LẬP TRÌNH DI ĐỘNG

Bài 3: các layout và một số loại widget thường dùng

Nhắc lại bài trước

- Giao diện phát triển ứng dụng của Android Studio
- Các thành phần của một project android
- File mô tả ứng dụng AndroidManifest.xml
- Bốn loại thành phần của ứng dụng android: activity, service, provider, receiver
- Khái niệm activity, cách tạo giao diện của activity bằng code và bằng xml
- Mã minh họa việc gọi một activity khác
- Vòng đời của activity: create -> start -> resume -> pause -> stop -> destroy

Nội dung

1. Khái niệm view & viewgroup
2. Làm việc với layout
3. Một số layout thông dụng
4. Tương tác với các điều khiển
5. Một số điều khiển đơn giản

Phần 1

Khái niệm view & view group

View - Widget

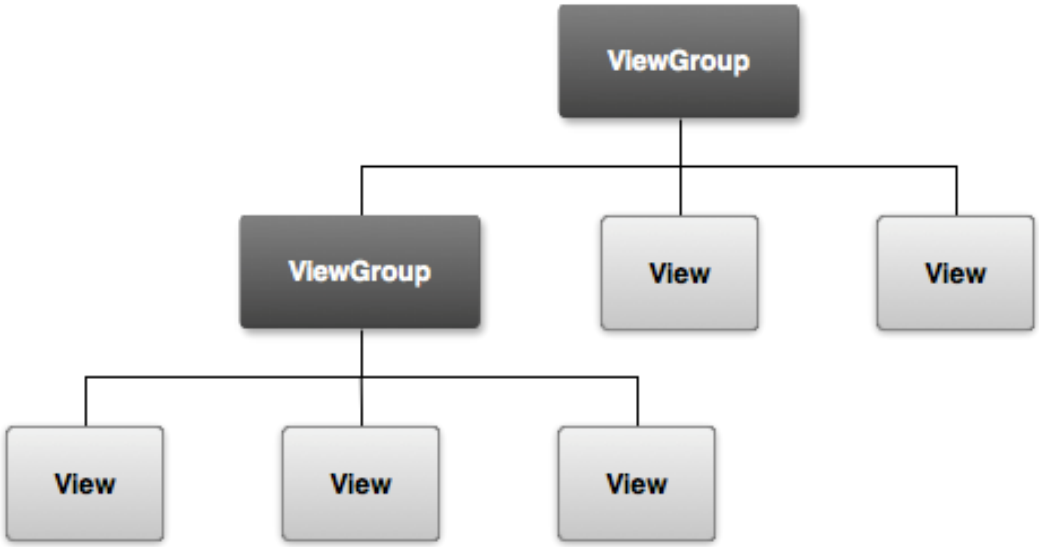
- View là đối tượng cơ bản để xây dựng mọi thành phần của giao diện đồ họa
- Hầu hết các thành phần cơ bản của giao diện đều kế thừa từ View: TextView, Button, Spinner, ToggleButton, RadioButton,…
- Các thành phần này hầu hết đều nằm trong gói android.widget nên thường gọi là widget
- Custom view: lập trình viên có thể tự tạo widget của riêng mình bằng cách tùy biến view để hoạt động theo cách của riêng mình

