



# **Lập trình Android**

**Bài 3: *Lập trình giao diện đồ họa người dùng***

---

**Phòng LT & Mạng**

*<http://csc.edu.vn/lap-trinh-va-cSDL>*

**2019**





# Nội dung

---

1. Mô hình Model – View – Controller (MVC)
2. View class
3. Các layout thường gặp
4. Các widget cơ bản
5. Làm việc với soft keyboard

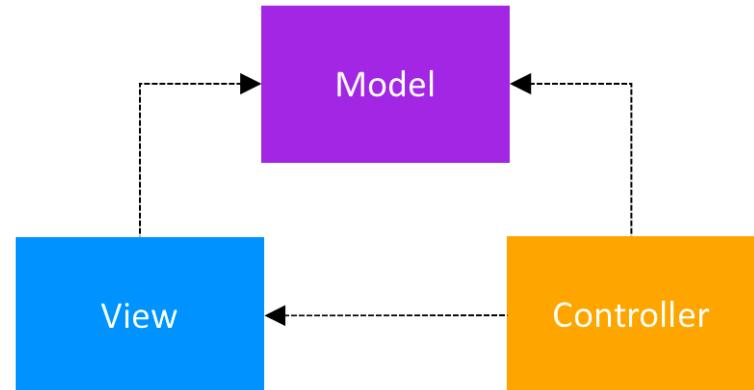




# Graphical User Interface

- Mô hình Model-View-Controller (MVC)

- Một design pattern quan trọng
- Chia chương trình thành 3 khối riêng biệt
  - User interface
  - Business
  - Input logic



- Mô hình MVC trong lập trình ứng dụng Android

- **Model:** mô tả các dữ liệu, hành vi của ứng dụng
- **View:** các màn hình mà người dùng có thể nhìn thấy và tương tác
- **Controller:** chịu trách nhiệm thông dịch các input từ người dùng và hệ thống và gửi tín hiệu tới Model hoặc View để thay đổi tương ứng. Các input có thể đến từ nhiều nguồn khác nhau: keyboard, touch-screen, GPS chip, các sensor, gia tốc kế...



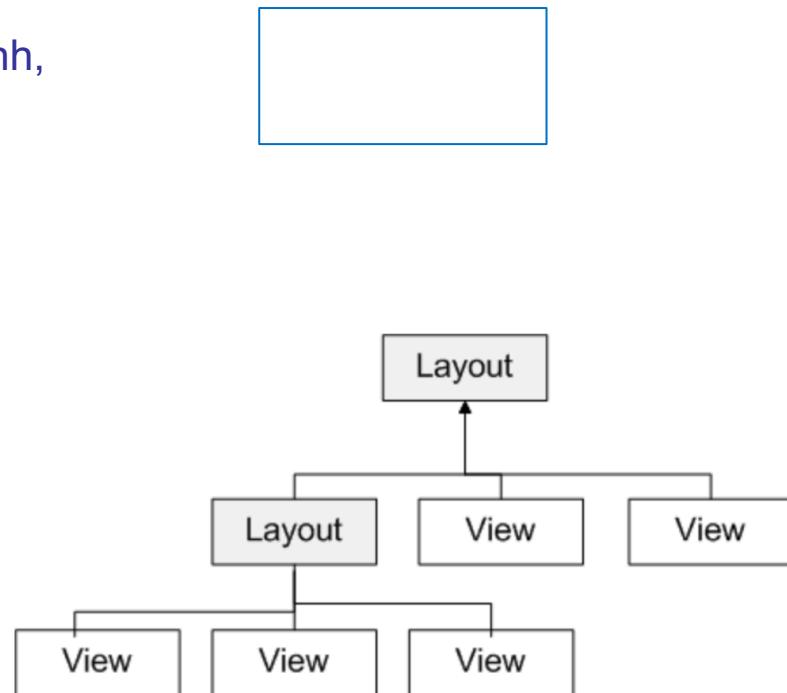
# View class

- **View** – còn gọi là Graphical User Interface (GUI)
- Android GUI thường được implement bởi các **XML** file (hoặc từ Java/Kotlin code)
- **View class** là thành phần cơ bản nhất của Android để tạo ra các đối tượng giao diện người dùng
- 1 **View** là 1 một vùng hình chữ nhật trên màn hình, chịu trách nhiệm việc vẽ và xử lý sự kiện
- **Widget** là các sub-class của View.

Thường là các đối tượng có thể tương tác với người dùng như button, checkbox, label, text field...

- **Layout** là một vật chứa (container) dùng để chứa các View hoặc Layout con.

Layout thường invisible với người dùng.





# XML file

## Sử dụng XML để đánh dấu GUI



UI được hiển thị bởi app

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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="csu.matos.gui_demo.MainActivity" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="36dp"
        android:text="@string/edit_user_name"
        android:ems="12" >
        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="48dp"
        android:text="@string/btn_go"  />

</RelativeLayout>
```



## XML file (2)

- Một XML view file thường bao gồm 1 layout để sắp xếp các thành phần con bên trong nó
- Các thành phần con bên trong có thể là các widget cơ bản hoặc do người dùng tự định nghĩa hoặc các layout con khác
- Activity dùng phương thức `setContentView(R.layout.xml_file_name)` để render 1 view lên màn hình thiết bị

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    Widgets and other nested layouts

```



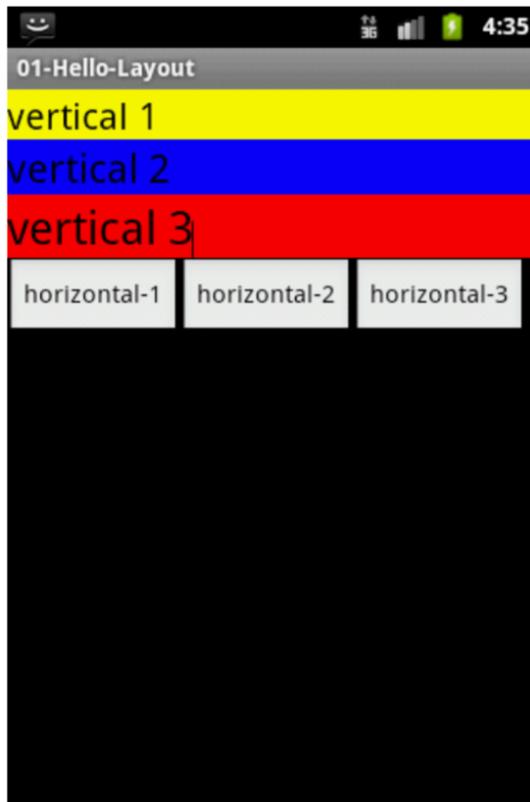
# Thiết lập cho View để có thể hoạt động

---

- Xử lý các widget và layout bao gồm các thao tác:
  - Thiết lập các **property** (thuộc tính). Ví dụ, với TextView cần thiết lập: background color, text, font, canh lề (alignment), size, padding, margin...
  - Thiết lập các **listener**. Ví dụ, một ImageView có thể được lập trình để phản hồi nhiều sự kiện khác nhau như: click (tap), long-tap...
  - Thiết lập **focus**. Để focus vào 1 view cụ thể, có thể gọi phương thức `requestFocus()` từ Java/Kotlin code hoặc dùng thẻ XML `<requestFocus />`
  - Thiết lập **visibility**. Dùng để show hoặc hide 1 view bằng phương thức `setVisibility(...)` từ Java/Kotlin code, hoặc thuộc tính XML `visibility`

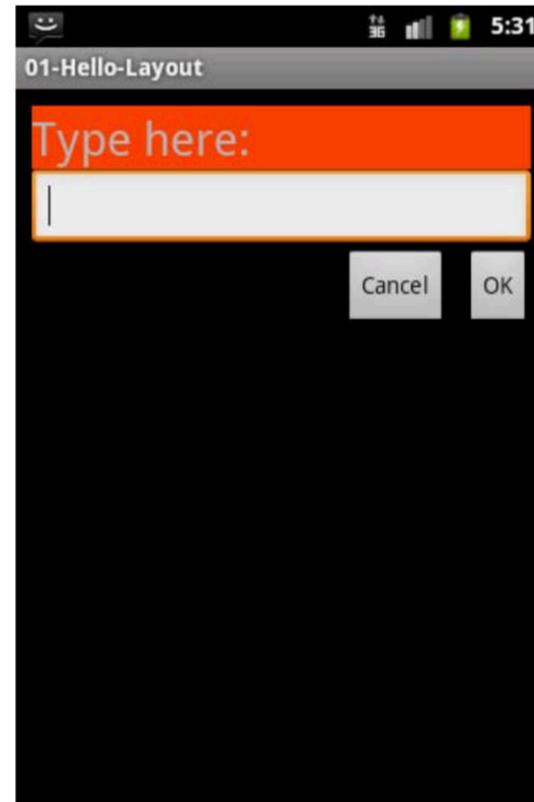


# Một số layout thường gặp



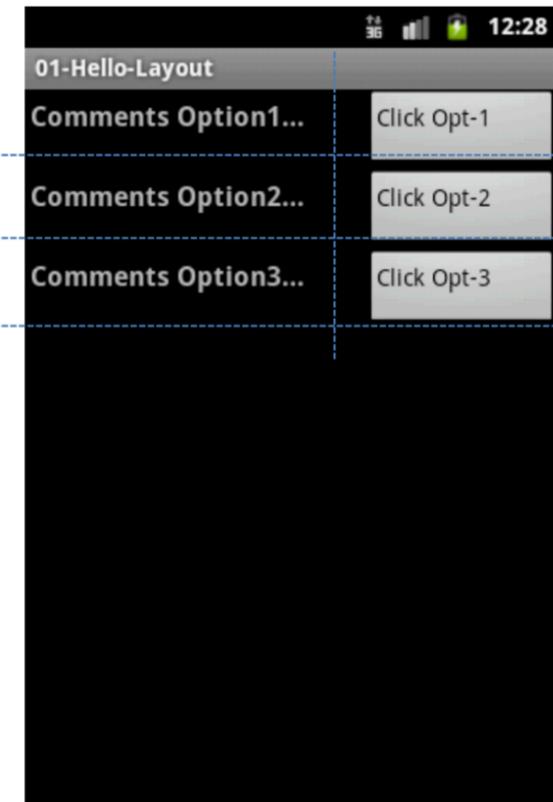
## Linear Layout

Là một ViewGroup sắp xếp các thành phần con bên trong theo chiều ngang hoặc chiều dọc



## Relative Layout

Là một ViewGroup cho phép đặt các thành phần con bên trong có quan hệ với nhau và có quan hệ với chính layout cha về vị trí

