

Lập trình trên thiết bị di động

# CÁC THÀNH PHẦN CƠ BẢN CỦA ỨNG DỤNG ANDROID

---

GV: Nguyễn Huy Cường

Email: [nh.cuong@hutech.edu.vn](mailto:nh.cuong@hutech.edu.vn)

# Nội dung

## 1. Các thành phần cơ bản của Android

 Activity

 Service


 ContentProvider

 BroadcastReceiver

 Intent

## 2. Activity và Controls cơ bản

 Activity - Vòng đời của Activity

 Thiết kế giao diện bằng **constraintLayout**

 Các widget cơ bản

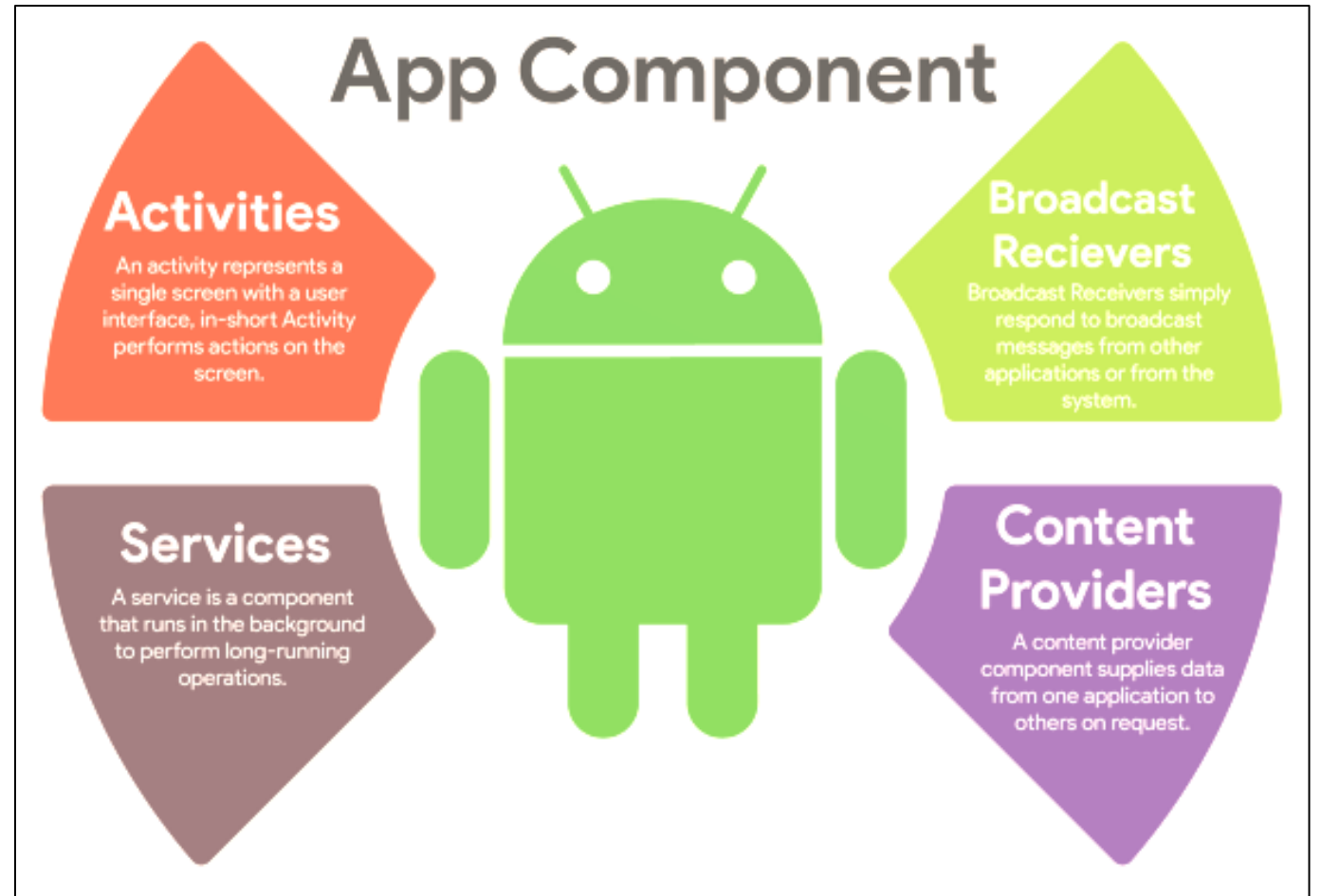
 Xử lý sự kiện

# CÁC THÀNH PHẦN CƠ BẢN CỦA ANDROID

---

# Các thành phần cơ bản

- Activity
- Service
- Content Provider
- Broadcast Receiver
- Intent



## Các thành phần cơ bản

### Activity

- **Activity** là giao diện màn hình, sử dụng **Fragment** và **View** để bố trí, hiển thị thông tin và tương tác với user. 1 ứng dụng (APP) có 1 hoặc nhiều **Activity**
- Activity bao gồm:
  - class.**java**: kế thừa từ lớp cha là Activity
  - file.**xml**: thiết kế giao diện người dùng.
- Để sử dụng các activity trong ứng dụng cần định nghĩa trong tệp kê khai (**AndroidManifest.xml**)

```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```



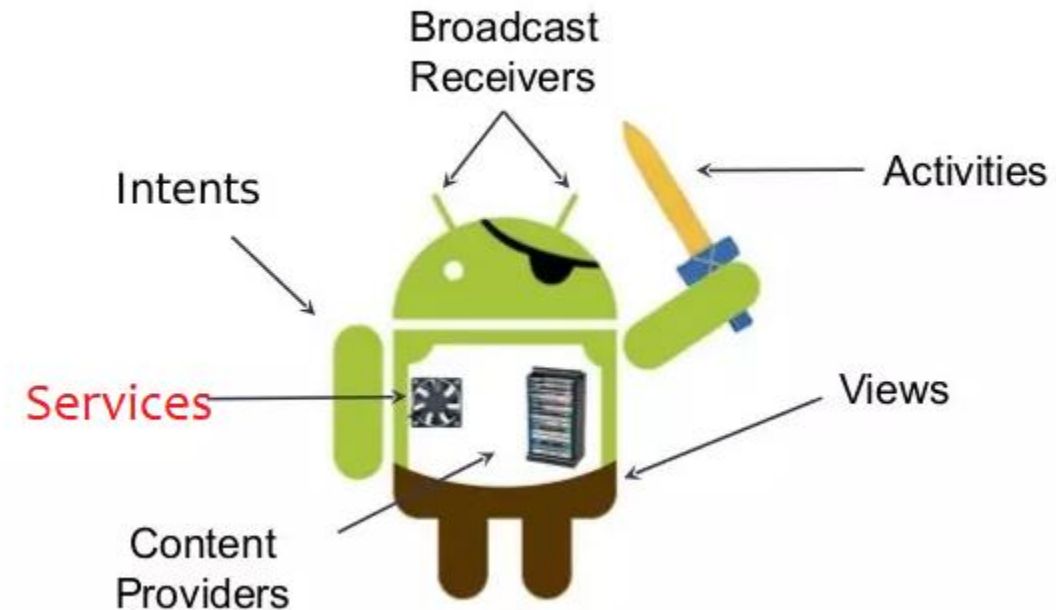
## Các thành phần cơ bản

### Service

- Service: không có phần giao diện, một thành phần chạy ngầm (chế độ nền) không có phần giao diện
- Kế thừa từ lớp cha là **Service**

Ví dụ:

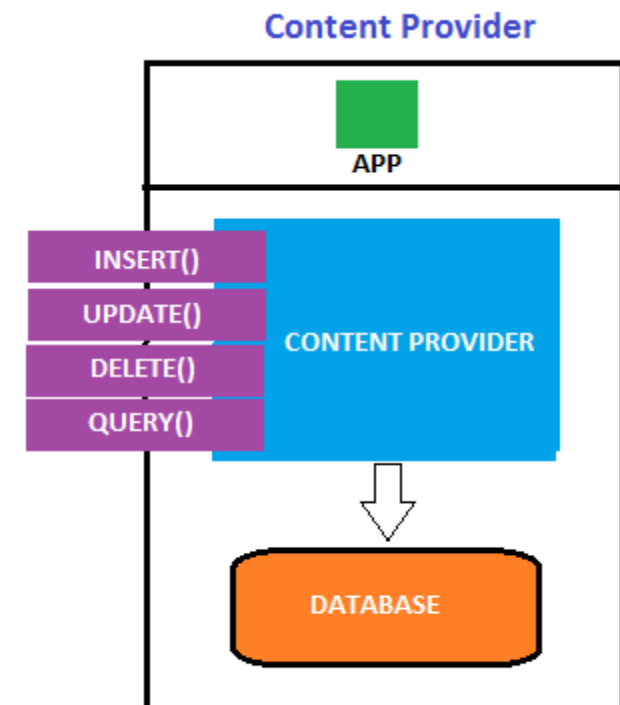
- Trình nghe nhạc
- Down File



## Các thành phần cơ bản

### Content Provider

- Là thành phần giúp ứng dụng có thể đọc, ghi dữ liệu từ một file hoặc từ SQLite của một ứng dụng khác trong cùng một hệ thống