ViewGroup - Layout

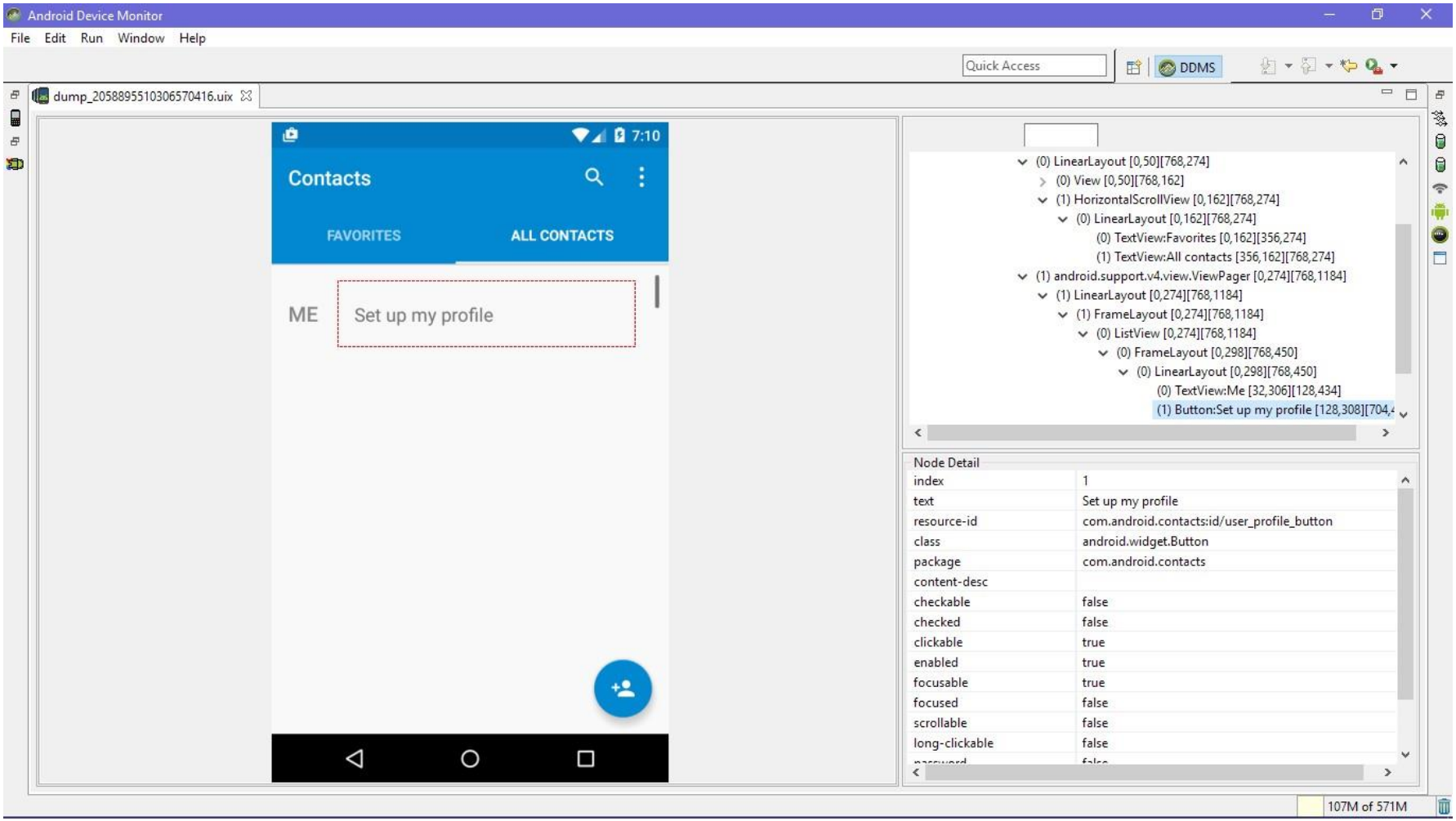
- ViewGroup là các view đặc biệt, có thể chứa bên trong nó các view khác
  - VD1: thông tin về ngày tháng gồm một số text
  - VD2: danh sách các ngày trong tháng gồm các button
- ViewGroup là cửa sổ cha của các view con
- Một view nằm trong ViewGroup cần phải có thông tin về vị trí của nó trong cửa sổ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