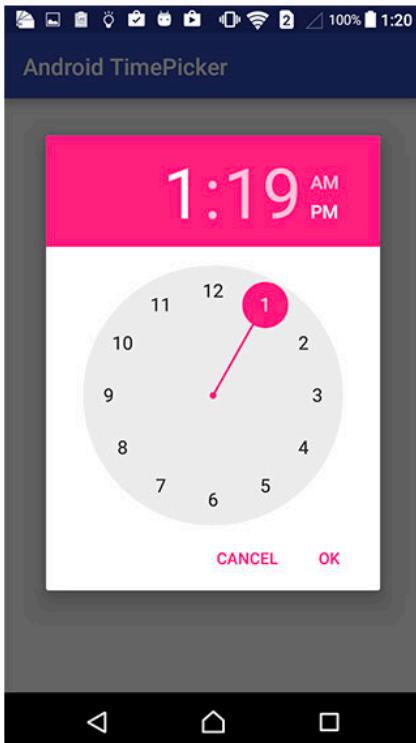


## Table Layout

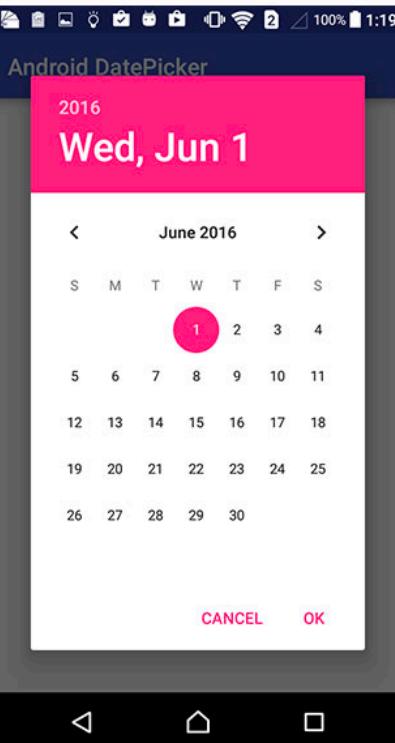
Là một ViewGroup sắp đặt các thành phần con bên trong theo dạng bảng (dòng & cột)



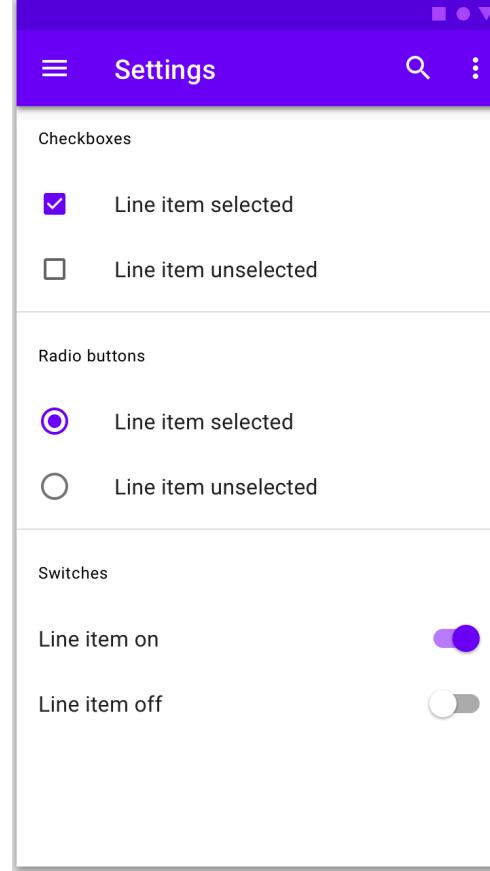
# Các widget thường gặp



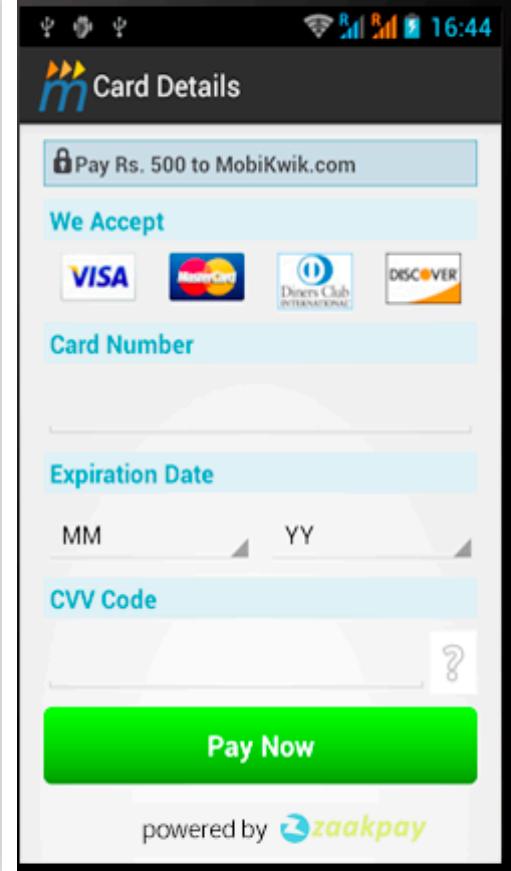
TimePicker



CalendarPicker

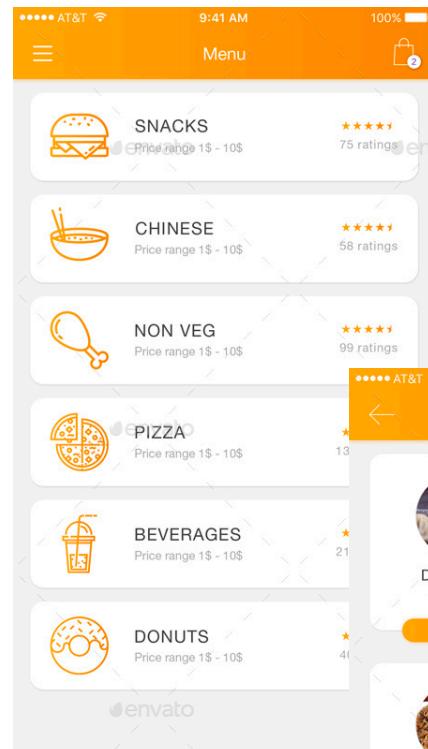
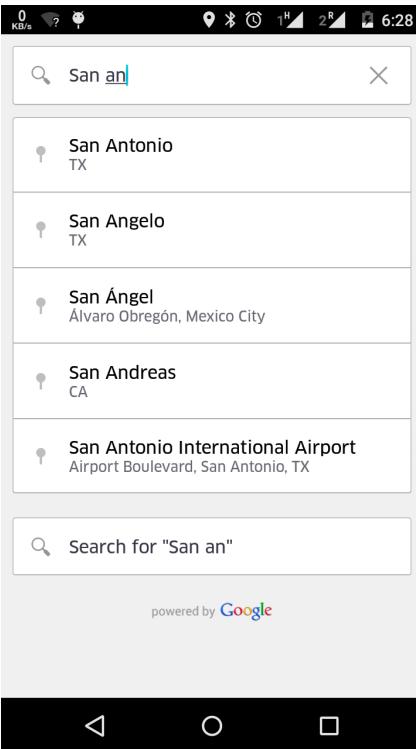


Các form controls: **Button**, **CheckBox**, **TextView**, **EditText**, **ImageView**, **ImageButton**, **Spinner**, **SeekBar**, **Switch**

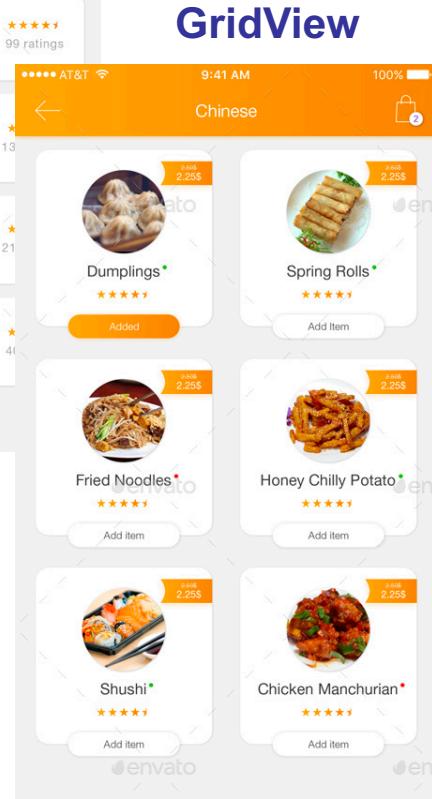




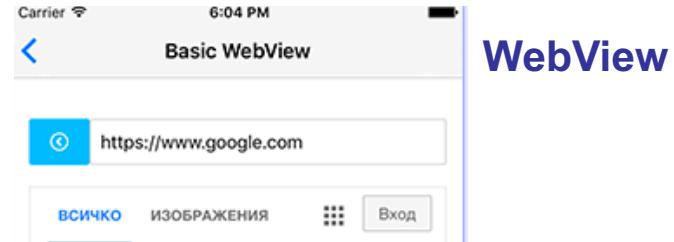
# Các widget thường gặp (2)



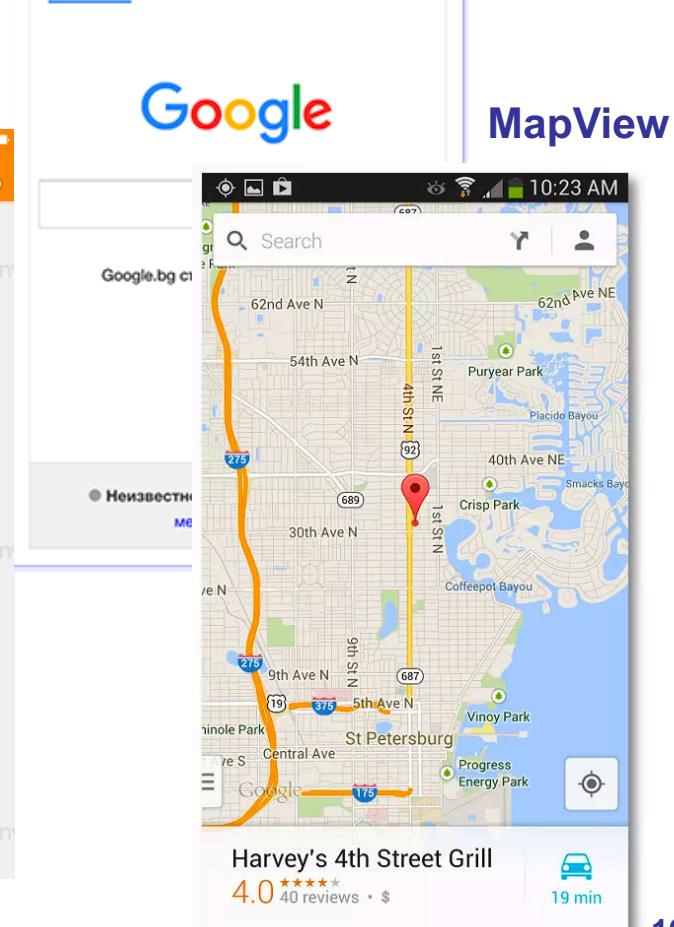
ListView



GridView



WebView



MapView

## AutoCompleteTextView

sub-class của **EditText**,  
cung cấp các gợi ý khi  
người dùng gõ phím





# XML Layout file

- Android xem XML layout file là 1 loại **resource**, do đó các XML layout file được chứa trong thư mục **res/layout** bên trong thư mục Android project

