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

Nội dung

1. Các thành phần cơ bản của Android

- Activity
- Service
- ContentProvider
- BroadcastReceiver
- Intent

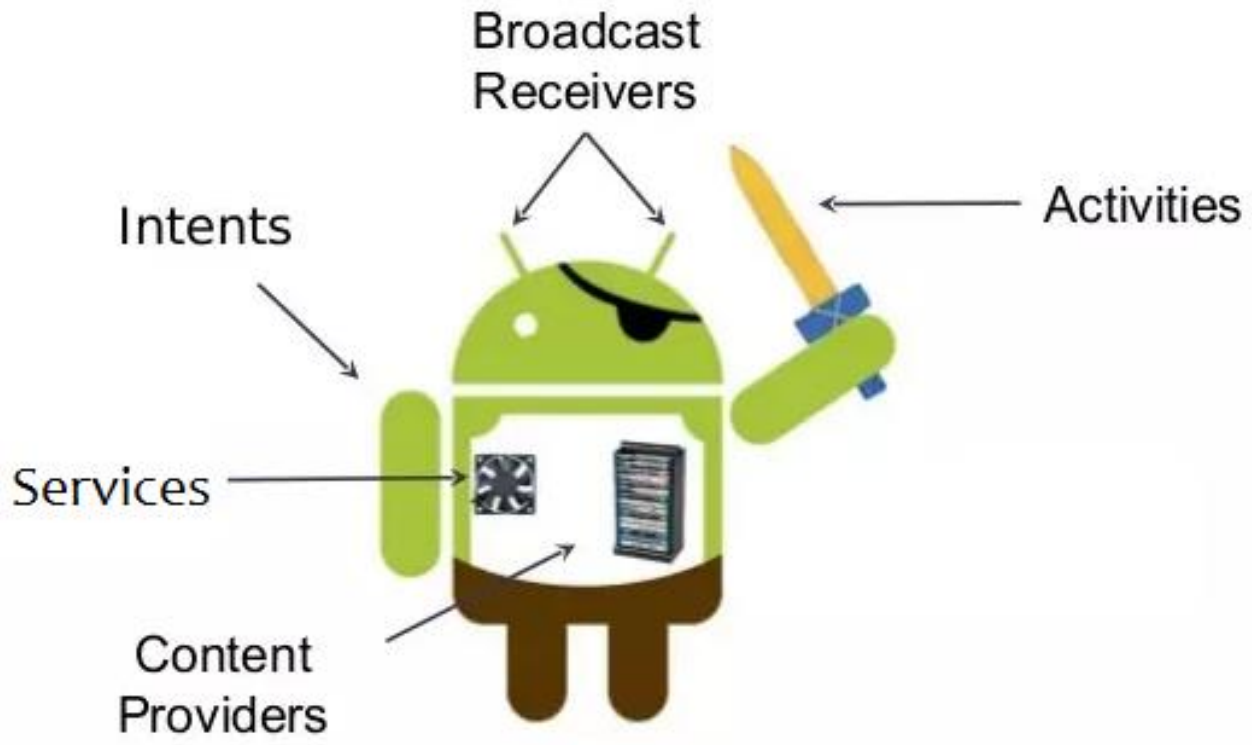
2. Activity và Controls cơ bản

- Activity - Vòng đời của Activity
- Thiết kế giao diện Sử dụng 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:

  lớp **.java:** kế thừa từ lớp cha của nó là Activity

  file **.xml** dùng để thiết kế giao diện người dùng.
- Để sử dụng các activity trong ứng dụng cần định nghĩa trong tệp kê khai (***AndroidMainfest.xml***)

