TextView  android:id="@+id/labelMail"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:text="Email" />

<EditText  android:id="@+id/inputEmail"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:layout_below="@id/labelMail " />

<Button  android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Register newAccount"
        android:layout_centerHorizontal="true"/>
```



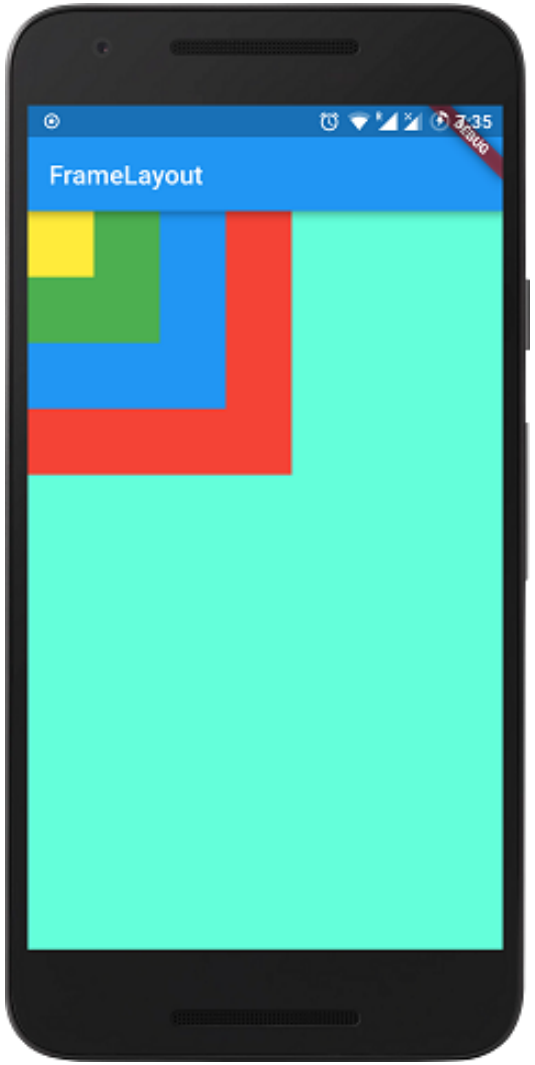
```
<Button  android:id="@+id/btnLogin"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_below="@d/inputEmail"
         android:layout_alignParentLeft="true"
         android:layout_marginRight="10px"
         android:text="Login"          />

<Button  android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_toRightOf="@id/btnLogin"
         android:layout_alignTop="@id/btnLogin"
         android:text="Cancel"         />
```



# FrameLayout

- Các view con được đặt liên tiếp chồng lên nhau, view sau đặt lên trên view trước
- Layout không tự động đổi kích thước của view con
- Có thể chuyển view con lên trên bằng code:

```
parent.bringChildToFront(child);
parent.invalidate();
```



# ScrollView & HorizontalScrollView

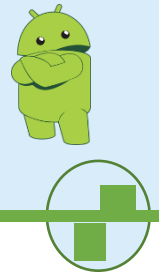
- ScrollView và HorizontalScrollView là trường hợp đặc biệt của FrameLayout
- Cho phép view con có thể có kích thước lớn hơn view cha
- Trong trường hợp
  - view con nhỏ hơn view cha,  người dùng chỉ nhìn và tương tác với view con
  - view con có kích thước lớn hơn view cha, ScrollView và HorizontalScrollView sẽ tự động xuất hiện các thanh cuộn phù hợp

# TableLayout

- TableLayout dùng để tổ chức các đối tượng view  dưới dạng một bảng gồm nhiều dòng và cột
- Mỗi dòng nằm trong một thẻ <TableRow>
- Mỗi đối tượng view đặt trên một dòng sẽ tạo thành một ô trong giao diện lưới do TableLayout tạo ra
- Chiều rộng của mỗi cột được xác định bằng chiều rộng lớn nhất của các ô nằm trên cùng một cột
- Kích thước của mỗi dòng cột không nhất thiết phải  bằng nhau

# TableLayout

- Thông thường mỗi view sẽ chiếm một ô trên lưới
- Trường hợp muốn để trống ô ta có thể đặt vào đó một textview trống (bằng thẻ <TextView />)
- Tuy nhiên ta cũng có thể chỉ định kích thước và vị trí của view thông qua các thuộc tính ẩn:
  - "android:layout_span": cỡ của view (bao nhiêu cột)
  - "android:layout_column": vị trí cột đặt view



# Contrainst layout

# Tương tác với các điều khiển

# Tương tác với các điều khiển

- Để tương tác được với các điều khiển (view), cần  làm 2 việc:
  1) Tìm đúng điều khiển cần xử lý
  2) Gọi các hàm phù hợp cho điều khiển đó (chặng hạn  như thiết lập màu chữ, nội dung hiển thị, font chữ,  các thuộc tính… hoặc xác định cách xử lý sự kiện)
- Tìm đúng điều khiển cần xử lý:
  - Nếu có một biến lưu trữ điều khiển cần xử lý thì bỏ qua
  - Nếu chưa có thì ta cần tìm điều khiển đó thông qua các hàm tìm kiếm của layout (thường là findViewById)

# Tương tác với các điều khiển ..

- Khi nạp layout từ XML, mỗi view sẽ có một mã số của riêng nó (giống như CMT), mã số này là một hằng số khai báo trong thuộc tính "android:id"
- Hàm "view findViewById(R.id.xyz)" cho phép tìm đối tượng có mã số là xyz, mã số này là một hằng số trong class con id thuộc class **R**
- Sau khi tìm được view, ta chuyển kiểu view về control đúng của nó để xử lý
- Chú ý quan trọng: mỗi lần nạp lại layout sẽ xóa toàn bộ các view cũ

# Tương tác với các điều khiển ..