Các khái niệm cơ bản

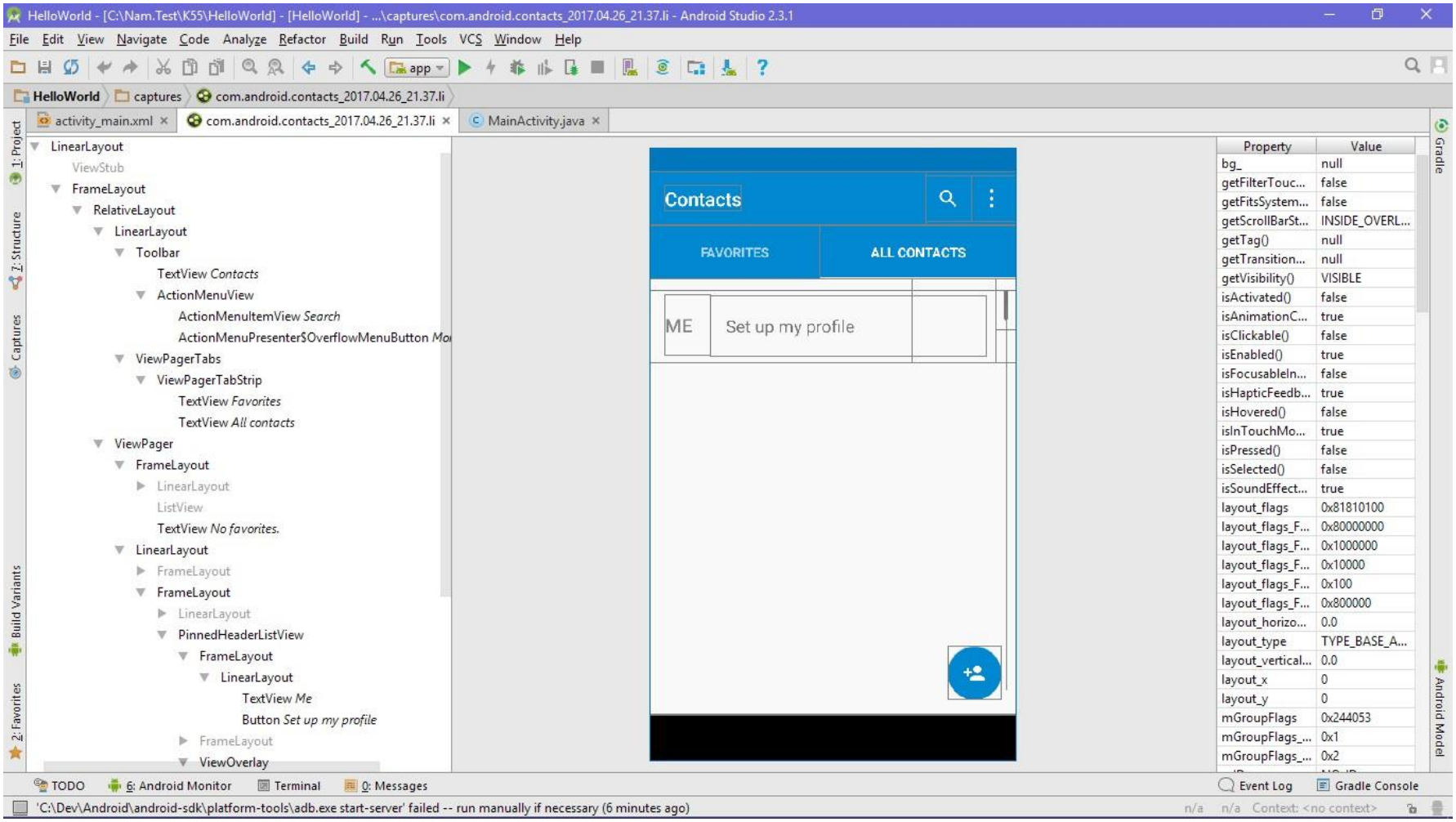
- View – Widget
- ViewGroup – Layout
- Tham khảo thiết kế của app:
  - Dùng công cụ Monitor của SDK
  - Dùng Layout Inspector



Công cụ Monitor (SDK)



Layout Inspector (Tools\Android)



Phần 2

Làm việc với layout

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 by 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 layout XML sau đó dùng bộ nạp LayoutInflater để tạo biến kiểu Layout

```
LayoutInflater.from(context)
        .inflate(R.layout.filename, null);
```