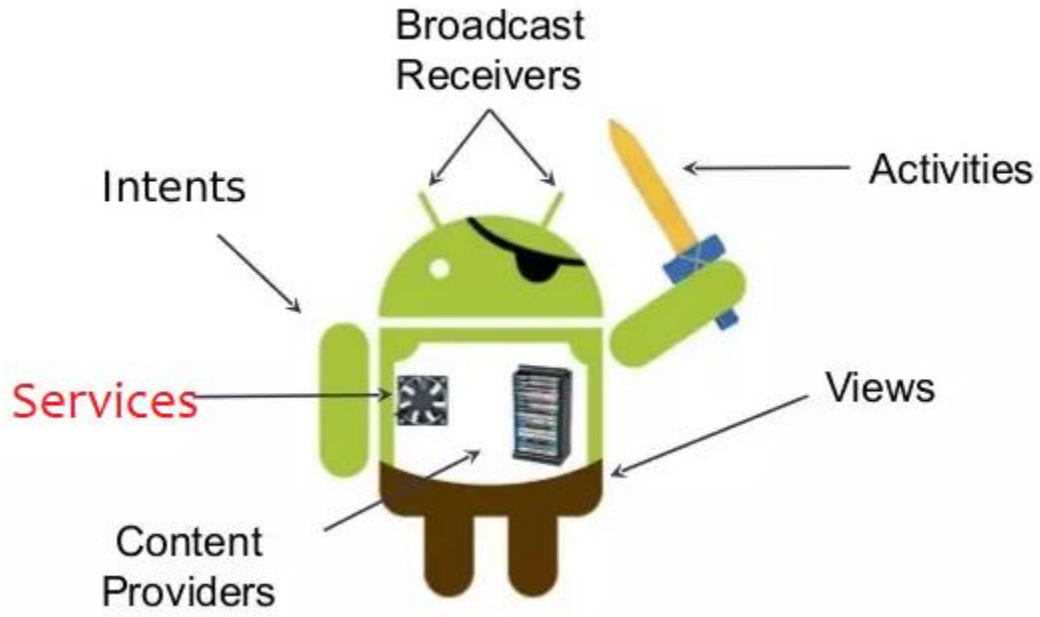
```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Service

- Service: không có phần giao diện, một thành phần chạy ngầm (chế độ nền) không có phần giao diện
  - ❑ Service được tạo và hủy như thế nào ?
- Kế thừa từ lớp cha là **Service**

Ví dụ:

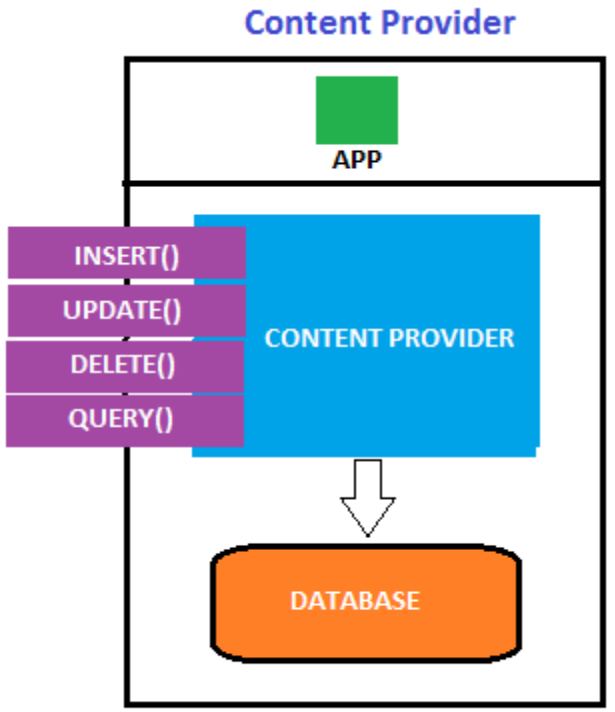
- Trình nghe nhạc
- Down File



Các thành phần cơ bản

Content Provider

- Quản lý lưu trữ dữ liệu trên 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

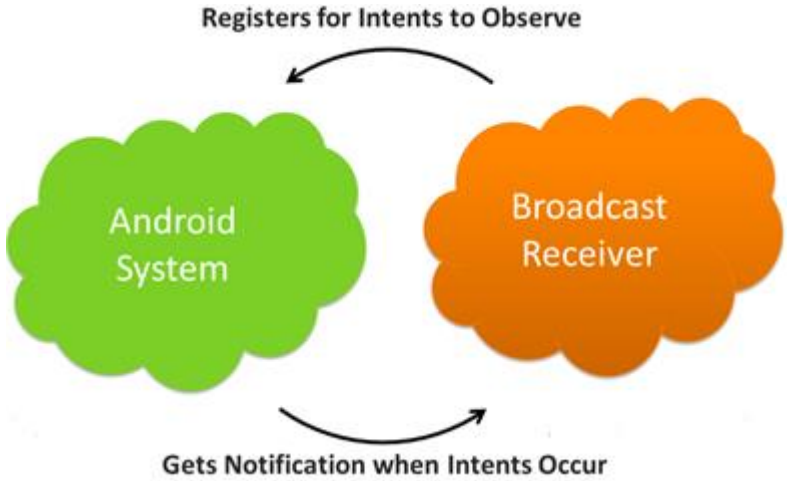
- Là một thành phần giao tiếp với **hệ thống** / **ứng dụng**.
  - Đăng ký **nhận** broadcast trong ứng dụng
  - **Gửi** broadcast từ hệ thống/ ứng dụng
- Sử dụng: kế thừa từ **BroadcastReceiver**
- **Ví dụ**:

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 là một đối tượng message có thể sử dụng để request một hành động từ một vài component trong ứng dụng
- 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)

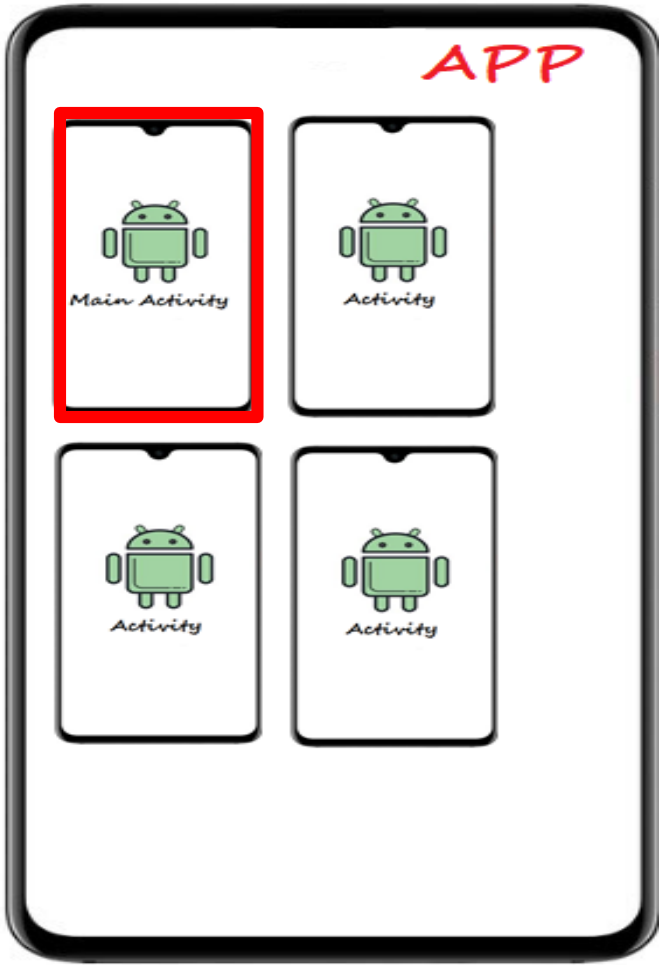
- Cấu trúc của Intent:

Component name	**Action**	**Data**
Category	**Extra**	**Flag**