App  
explorer

Resource  
folder

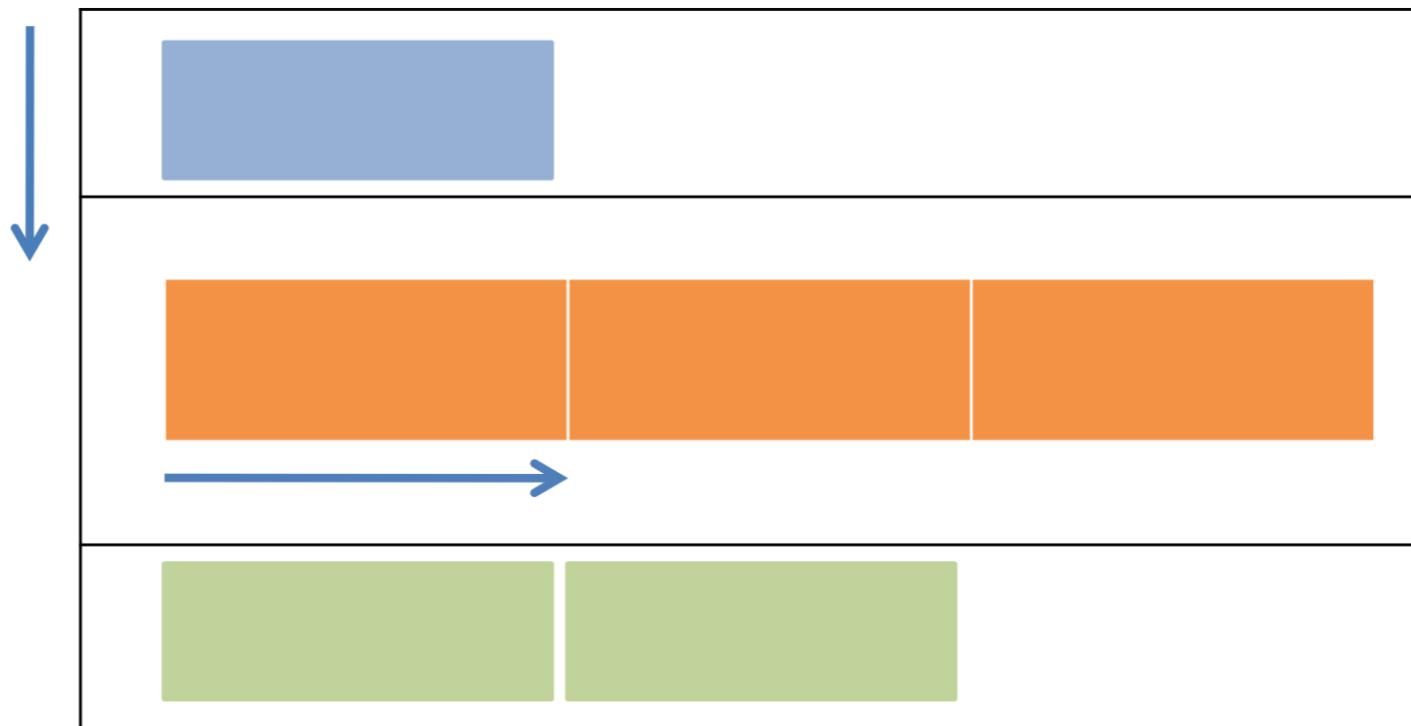
XML version  
of a window

```
<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"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context=".MainActivity" >  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/hello_world" />  
  
    </RelativeLayout>
```



# LinearLayout

- LinearLayout có chiến lược **lấp đầy**, trong đó các phần tử mới được sắp chồng lên nhau theo chiều **ngang (horizontal)** hoặc chiều **dọc (vertical)**





## LinearLayout (2)

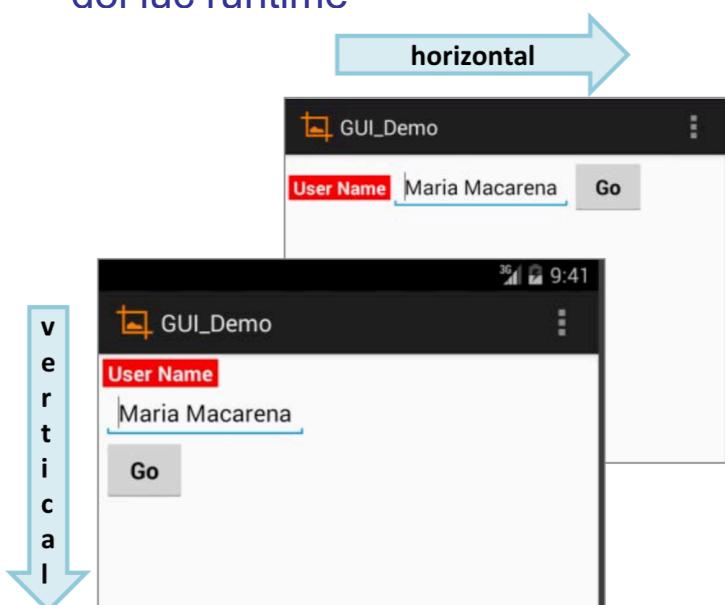
- Các thuộc tính cần thiết lập cho một LinearLayout

- **orientation** *(vertical, horizontal)*
- **fill model** *(match\_parent, wrap\_contents)*
- **weight** *(0, 1, 2, ...n )*
- **gravity** *(top, bottom, center,...)*
- **padding** *( dp – dev. independent pixels )*
- **margin** *( dp – dev. independent pixels )*

# LinearLayout (3)

## Orientation

- Thuộc tính `android:orientation` có thể được thiết lập là `horizontal` để sắp xếp các phần tử con theo chiều ngang, hoặc `vertical` theo chiều dọc
- Sử dụng `setOrientation(...)` để thay đổi lúc runtime



Lập trình AI

```
<LinearLayout
    xmlns:android="http://schemas.android.com/ap
    k/res/android"
    android:id="@+id/myLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" ←
    android:padding="4dp" >

    <TextView
        android:id="@+id/LabelUserName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#ffff0000"
        android:text=" User Name "
        android:textColor="#ffffffff"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/ediName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Maria Macarena"
        android:textSize="18sp" />

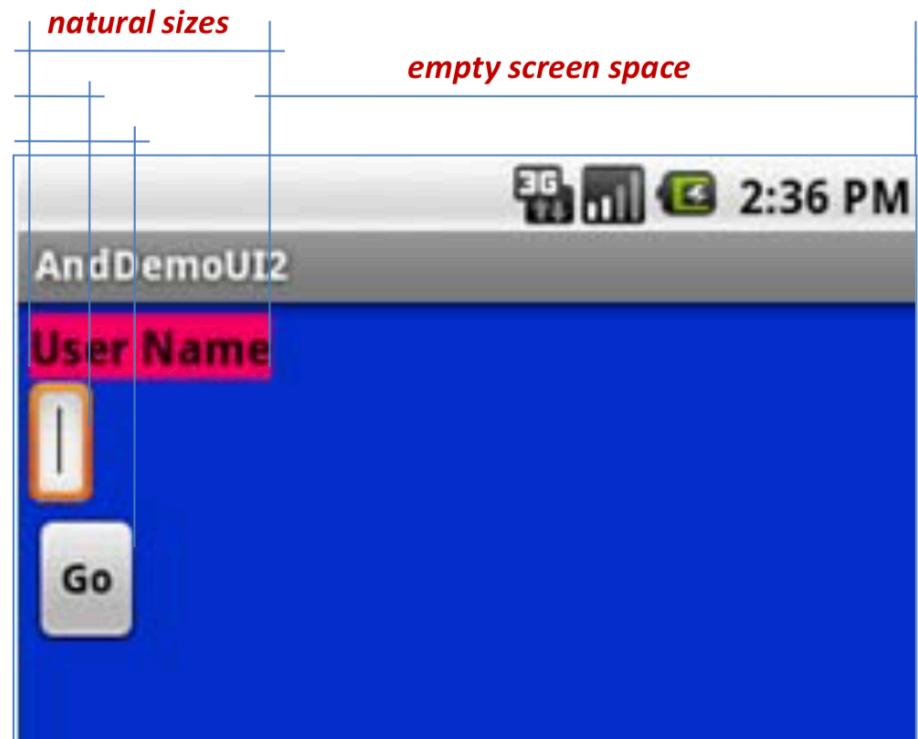
    <Button
        android:id="@+id/btnGo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go"
        android:textStyle="bold" />

</LinearLayout>
```



# LinearLayout - Fill model

- Widget có “natural size” dựa trên text mà nó chứa
- Trong một số trường hợp, widget cần được xác định kích thước (`layout_width`, `layout_height`) cụ thể ngay cả khi nó không chứa text nào cả





# LinearLayout - Fill model

---

- Tất cả các widget bên trong 1 layout **bắt buộc** phải có các thuộc tính:
  - android:layout\_width
  - android: layout\_height
- Giá trị của các thuộc tính này có thể là:
  - Kích thước xác định, ví dụ: **125dp** (device independent pixels - dip)
  - **wrap\_content**: widget sẽ co giãn theo natural space
  - **match\_parent**: widget muốn có kích thước lớn nhất mà layout chứa nó cho phép



# LinearLayout Fill model



```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/myLinearLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff0033cc"
    android:orientation="vertical"
    android:padding="6dp" >

    <TextView
        android:id="@+id/LabelUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0066"
        android:text="User Name"
        android:textColor="#ff000000"
        android:textSize="16sp"
        android:textStyle="bold" />

    <EditText
        android:id="@+id/ediName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp" />

    <Button
        android:id="@+id/btnGo"
        android:layout_width="125dp"
        android:layout_height="wrap_content"
        android:text="Go"
        android:textStyle="bold" />

</LinearLayout>
```

Row-wise

Use all the row

Specific size: 125dp



# LinearLayout: Weight

- **Weight** - Trọng số
- Không gian thừa còn lại trong 1 layout có thể được dành cho 1 thành phần nào đó bên trong layout

