

**Lập trình ứng dụng trên đầu cuối di động**

## **CHƯƠNG 3. LẬP TRÌNH ỨNG DỤNG DI ĐỘNG ANDROID**

# Nội dung

1. Lập trình quản lý vòng đời ứng dụng di động
2. Lập trình giao diện ứng dụng di động
3. Lập trình quản lý dữ liệu
4. Lập trình kết nối mạng
5. Lập trình chức năng cuộc gọi
6. Lập trình chức năng nhắn tin ngắn (SMS)
7. Lập trình xử lý đa phương tiện
8. Lập trình điều khiển các thiết bị phần cứng tích hợp trên đầu cuối di động

# **1. Lập trình quản lý vòng đời ứng dụng di động Android**

# Tiến trình (process)

- Android cung cấp máy ảo Dalvik cho phép tất cả các ứng dụng chạy trong tiến trình riêng của nó. Đồng thời Android có cơ chế quản lý các tiến trình (process) riêng này theo chế độ ưu tiên (priority). Các tiến trình có độ ưu tiên thấp sẽ bị Android giải phóng mà không hề cảnh báo nhằm đảm bảo tài nguyên.
- Thời gian sống của tiến trình ứng dụng không được điều khiển trực tiếp bởi chính nó. Thay vào đó, nó được xác định bởi hệ thống qua sự kết hợp của:
  - Những phần của ứng dụng mà hệ thống biết đang chạy.
  - Những phần đó quan trọng như thế nào đối với người dùng.
  - Bao nhiêu vùng nhớ chiếm lĩnh trong hệ thống.

# Các loại tiến trình(process) của Android

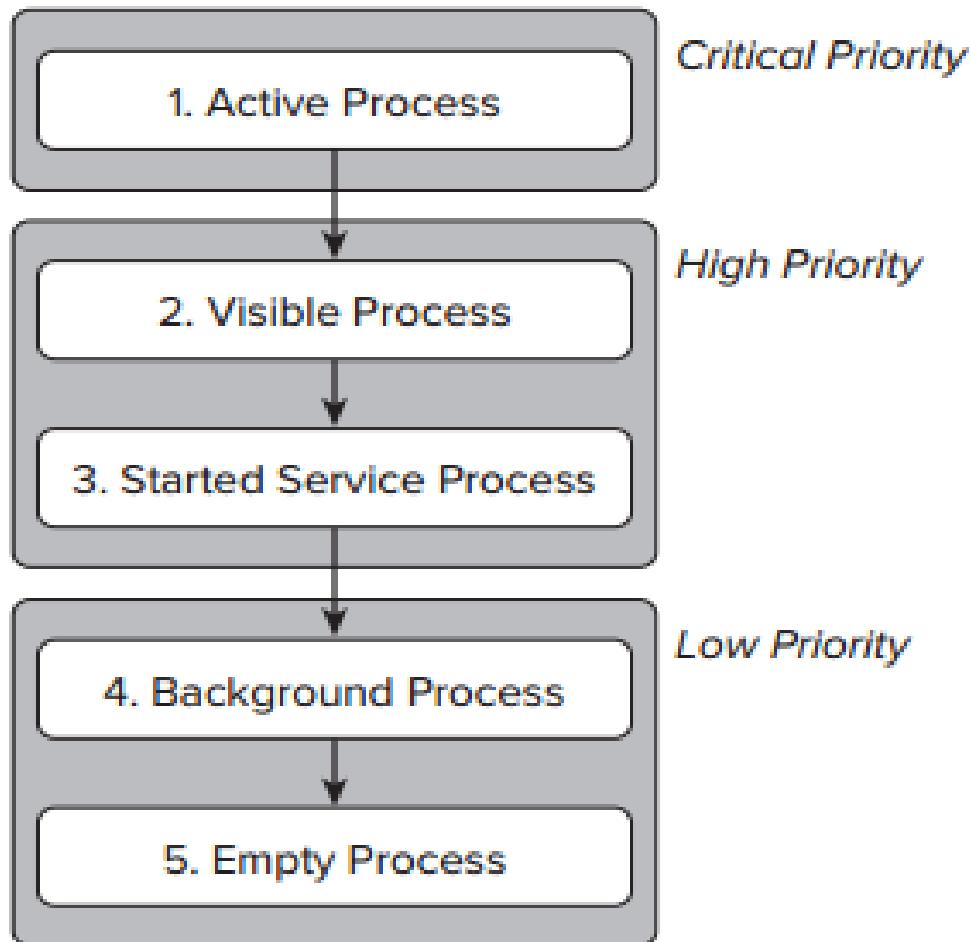
- **Active process:** là process của ứng dụng hiện thời đang được người dùng tương tác. Active process bao gồm:
  - Các Activity đang ở trạng thái hoạt động (active).
  - Broadcast Receivers đang thực thi sự kiện onReceive.
  - Các service đang thực thi các sự kiện onStart, onCreate, onDestroy.
  - Các service được chỉ định để chạy ở chế độ hiển thị trên màn hình (foreground).
- **Visible process:** là process của ứng dụng mà Activity đang hiển thị đối với người dùng, nhưng không chạy ở chế độ foreground cũng như không nhận tương tác từ người dùng. Điều này xảy ra khi Activity không hiển thị toàn màn hình hoặc trong suốt với người dùng (Khi sự kiện OnPause của Activity được gọi).
- **Started Service process:** là service đang chạy. Những service này bị Android giải phóng khi active process hoặc visible process cần sử dụng tài nguyên của hệ thống