ACTIVITY VÀ CONTROLS CƠ BẢN

Tổng quan Activity

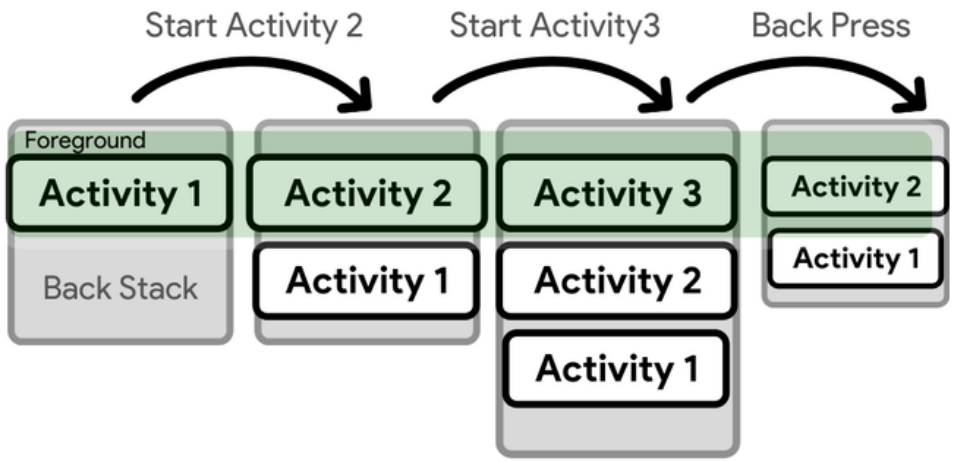
- Activity đại diện cho một màn hình với giao diện người dùng (UI) của một ứng dụng
- Từ nhiều activity trong ứng dụng android, có 1 **activity chính** và đó là màn hình đầu tiên xuất hiện khi khởi chạy ứng dụng. Các thông tin trong tệp kê khai của ứng dụng (***AndroidMainfest***.xml)
- Cần quản lý vòng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



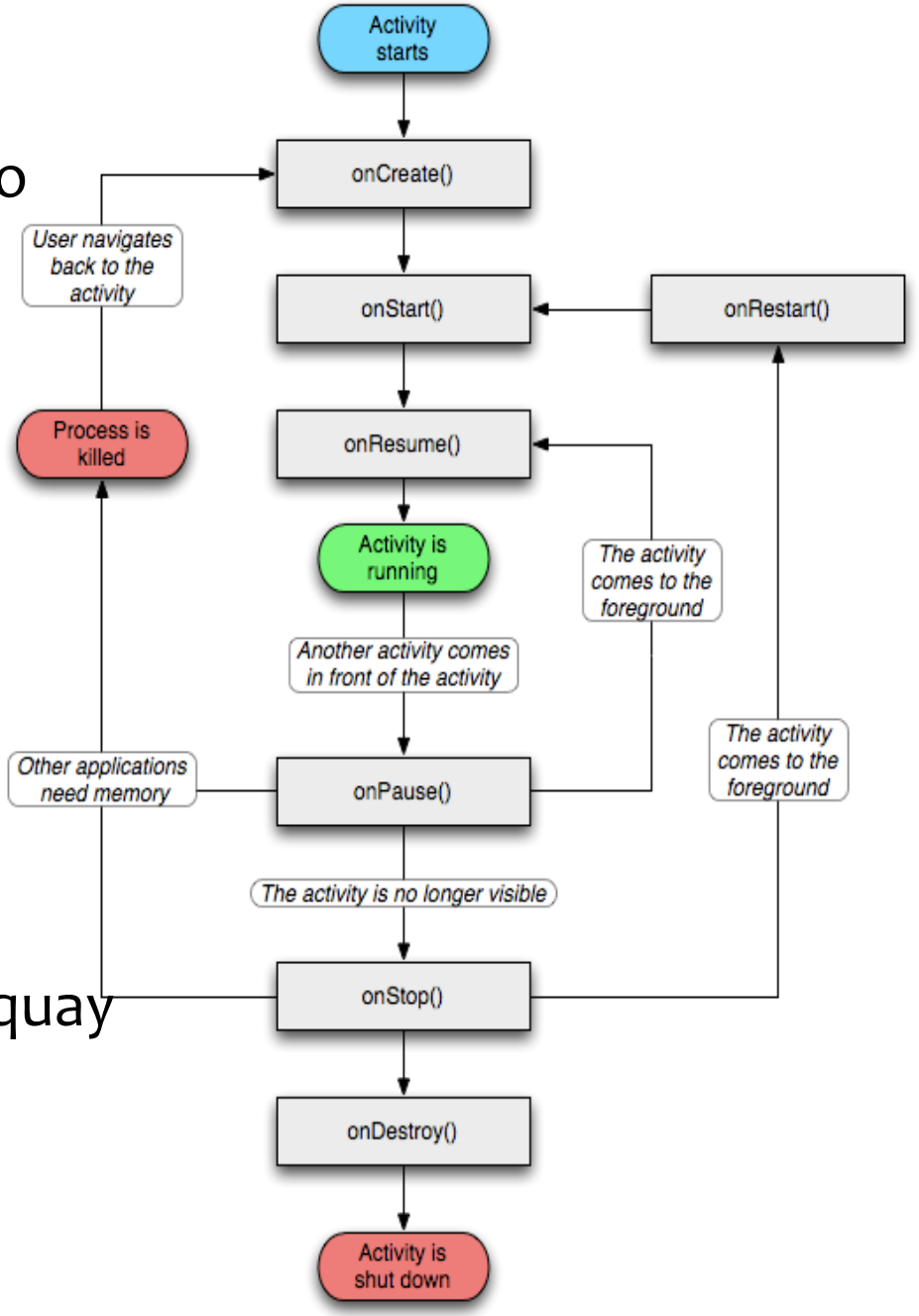
Vòng đời của Activity

- 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



DEMO

1. Xây dựng project "Lab02.Demo" có **MainActivity** và **Acitivity2**

EditText: Với giá trị ban đầu "Activity"

Button Mở hộp thoại: **AlertDialog** (OK)

Button Mở Activity2: Tới **Activity2** (gồm Text và Nút quay lại Activity1)

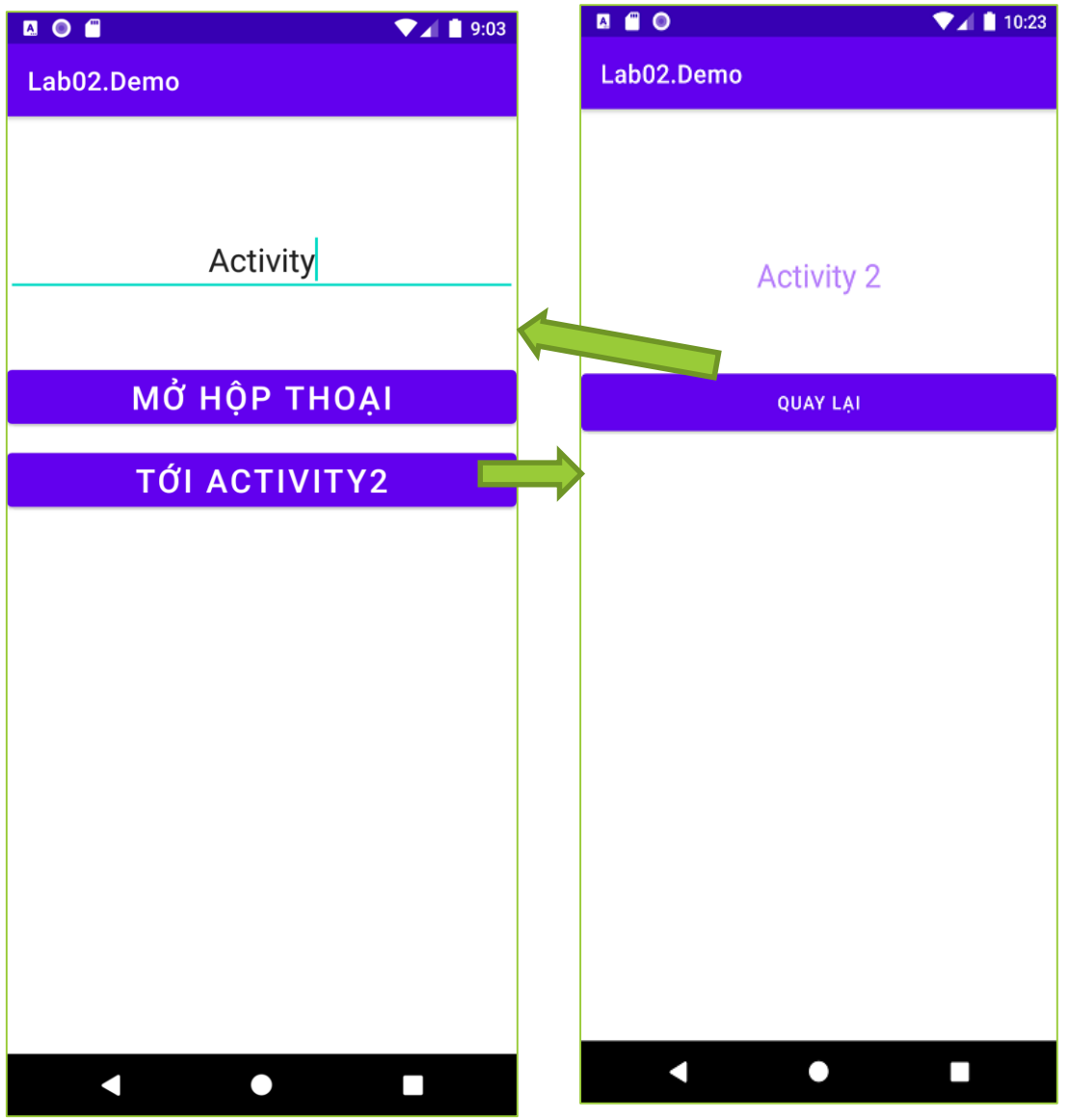
2.Override tất cả các hàm trong vòng đời của **MainActivity** ?

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

Với mỗi hàm **Override, thực hiện việc** nối chuỗi editText với "_TenHamOverride"

VD:

```
editText.setText(editText.getText().toString() + "_" + "onStart");
```



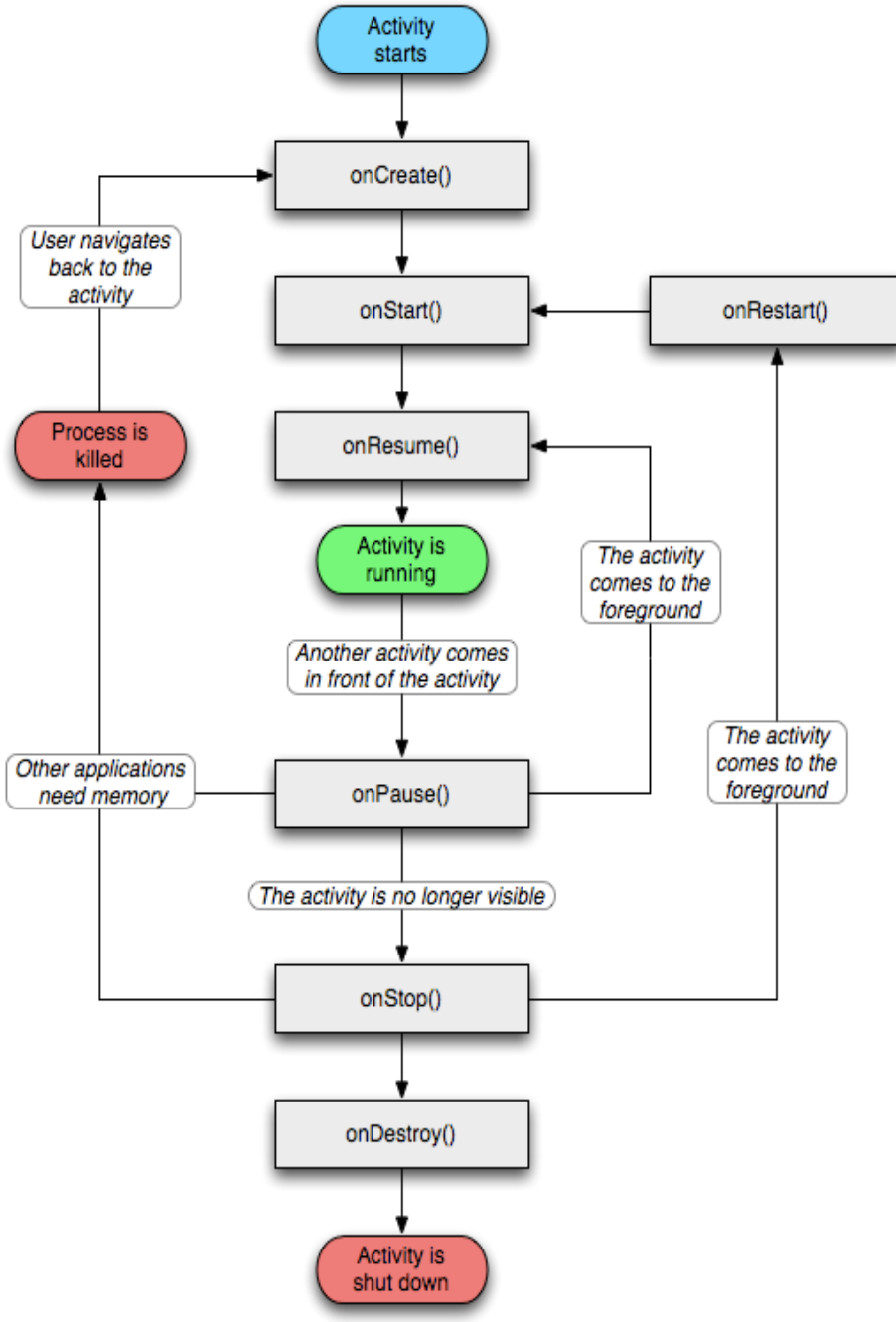
3. Tìm hiểu vòng đời của Activity – Khi bắt đầu launch
   3.1 Khi Mở hộp thoại
   3.2 Khi Tới Activity2 và Quay lại
   3.3 Khi mở 1 Tab ứng dụng khác
   3.4 Khi Quay màn hình ?
4. Đăng kí sự kiện cho Button Gọi Activity2