Trong ví dụ này, TextView và Button

có **android:layout\_weight="1"**,

EditText có **android:layout\_weight="2"**



*Chú ý: mặc định layout\_weight="0"*



# LinearLayout - Gravity

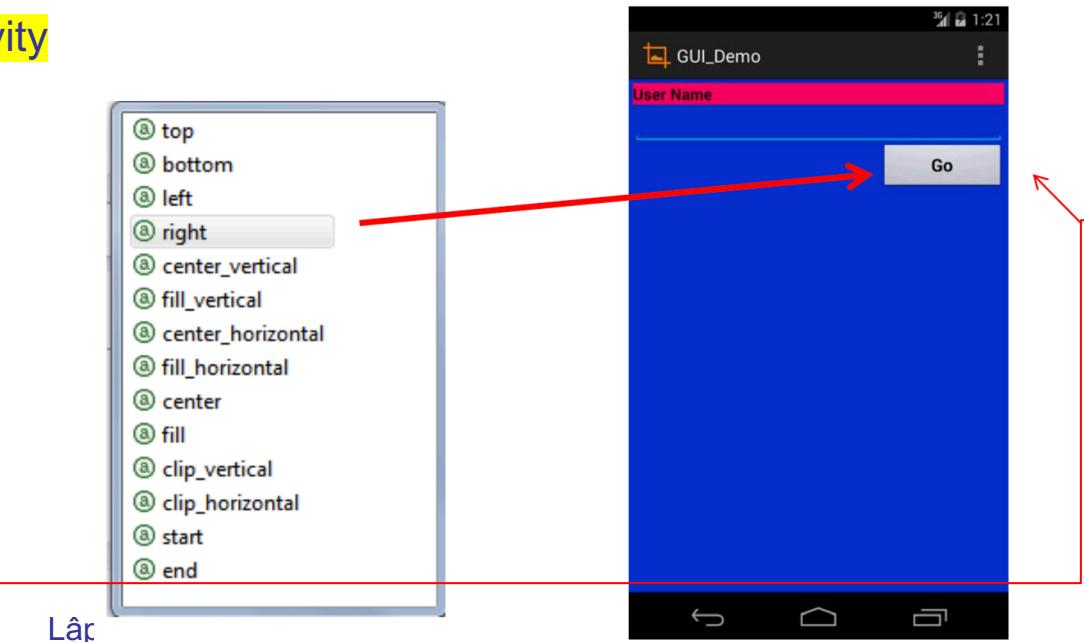
- Thuộc tính `android:gravity` và `android:layout_gravity` dùng để căn chỉnh vị trí của các thành phần con bên trong 1 linear layout, mặc định là **top-left**.
  - gravity** thiết lập trên linear layout cha để định vị trí các thành phần con bên trong. VD: thiết lập text view bên trong 1 linear layout ở chính giữa layout, đặt thuộc tính `android:gravity="center"` cho linear layout
  - layout\_gravity** thiết lập trên view con để định vị trí của view này bên trong 1 linear layout. VD: thiết lập button Go nằm về phía phải của màn hình (xem hình bên dưới), đặt thuộc tính `android:layout_gravity="right"` cho button Go.
- Thuộc tính `gravity, layout_gravity`

có thể nhận các giá trị:

**left, center, right,**

**top, bottom...**

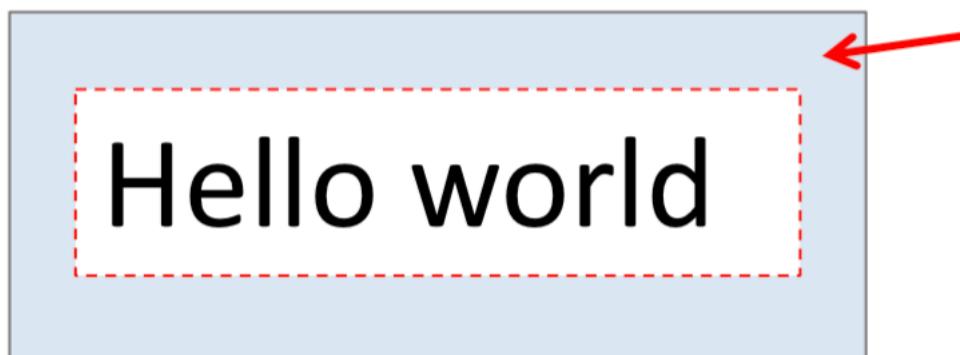
Button có  
`android: layout_gravity="right"`





# LinearLayout: Padding

- **Padding** cho biết khoảng cách giữa layout ngoài và các thành phần bên trong (sử dụng đơn vị **dp**)
- Đối với widget, padding là khoảng trống giữa khung bên ngoài (border) của nó với nội dung bên trong
- Thuộc tính XML: **android:padding**
- Phương thức Java: **setPadding(...)**

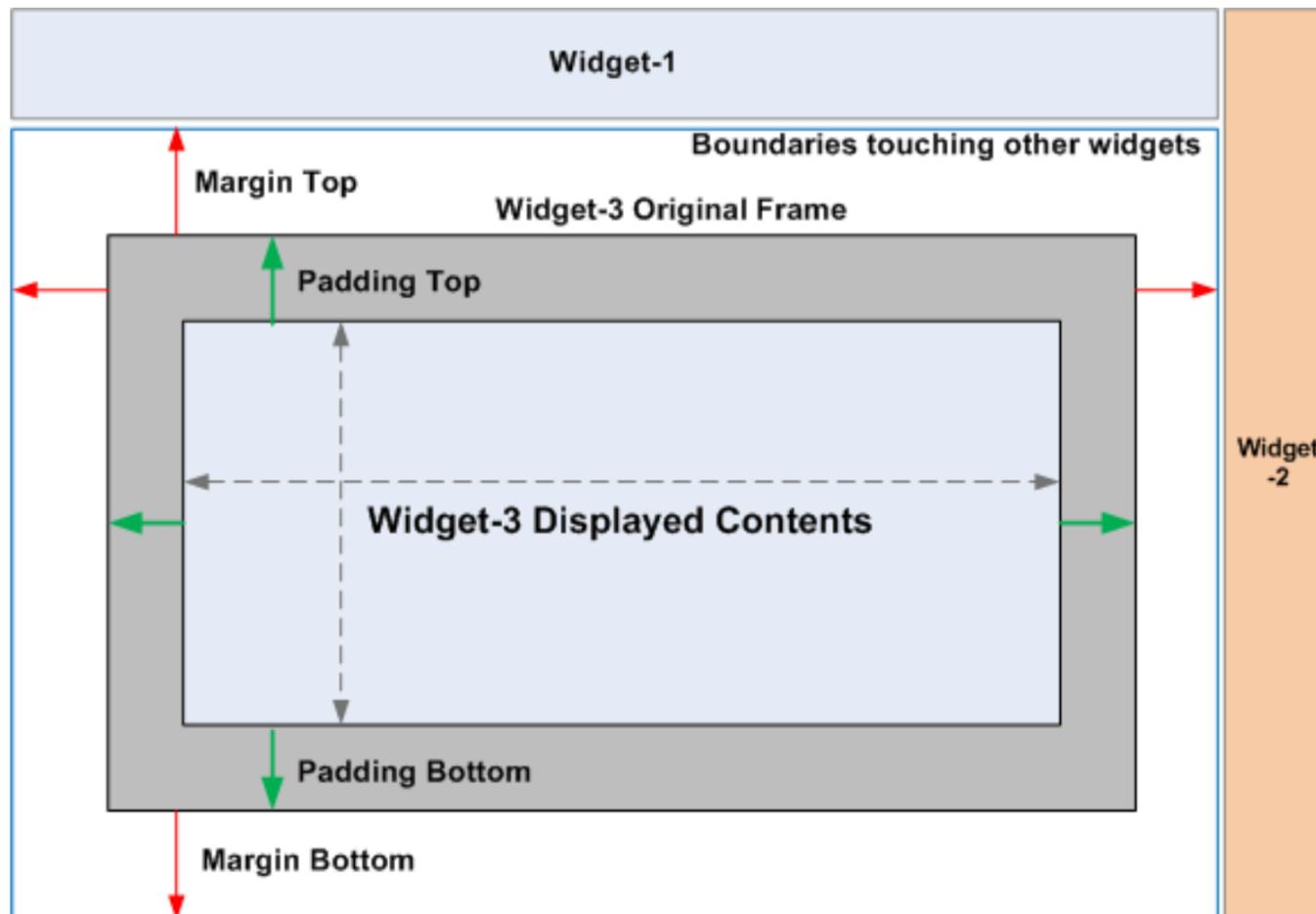


Padding - Phần màu xám



# LinearLayout: Padding và Margin

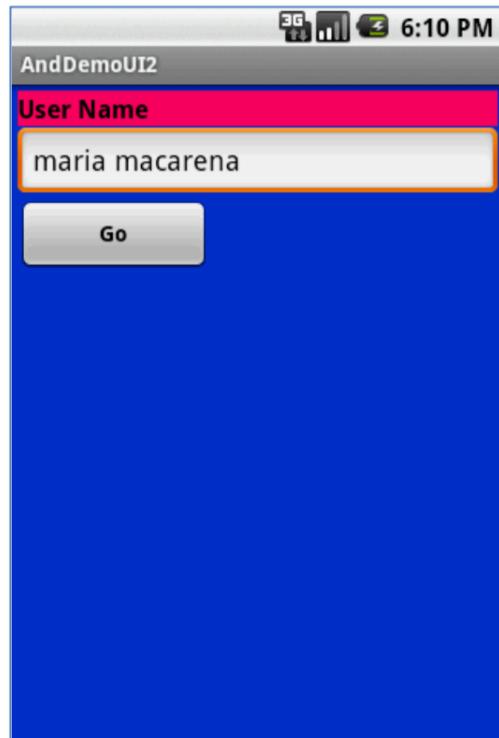
- **Padding:** khoảng cách từ khung bên ngoài đến nội dung bên trong
- **Margin:** khoảng cách đến đối tượng bên ngoài





# Ví dụ Padding

- EditText có padding bằng 30dp cho cả 4 hướng left, top, right, bottom
- Có thể thiết lập padding theo các hướng riêng bằng: android:padding\_Left, android:padding\_Top, android:padding\_Right, android:padding\_Bottom



```
<EditText  
    android:id="@+id/ediName"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textSize="18sp"  
    android:padding="30dp"  />
```

...



# Ví dụ Margin

- Dùng android:layout\_margin để tăng khoảng cách giữa các widget
- android:layout\_marginLeft, android:layout\_marginTop, android:layout\_marginRight, android:layout\_marginBottom

Using default spacing between widgets

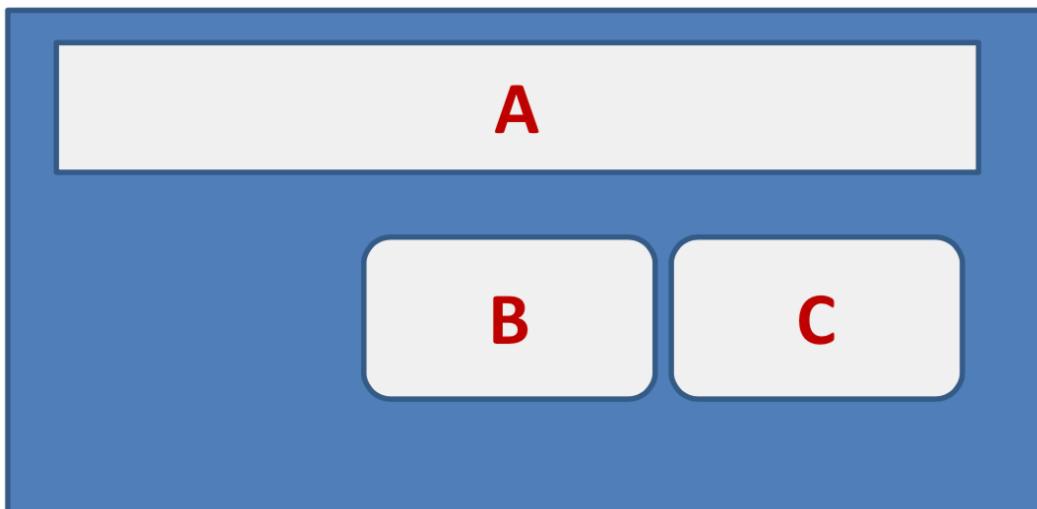
Increased inter-widget space

```
<EditText  
    android:id="@+id/ediName"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:textSize="18sp"  
    android:layout_margin="6dp"  
    >  
    </EditText>  
    ...
```