# Các loại tiến trình(process) của Android

- **Background process:** là process của ứng dụng mà các activity của nó ko hiển thị với người dùng và không có bất kỳ service nào đang chạy (Khi sự kiện OnStop của Activity được gọi).
- **Empty process:** process không có bất cứ 1 thành phần nào active.
  - Để cải thiện hiệu suất tổng thể hệ thống, Android thường giữ lại một ứng dụng trong bộ nhớ ngay cả khi nó không còn sử dụng, nhằm duy trì vùng nhớ tạm thời để quá trình khởi động lại ứng dụng được nhanh chóng. Các process này sẽ bị giải phóng khi có yêu cầu.

# Thứ tự ưu tiên của các process

Theo chế độ ưu tiên thì khi cần tài nguyên, Android sẽ tự động giải phóng process có độ ưu tiên từ thấp lên cao, trước tiên là các empty process.

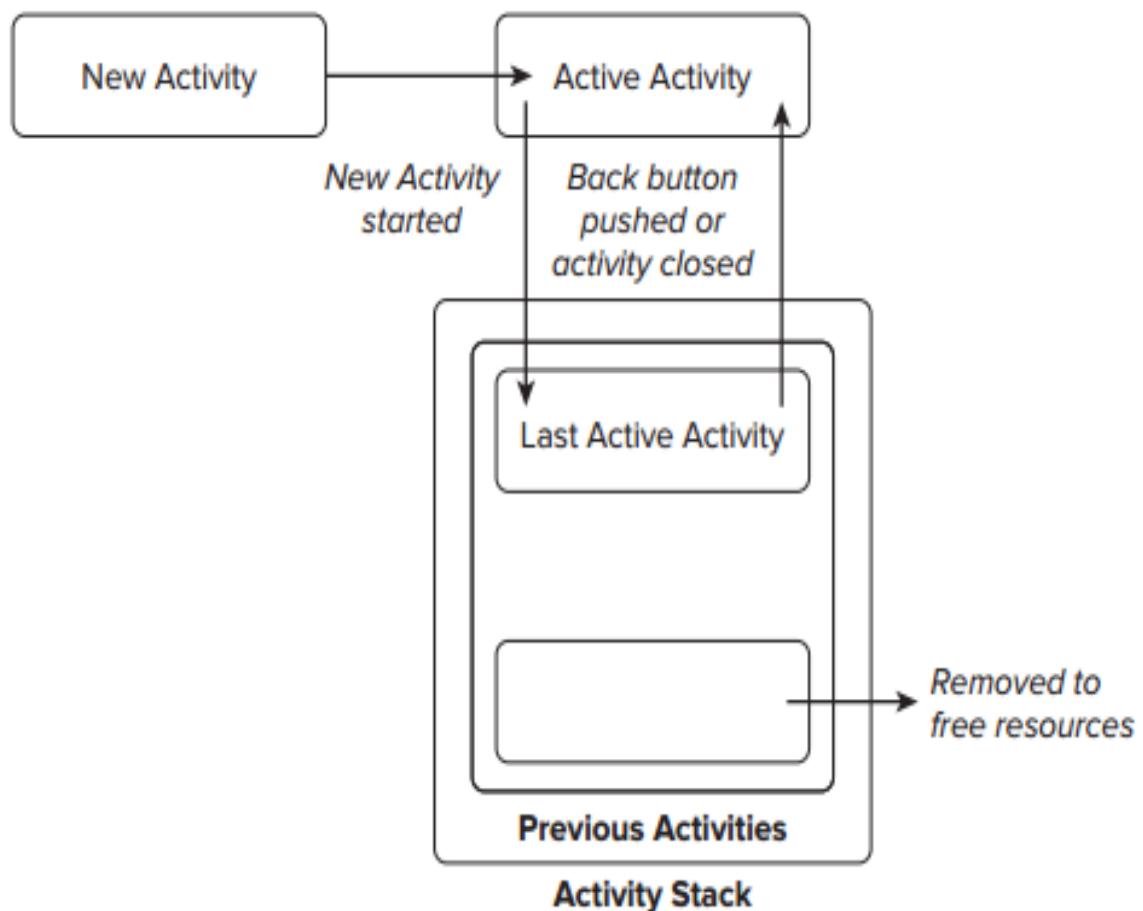


# Chu kỳ sống của ứng dụng Android

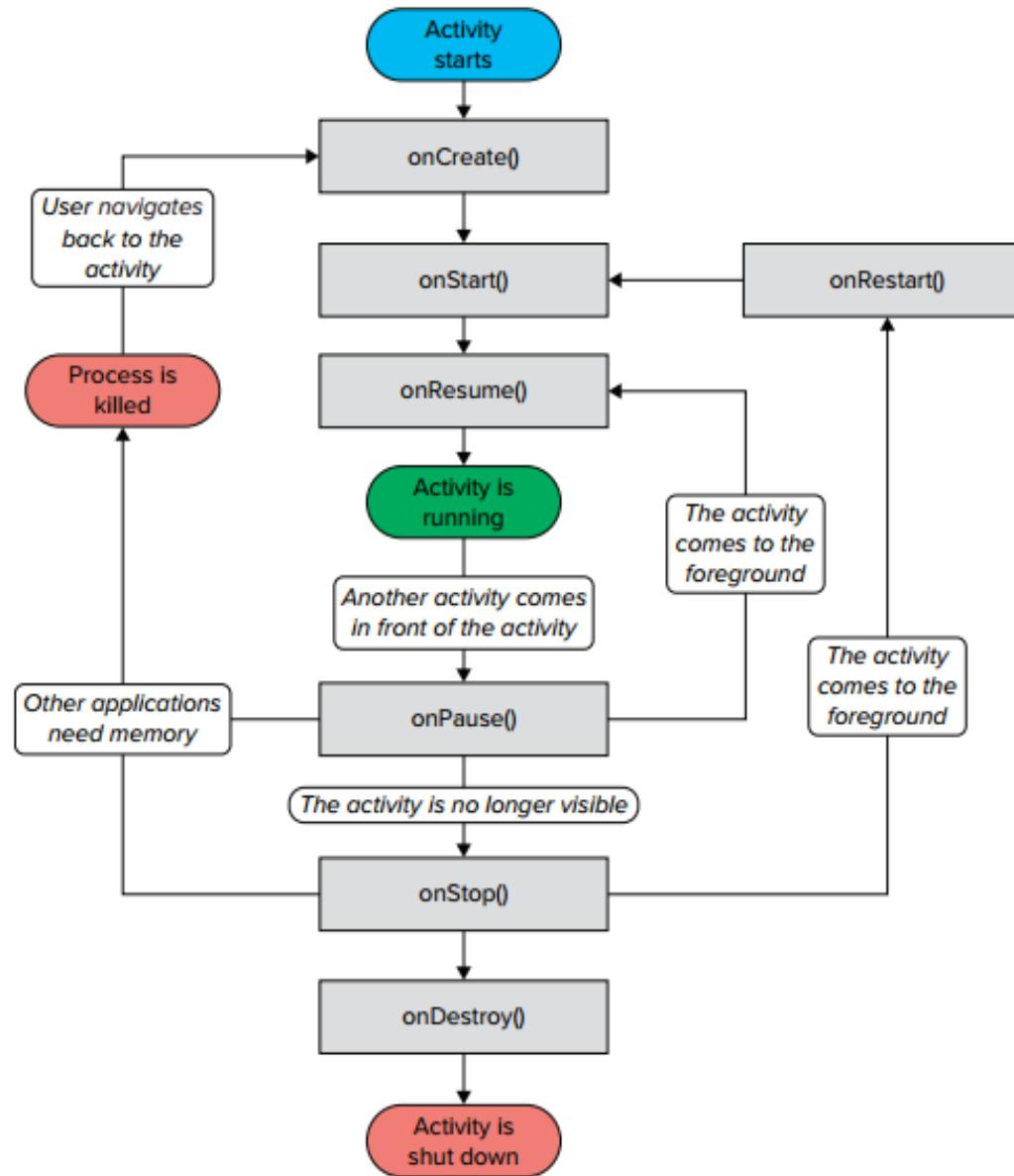
- Mỗi ứng dụng chạy trong tiến trình riêng của mình do máy ảo của Android quản lý đều có chu kỳ sống của nó.
- Chu kỳ sống của ứng dụng Android phụ thuộc vào chu kỳ sống của các thành phần tạo nên ứng dụng Android, mỗi thành phần trong ứng dụng Android đều có một chu kỳ sống riêng.
- Các ứng dụng chỉ được gọi là kết thúc khi tất cả các thành phần trong ứng dụng kết thúc.
- Trong các thành phần tạo nên ứng dụng Android thì một thành phần tối quan trọng của bất kỳ một ứng dụng Android là Activity

# Activity

- Thông thường trong một ứng dụng sẽ có một hoặc nhiều Activity. Mỗi một Activity này sẽ có một vòng đời riêng độc lập hoàn toàn với các Activity khác
- **Activity stack**



# Mô hình vòng đời của Activity



# Lập trình quản lý vòng đời của Activity trong ứng dụng Android

- ✓ Bước 1: Tạo Android project có tên Activity101
- ✓ Bước 2: Tạo một activity mới
- ✓ Bước 3: Quản lý vòng đời Activity qua các hàm sự kiện trong mô hình vòng đời Activity.
- ✓ Bước 4: Bấm F11 để debug ứng dụng trên Emulator