```
Button btnOpenActivity2 = findViewById(R.id.btnOpenActivity2);
btnOpenActivity2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(MainActivity.this, Activity2.class);
        startActivity(intent);
    }
});
```

5. Ở Second Activity, muốn đóng Activity

```
finish();
```

6. Hiển thị "**Back**" trên Bar ở Activity2?



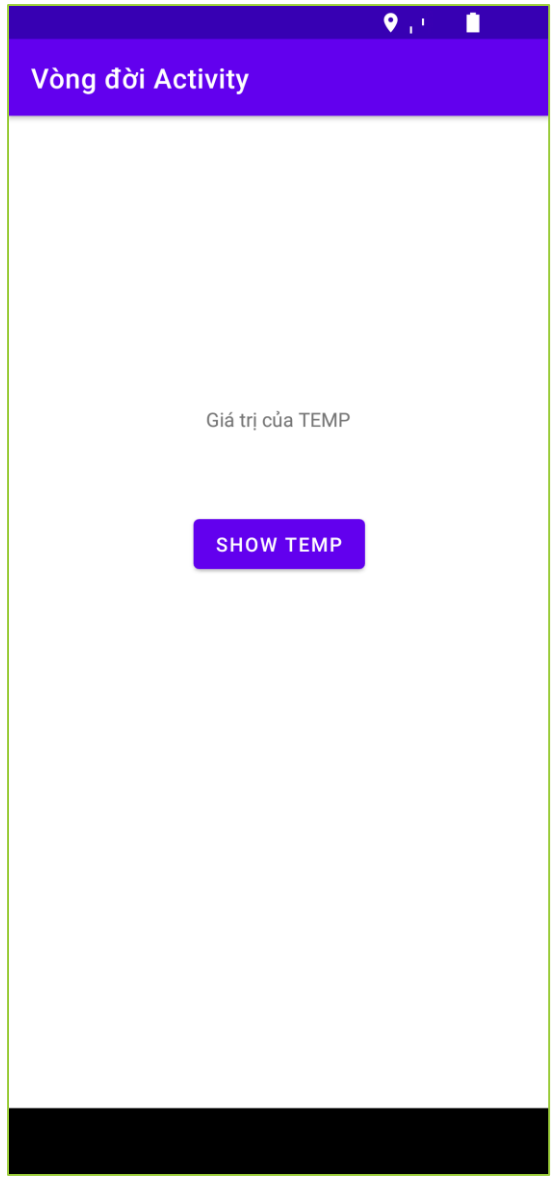
DEMO

❖ Giả sử có biến TEMP= 0 ban đầu ở MainActivity.
Cho biết các thay đổi của biến ở trong MainActivity được implement như sau

- onCreate() { TEMP = TEMP +1 }
- onPause() { TEMP = TEMP * 2 }
- onStart() { TEMP = TEMP + 10 }
- onResume() {TEMP = TEMP - 3 }

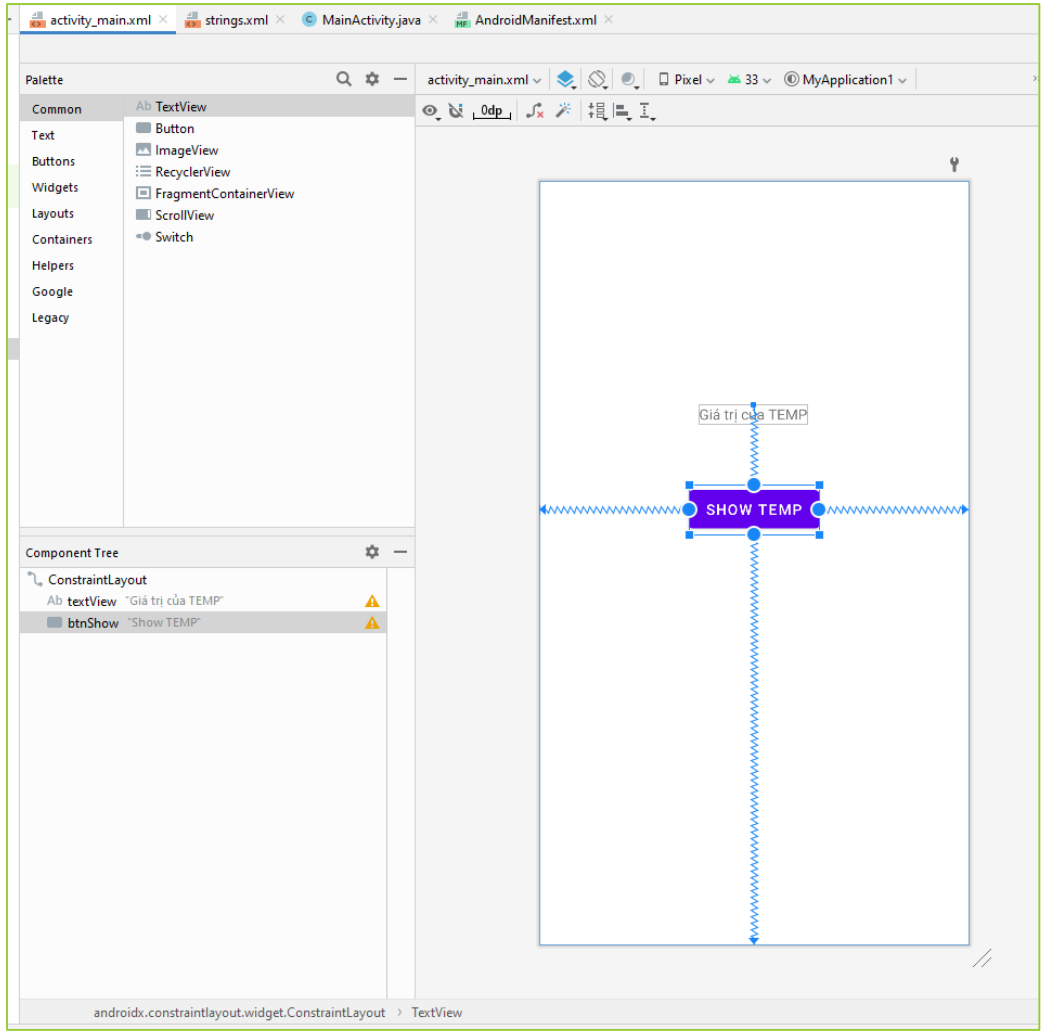
1. Cho biết giá trị của biến TEMP khi chạy MainActivity lần đầu tiên ?
2. Nếu user đi tới 1 Activity tiếp theo (hoặc mở 1 tab khác) và rồi quay lại, giá trị của TEMP là bao nhiêu ?
3. Khi Xoay màn hình thì kết quả TEMP ntn?

Viết chương trình minh họa.



Hướng dẫn

1. Thiết kế giao diện



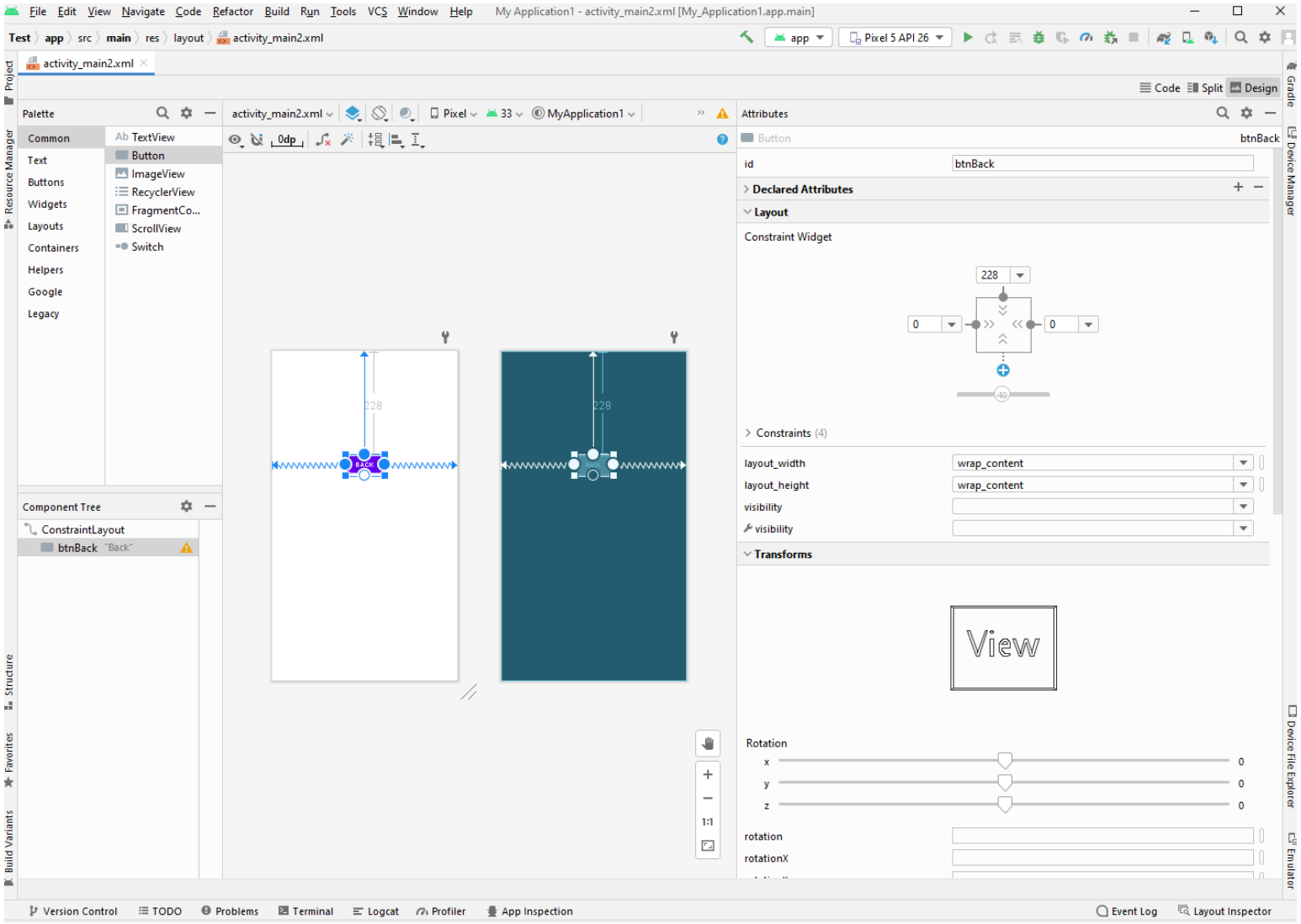
2. Code ở MainActivity.java