# RelativeLayout

- Sự sắp đặt 1 widget bên trong 1 **RelativeLayout** dựa trên *quan hệ vị trí* của nó với các widget khác và với chính layout cha



**Ví dụ:**

A ở vị trí trên cùng của layout cha

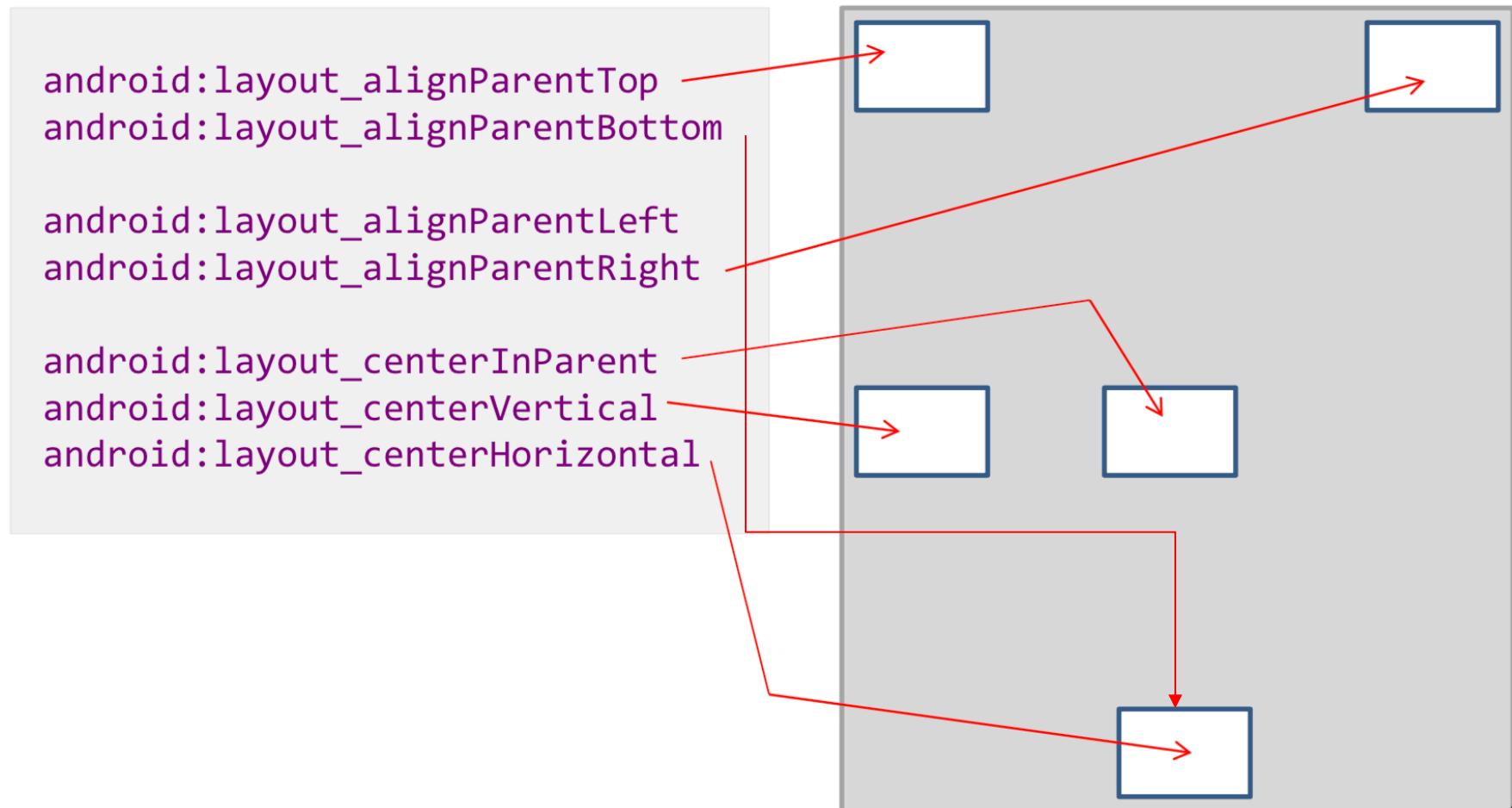
C ở bên dưới A, lệch về bên phải

B ở bên dưới A, ở phía trái của C



# RelativeLayout – Tham chiếu tới container

- Các thuộc tính XML dùng để sắp xếp các thành phần con bên trong RelativeLayout (layout cha). Các thuộc tính này nhận giá trị **true/false**





# RelativeLayout – Tham chiếu tới widget khác

android:layout\_alignTop="@+id/wid1"

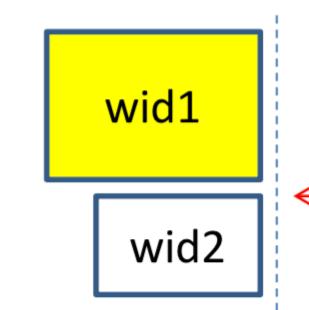
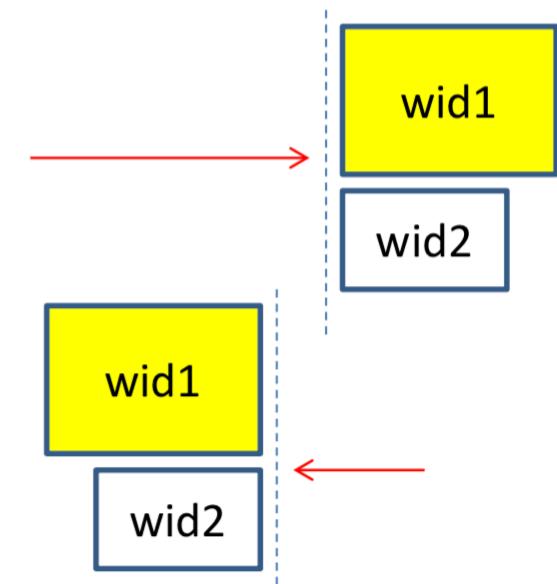
android:layout\_toRightOf="@+id/wid1"

android:layout\_alignBottom="@+id/wid1"

android:layout\_alignLeft="@+id/wid1"

android:layout\_below="@+id/wid1"

android:layout\_alignRight="@+id/wid1"





# RelativeLayout – Tham chiếu tới widget khác

---

- Khi tham chiếu đến widget khác, widget đó cần được khai báo thuộc tính **android:id**
- Một widget/layout được khai báo id bằng prefix **@+id/...**
- Ví dụ:
  - một EditText có thể được đặt id như sau:  
**android:id="@+id/txtUserName"**
  - Một widget khác có vị trí nằm bên dưới EditText txtUserName sẽ tham chiếu đến bởi  
**android:layout\_below="@+id/txtUserName"**

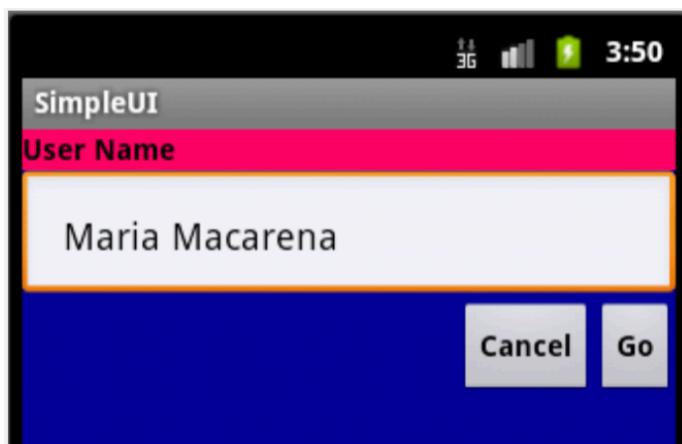




# RelativeLayout – Ví dụ

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/myRelativeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff000099"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <TextView
        android:id="@+id/tvUserName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0066"
        android:text="User Name"
        android:textStyle="bold"
        android:textColor="#ff000000"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true">
    </TextView>
```



```
<EditText
    android:id="@+id/txtUserName"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/tvUserName"
    android:layout_alignParentLeft="true"
    android:layout_alignLeft="@+id/myRelativeLayout"
    android:padding="20px">
</EditText>

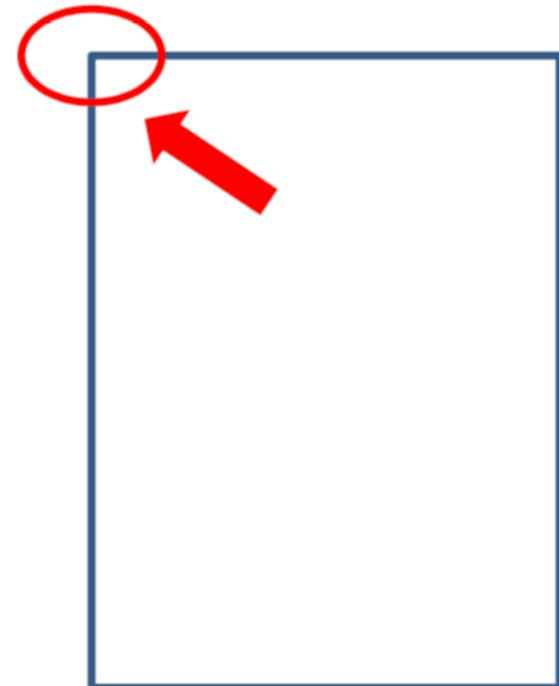
<Button
    android:id="@+id/btnGo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/txtUserName"
    android:layout_alignRight="@+id/txtUserName"
    android:text="Go"
    android:textStyle="bold">
</Button>