```
Button btnOpen = (Button) findViewById(R.id.btnOpen);
btnOpen.setOnClickListener(new View.OnClickListener() {
     publicvoid onClick(View v) {

          //CODE KHI NÚT OPEN ĐƯỢC NGƯỜI DÙNG CLICK

}
});
```

```
//Creating TextView Variable
TextView text = (TextView) findViewById(R.id.tv);


//Sets the new text to TextView (runtime click event)
text.setText("You Have click the button");
```

# Một số điều khiển đơn giản

# TextView

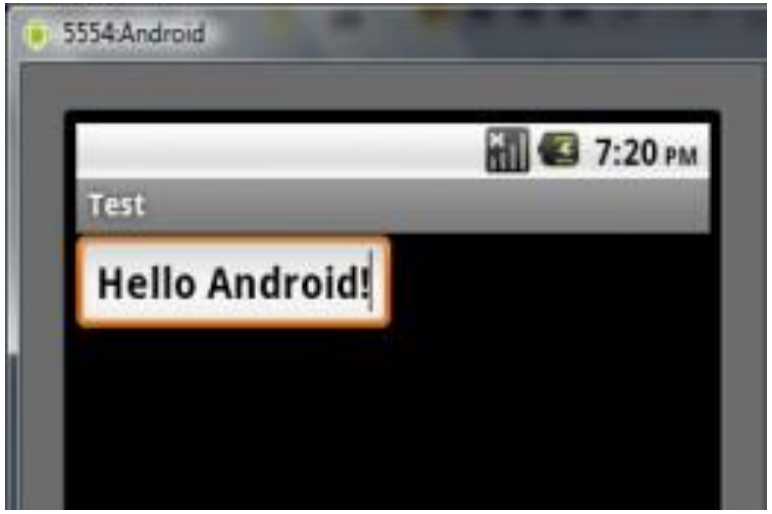
- Mục đích: hiển thị văn bản
- Một số thuộc tính hay được sử dụng:
  - android:layout_width
  - android:layout_height
  - android:text
  - android:textColor
  - android:textSize
  - android:gravity

```
<TextView
Android:layout_width="fill_parent"
Android:layout_height="wrap_content"
Android:text="Hello World! Demo TextView"
Android:textColor="#07a931"
Android:textSize="20px"
Android:gravity="center_horizontal"
/>
```

# EditText

- Mục đích: hiển thị và cho phép nhập dữ liệu
- Chú ý:
  - Thuộc tính "android:singleLine" bằng false thì EditText sẽ là một Textbox, ngược lại nó là một TextField (cho phép nhập liệu nhiều dòng)
  - Lấy nội dung: editText.getText().toString()

```
<EditText
        Android:id="@+id/EditText01"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:textStyle="bold"
        Android:textSize="20dip"
        Android:textColor="#000000"
        Android:text="Hello  Android!"
        Android:singleLine="true"
        Android:inputType="textCapWords"
        />
```

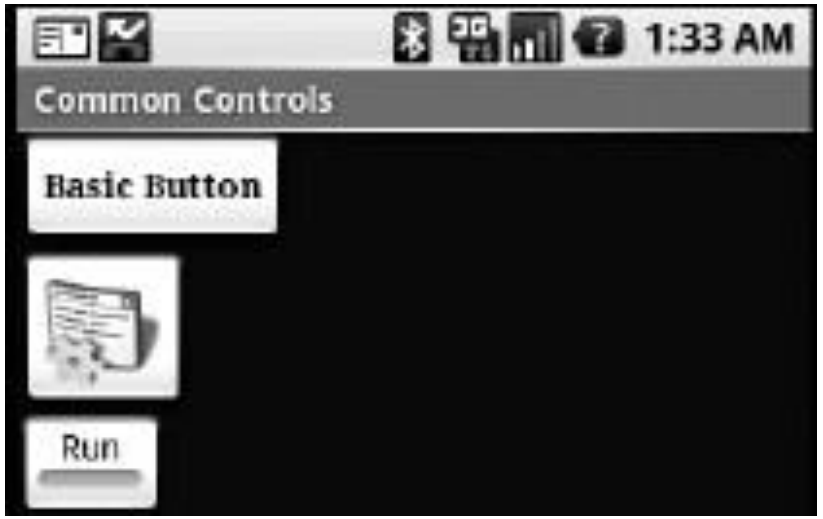


# Button & ImageButton

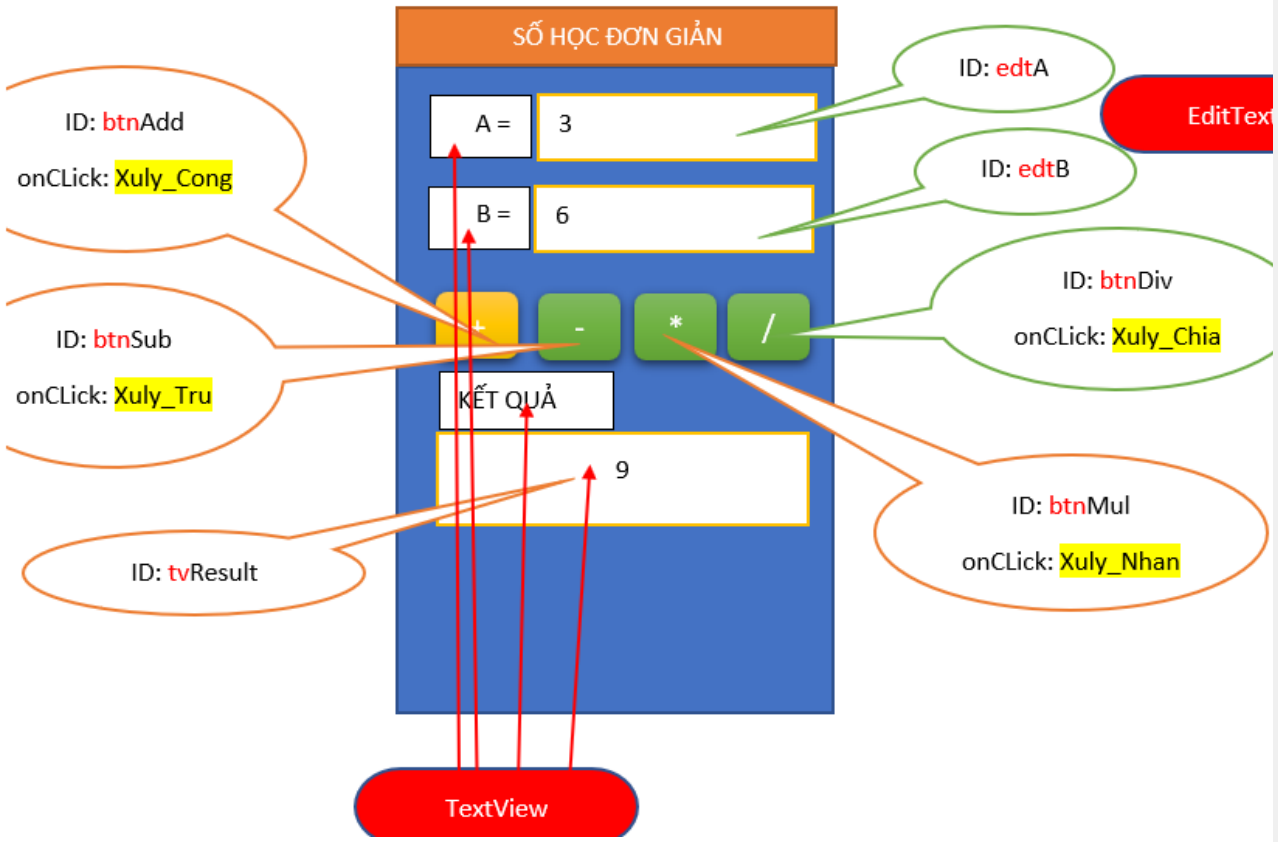
- Mục đích: nhận lệnh bấm từ người dùng
- Thuộc tính "android:**onClick**" chỉ ra phương thức sẽ  khởi chạy khi nút được bấm
- ImageButton sử dụng hình ảnh thay vì text