Layout by XML

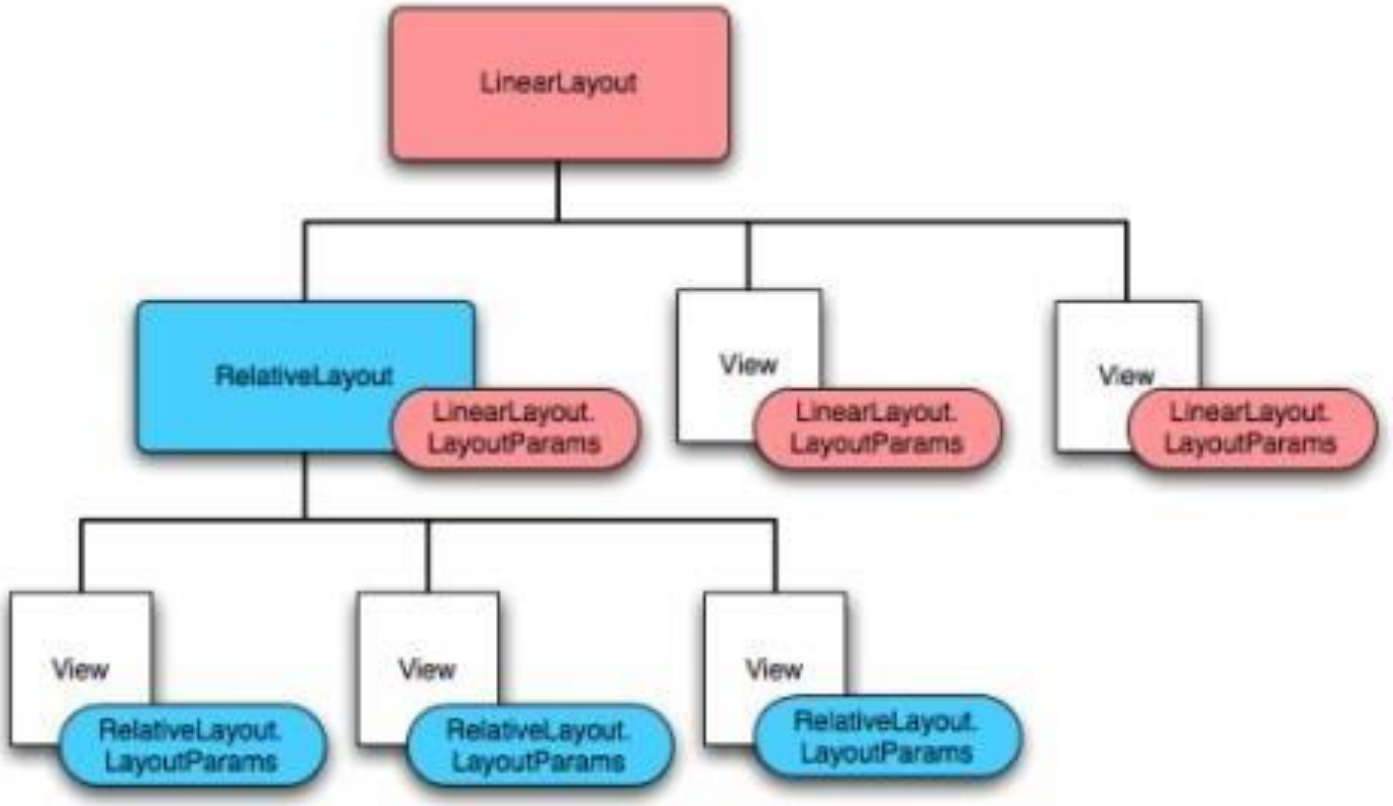
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://.../res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am 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>
```

Layout by 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

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

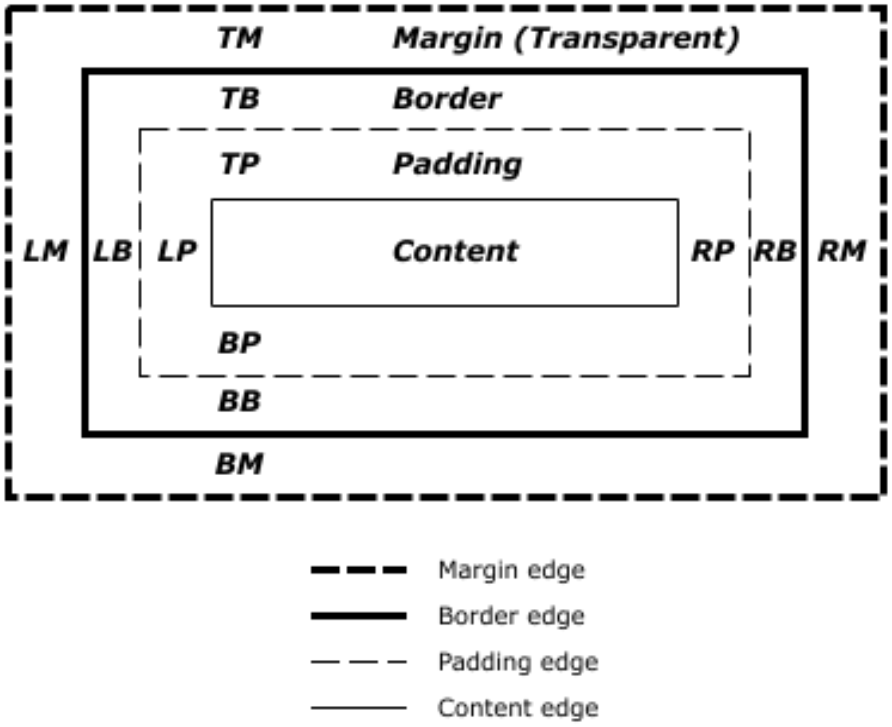


Tham số của layout và view