<Button
    android:id="@+id btnCancel"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toLeftOf="@+id/btnGo"
    android:layout_below="@+id/txtUserName"
    android:text="Cancel"
    android:textStyle="bold">
</Button>
</RelativeLayout>
```



# FrameLayout

- **FrameLayout** là layout đơn giản nhất
- Tất cả các element con được sắp xếp ở góc trái-trên của layout này





# Các layout thường gặp

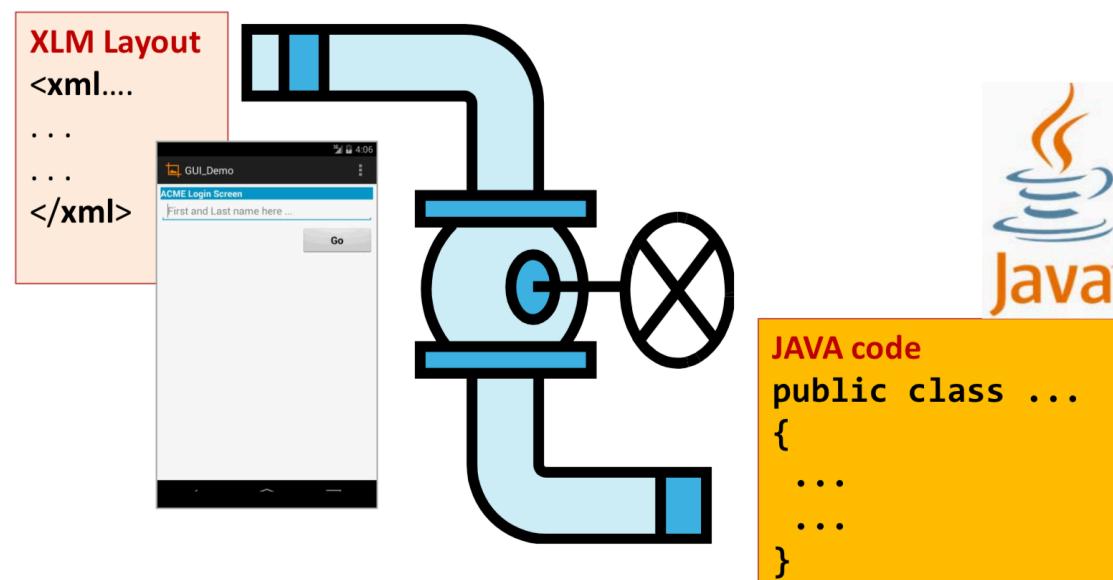
---

- TableLayout, ConstraintLayout → Xem thêm ở bài tập



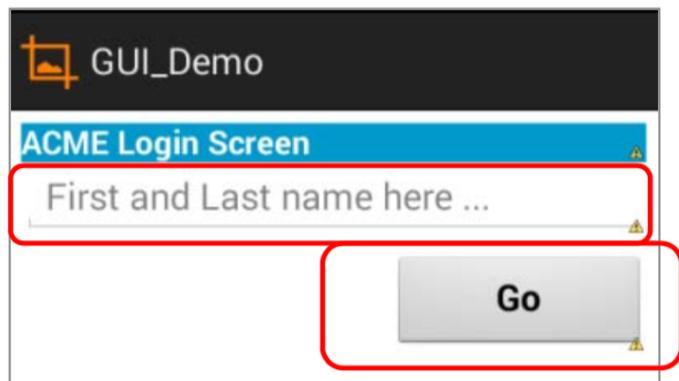
# Kết nối XML layout với Java code

- Phải kết nối các XML layout element (button, edittext...) với đối tượng Java tương ứng
- Thực hiện trong hàm **onCreate(...)** của một Activity
  - `setContentView(R.layout.xml_layout_file)`
  - `findViewById(R.id.id_of_element)`
- Sau khi các kết nối được hoàn tất, ứng dụng có thể sẵn sàng giao tiếp với người dùng





# Kết nối XML layout với Java code (2)



```
<!-- XML LAYOUT -->
<LinearLayout
    android:id="@+id/myLinearLayout"
    ... >

    <TextView
        android:text="ACME Login Screen"
        ... />

    <EditText
        android:id="@+id/edtUserName"
        ... />

    <Button
        android:id="@+id/btnGo"
        ... />

</LinearLayout>
```

res/layout/activity\_main.xml

→ Cần khai báo **id** cho các XML layout element

## Java code

```
package edu.csc.gui;
import android...;

public class MainActivity extends Activity {

    EditText edtUserName;
    Button btnGo;

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

        setContentView(R.layout.activity_main);

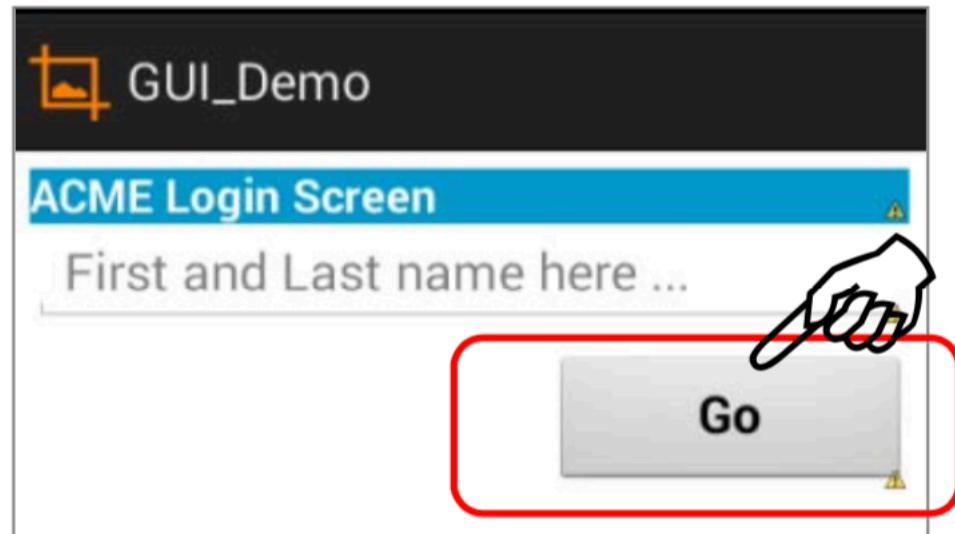
        edtUserName = (EditText) findViewById(R.id.edtUserName);
        btnGo = (Button) findViewById(R.id.btnGo);
        ...
    }
    ...
}
```

MainActivity.java



# Kết nối XML layout với Java code (2)

- Gắn bộ lắng nghe các sự kiện (event listener) vào widget
- Ví dụ: **click event** tương ứng với button “Go”



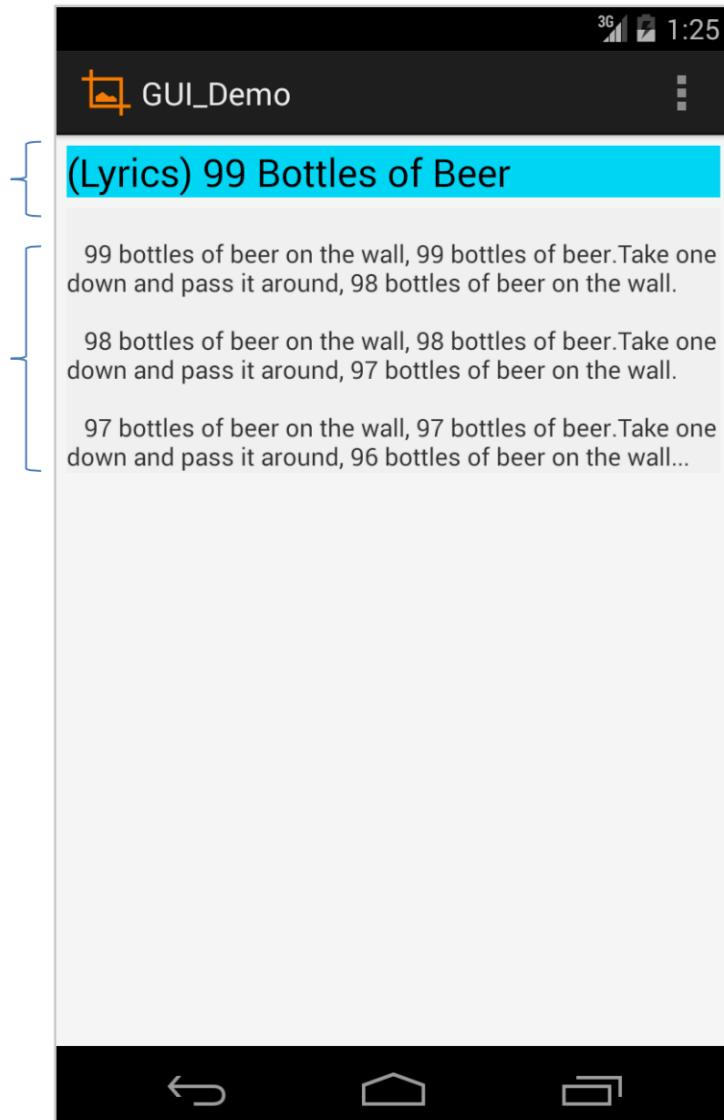
```
Button btnGo = (Button) findViewById(R.id.btnGo);  
  
btnGo.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // get username from EditText  
        // compare with database  
    }  
});
```



Ngoài ra còn có các sự kiện khác, tương ứng với từng loại widget



# Các widget cơ bản: TextView



- Còn gọi là label, dùng để trình bày một đoạn văn bản
- Không chỉnh sửa được nội dung văn bản, nên không dùng để nhập liệu
- Có thể trình bày được các ký tự đặc biệt, ví dụ '\n' (xuống dòng)
- Có thể trình bày được nội dung HTML, định dạng bởi  
`Html.fromHtml("<b>bold</b> string")`



# Các widget cơ bản: TextView

```
<?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"
    android:padding="@dimen/mediumSpace">
```

padding/margin: nên khai báo trong res/values/dimens.xml  
đơn vị: dp

```
<TextView
    android:id="@+id/lblTitle"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/blue"
    android:text="(Lyrics) 99 Bottles of Beer"
    android:textSize="@dimen/largeFontSize" />
```

màu sắc: nên khai báo trong res/values/colors.xml  
giá trị màu: mã hex

```
<TextView
    android:id="@+id/lblDescription"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="6dp"
    android:background="@color/gray"
    android:text="@string/text_description"
    android:textSize="14sp" />
```

kích cỡ chữ: nên khai báo trong res/values/dimens.xml  
đơn vị: sp

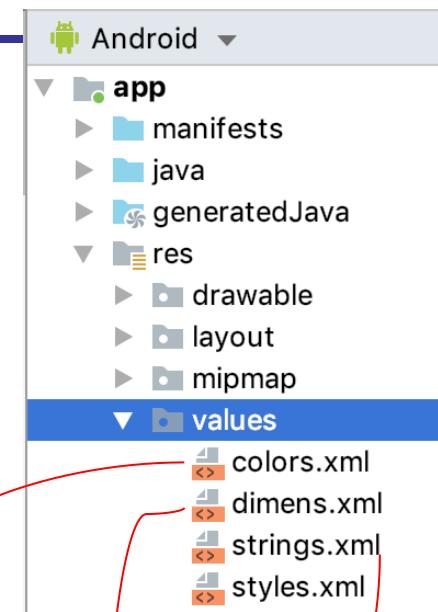
văn bản: nên khai báo trong res/values/strings.xml



# Các widget cơ bản: TextView

## o Các thuộc tính thường gặp

Thuộc tính	Ý nghĩa	Giá trị/Tham chiếu
text	Nội dung văn bản hiển thị	@string/... - strings.xml
textSize	Kích thước văn bản	@dimen/... - dimens.xml
textColor	Màu sắc văn bản	@color/... - colors.xml
textStyle	Kiểu văn bản	regular/bold/italic
background	Màu nền của textView	@color/... - colors.xml



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="blue">#3498db</color>
    <color name="gray">#bdc3c7</color>
</resources>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="largeFontSize">18sp</dimen>
    <dimen name="mediumSpace">6dp</dimen>
</resources>
```