  - ❑ Sqlite và thao tác với content Provider
  - ❑ Thêm xóa sửa dữ liệu.
  - ❑ Permission: quyền truy cập đến nguồn dữ liệu



## Các thành phần cơ bản

### BroadcastReceiver

- Là một thành phần giao tiếp với **hệ thống / ứng dụng**.

- ☐ Đăng ký **nhận** broadcast trong ứng dụng
- ☐ **Gửi** broadcast từ hệ thống/ ứng dụng

- Sử dụng: kế thừa từ **BroadcastReceiver**

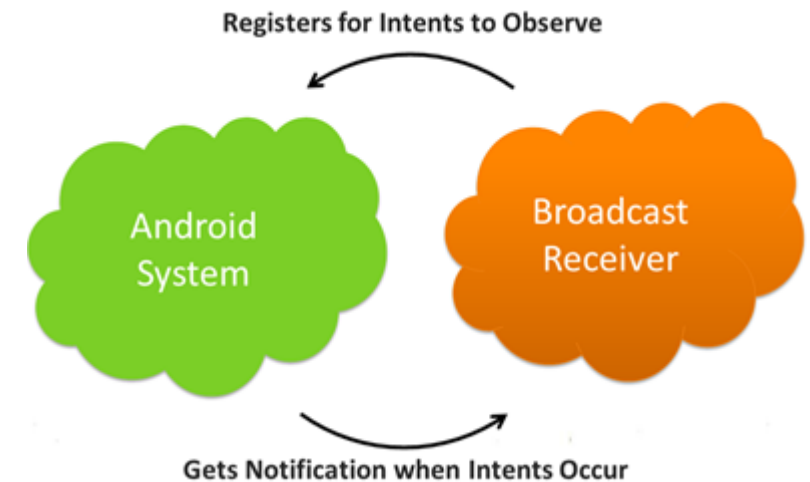
- **Ví dụ:**

Một số broadcast từ hệ thống:

- ☐ Thông báo pin yếu
- ☐ kết nối hay ngắt kết thiết bị ngoại vi...

Một số broadcast từ ứng dụng:

- ☐ Hẹn giờ, khi đến giờ hẹn, ứng dụng sẽ sử dụng broadcast báo thức, tạo ra notification trên màn hình để báo cho người dùng biết...





## Intent

- Intent cho phép các thành phần ứng dụng có thể yêu cầu các hàm từ các thành phần ứng dụng Android khác
- Có 2 loại chính là Explicit Intent (tường minh) và Implicit Intent

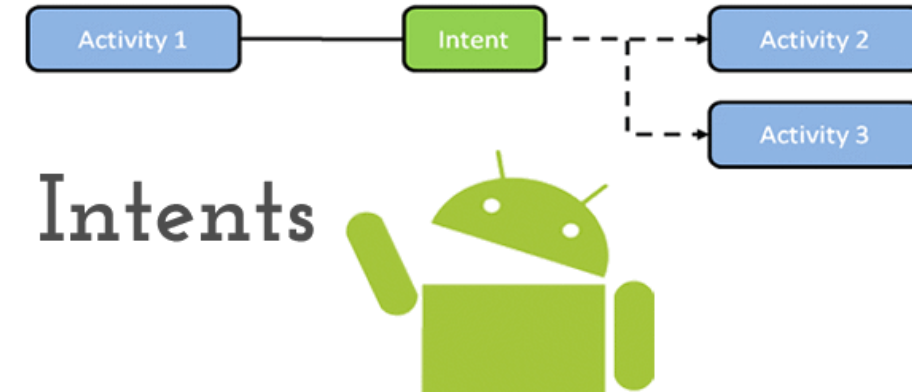
❑ **Explicit Intent** xác định cụ thể các thành phần tham gia hành động:

```
Intent intent = new Intent(FromActivity.this, ToActivity.class);
startActivity(intent);
```

❑ **Implicit Intent:** Loại Intents này chỉ ra hành động cần được thực hiện (action) và dữ liệu cho hành động đó (data)

- Cấu trúc của Intent:

Component name	Action	Data
Category	Extra	Flag



## Intent – ví dụ

- Ví dụ: Thông qua startActivity() bạn có thể xác định một Intent sử dụng để gọi chạy một Activity khác. Tại Activity mục tiêu, với startActivity() bạn có thể xác định được Intent của người gửi đến để khởi động Activity này.