```
<Button
   Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:id="@+id/cmdButton1"
    Android:text="Touch me!"
/>
```

```
<ImageButtonandroid:id="@+id/btnImg1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:src="@drawable/icon"
/>
```

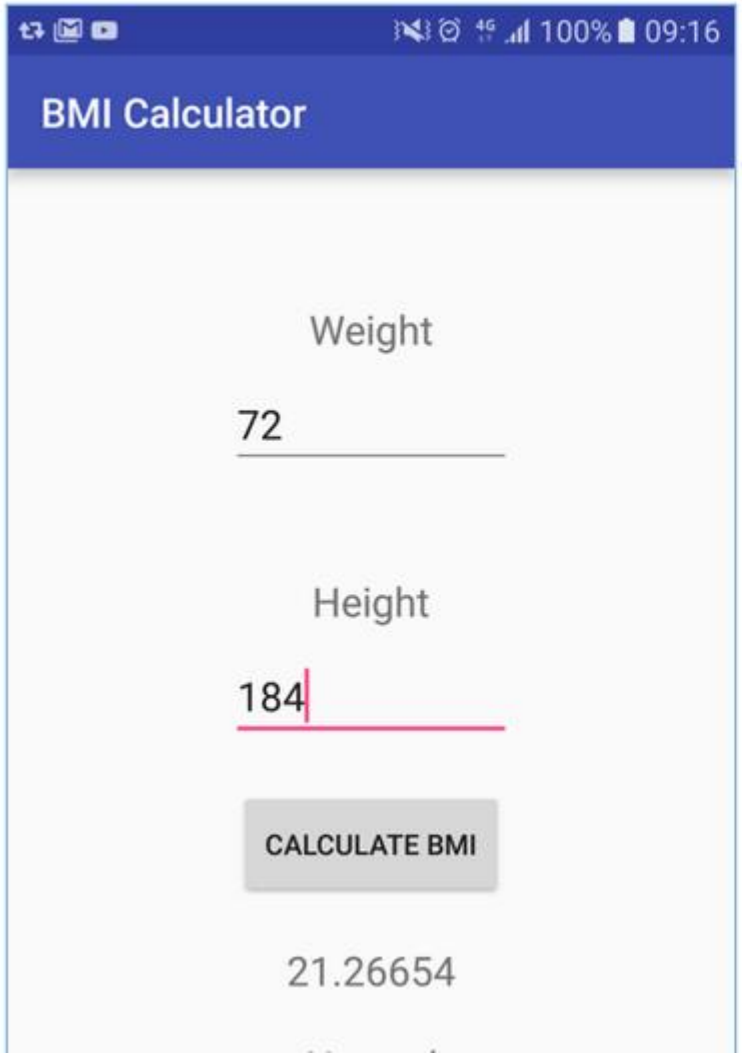


# Ví dụ: App các phép toán số học cơ bản




```
        setContentView(R.layout.activity_main);
    }

    // Hàm xử lý Cộng, hàm đáp ứng sự kiện nhấn lên nút Cộng
    void Xuly_Cong (View view)  {
        //=============================================
        // Lấy số thứ nhất A
            // Tìm điều khiển chứa dữ liệu số A
            EditText dk_soA = (EditText)  findViewById(R.id.edtA);
            // Lấy dữ liệu chứa trong điều khiển, chuyển sang kiểu số nguyên
            int soA=  Integer.parseInt(   dk_soA.getText().toString()    ) ;
        // Lấy số thứ nhất B
            // Tìm điều khiển chứa dữ liệu số B
            EditText dk_soB = (EditText)  findViewById(R.id.edtB);
            // Lấy dữ liệu chứa trong điều khiển, chuyển sang kiểu số nguyên
            int soB=  Integer.parseInt(   dk_soB.getText().toString()    ) ;
        //=============================================
            // Xử lý
            int KetQua = soA + soB;