```
11  public class MainActivity extends AppCompatActivity {
12      Integer TEMP =0;
13      @Override
14      protected void onCreate(Bundle savedInstanceState) {
15          super.onCreate(savedInstanceState);
16          setContentView(R.layout.activity_main);
17          Button btnShow = findViewById(R.id.btnShow);
18          TextView txtText = findViewById(R.id.textView);
19          btnShow.setOnClickListener(new View.OnClickListener() {
20              @Override
21              public void onClick(View view) {
22                  txtText.setText(String.format("Giá trị của TEMP =%d", TEMP));
23              }
24          });
25          TEMP =  TEMP+1;
26      }
27      @Override
28      protected void onPause()
29      {
30          super.onPause();
31          TEMP = TEMP * 2;
32      }
33      @Override
34      protected void onStart()
35      {
36          super.onStart();
37          TEMP = TEMP + 10;
38      }
39      @Override
40      protected void onResume() {
41          super.onResume();
42          TEMP = TEMP - 3;
43      }
44  }
```

1
2
3
4

Thiết kế giao diện ConstraintLayout

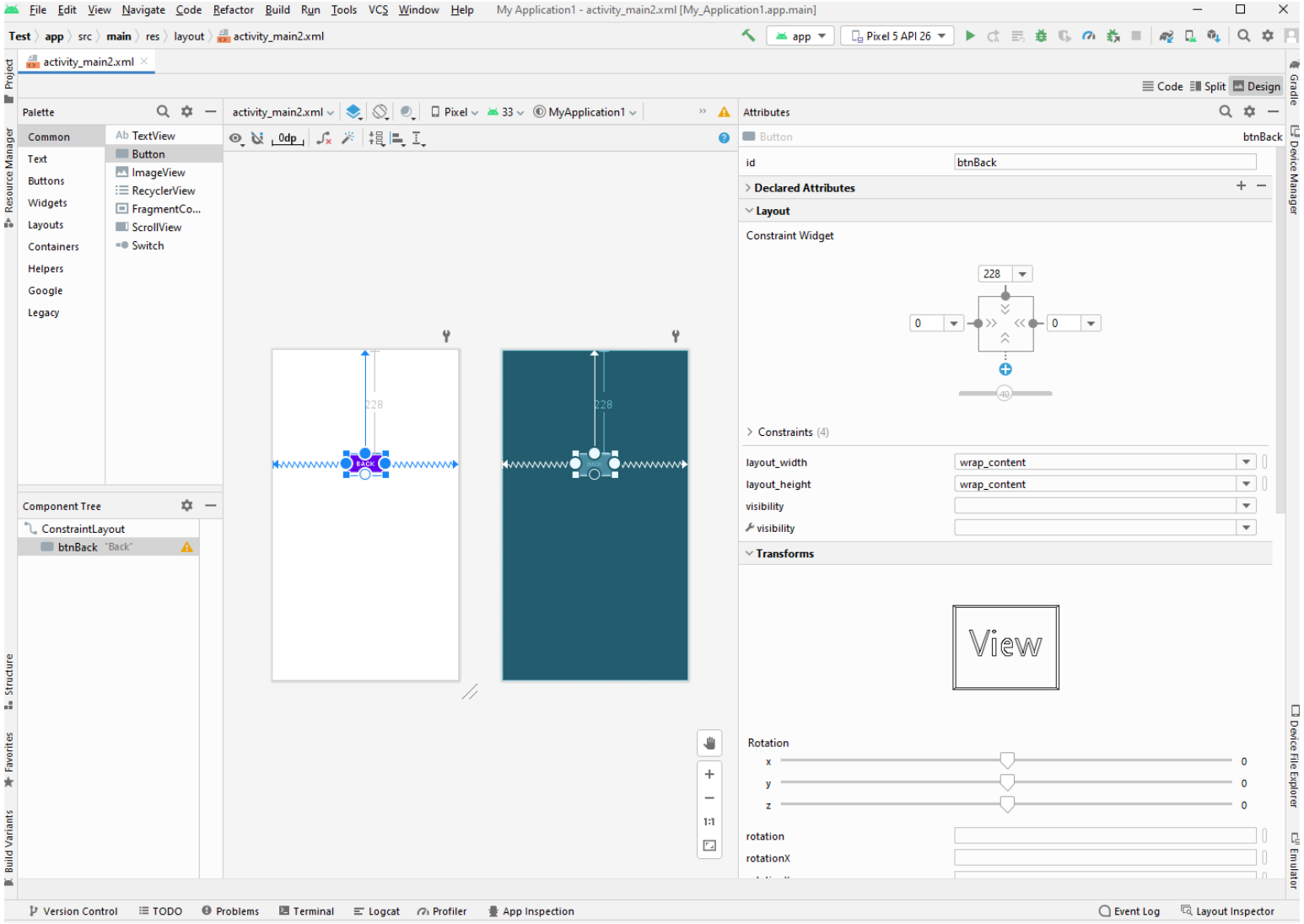
- **ConstraintLayout** (2018) là một layout để xây dựng giao diện nhanh và dễ dùng hơn so với kiểu code giao diện bằng XML truyền thống (RelativeLayout /LinearLayout)
- Mỗi một view phải có ít nhất một điểm neo theo chiều ngang và một điểm neo theo chiều dọc, nếu không đủ các điểm neo tối thiểu, hệ thống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



Thiết kế giao diện trên Activity sử dụng ConstraintLayout

- Một số thuộc tính cơ bản của layout

1. **id** của View
2. **margin** của View
3. Các khoảng cách tới các thành viên bên trong
4. **layout_width** và **layout_height**



Một số controls cơ bản

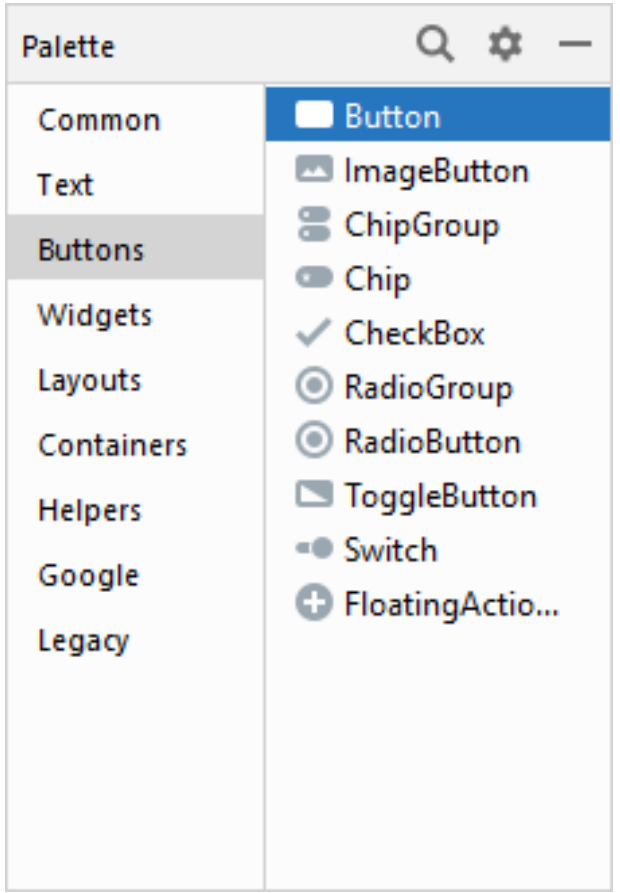
- Các Controls cơ bản: Button, TextView, ImageView, Spinner,…
- Các thuộc tính cơ bản
  - **id** cho control.
  - **layout_width, layout_height** (bắt buộc)
  - gravity
  - Background: màu nền
  - textColor: màu chữ
- Ánh xạ trong Java