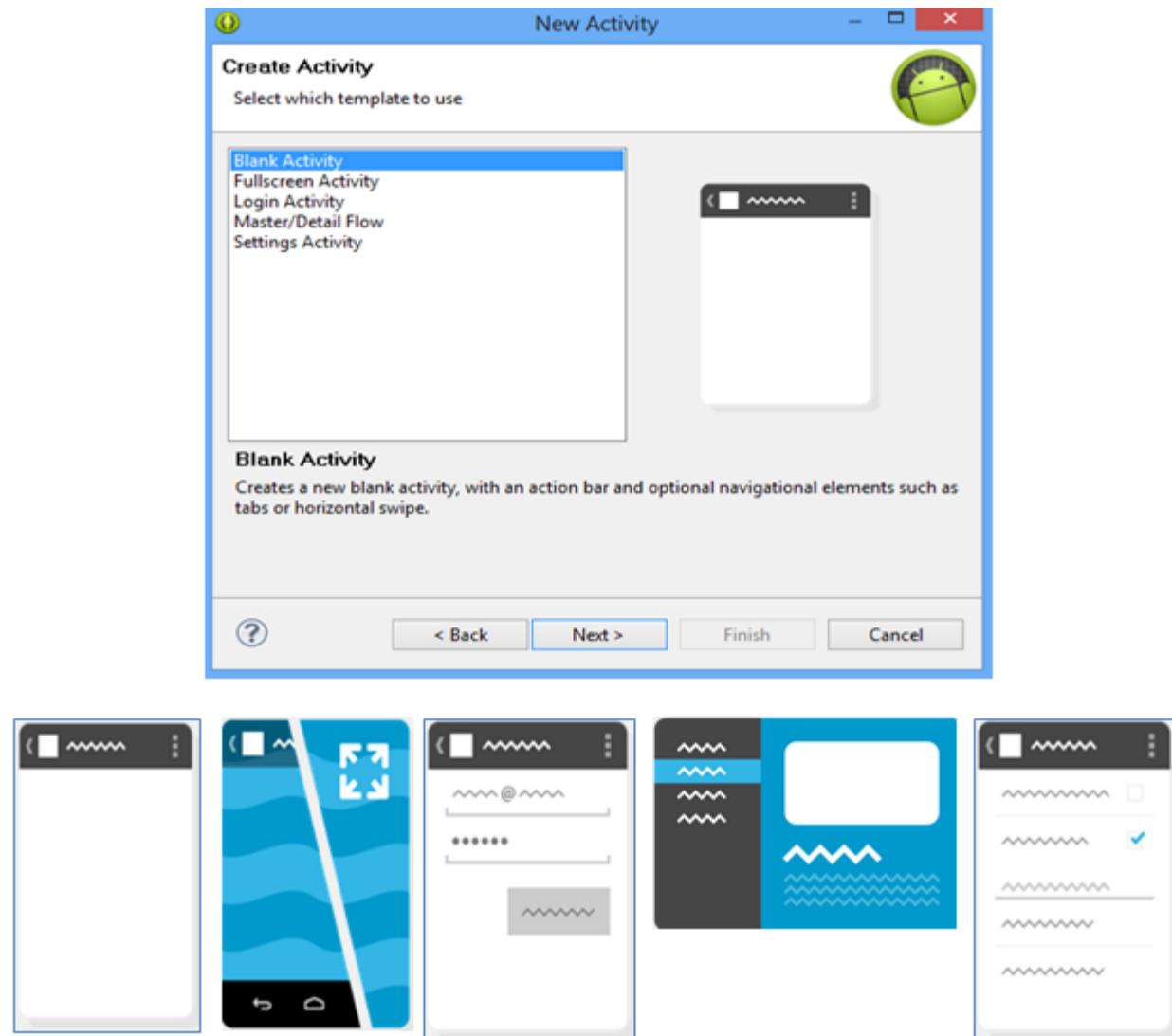
# BTVN 1

- Tạo ứng dụng CheckLifeTimeCycle
  - Start: onCreate, onStart, onResume
  - Bấm nút Home: onPause, onStop
  - Bấm nút Back: onPause, onStop, onDestroyed
  - Có ứng dụng khác kích hoạt: onPause, onStop
  - Xoay màn hình: onP, onS, onD, onC, onStart, onRe
  - Tắt nguồn: onP, onStop
- Chọn menu Source/ Override/Implement methods để override các phương thức onStart, onRestart, onResume, onPause, onStop, onDestroy
- Trong các hàm trên Viết lệnh hiển thị message cho mỗi sự kiện  
`Toast.makeText(this,"onResume", Toast.LENGTH_SHORT).show();`

## **2. Lập trình giao diện ứng dụng di động Android**

# GUI Template /Android

- 5 mẫu giao diện cho Activity:
  - Blank Activity
  - Fullscreen Activity
  - Login Activity
  - Master/Detail Flow
  - Settings Activity