```
Intent intent = new Intent(MainActivity.this, LoginActivity.class);  
startActivity(intent);
```



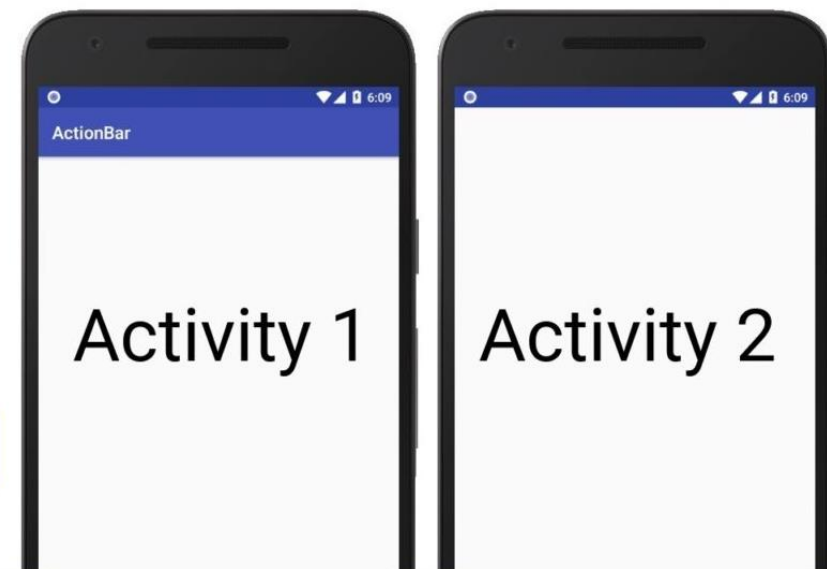
- **intents thường được sử dụng chính:** Start dịch vụ, Gọi một activity, Hiển thị một trang web, Hiển thị danh sách liên hệ, Gửi một tin nhắn, Gọi điện thoại...

# ACTIVITY VÀ CONTROLS CƠ BẢN

---

# Tổng quan Activity

- Activity đại diện cho một màn hình với giao diện người dùng (UI) của một ứng dụng
- Từ nhiều activity trong ứng dụng android, có 1 **activity chính** và đó là màn hình đầu tiên xuất hiện khi khởi chạy ứng dụng. Các thông tin trong tệp kê khai của ứng dụng (***AndroidManifest.xml***)
- Cần quản lý vòng đời hoạt động của Activity đúng cách.



# Tổng quan Activity

- **Back stack:** Các activity được sắp xếp trong một stack được gọi là **Back stack**, theo thứ tự **mở** của mỗi activity

❑ *Không nên đưa bản copy* của một Activity

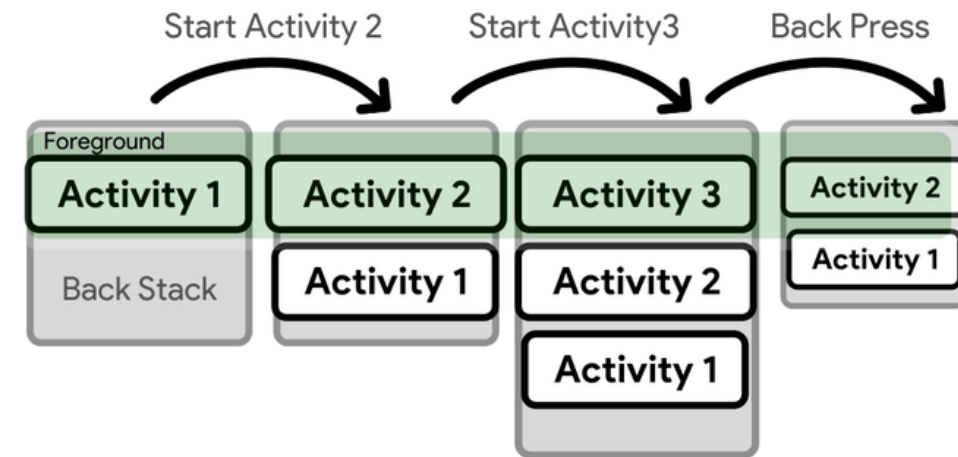
vào **Back Stack**

❑ Có thể code *thay đổi thứ tự backstack*

- Khởi tạo Activity bằng cách gọi **startActivity(Intent)**
- Sub-activity: Là activity được gọi bởi activity khác.

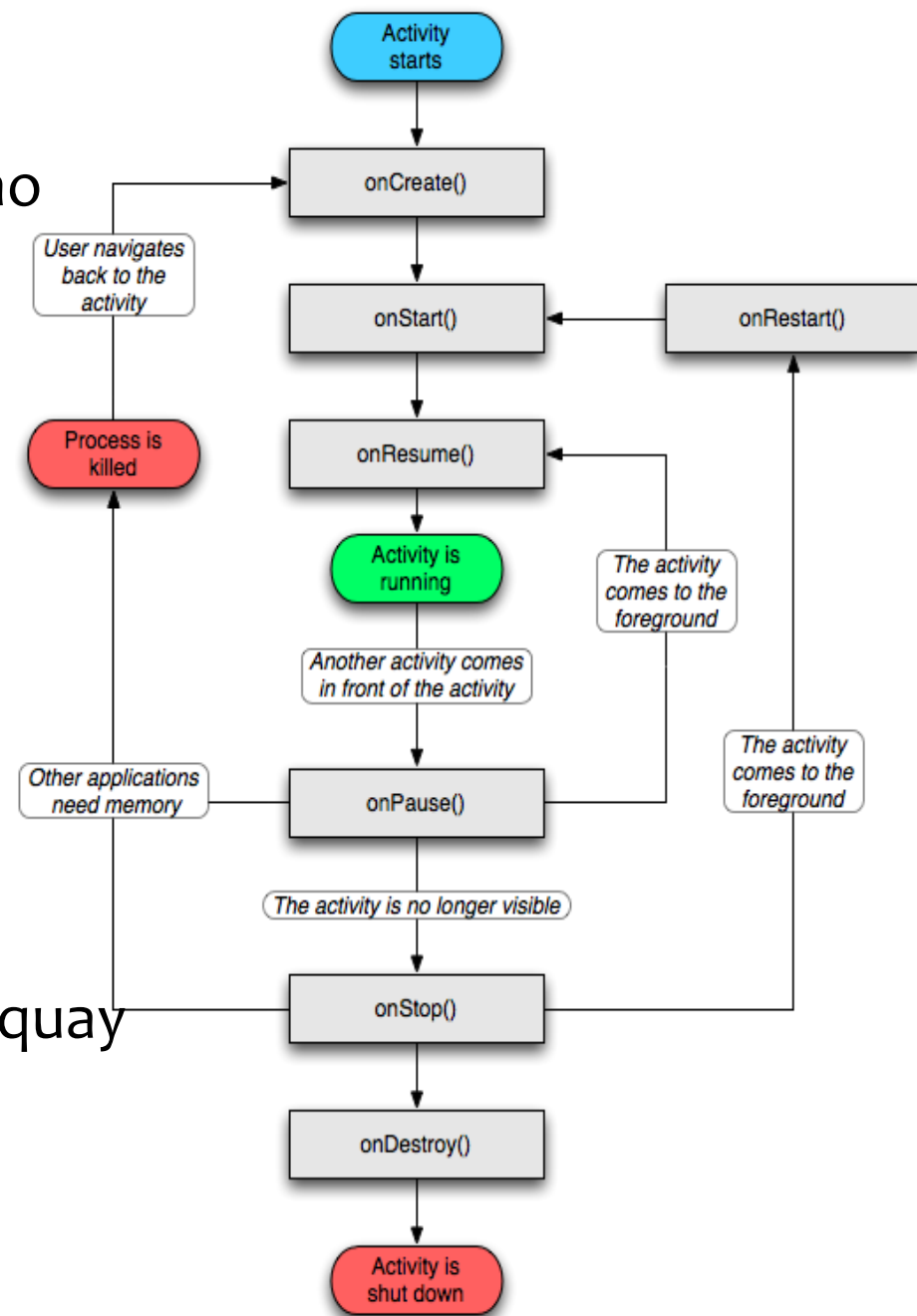
Có 2 kiểu gọi sub-activity:

- ❑ Không cần đợi kết quả trả về
- ❑ Gọi Sub-Activity sử dụng **startActivityForResult** thay thế bằng **registerForActivityResult**



# Vòng đời của Activity

- Khi activity được kích hoạt (**launched**), hệ thống đẩy vào Backstack
- Lần lượt các callback: **onCreate** – **onStart** – **onResume**  
Gọi là **Running**
- Khi bị chiếm quyền hiển thị **onPause**
  - ❑ Nếu không nhìn thấy nữa thì **onStop**
    - ❑ Quay lại: **OnRestart**– **onStart**– **onResume**
    - ❑ Bị thu hủy: **onCreate** – **onStart** – **onResume**
  - ❑ Đang bị Activity khác đè lên, mà người dùng sau đó quay về lại Activity cũ, thì **onResume()**.
- Bị hủy có chủ đích: **onDestroy()** và kết thúc vòng đời  
VD: nhấn nút **Back** ở System Bar, hay hàm **finish()** được gọi





## 1. Xây dựng project “Lab02.Demo” có **MainActivity**

**EditText:** Với giá trị ban đầu “Activity”

Button mở hộp thoại: **AlertDialog**