```
view???  =  findViewById(R.id.view_id_name)
```

1. Button

- Một Button cho phép người dùng thực hiện một hành động click vào giao diện. Button có thể chứa text, icon, hoặc cả text cả icon.
- Layout của Button

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



2. TextView / EditText

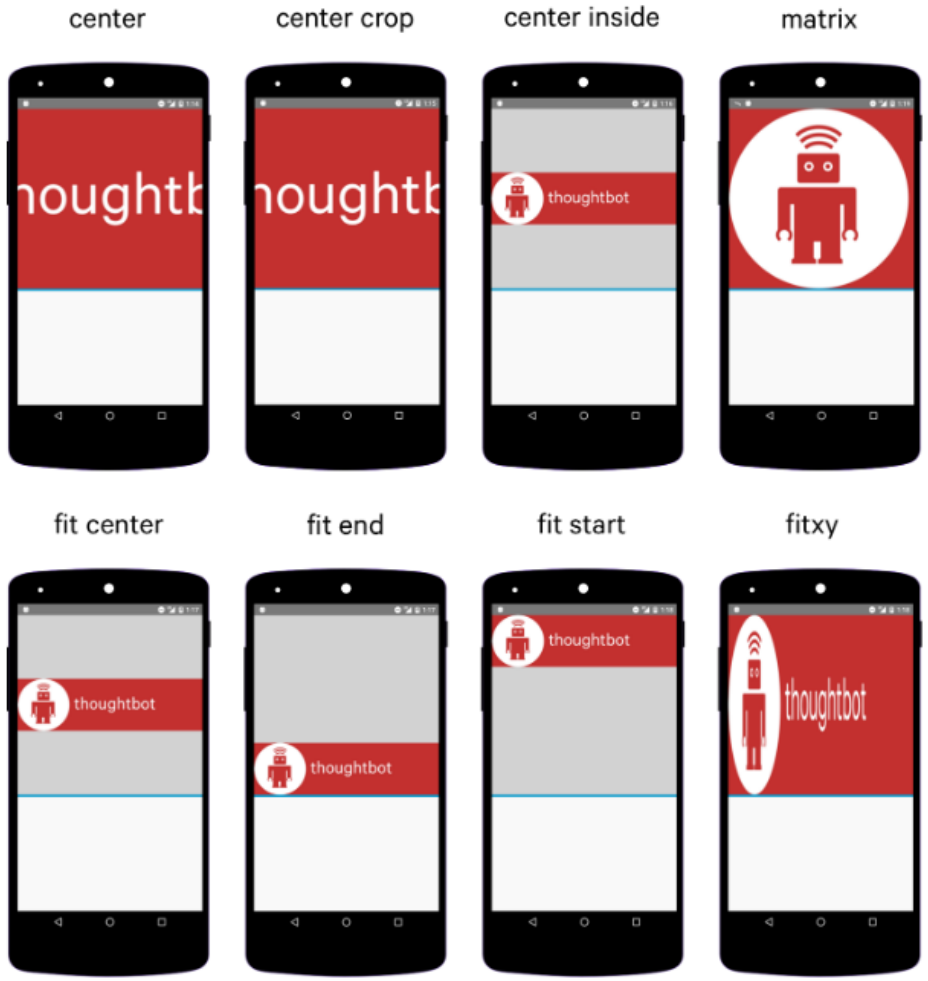
- Một TextView/ EditText dùng để hiển thị thông tin
  - TextView không cho phép người dùng chỉnh sửa.
  - EditText là mở rộng cho phép chỉnh sửa
- Một số thuộc tính của control
  - Để thay đổi màu nền dùng background, thay đổi màu chữ dùng textColor
  - **InputType**
- Phương thức **setText**

3. ImageView

- ImageView dùng để hiển thị ảnh
  - Nguồn ảnh: trên ứng dụng/ thiết bị/ URL
- Layout của ImageView

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/img" />
```