Hình 3.6. Các mẫu giao diện ứng dụng Android

Blank, Fullscreen, Login, Master/Detail Flow, Settings

# ACTIVITY

- Activity dùng để hiển thị màn hình giao diện ứng dụng Android
- Giao diện ứng dụng được xây dựng từ các đối tượng **View** và **ViewGroup**
- Có nhiều kiểu View và ViewGroup. Mỗi một kiểu là một hậu duệ của class View và tất cả các kiểu đó được gọi là các **widget**.
- Tất cả mọi widget đều có chung các thuộc tính cơ bản như là cách trình bày vị trí, background, kích thước, lề...

Thuộc tính	Miêu tả
layout_width	Chiều rộng của View/ ViewGroup
layout_height	Chiều cao của View/ ViewGroup
layout_marginTop	Khoảng cách trống tính từ mép phía trên của View/ ViewGroup
layout_marginBottom	Khoảng cách trống tính từ mép phía dưới của View/ ViewGroup
layout_marginLeft	Khoảng cách trống tính từ mép bên trái của View/ ViewGroup
layout_marginRight	Khoảng cách trống tính từ mép bên phải của View/ ViewGroup
layout_gravity	Qui định vị trí của View đặt bên trong so với ViewGroup
layout_weight	Khoảng cách trống của layout cấp phát cho View
layout_x	Tọa độ x của View/ ViewGroup
layout_y	Tọa độ y của View/ ViewGroup

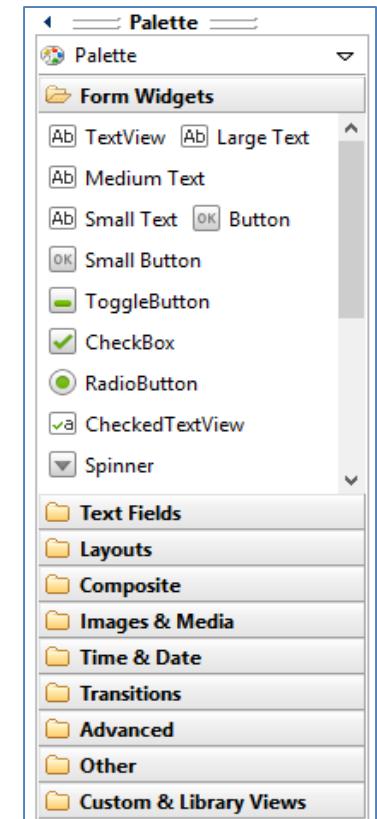
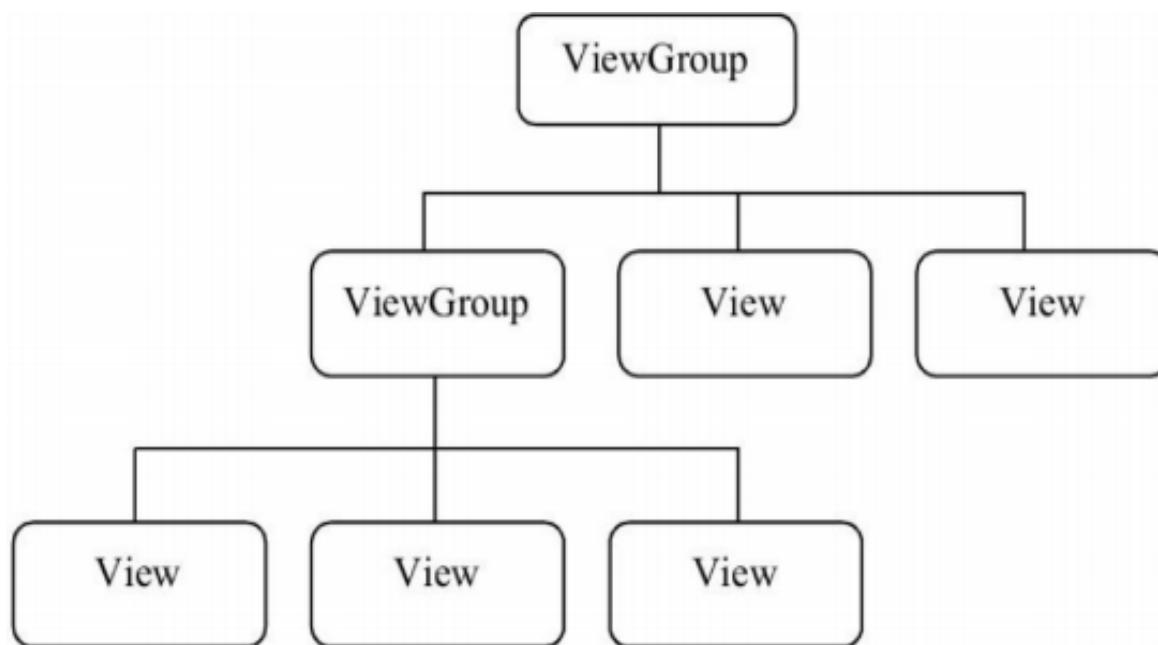
## Các thuộc tính khoảng cách có giá trị với đơn vị đo lường thuộc 1 trong các loại sau:

- **dp (Density-independent pixel)**: đây là đơn vị đo lường được khuyến cáo sử dụng. 1dp tương ứng với 1 pixel trên màn hình phổ biến của thiết bị Android 160 dpi.
- **sp (Scale-independent pixel)**: khuyến cáo sử dụng làm đơn vị của kích cỡ font.
- **pt (Point)**:  $1\text{pt} = 1/72 \text{ inch}$ .
- **px (Pixel)**: Đơn vị đo số lượng điểm ảnh thực tế trên màn hình thiết bị. Đơn vị này không được khuyến cáo sử dụng vì kích cỡ các phần tử trên giao diện ứng dụng có thể không cố định trên các thiết bị có độ phân giải màn hình khác nhau.

- Chú ý rằng chia thiết bị Android thành 4 loại với mật độ điểm ảnh trên màn hình khác nhau:

- ✓ Màn hình có mật độ điểm ảnh thấp (ldpi) -120 dpi.
- ✓ Màn hình có mật độ điểm ảnh trung bình (mdpi) -160 dpi.
- ✓ Màn hình có mật độ điểm ảnh cao (hdpi) -240 dpi.
- ✓ Màn hình có mật độ điểm ảnh cao nhất (xhdpi) -320 dpi.

# Cấu trúc một giao diện ứng dụng Android



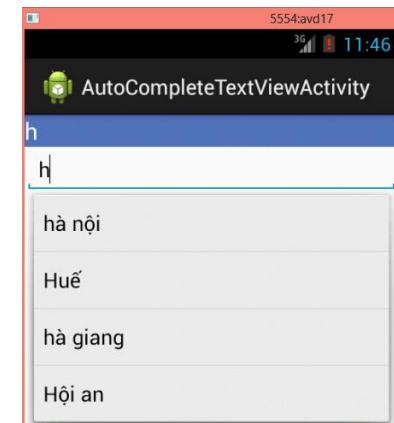
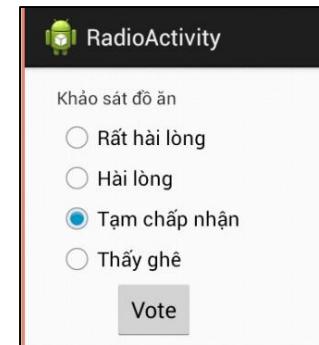
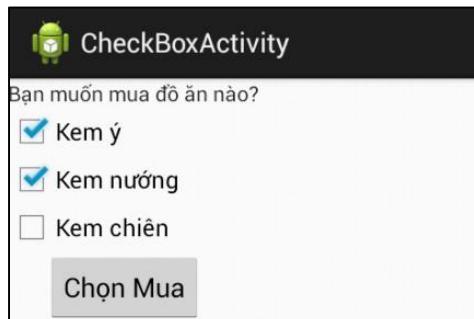
# View