```
AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(view.getContext());
alertDialogBuilder.setTitle("hi");
alertDialogBuilder.setMessage("this is my app");
alertDialogBuilder.show();
```

## 2. Override tất cả các hàm trong vòng đời của **MainActivity** ?

**onCreate - onStart - onResume – onPause- onStop –onRestart- onDestroy.**

Với mỗi hàm **Override**, thực hiện việc nối chuỗi editText với các hàm trong vòng đời

VD: **editText.setText(editText.getText().toString() + "\_" + "onStart");**

## 3. Tìm hiểu vòng đời của Activity – Khi bắt đầu launch

3.1 Khi Mở hộp thoại

3.2 Khi mở 1 tab ứng dụng khác

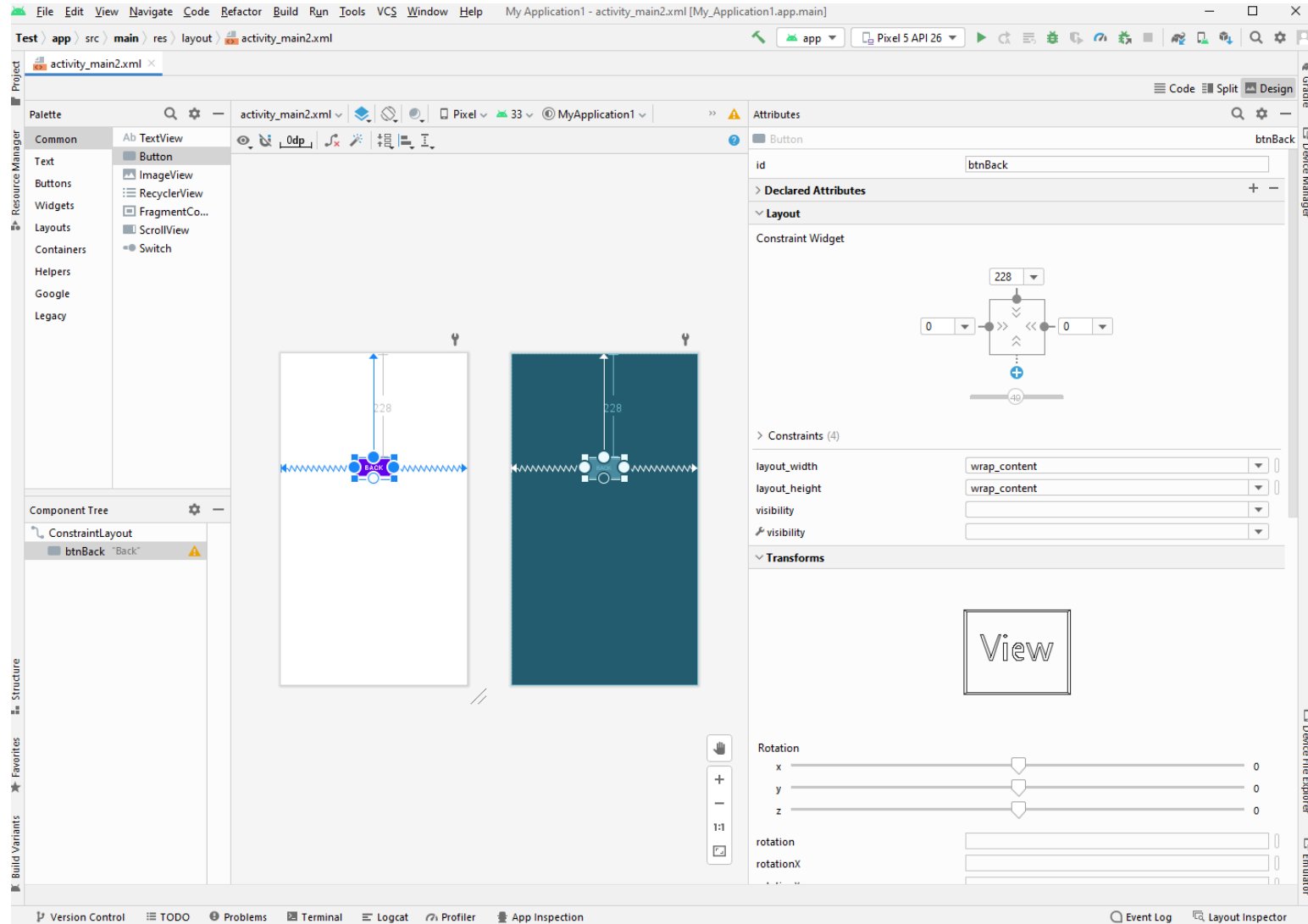
3.3 khi mở 1 hộp thoại (màn hình khác)

3.4 Khi mở activity khác, và có thể back về.



# Thiết kế giao diện ConstraintLayout

- **ConstraintLayout** (2018) là một layout để xây dựng giao diện nhanh và dễ dùng hơn so với kiểu code giao diện bằng XML truyền thống (RelativeLayout /LinearLayout)
- Mỗi một view phải có ít nhất **một điểm neo theo chiều ngang** và **một điểm neo theo chiều dọc**, nếu không đủ các điểm neo tối thiểu, hệ thống sẽ báo lỗi ở cửa sổ **Component tree**





# Thiết kế giao diện trên Activity với ConstraintLayout

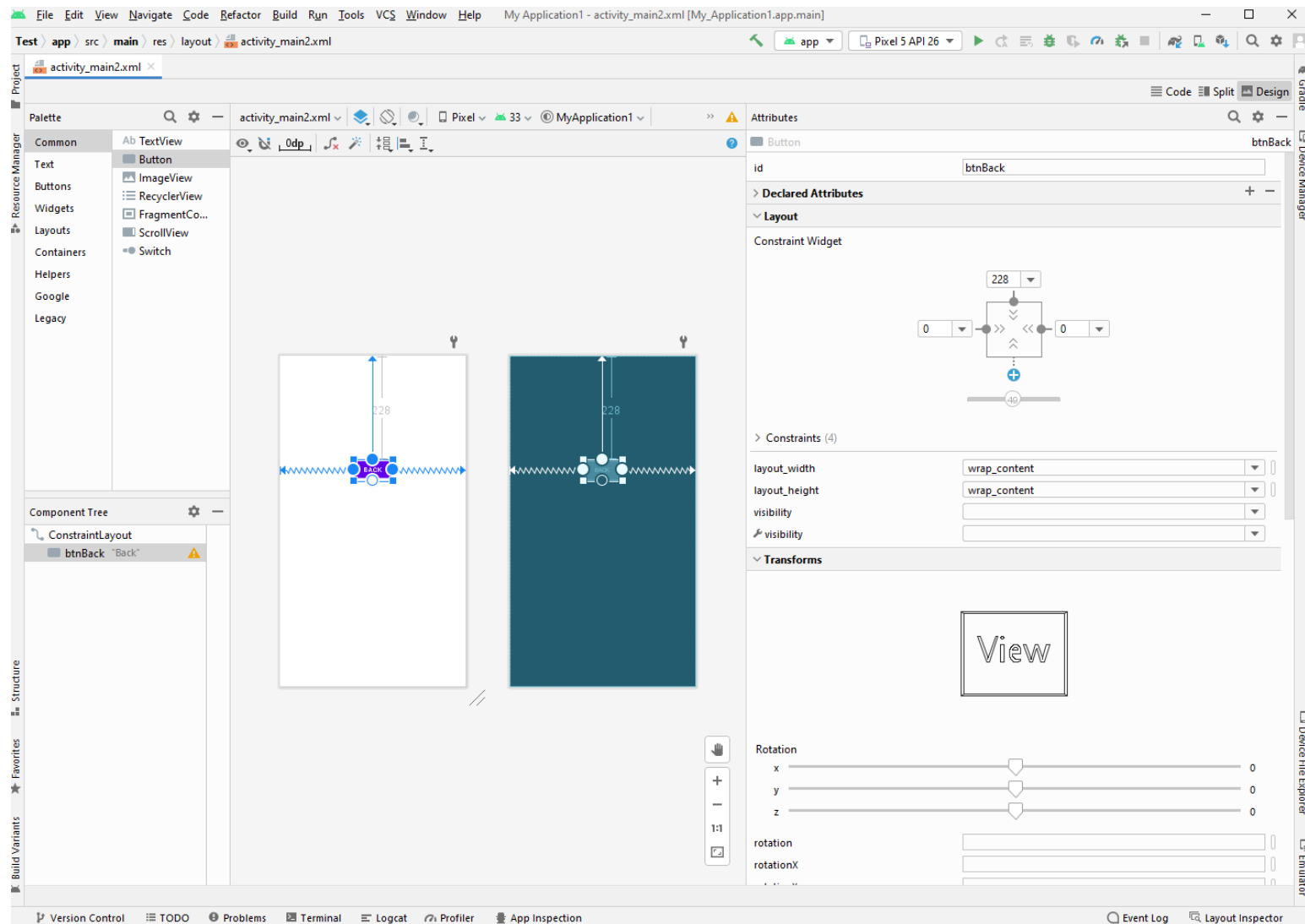
## ● Một số thuộc tính cơ bản của layout

1. **id** của View

2. **padding** của View

Các khoảng cách tới các thành viên bên trong

4. **layout\_width** và **layout\_height**



## Một số controls cơ bản

- Các Controls cơ bản: **TEXT**, **IMAGEVIEW**, **BUTTON**, **SPINNER**
- 1 số thuộc tính
  - ❑ **layout\_width**, **layout\_height** (bắt buộc)  
có các giá trị dp, px, in, mm, sp... hoặc **match\_parent**/ **wrap\_content**
  - ❑ **id**: thuộc tính xác định của View, được sử dụng lại trong code Java để ánh xạ đối tượng, tìm kiếm khi cần  