- 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



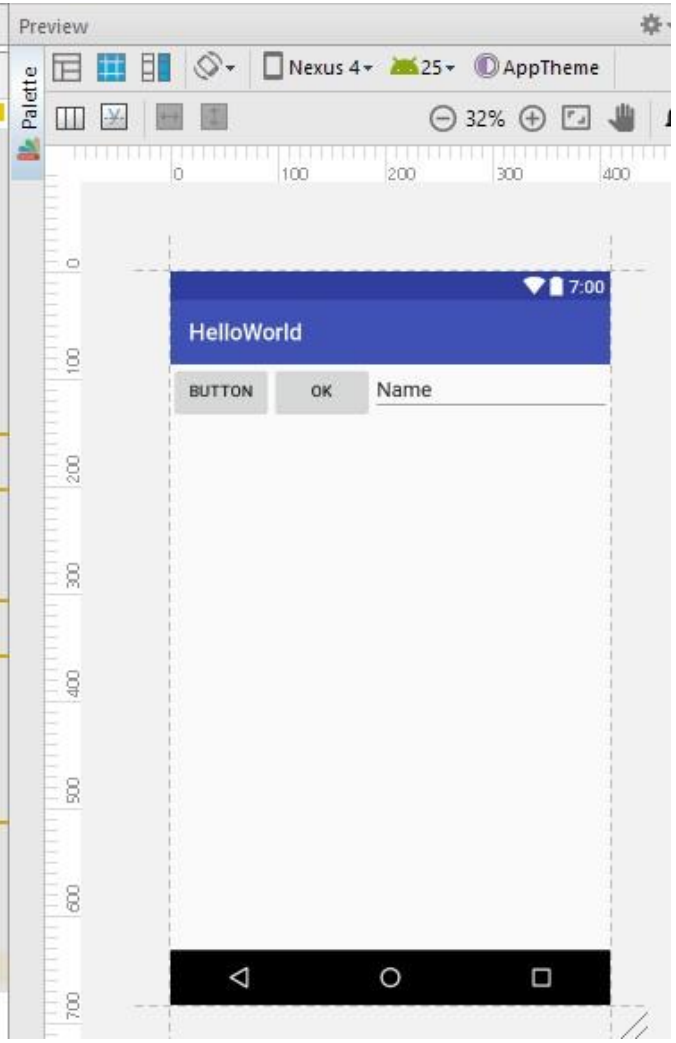
Phần 3

Một số layout thông dụng

LinearLayout

activity_main.xml | MainActivity.java

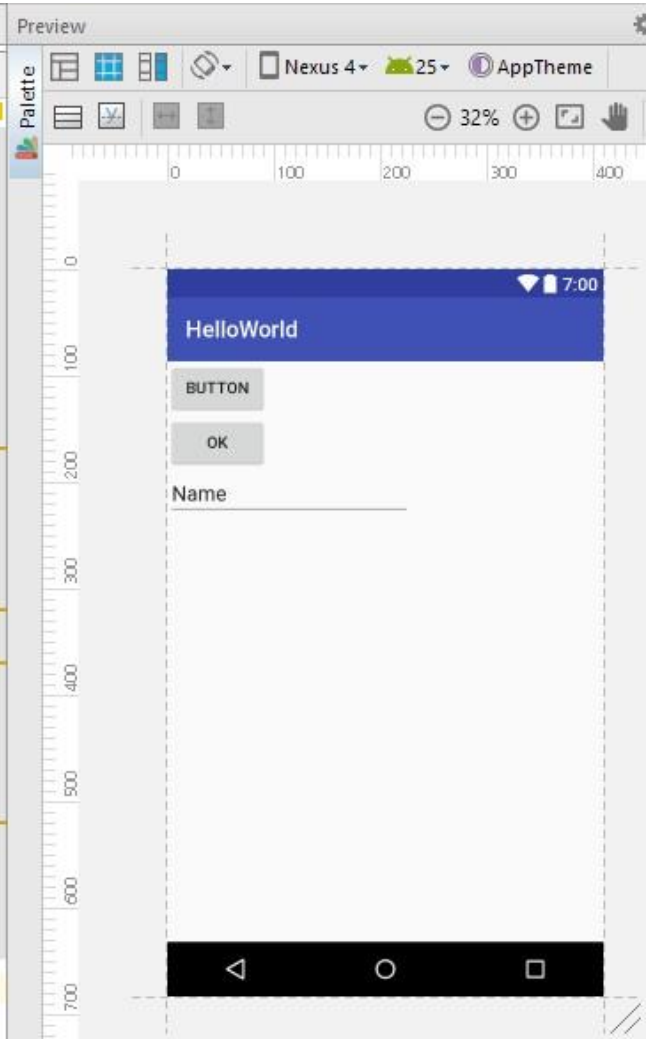
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.xuannam.helloworld.MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OK" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name" />
</LinearLayout>
```