        //=============================================
        // Hiện kết quả
            // Tìm điều khiển chứa kết quả
            TextView dk_KQ = (TextView)  findViewById(R.id.tvResult);
            // Qui định Text cho điều khiển này là kết quả tính toán được
                // Chú ý, chuyển sang kiểu chuỗi trước khi set
            dk_KQ.setText(   String.valueOf(KetQua)         );
    }
```

# Ví dụ: App tính BMI  đơn giản




$$BMI = \frac{\text{cân nặng (kg)}}{\text{chiều cao x chiều cao (m)}}$$

**Chiều cao thường được đo bằng cm, nên phải đổi bằng mét**

**VD: Cân nặng = 68kg, chiều cao = 165cm (1.65m)**

$$BMI = \frac{68}{1.65 \times 1.65} = 24.98$$

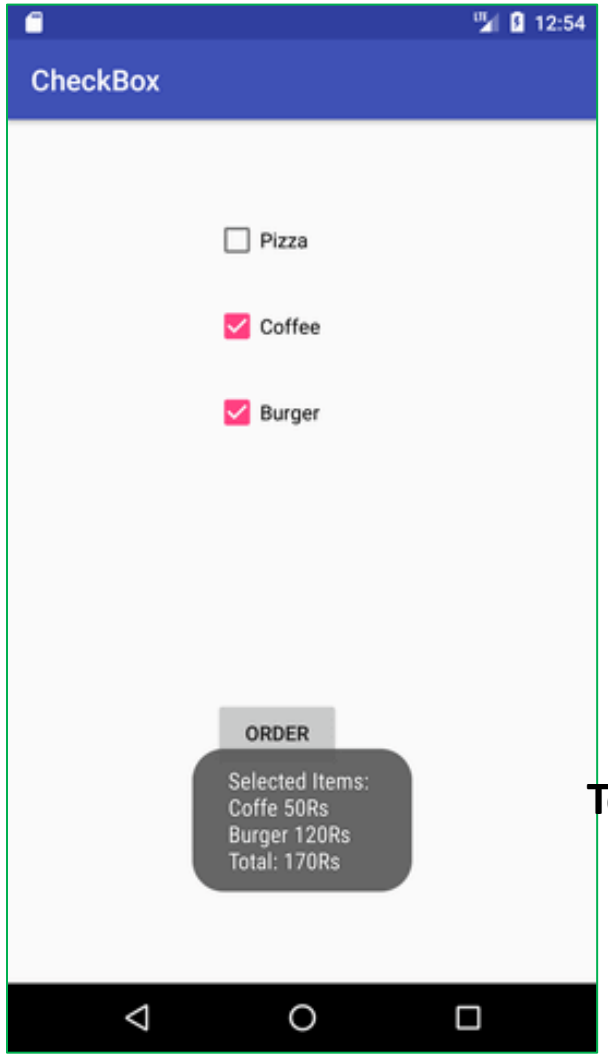
Bảng đánh giá theo chuẩn của Tổ chức Y tế thế giới(WHO) và dành riêng cho người châu Á ( IDI&WPRO):

| Phân loại | WHO BMI (kg/m2) | IDI & WPRO BMI (kg/m2) |
|---|---|---|
| Cân nặng thấp (gầy) | <18.5 | <18.5 |
| Bình thường | 18.5 - 24.9 | 18.5 - 22.9 |
| Thừa cân | 25 | 23 |
| Tiền béo phì | 25 - 29.9 | 23 - 24.9 |
| Béo phì độ I | 30 - 34.9 | 25 - 29.9 |
| Béo phì độ II | 35 - 39.9 | 30 |

# CheckBox

- Mục đích: đưa ra một ô check để người dùng có thể xác nhận có lựa chọn hay không
- Thuộc tính quan trọng
  - "android:checked": thiết lập trạng thái ban đầu
  - "android:text": nội dung đi kèm với check box
  - "android:onClick": tương tự như ở Button
- Phương thức "bool isChecked()" trả về trạng thái on/off
- Phương thức "void setChecked(bool)" để thiết lập trạng thái on/off

# Ví dụ sử dụng Checkbox

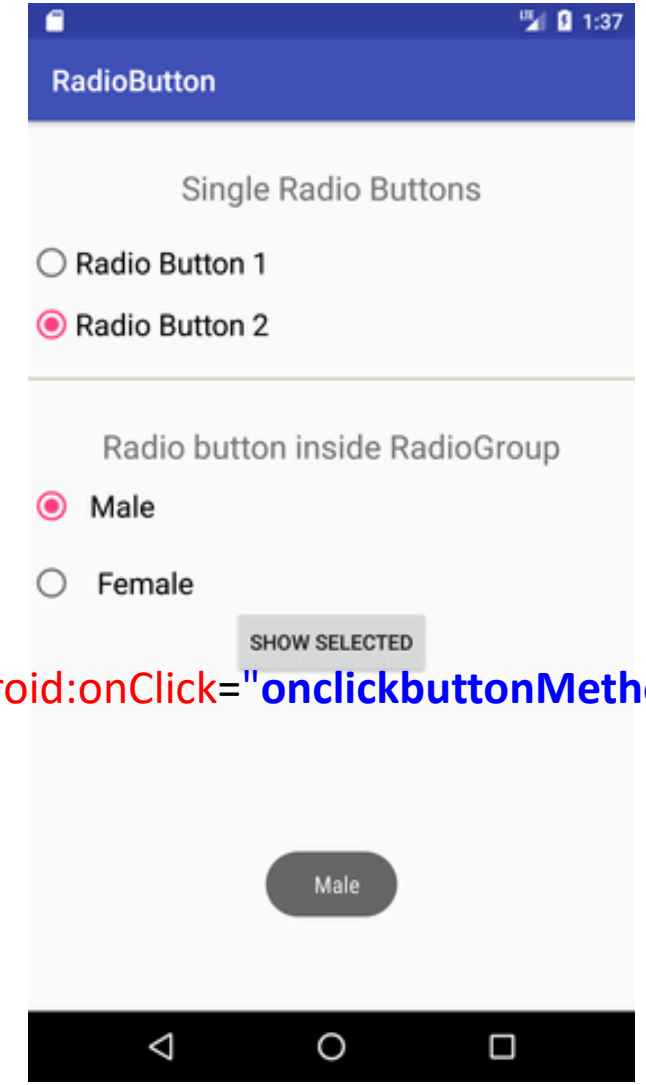