```
view??? = findViewById(R.id.view_id_name)
```
  - ❑ **gravity**: căn chỉnh text của nó sao cho canh trái, phải, giữa,... so với không gian của chính nó
  - ❑ **background**: màu nền

# Một số thuộc tính của View

## ● 1 số thuộc tính

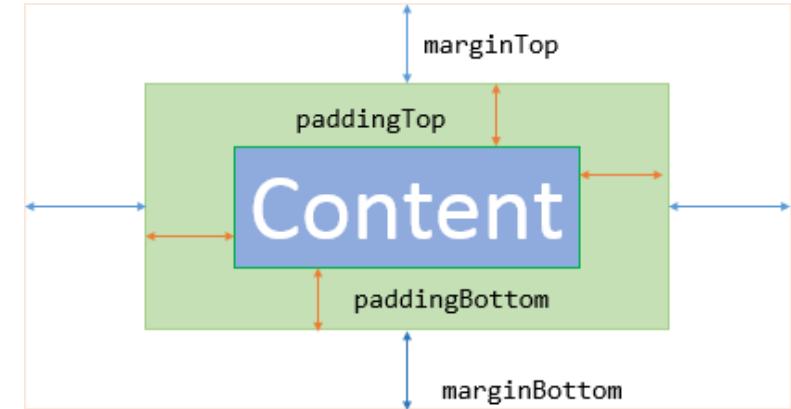
- ❑ **textColor**: màu chữ
- ❑ **text**: nội dung văn bản hiển thị
- ❑ **padding**: set khoảng cách giữa biên của view đến các thành phần con của nó. Đơn vị tính dp hoặc dip.

Nếu muốn khoảng cách riêng cho từng cạnh biên -> tách biệt từng thuộc tính cụ thể của padding như **paddingTop**, **paddingBottom**, **paddingStart** (**paddingLeft**) và **paddingEnd** (**paddingRight**).

- ❑ **margin** – set khoảng cách giữa biên của view đến các thành phần bên ngoài của nó. Đơn vị tính dp hoặc dip.

Tương tự có thể set **marginTop**, **marginBottom**, **marginStart** (**marginLeft**) và **marginEnd** (**marginRight**).

- ❑ **ellipsize**: text bị cắt và hiển thị “...” khi không đủ không gian để chứa



## Một số controls cơ bản

### 1- Text: TextView, EditText, Password, E-mail, Phone

- Text dùng để hiển thị thông tin

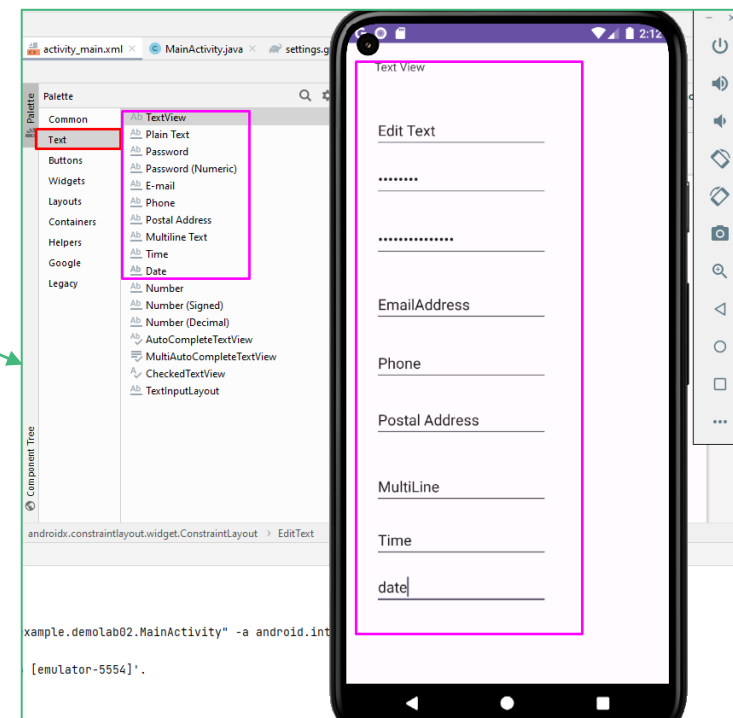
- ☐ TextView không cho phép người dùng chỉnh sửa.
- ☐ EditText là mở rộng cho phép chỉnh sửa

- Thuộc tính:

- ☐ **inputType**: kiểu nhập liệu
- ☐ **textSize**: Kích cỡ của text, kích cỡ này được tính theo đơn vị sp
- ☐ **hint, maxLines, lines, textAllCaps ...**

- Phương thức: **setText**

Giá trị	Ý nghĩa
<b>date</b>	Nhập ngày tháng
<b>datetime</b>	Nhập ngày tháng, giờ
<b>number</b>	Nhập số
<b>numberDecimal</b>	Nhập số thập phân
<b>numberSigned</b>	số nguyên không dấu
<b>phone</b>	Nhập số điện thoại
<b>text</b>	Nhập văn bản
<b>textMultiLine</b>	Chữ trên nhiều dòng
<b>textPassword</b>	Nhập password
<b>textUri</b>	Địa chỉ URL
<b>time</b>	Thời gian



## Một số controls cơ bản

### 1- Text: TextInputLayout, TextInputEditText

- **TextInputLayout:** Hiển thị và nổi lên đoạn text khi người dùng gõ vào EditText

- ❑ TextInputEditText là làm phần tử con của phần tử

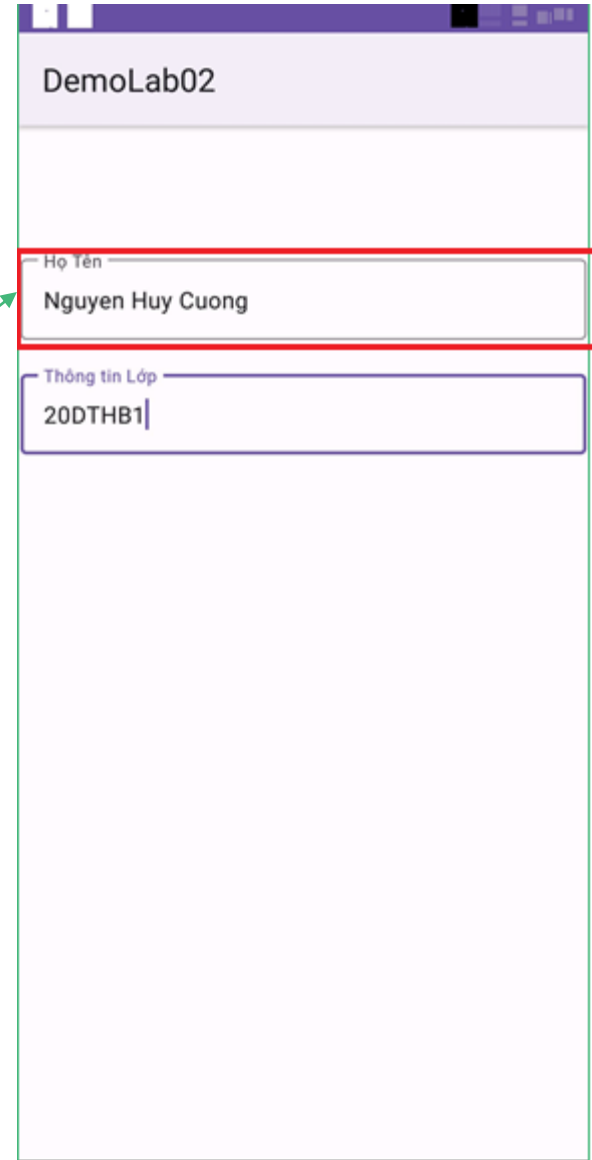
#### TextInputLayout

- ❑ Hiển thị các thông tin như: gợi ý (hint)

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/txtHoTen"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="1dp"
    android:layout_marginTop="87dp"
    android:layout_marginEnd="1dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
```

```
<com.google.android.material.textfield.TextInputEditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Họ Tên" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```



## ❑ Hiển thị số ký tự nhập

`app:counterEnabled="true"`

## ❑ Để thiết lập số ký tự lớn nhất `app:counterMaxLength="7"`

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="1dp"
    android:layout_marginTop="18dp"
    android:layout_marginEnd="1dp"
    app:counterEnabled="true"
    app:counterMaxLength="7"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/txtHoTen">
```

```
<com.google.android.material.textfield.TextInputEditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Thông tin Lớp"
    android:maxLength="7" />
```

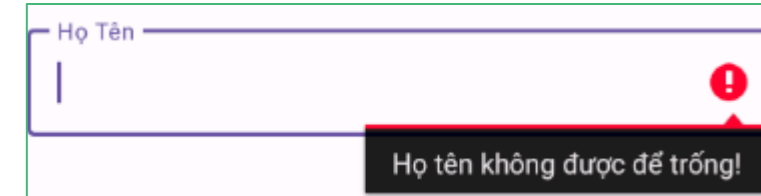
```
</com.google.android.material.textfield.TextInputLayout>
```

Thông tin Lớp

20DTHB

6/7

❑ Để hiển thị dòng thông báo lỗi, thiết lập `app:errorEnabled="true"` và dòng thông báo lỗi được thiết lập bằng code Java: `setError()`



```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/txtHoTen"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="1dp"
    android:layout_marginTop="87dp"
    android:layout_marginEnd="1dp"
    app:errorEnabled="true"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/errortxtHoten"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Họ Tên" />
</com.google.android.material.textfield.TextInputLayout>
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    TextInputEditText errortxtHoTen = findViewById(R.id.errortxtHoten);
    errortxtHoTen.addTextChangedListener(new TextWatcher() {

        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            if (s.length() == 0)
                errortxtHoTen.setError("Họ tên không được để trống!");
            else
                errortxtHoTen.setError(null);
        }

        @Override
        public void afterTextChanged(Editable s) {

        }
    });
}
```

## 2- ImageView

- ImageView là loại View dùng để hiển thị tài nguyên hình ảnh: ảnh Bitmap/ Drawable. Cung cấp các chức năng tùy biến khác nhau như đổ màu (tint) vào ảnh, co/kéo/cắt ảnh khi hiển thị trên View...
- các thuộc tính cần lưu ý:
  - ❑ **src** – Gán tài nguyên ảnh vào ImageView
  - ❑ **adjustViewBounds** - Nếu nhận giá trị **true** thì *ImageView* tự động co biên vừa với ảnh.  
(cần có thiết lập chiều rộng hoặc cao là *wrap\_content*)
  - ❑ **scaleType** - thiết lập thu phóng ảnh, nhận các giá trị như: **center**, **centerInside**, **centercrop**, **fitcenter**, **fitStart**, **fitEnd**, **fitXY**,...