LinearLayout

activity_main.xml | MainActivity.java

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

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="OK" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name" />
</LinearLayout>
```



LinearLayout

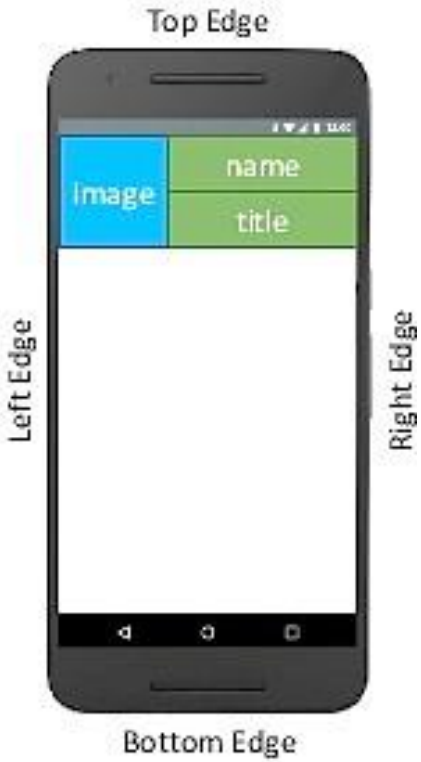
- Các view bên trong nó được xếp liên tiếp thành một hàng hoặc một cột
- Thuộc tính android:orientation quy định cách bố cục theo hàng hay theo cột, giá trị có thể là:
  - "vertical": các view bên trong LinearLayout được sắp xếp theo chiều dọc
  - "horizontal": các view bên trong sắp xếp theo chiều ngang
- LinearLayout không thay đổi kích thước các view con, chỉ điều chỉnh vị trí của chúng

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

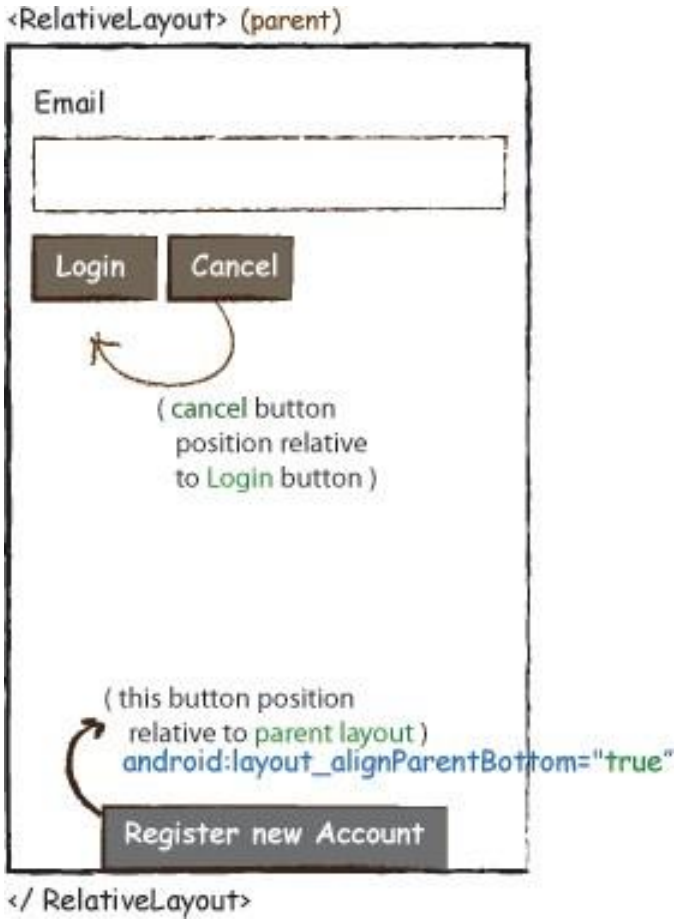
RelativeLayout

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/image"
        android:src="@drawable/thumbnail"
        android:layout_width="64dp"
        android:layout_height="64dp"
        android:scaleType="centerCrop"/>

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Mohammad Tarek"
        android:layout_toRightOf="@id/image" />

    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Software Engineer"
        android:layout_below="@id/name"
        android:layout_toRightOf="@id/image" />
</RelativeLayout>
```