```
buttonOrder.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        int totalamount=0;
        StringBuilder result=new StringBuilder();        result.append("Selected Items:");
        if(pizza.isChecked())   {   result.append("\nPizza 100Rs");
            totalamount+=100;              }
        if(coffe.isChecked())  {       result.append("\nCoffe 50Rs");
            totalamount+=50;               }
        if(burger.isChecked()) {       result.append("\nBurger 120Rs");
            totalamount+=120;              }
        result.append("\nTotal: "+totalamount+"Rs");
        //Displaying the message on the toast
Toast.makeText(getApplicationContext(), result.toString(), Toast.LENGTH_LONG).show();
    }
} );
```

# RadioGroup & RadioButton

- RadioGroup kế thừa từ LinearLayout (mặc định là căn theo cột – vertical)
- RadioButton là các view cho phép lựa chọn on/off (dùng "bool isChecked()" để kiểm tra)
- RadioGroup đảm bảo chỉ tối đa một RadioButton được chọn vào một thời điểm
- Từ RadioGroup để lấy ID của RadioButton đang bật, dùng hàm "int getCheckedRadioButtonId()" (nếu không có RadioButton nào được lựa chọn thì hàm trả về -1)

# Ví dụ sử dụng RadioGroup/ radio button

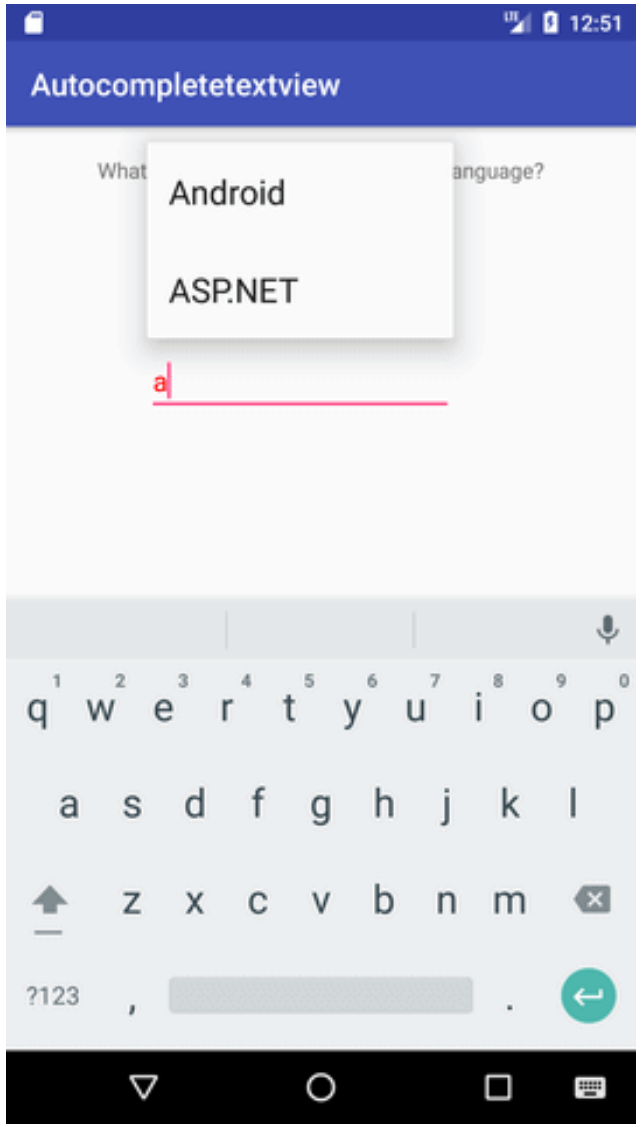



android:onClick="**onclickbuttonMethod**"

```
public void onclickbuttonMethod(View v){
    int selectedId = radioGroup.getCheckedRadioButtonId();
    genderradioButton = (RadioButton) findViewById(selectedId);
    if(selectedId==-1){
        Toast.makeText(MainActivity.this,"Nothing selected",
                                        Toast.LENGTH_SHORT).show();
    }
    else{
        Toast.makeText(MainActivity.this,genderradioButton.getText(),
                                        Toast.LENGTH_SHORT).show();
    }