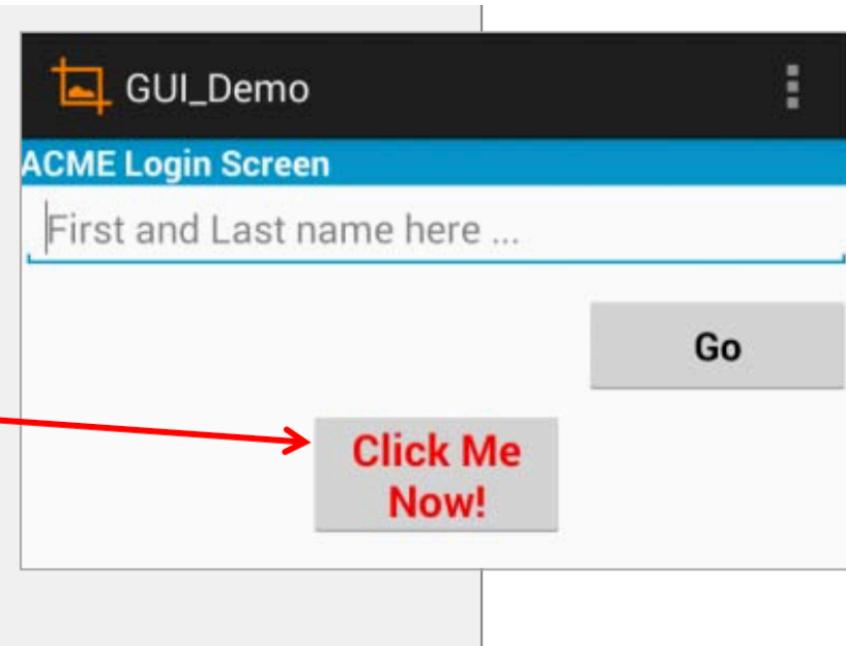
```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="text_description">\n\t99 bottles of beer on the wall, 99 bottles of beer...</string>
</resources>
```



# Các widget cơ bản: Button

- Là sub-class của TextView → có các thuộc tính giống TextView
- Cho phép người dùng click (tap) vào → sự kiện cơ bản: **onClick**
- Có thể thay đổi thuộc tính background của thông qua các file *drawable.xml*, thay đổi hành vi của button dựa trên các trạng thái (pressed, focused) → Xem ở các slide sau.

```
<Button  
    android:id="@+id/btnClickMeNow"  
    android:layout_width="120dp"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:layout_marginTop="5dp"  
    android:gravity="center"  
    android:padding="5dp"  
    android:text="Click Me Now!"  
    android:textColor="#ffff0000"  
    android:textSize="20sp"  
    android:textStyle="bold" />
```



Chú ý: Một số thuộc tính nên tham chiếu đến các giá trị ở *res/values.xml*, *res/dimens.xml*, *res/colors.xml*



# Button – Ví dụ minh họa

```
<LinearLayout
```

```
    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"  
    android:orientation="vertical"  
    android:padding="@dimen/mediumSpace" >  
    <TextView  
        android:id="@+id/txtMsg"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:background="@color/light_red" />
```

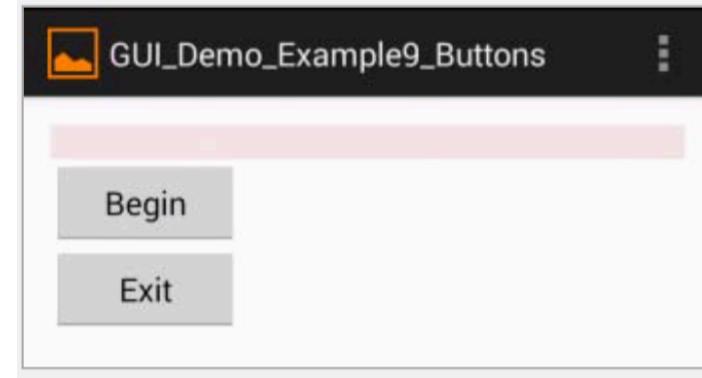
```
    <Button
```

```
        android:id="@+id/btnBegin"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:ems="5" ←  
        android:text="@string/begin" />
```

```
    <Button
```

```
        android:id="@+id/btnExit"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:ems="5" ←  
        android:text="@string/exit" />
```

```
</LinearLayout>
```



width tương ứng với 5 characters



# Button – Ví dụ minh họa

```
public class MainActivity extends Activity implements View.OnClickListener {
```

```
    TextView lblMessage;  
    Button btnBegin, btnExit;
```

```
@Override
```

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

```
    lblMessage = findViewById(R.id.lblMessage);  
    btnBegin = findViewById(R.id.btnBegin);  
    btnExit = findViewById(R.id.btnExit);
```

```
    btnBegin.setOnClickListener(this);  
    btnExit.setOnClickListener(this);
```

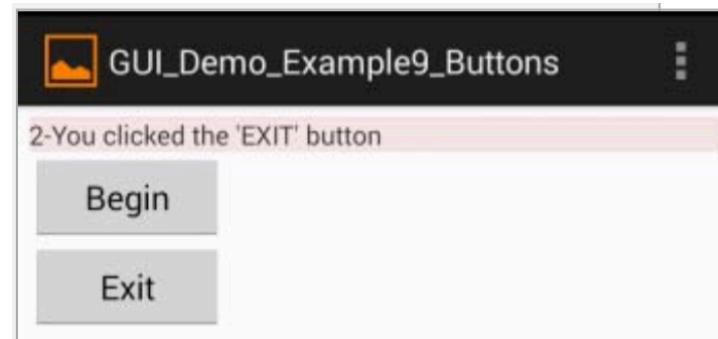
```
}
```

```
@Override
```

```
public void onClick(View v) {  
    int id = v.getId();  
    if (id == R.id.btnBegin) {  
        lblMessage.setText("1 - You clicked the 'BEGIN' button");  
    } else if (id == R.id.btnExit) {  
        lblMessage.setText(getString(R.string.text_click_exit));  
    }  
}
```

MainActivity implement **OnClickListener** interface, nó override phương thức **onClick** của interface này.

Khi có sự kiện click lên các button, sẽ có tín hiệu gửi tới MainActivity tương ứng với hàm onClick



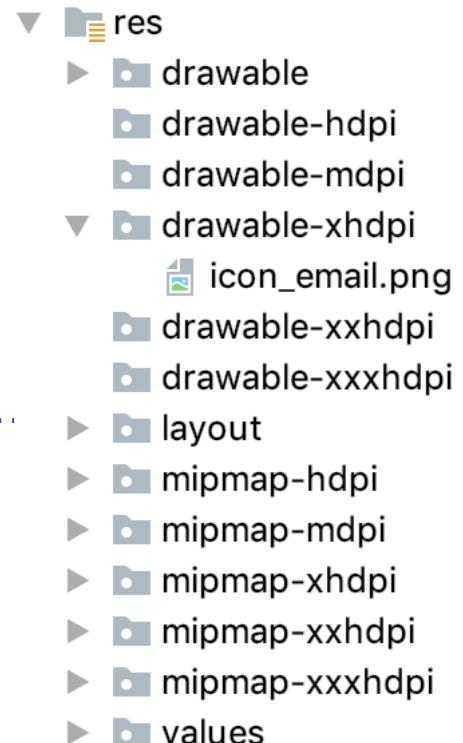
strings.xml

```
<resources>  
    <string name="text_click_exit">2 - You clicked the 'EXIT' button</string>  
</resources>
```



# Các widget cơ bản: ImageView & ImageButton

- Giúp nhúng hình ảnh (png, jpg, gif...) vào ứng dụng
- Tương tự Button, TextView
- Các thuộc tính **android:src** và **android:background** giúp thiết lập hình ảnh
- Hình ảnh được lưu trữ trong thư mục **res/drawable**. Các thư mục **drawable-mdpi**, **drawable-hdpi**, **drawable-xhdpi**... được tạo ra để chứa các hình ảnh tương ứng với các kích thước khác nhau của màn hình thiết bị





# Các widget cơ bản: ImageView & ImageButton

```
<LinearLayout
```

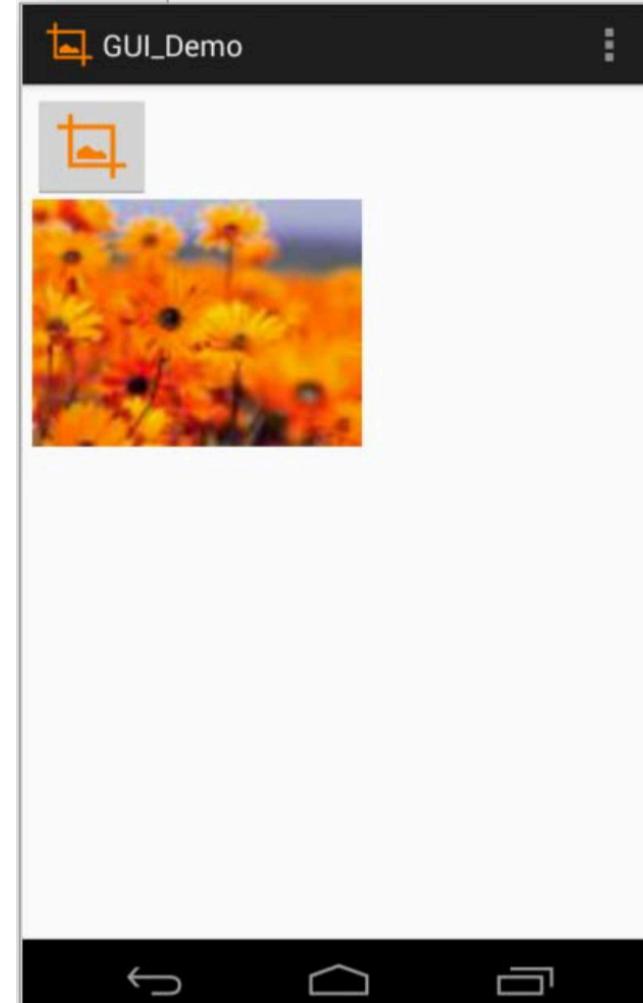
```
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:padding="@dimen/mediumSpace"  
    android:orientation="vertical" >  
    <ImageButton
```

```
        android:id="@+id/ibCrop"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:src="@drawable/ic_crop" />
```

```
    <ImageView  
        android:id="@+id/ivPhoto"  
        android:layout_width="200dp"  
        android:layout_height="150dp"  
        android:scaleType="fitXY"  
        android:src="@drawable/flowers" />
```

```
</LinearLayout>
```

scaleType có thể nhận các giá trị center, centerInside, centerCrop, fitcenter, fitStart, fitEnd, fitXY, matrix