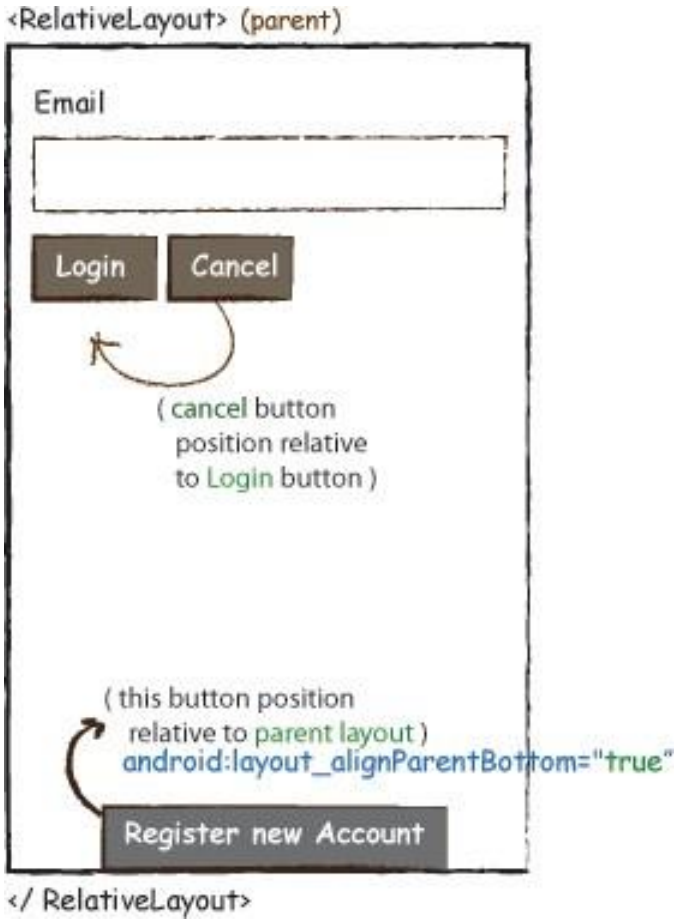
RelativeLayout

```
<TextView    android:id="@+id/label"
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/label"              />
<Button  android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
android:text="Register          new   Account"
android:layout_centerHorizontal="true"/>
```



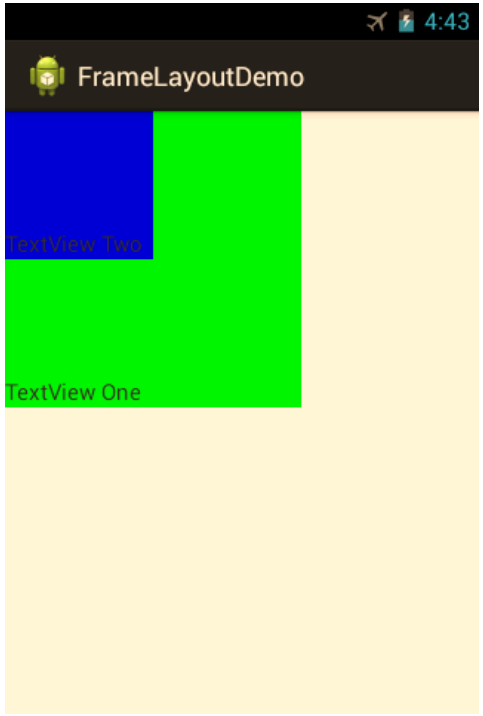
RelativeLayout

```
<Button     android:id="@+id/btnLogin"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/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
- Trường hợp 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

```
<?xmlversion="1.0"encoding="utf-8"?>
<TableLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_height="fill_parent"
  android:layout_width="fill_parent"
  android:background="#000044">
  <TableRow>
    <TextView android:text="User Name:"
    android:width="120px"
  />
   <EditText android:id="@+id/txtUserName"
    android:width="200px"/>
  </TableRow>
  <TableRow>
    <TextView android:text="Password:"
  />
   <EditText android:id="@+id/txtPassword"
    android:password="true"
  />
  </TableRow>
  <TableRow>
    <TextView/>
    <CheckBox android:id="@+id/chkRememberPassword"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Remember Password"
  />
  </TableRow>
  <TableRow>
    <Button android:id="@+id/buttonSignIn"
    android:text="Log In"/>
  </TableRow>
</TableLayout>
```

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

Phần 4

Tương tác với các view

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 (chẳng 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");
```