}
```

# AutoComplete TextView

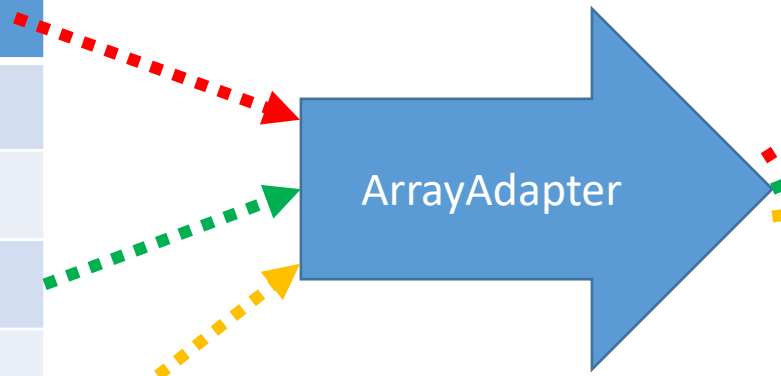
- Tương tự EditText, nhưng
- Có khả năng đưa ra một danh sách các gợi ý tương tự như phần text mà người dùng nhập vào
- Thuộc tính: "android:completionThreshold": số kí tự tối thiểu để bắt đầu hiện các gợi ý (nếu code thì dùng setThreshold)
- Dùng setAdapter để thiết lập các chuỗi gợi ý cho người nhập liệu

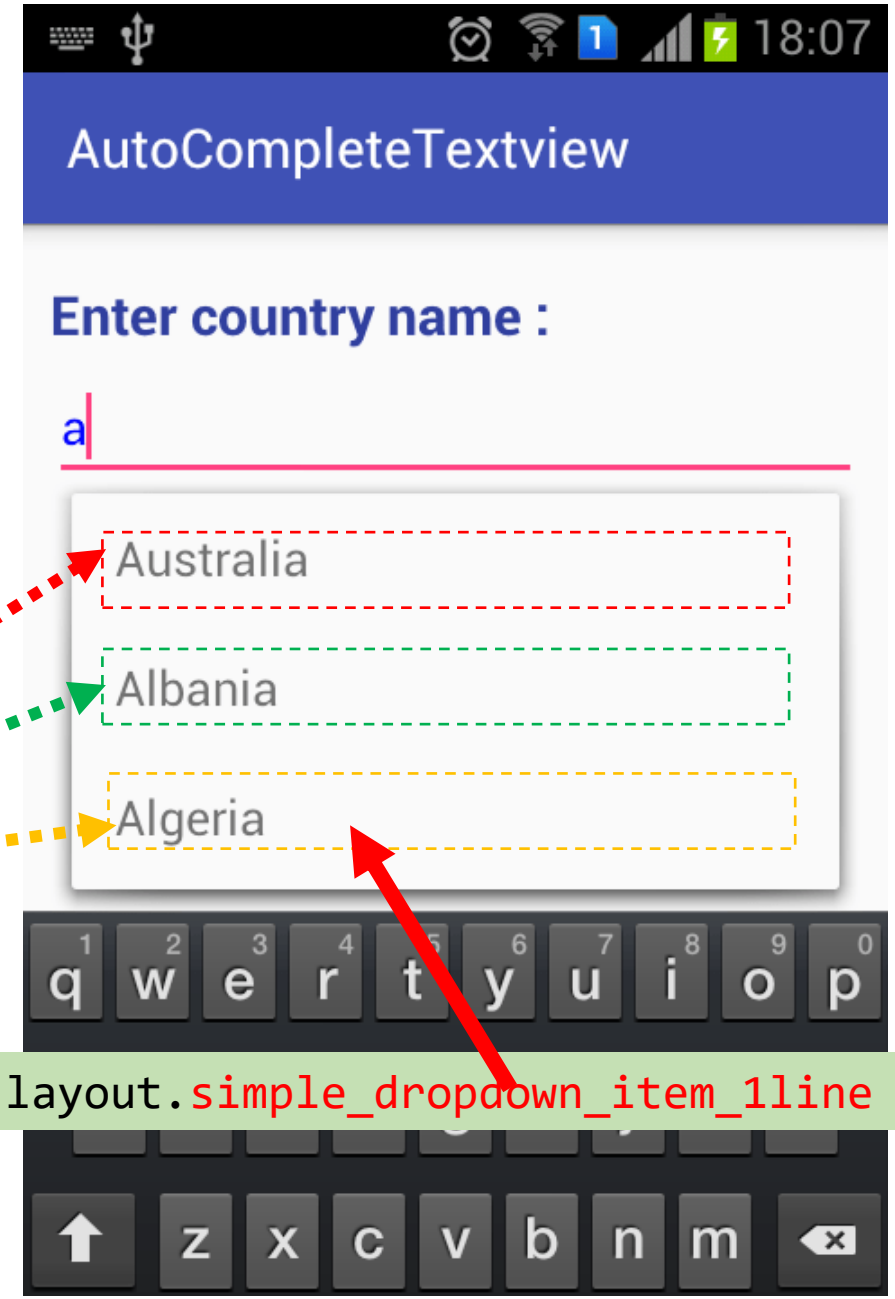



*Example:*
*Array, Cursor,..*

*Example:*
*GridView, ListView,*
*Spinner, StackView, ...*

**ArrayList<String>** nation


`android.R.layout.simple_dropdown_item_1line`

**String[]** nation {“Australia”, “Englenad”, “.. ”, .. }

# Ví dụ: AutoComplete TextView

```
public class CountriesActivity extends Activity {
    static final String[] NATIONS = new String[] {
        "Australia", "England", "Viet Nam", "Albania", "USA", "Algeria"};
  protected void onCreate(Bundle icicle) { super.onCreate(icicle);
        setContentView(R.layout.countries);
        ArrayAdapter<String> adapter;
        adapter = new ArrayAdapter<String>(this,
                                    android.R.layout.simple_dropdown_item_1line,
                                   COUNTRIES);
        AutoCompleteTextView autoTV= (AutoCompleteTextView)
                                              findViewById(R.id.countries_list);
        autoTV.setAdapter(adapter);
} }
```

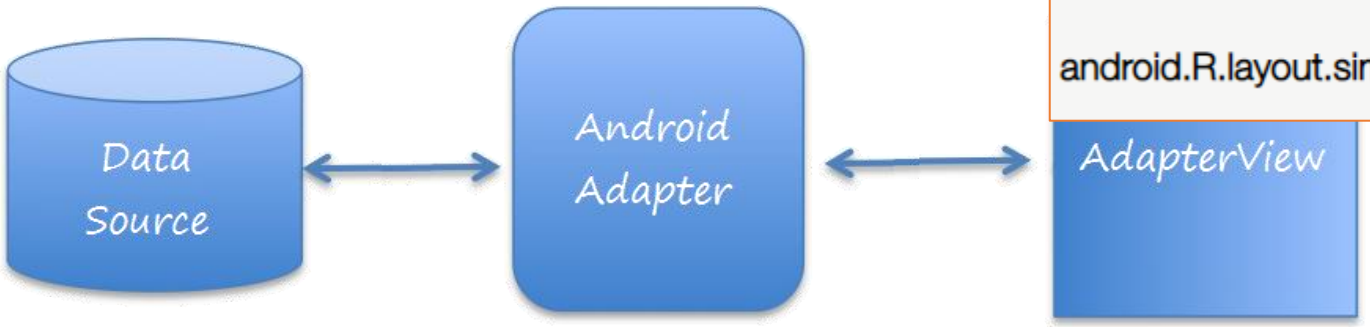
# ListView

- hiển thị một danh sách các phần tử trên một giao diện cho phép cuộn theo chiều dọc

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="listview.example.com.listview.MainActivity">

   <ListView
      android:id="@+id/listView"
      android:layout_width="match_parent"
      android:layout_height="fill_parent"
       />
</android.support.constraint.ConstraintLayout>
```

Android xây dựng sẵn một số Layout (cho ListItem) có thể làm việc với ArrayAdapter.