# Một số controls cơ bản

## 2- ImageView

Giá trị	Ý nghĩa
<b>center</b>	Đặt ảnh vào giữa ImageView, không có thay đổi tỷ lệ ảnh.
<b>centerCrop</b>	Đặt ảnh vào giữa ImageView, có thu phóng ảnh (nhưng giữ nguyên tỉ lệ cao / rộng) sao cho ảnh phủ kín hết cả ImageView (phần thừa bị cắt)
<b>centerInside</b>	Đặt ảnh vào giữa ImageView, có thu phóng ảnh (nhưng giữ nguyên tỉ lệ cao / rộng) sao cho toàn bộ các phần của ảnh hiện thị trên ImageView.
<b>fitCenter</b>	Đặt ảnh vào giữa ImageView, có thu phóng ảnh (nhưng giữ nguyên tỉ lệ cao / rộng) sao cho toàn bộ các phần của ảnh hiện thị trên ImageView.
<b>fitEnd</b> <b>fitStart</b>	Co ảnh vừa View, vị trí ảnh ở cuối (ở đầu) ImageView
<b>fitXY</b>	Co ảnh vừa khít cả chiều rộng và cao.



Một số controls cơ bản

### 3- Button: Button

- Button là 1 View, hiển thị nút bấm để chờ người dùng nhấn vào.
- Mở rộng từ **TextView** nên nó có đầy đủ các thuộc tính của TextView như: **text, textColor, textSize, textStyle, textAllCaps, ...**
- Ngoài thuộc tính được kế thừa còn 1 số thuộc tính:
  - ❑ **drawableStart, drawableEnd, drawableTop, drawableBottom**: gán các ảnh Drawable vào biên **trái, phải, trên, dưới** của button.
  - ❑ **minHeight, minWidth**: giảm kích thước của Button (dp)
  - ❑ **maxLines**: số dòng text
  - ❑ **backgroundTint**: màu nền



Một số controls cơ bản

## 2- Button: Button

- Sự kiện

*//Ảnh xạ View*

```
Button btnAddToCart = findViewById(R.id.btnAddToCart);
```

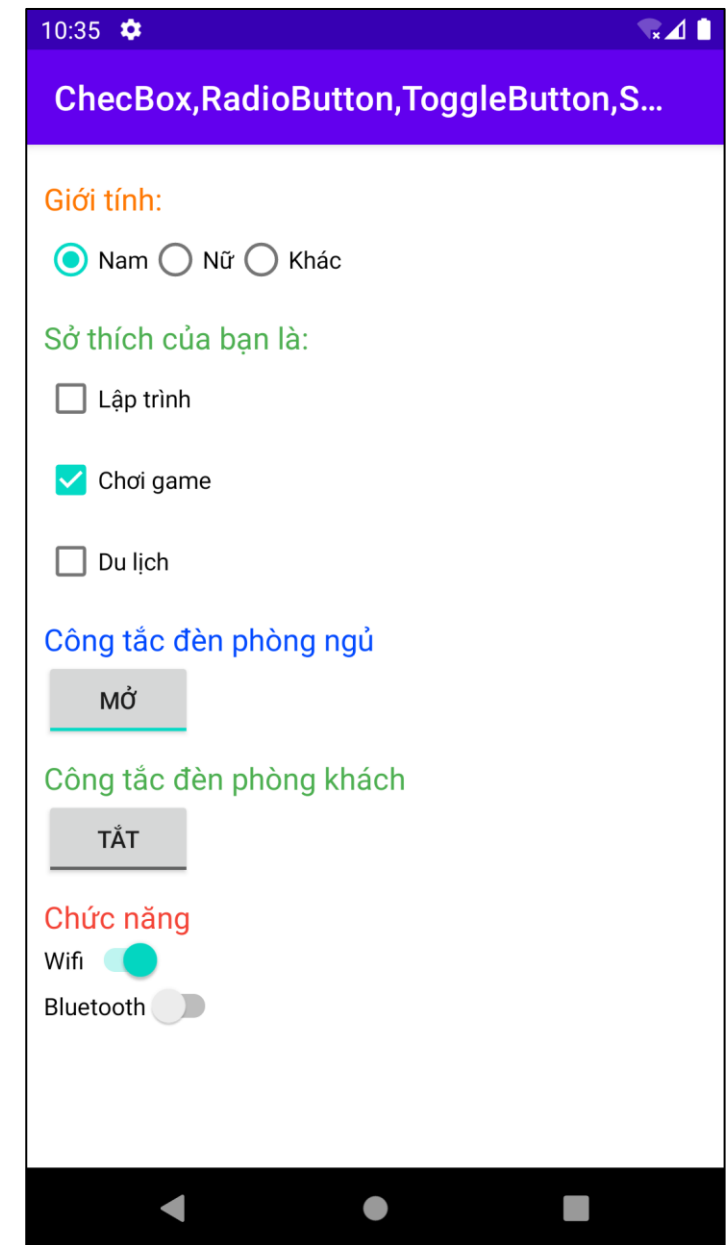
*//Bắt sự kiện click*

```
btnAddToCart.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) {  
        //Thực hiện xử lý khi nhấn vào nút  
        Toast.makeText(MainActivity.this,  
            "Đã thêm sản phẩm vào giỏ hàng",  
            Toast.LENGTH_SHORT).show();  
    }  
});
```

Một số controls cơ bản

### 3- Button: CheckBox, RadioButton, ToggleButton, Switch

- 4 view đều có 2 trạng thái **checked** và **unchecked**
  - ☐ **Checkbox** cho phép chọn nhiều lựa chọn
  - ☐ **RadioButton** cho phép chọn 1 trong nhiều lựa chọn (phải bỏ vào một RadioGroup)
  - ☐ **ToggleButton** và **Switch** chỉ khác nhau về hình dạng. Có thêm 2 thuộc tính là `textOn` và `textOff` cho text ở 2 trạng thái
- Gán trạng thái mặc định trong layout dùng: `android:checked="true"`
- Một số sự kiện
  - ☐ Xét trạng thái hiện tại: `isChecked()`
  - ☐ Gán trạng thái hiện tại: `setChecked(true/false)`



### 3- Button: CheckBox, RadioButton, ToggleButton, Switch

- Một số sự kiện

❑ Đảo trạng thái hiện tại: `.toggle();`

❑ Bắt sự kiện khi có sự thay đổi trạng thái: `setOnCheckedChangeListener`

```
cb.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {

    }
});
```

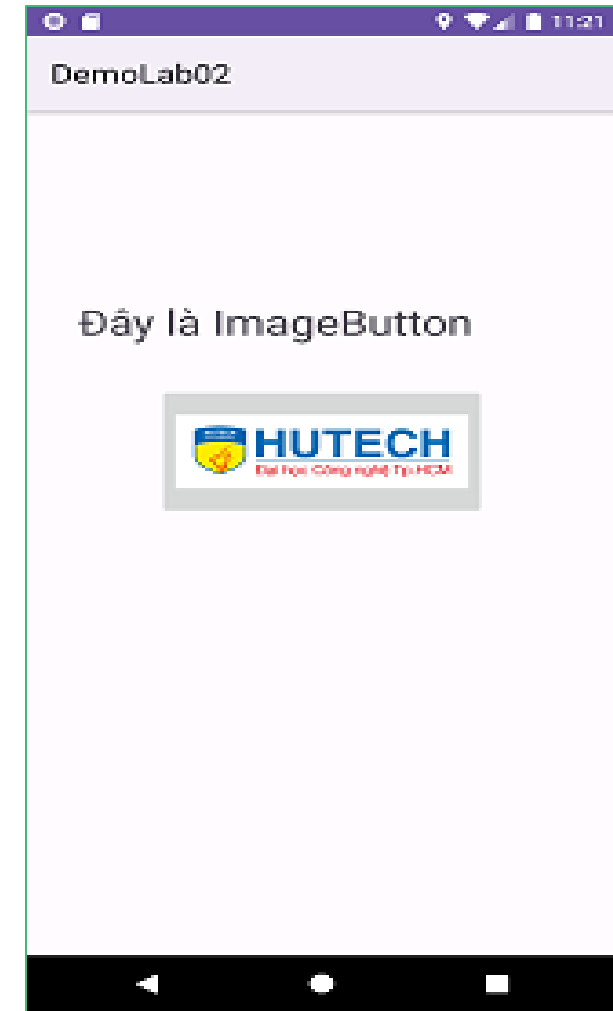
❑ Xét trạng thái theo RadioGroup:

```
RadioGroup group= (RadioGroup) findViewById(R.id.radiogroup1);
int idChecked = group.getCheckedRadioButtonId();
switch (idChecked)
{
    case R.id.radioButton: break;
}
```

Một số controls cơ bản

### 3- Button: ImageButton

- ImageButton kế thừa từ **ImageView** nhưng hiển thị như một nút bấm, có một chút.
  - ❑ khác với Button đã biết là nó hiện thị ảnh ở giữa nút bấm thay vì **text**



## 4- Spinner

- Spinner: Cho phép chọn giá trị từ 1 tập danh sách thả xuống
- Code Java

Tạo **ArrayAdapter**  
và **setAdapter**

- Sự kiện

- ☐ onItemClick
- ☐ onNothingSelected

Bạn học chuyên ngành nào?