Phần 5

Một số điều khiển đơn giản

TextView

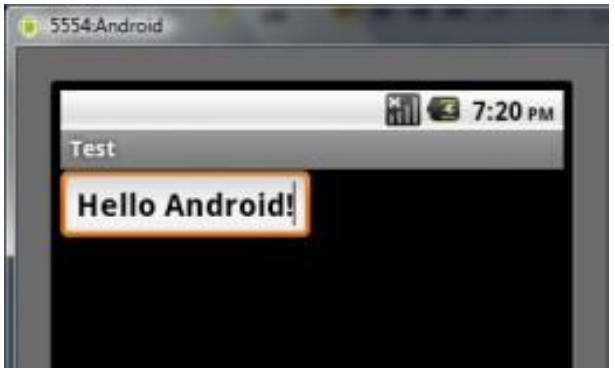
- Mục đích: hiển thị văn bản
- Một số thuộc tính hay được sử dụng:
  - android:layout_width
  - android:layout_height
  - android:text
  - android:textColor
  - android:textSize
  - android:gravity

```
<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()

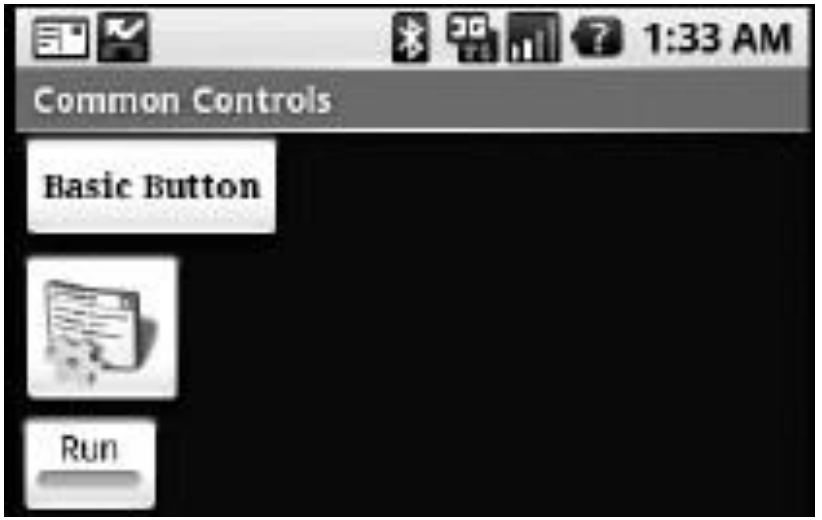
```
<EditText
        Android:id="@+id/EditText01"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:textStyle="bold"
        Android:textSize="20dip"
        Android:textColor="#000000"
        Android:text="Hello  Android!"
        Android:singleLine="true"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



ToggleButton

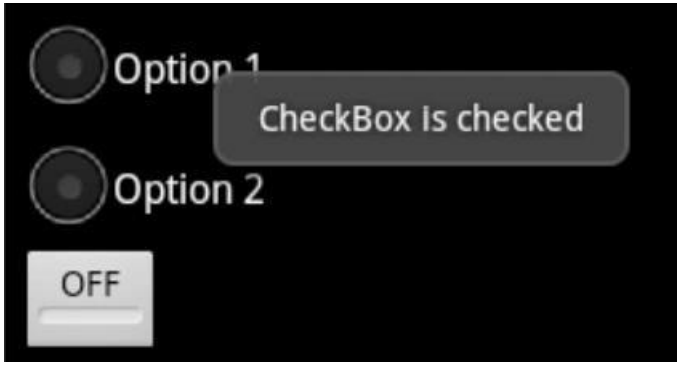
- Mục đích: hiển thị trạng thái on/off
- Thuộc tính quan trọng
  - "android:textOn": text hiện ở trạng thái on
  - "android:textOff": text hiện ở trạng thái off
  - "android:onClick": tương tự như ở Button
- Phương thức "bool isChecked()" trả về trạng thái on/off
- Từ phiên bản 4.0, android có điều khiển Switch tương tự như ToggleButton nhưng khác về cách hiển thị thông tin

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

RadioGroup & RadioButton

```
<RadioGroupandroid:id="@+id/rdbGp1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
>
<RadioButtonandroid:id="@+id/rdb1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Option 1"
/>
<RadioButtonandroid:id="@+id/rdb2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Option 2"
/>
</RadioGroup>
```



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)