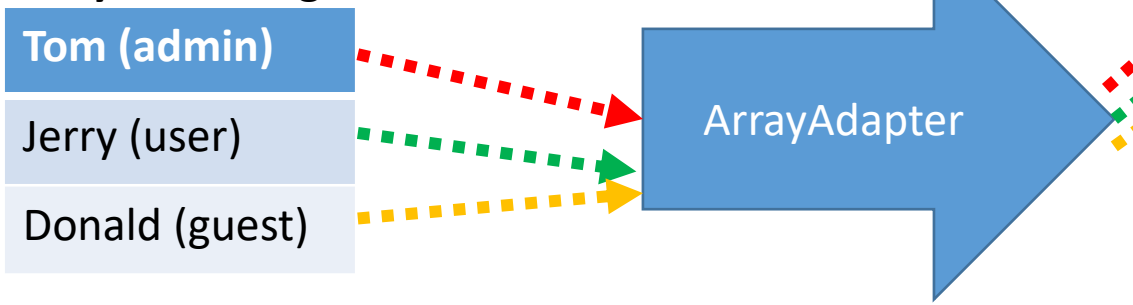
android.R.layout.simple_list_item_1




*Example:*
*Array, Cursor,..*

*Example:*
*GridView, ListView,*
*Spinner, StackView, ...*

**ArrayList<String>**





Đây là Layout đơn giản của ListItem, được tạo bởi duy nhất một TextView

Có id là android.R.id.text1

# ListView: thiết lập nội dung

- Sử dụng setAdapter để đổ nội dung vào các Item của listview như cách làm của AutoComplete Textview

```
// mỗi string là 1 dòng trong ListView
String[] values = new String[] {
"Việt Nam", "Trung Quốc", "Triều Tiên", "Cuba", "Hoa Kỳ"};
// Adapter: chứa dữ liệu và view của 1 dòng
// 1. Context → // 2. Layout cho 1 dòng → // 3. ID của TextView để hiện dữ liệu
trong layout(2)→ // 4. Mảng các dữ liệu
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                                        android.R.layout.simple_list_item_1,
                                        android.R.id.text1, values);
listView = (ListView) findViewById(R.id.list);
listView.setAdapter(adapter);
```