Công nghệ phần mềm  
Mạng máy tính và TT  
An Toàn TT

```
Spinner spinnerMajor = findViewById(R.id.spinnerMajor);
List<String> listMajor = Arrays.asList("Công nghệ phần mềm", "Mạng máy tính và TT", "An Toàn TT");

//1. đưa dữ liệu vào spinner
ArrayAdapter<String> stringArrayAdapter = new ArrayAdapter<String>(context: this, android.R.layout.simple_spinner_item, listMajor);
spinnerMajor.setAdapter(stringArrayAdapter);

//2. sự kiện trong spinner
spinnerMajor.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        String strMsg = "Đã chọn " + listMajor.get(position);
        Toast.makeText(context: MainActivity.this, strMsg, Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
});
```

# Xử lý sự kiện trên Android

## ● Một số Event Listeners

- ☐ onClick(): Tương tự như hàm onTouch() nhưng chỉ có kích bản chạm và nhấc tay lên.
- ☐ onLongClick(): Khi người dùng chạm và giữ tay ở màn hình.
- ☐ onFocusChange(): Khi focus hoặc mất focus vào một view.
- ☐ onKeyDown():khi người dùng focus trên widget và nhấn (presse) hoặc thả (release) một phím trên thiết bị.
- ☐ onTouch(): khi người dùng chạm vào màn hình với nhiều kích bản chạm.
- ☐ onOptionsItemSelected(): khi người dùng chọn một mục trong menu.
- ☐ onCreateContextMenu():khi người dùng chọn một mục trong menu ngữ cảnh (Context Menu)

## ● Các cách xử lý Event Listeners trong Android

### ☐ Cách 1. Implement trực tiếp trên Activity

```
Button btn = findViewById(R.id.btnButton1);
btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //C1: Implement click vào Button1
    }
});
```



# Xử lý sự kiện trên Android

## ● Các cách xử lý Event Listeners trong Android

### ❑ Cách 2. Implement trực tiếp trên Activity

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout);

        findViewById(R.id.btnButton1).setOnClickListener(onclick1);
    }

    View.OnClickListener onclick1 = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //Implement Click button
        }
    };
}
```

### ❑ Cách 3. Implement trên Interface

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout);
    }


    @Override
    public void onClick(View view) {
        if(view.getId() == R.id.btnButton1)
        {
            //Implement onClick cho Button có id = btnButton1
        }
    }
}
```



- ☐ Thiết kế giao diện
- ☐ Xử lý hiển thị nội dung sau khi lưu thông tin

9:51

Lab2\_PS23456

  
android

Nhập tên bạn

Nhập MSSV

Nhập tuổi

Giới tính: ☐ Nam ☐ Nữ

Sở thích:

☐ Đá bóng

☐ Chơi game

LƯU

Nội dung hiển thị sau khi nhấn nút Lưu

10:10

Lab2\_PS23456

Nguyen Van A

PS123456

20

Giới tính: ☒ Nam ☐ Nữ

Sở thích:

☒ Đá bóng

☐ Chơi game

LƯU

Tôi tên: Nguyen Van A  
MSSV: PS123456  
Tuổi: 20  
Giới tính: Nam  
Sở thích: Đá bóng

## ASM 2.1

1. Thiết kế giao diện cho Activity “**Đăng Nhập**” và “**Đăng ký**”
2. Click vào **đăng ký** tài khoản sẽ mở ra Activity “Đăng ký”
3. Ở Activity “Đăng Ký”
  - 3.1 Đã có tài khoản?: Quay lại trang đăng nhập
  - 3.2 Click Button “Đăng ký” **kiểm tra** các giá trị nhập và cho phép đăng ký. Thông báo thành công!
  - 3.3 Khi đăng nhập đúng thông tin đăng ký (email/password)  
Thông báo cho người dùng: Thành công hoặc chưa có thông tin đăng ký

**Đăng Nhập**

Nhập Email

Nhập Password

[Quên Password?](#)

LOGIN

[Chưa có tài khoản? Đăng ký](#)

Google Facebook

**Đăng ký**

Nhập Username

Nhập Email

\*\*\*\*\*

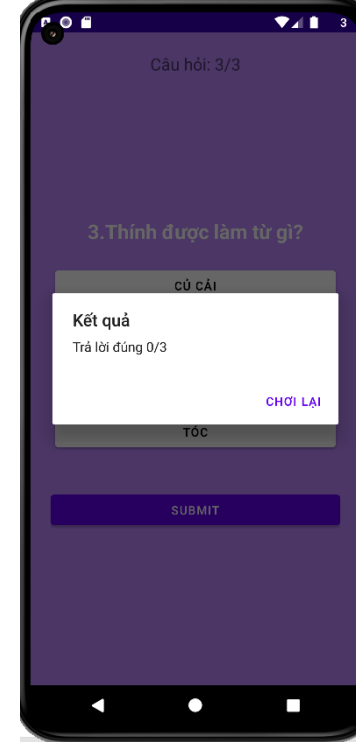
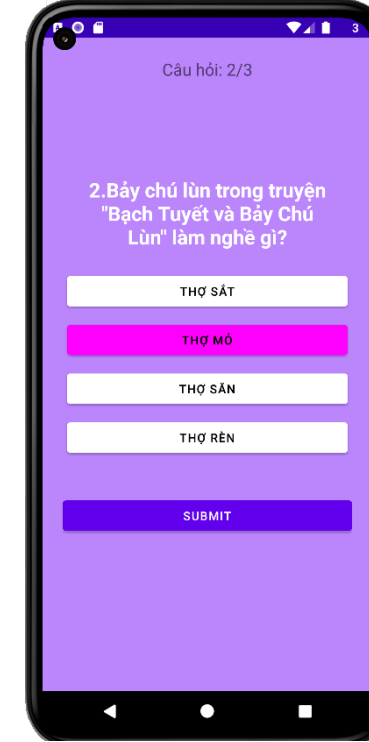
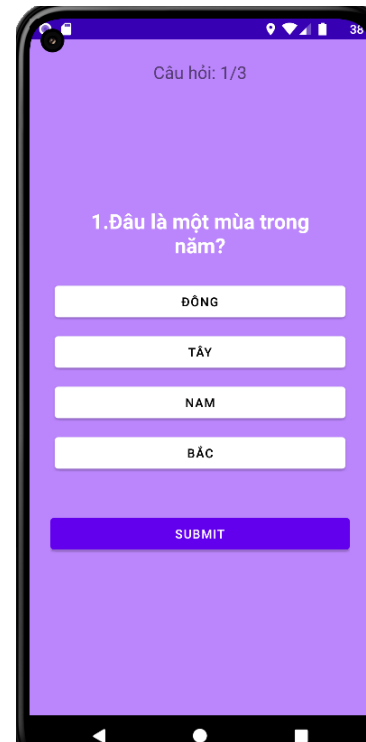
\*\*\*\*\*

ĐĂNG KÝ

[Đã có tài khoản?](#)

## ASM 2.2

1. **Thiết kế giao diện** chương trình trắc nghiệm ? Hiển thị câu hỏi đầu tiên và các đáp án?
2. **Thay đổi màu** đáp án khi người dùng lựa chọn ?
3. Giả sử chương trình có 3 câu hỏi:
  - 3.1 Nhấn “**Submit**” đi tới câu hỏi tiếp theo ?
  - 3.2 Xử lý kết quả **trả lời ở câu cuối cùng** ?  
(Đưa ra câu trả lời đúng/Tổng)
  - 3.3 Nhấn “**chơi lại**” để về câu hỏi đầu tiên ?
4. Thêm nút “**Back**” để quay lại câu trả lời ở trạng thái trước đó (chỉ **Enable** nút Back từ câu hỏi số 2)