- Một số thuộc tính ở ImageView
  - ScaleTypes (center, fitxy, matrix…)
- Code Java set hình ảnh



4. Spinner

- Spinner: Cho phép chọn giá trị từ 1 tập hợp
- Layout của Spinner

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

- Code Java

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

- **OnItemSelectedListener**

  Xử lý sự kiện onItemSelected

```
public class MainActivity extends AppCompatActivity  {

    List<String> listStr;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.layout);

        listStr = new ArrayList<>();
        listStr.add("Spinner 1");
        listStr.add("Spinner 2");
        listStr.add("Spinner 3");

        Spinner spinner = findViewById(R.id.spinner);
        ArrayAdapter<String> spinnerArrayAdapter = new ArrayAdapter<String>( context: this,
                                                                android.R.layout.simple_spinner_item,
                                                                    listStr);
        spinner.setAdapter(spinnerArrayAdapter);
        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
                String str = listStr.get(i);
                Toast.makeText( context: MainActivity.this, str, Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {

            }
        });
    }
}
```

Xử lý sự kiện trên Android

- Một số Event Listeners
  - onClick(): Tương tự như hàm onTouch() nhưng chỉ có kịch 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.
  - onKey():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 kịch bản chạm.
  - onMenuItemClick(): 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
        }
    }
}
```

Exercise 2.1

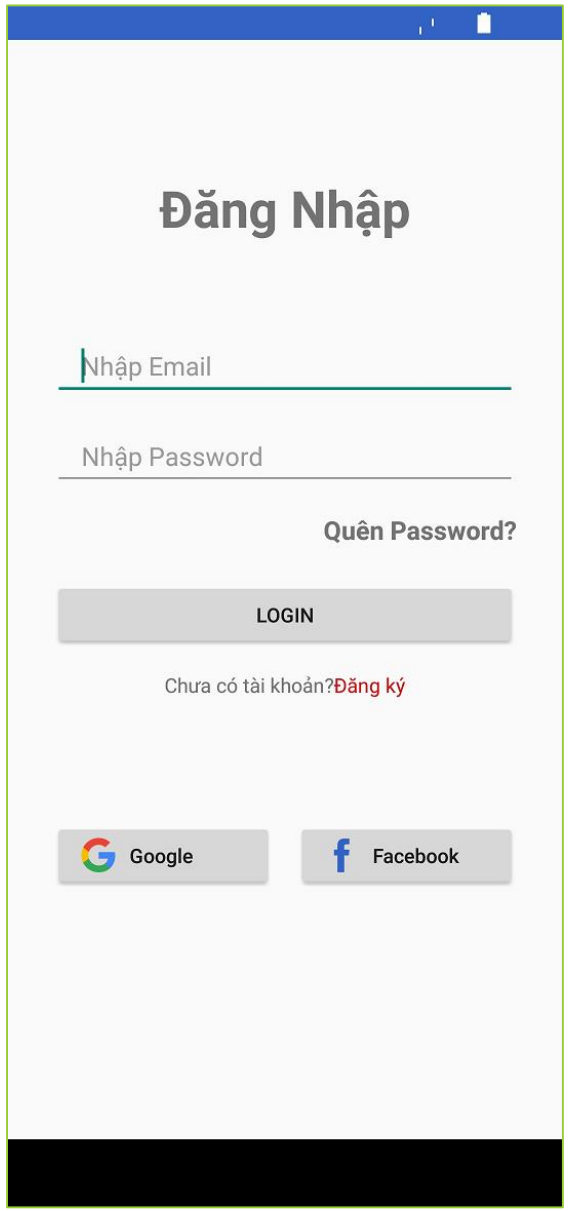
1. Thiết kế giao diện cho Activity "**Đăng Nhập**" và "**Đăng ký**"
2. Click vào **đăng ký** tài khoản sẽ mở ra Activity "Đăng ký"
3. Ở Activity "Đăng Ký"

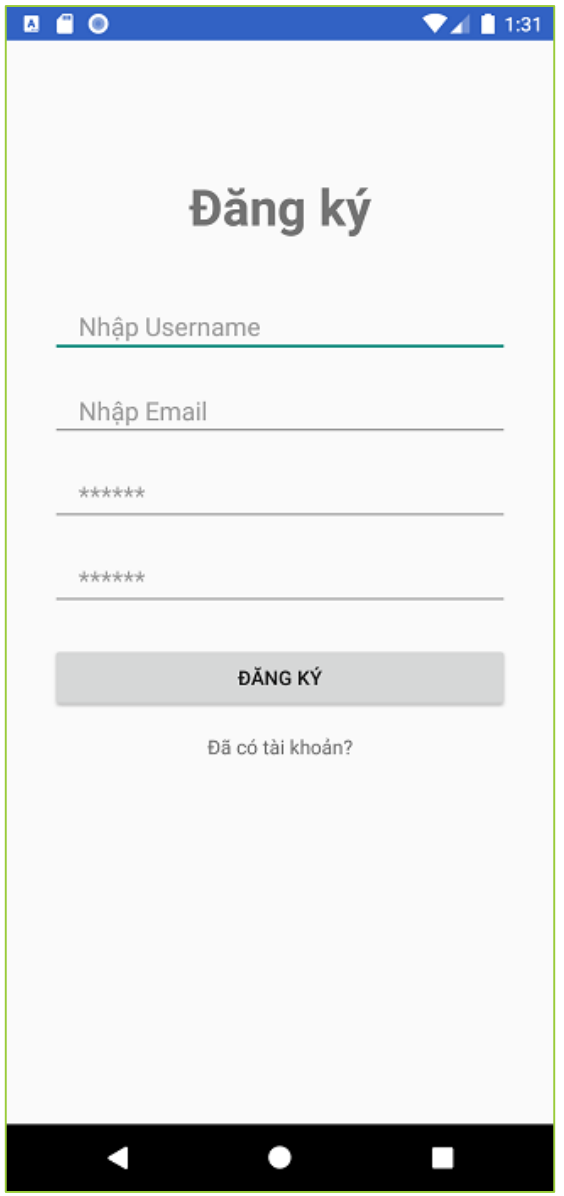
3.1 Đã có tài khoản?: Quay lại trang đăng nhập

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





Exercise 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)