# Listview: xử lý sự kiện

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
     public void onItemClick(AdapterView<?> adapterView, View view, int position, long l)
            {
             String value=adapter.getItem(position);
             Toast.makeText(getApplicationContext(),value,Toast.LENGTH_SHORT).show();
    }
```

# Custom Listview

Listview nâng cao


Example:
Array, Cursor,..

Example:
GridView, ListView,
Spinner, StackView, ...

ListView

Step 2

CustomList Adapter

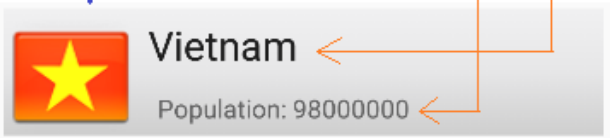
Step1

ArrayList<Country>

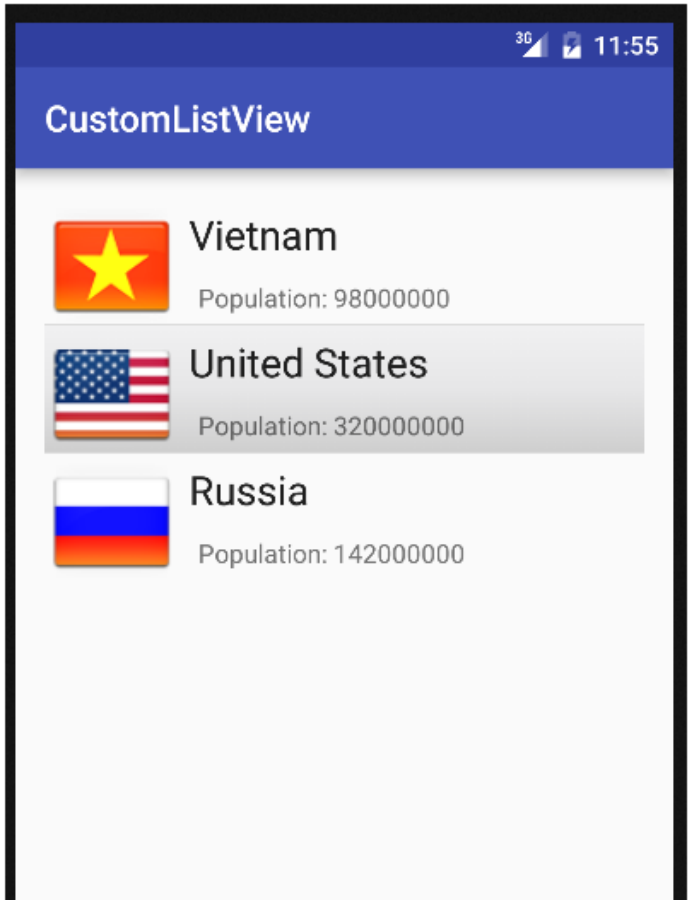
| Object 1 |
| --- |
| Object 2 |
| Object 3 |

ArrayList<CustomClass>


Step 3

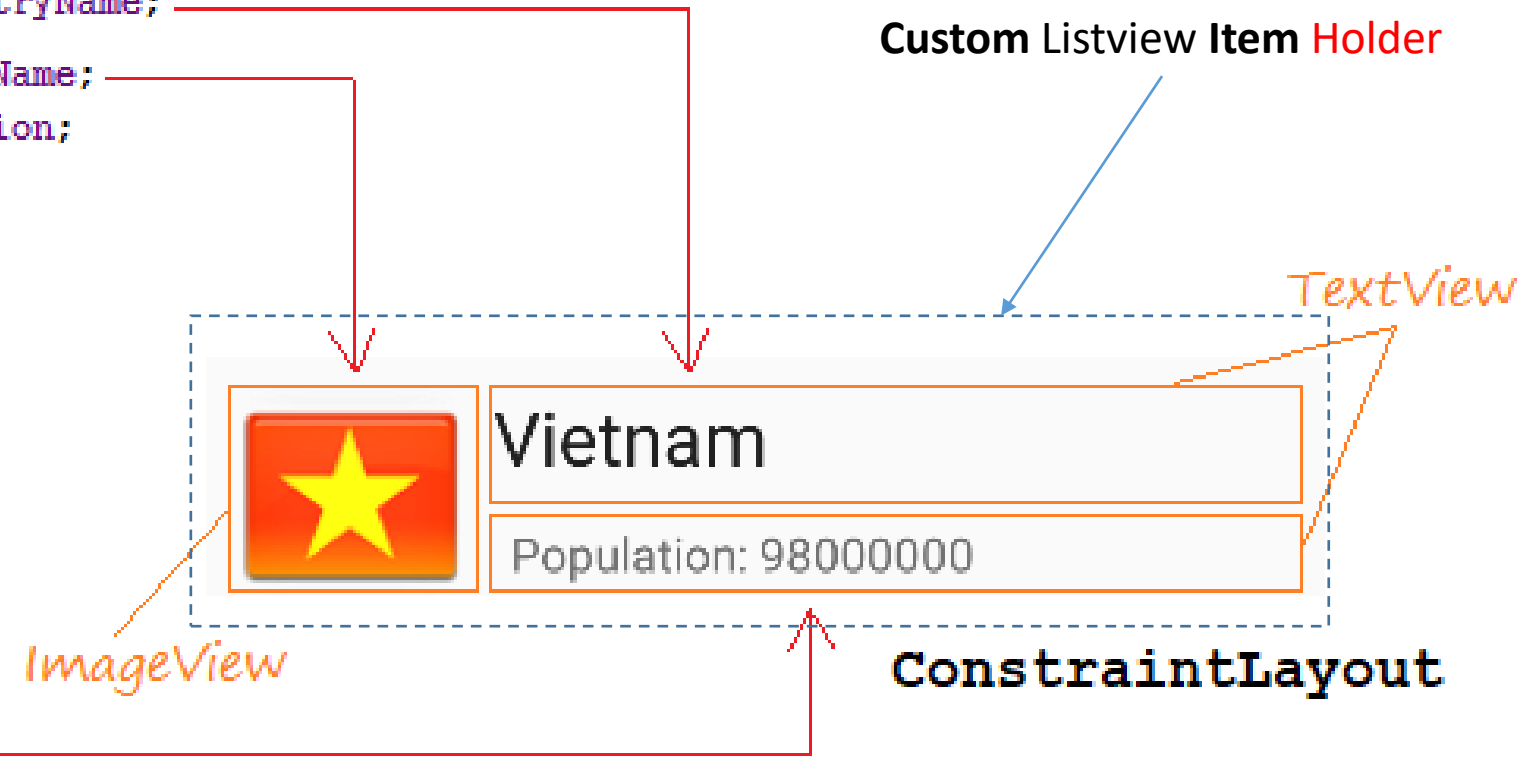



Custom View List Item Layout



# Step 3. Custom Listview Item Layout

```
public class Country {

    private String countryName;
    private String flagName;
    private int population;

}
```



# Step 2: Xây dựng CustomListAdapter

- Bằng cách kế thừa lớp BaseAdapter
- Và ghi đè các phương thức
  - **public int** getCount()
  - **public Object** getItem(**int** position )
  - **public int** getItemId(**int** position )
  - **public** View getView(**int** position, View convertView, ViewGroup parent)

# Ví dụ: Custom Listview

```
public class CustomListAdapter  extends BaseAdapter {

    private List<Country> listData;    // Lưu dữ liệu được nạp từ DataSource  cho Adapter
    private LayoutInflater layoutInflater;  // Chuyển đổi Layout XML thành View java
    private Context context;

    public CustomListAdapter(Context aContext,  List<Country> listData) {
        this.context = aContext;
        this.listData = listData;
        layoutInflater = LayoutInflater.from(aContext);
    }

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

    @Override
    public Object getItem(int position) {
        return listData.get(position);
    }
```

# Ví dụ: Custom Listview ..

```
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    if (convertView == null) {
        convertView = layoutInflater.inflate(R.layout.list_item_layout, null);
        holder = new ViewHolder();
        holder.flagView = (ImageView) convertView.findViewById(R.id.imageView_flag);
        holder.countryNameView = (TextView) convertView.findViewById(R.id.textView_countryName);
        holder.populationView = (TextView) convertView.findViewById(R.id.textView_population);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
    }

    Country country = this.listData.get(position);
    holder.countryNameView.setText(country.getCountryName());
    holder.populationView.setText("Population: " + country.getPopulation());

    int imageId = this.getMipmapResIdByName(country.getFlagName());

    holder.flagView.setImageResource(imageId);

    return convertView;
}
```

# Ví dụ: Custom Listview ..

```
// Find Image ID corresponding to the name of the image (in the directory mipmap).
public int getMipmapResIdByName(String resName)  {
    String pkgName = context.getPackageName();
    // Return 0 if not found.
    int resID = context.getResources().getIdentifier(resName , "mipmap", pkgName);
    Log.i("CustomListView", "Res Name: "+ resName+"==> Res ID = "+ resID);
    return resID;
}

static class ViewHolder {
    ImageView flagView;
    TextView countryNameView;
    TextView populationView;
}
```