## ASM 2.3

Thiết kế và thực hiện chương trình trò chơi **đoán số**:

(1) Khi nhấn nút New (bắt đầu trò chơi mới)

- Máy tính sẽ phát sinh ngẫu nhiên một số có 3 chữ số từ 100 đến 999
- Người chơi sẽ đoán số này bằng cách nhập vào một số có 3 chữ số.
- Người chơi sẽ đoán số này bằng cách nhập vào một số có 3 chữ số.
- Sau mỗi lần đoán, máy tính sẽ phản hồi dựa trên số đã đoán của người chơi:
  - ❑ Dấu '+' được sử dụng để chỉ ra rằng một chữ số trong số đoán của người chơi là chính xác và nằm ở đúng vị trí tương ứng.
  - ❑ Dấu '?' được sử dụng để chỉ ra rằng một chữ số trong số đoán của người chơi là chính xác, nhưng nằm ở một vị trí khác so với vị trí tương ứng trong số mà máy tính đã phát sinh. Các vị trí còn lại sẽ không có phản hồi nào. Trường hợp có các số đoán có chữ số trùng nhau thì máy tính vẫn phản hồi ở tất cả các vị trí số đã đoán.
  - ❑ Thông báo chiến thắng/thất bại sau **7 lần đoán tối đa** ?

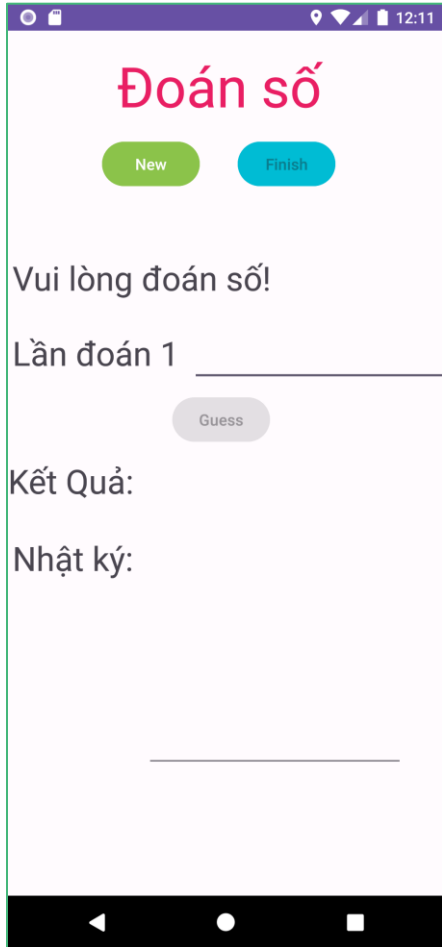
(2) Khi finish (kết thúc)

- Thông báo thất bại, cho biết số cần đoán

(3) Đoán số: cho phép nhập số cần đoán và đưa ra feedback tương ứng

# ASM 2.3

## (1) Default



Đoán số

New Finish

Vui lòng đoán số!

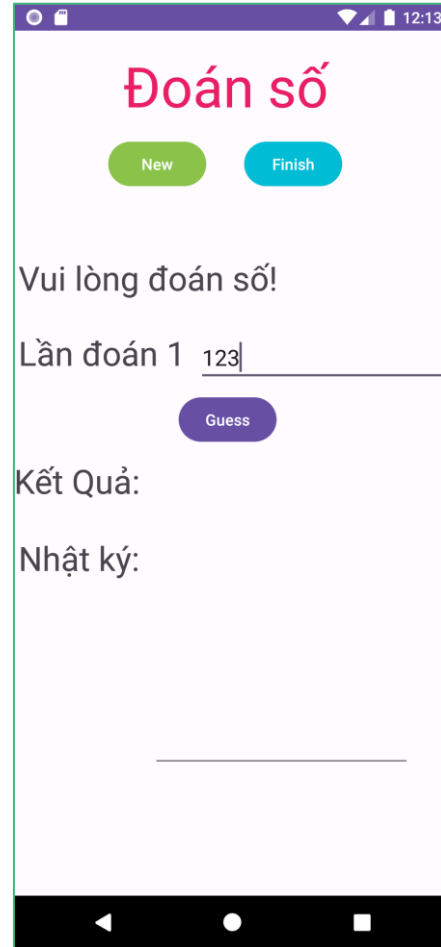
Lần đoán 1

Guess

Kết Quả:

Nhập ký:

## (2) New



Đoán số

New Finish

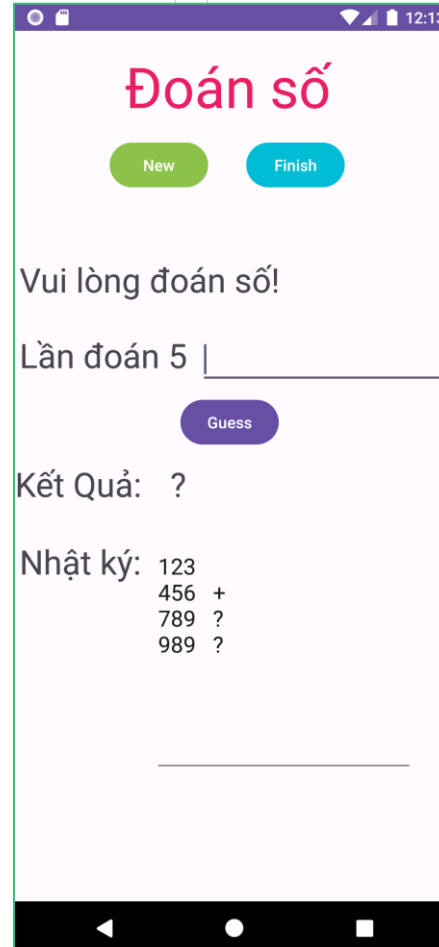
Vui lòng đoán số!

Lần đoán 1

Guess

Kết Quả:

Nhập ký:



Đoán số

New Finish

Vui lòng đoán số!

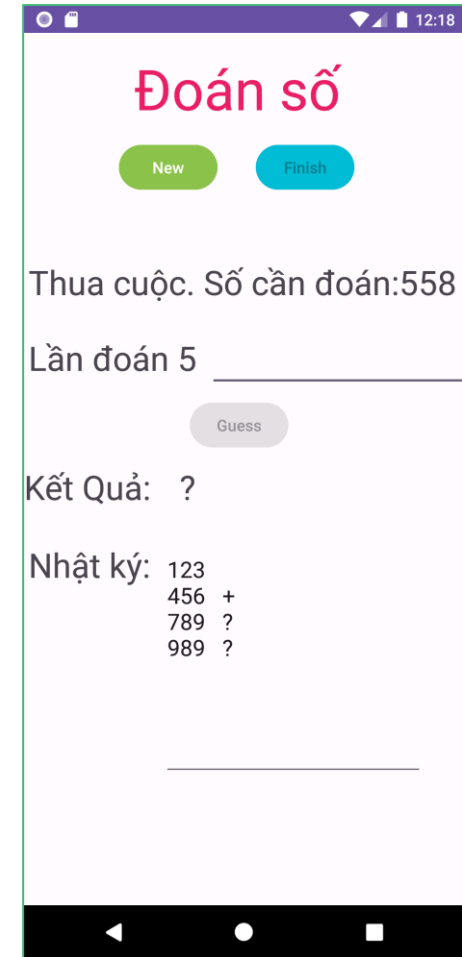
Lần đoán 5

Guess

Kết Quả: ?

Nhập ký: 123  
456 +  
789 ?  
989 ?

## (3) Finish



Đoán số

New Finish

Thua cuộc. Số cần đoán:558

Lần đoán 5

Guess

Kết Quả: ?

Nhập ký: 123  
456 +  
789 ?  
989 ?

# Tổng kết

- Các thành phần cơ bản của Android ?
- TextView là gì ?
- Button là gì ?
- EditText là gì ?
- Lấy giá trị từ EditText mình dùng hàm nào?
- Ánh xạ đến widget mình dùng câu lệnh nào?
- ImageView là gì ?
- 1 số thuộc tính: text, textcolor, gravity, textAllCaps, counterMaxLength, errorEnabled
- Phân biệt match\_parent và wrap\_content
- Toast trong Android Studio ?