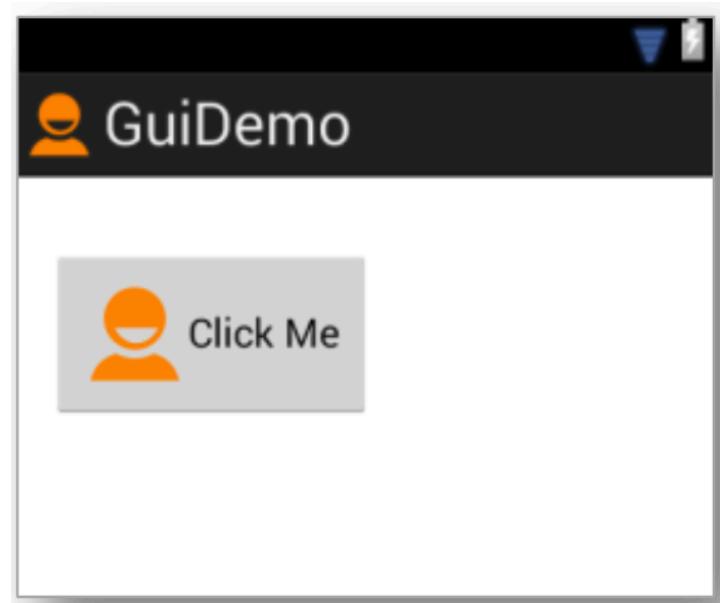


# Các widget cơ bản: Button

- Button có thể kết hợp trình bày cả text và image

```
<LinearLayout  
    <Button  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:drawableLeft="@drawable/ic_user"  
        android:gravity="left/center_vertical"  
        android:padding="@dimen/LargeSpace"  
        android:text="@string/click_me" />  
</LinearLayout>
```

android:drawableLeft  
android:drawableTop  
android:drawableRight  
android:drawableBottom





# Các widget cơ bản

- Tuỳ vào kích thước màn hình thiết bị khác nhau, các hình ảnh được sử dụng cũng cần có kích thước tương ứng
- Xem chi tiết hơn ở bài quản lý **resources**



**mdpi** (761 bytes)  
1x = 48 x 48 pixels  
BaseLine



**hdpi** (1.15KB)  
1.5x = 72 x 72 px



**x-hdpi** (1.52KB)  
2x = 96 x 96 px

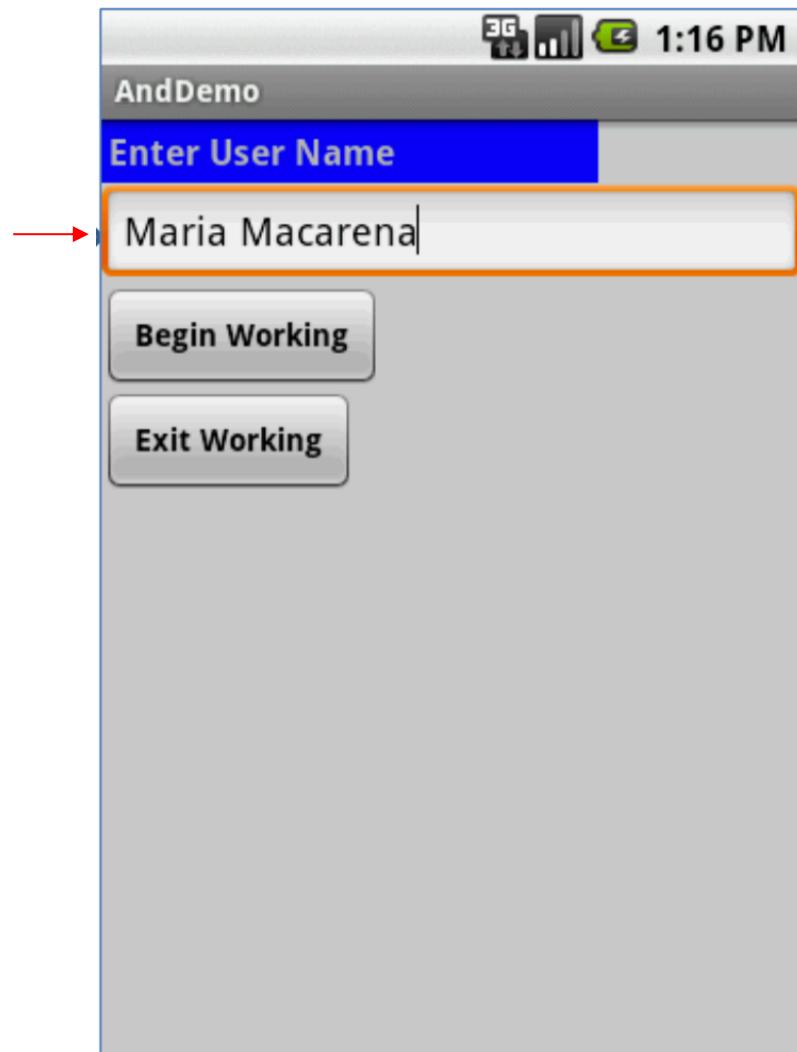


**xx-hdpi** (2.47KB)  
3x = 144 x 144 px



# Các widget cơ bản: EditText

- **EditText** mở rộng từ **TextView**, cho phép người dùng nhập liệu
- Ngoài việc hiển thị plain text, **EditText** còn có thể định dạng văn bản theo HTML, dùng `Html.fromHtml(html_text)`
- Từ Java code có thể gán text hoặc lấy text từ **EditText** bằng:
  - `txtBox.setText("some string")`
  - `txtBox.getText().toString()`

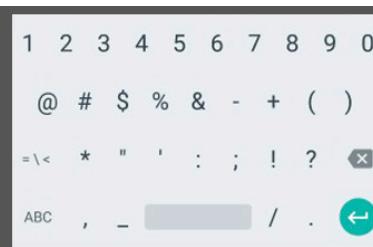


# Các widget cơ bản: EditText

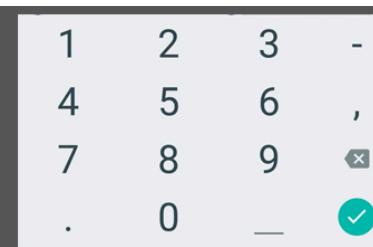
- Định dạng kiểu dữ liệu nhập
  - EditText có thể chấp nhận các chuỗi theo mẫu như: số, số điện thoại, ngày, giờ, uri...
  - Thông qua thuộc tính `android:inputType` chọn →
  - Có thể kết hợp 2 hay nhiều thuộc tính. Ví dụ: `textCapWords|textAutoCorrect` sẽ giúp nhập chuỗi sao cho viết hoa ký tự đầu tiên của mỗi từ, đồng thời nếu từ nhập bị sai chính tả sẽ tự động sửa lỗi (nhập 'teh' → sửa thành 'the')
  - Bàn phím ảo (soft keyboard) cũng tự động điều chỉnh tương ứng với `inputType`. Ví dụ:



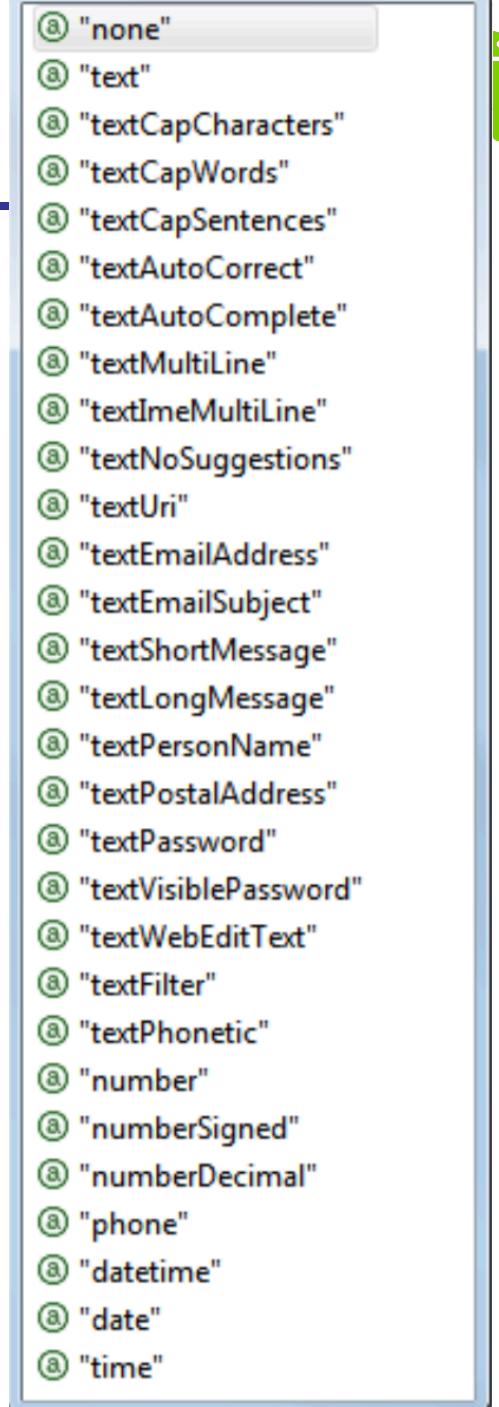
Default  
Alphabet



Default  
Numeric



Keypad  
Numeric Only





# Các widget cơ bản

---

- CheckBox, RadioButton... → Xem thêm ở bài tập



# Tuỳ chỉnh các widget

---

- Tuỳ chỉnh các thuộc tính background, border... →  
Xem thêm ở bài tập



# Làm việc với soft keyboard

- **Hiện** soft keyboard trên 1 view cụ thể (thông thường là EditText và các widget dùng để nhập văn bản)

```
public void showSoftKeyboard(View view) {  
    if (view.requestFocus()) {  
        InputMethodManager imm = (InputMethodManager)  
            getSystemService(Context.INPUT_METHOD_SERVICE);  
        imm.showSoftInput(view, InputMethodManager.SHOW_IMPLICIT);  
    }  
}
```

- **Ẩn** soft keyboard

```
public void hideSoftKeyboard(View view) {  
    InputMethodManager imm = (InputMethodManager)  
        getSystemService(Context.INPUT_METHOD_SERVICE);  
    imm.hideSoftInputFromWindow(view.getWindowToken(), 0);  
}
```

- Thêm “**Done**” key

- Sử dụng thuộc tính XML

```
<EditText android:imeOptions="actionDone" />
```

- Thiết lập từ Java/Kotlin code

```
myEditText.setImeOptions(EditorInfo.IME_ACTION_DONE);
```

Lập trình Android





# Làm việc với soft keyboard (2)

- Hiện keyboard khi vừa mở activity

```
<application ... >
  <activity
    android:name="edu.csc.MainActivity"
    android:windowSoftInputMode="stateVisible">
  </activity>
</application>
```

{ AndroidManifest.xml }

stateUnchanged, stateHidden

- Điều chỉnh UI để dành không gian cho keyboard

```
<application ... >
  <activity
    android:windowSoftInputMode="adjustResize" ... > ←
  </activity>
</application>
```

adjustPan, adjustUnspecified

- Có thẻ kết hợp thuộc tính visibility và adjustment

```
<activity
  android:windowSoftInputMode="stateVisible|adjustResize" ... >
</activity>
```





## Q&A

---