- Các view trong Android được định nghĩa sẵn trong lớp thư viện **android.view.View**.
- Các view này được chia thành 3 loại chính sau :
  - Các view cơ bản
  - Các picker view
  - Các view hiển thị danh sách

# View

-- Các view cơ bản --

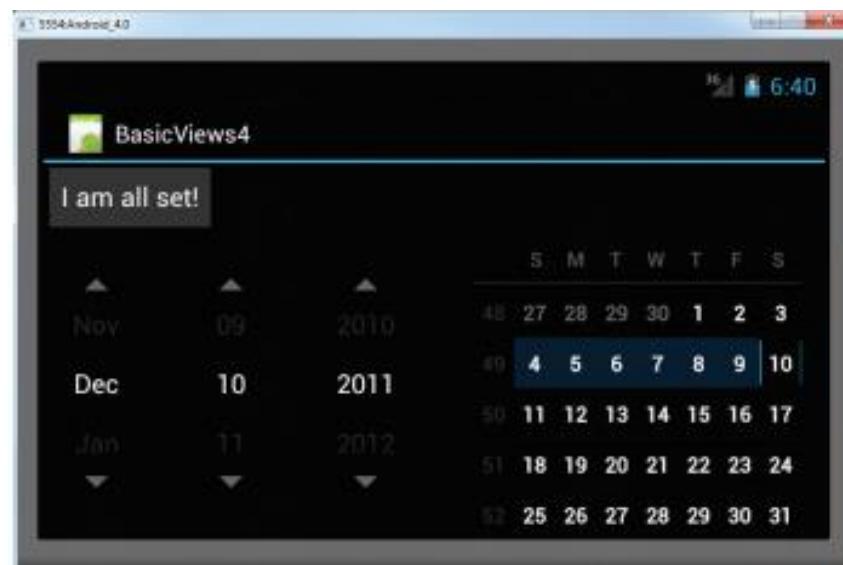
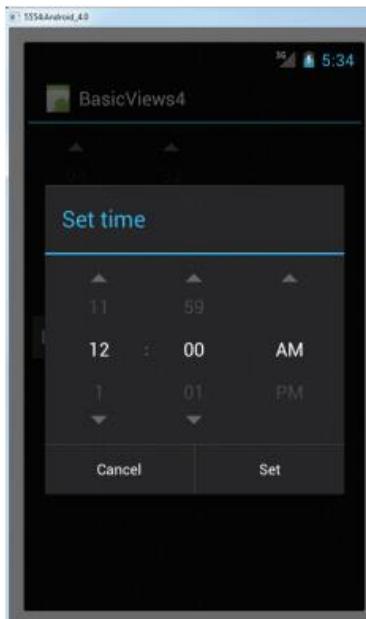
- Các view cơ bản trong Android nhằm hiển thị thông tin văn bản hoặc cho phép lựa chọn thông tin
- Bao gồm : TextView, EditText, Button, ImageButton, CheckBox, ToggleButton, RadioButton, RadioGroup, Progressbar, AutoCompleteTextView



# View

-- Các picker view --

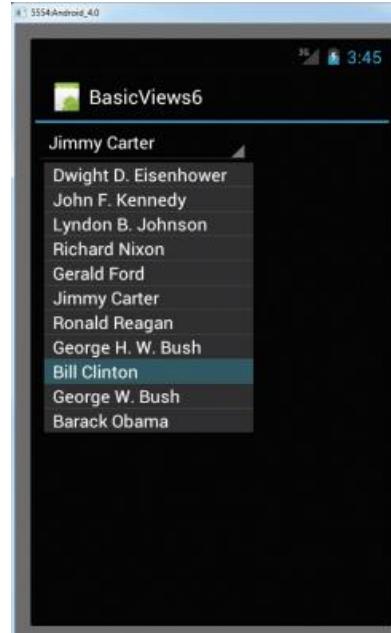
- Picker được chia làm 2 loại : TimePicker, DatePicker. Việc sử dụng 2 loại picker này cho phép người dùng chọn thời gian, ngày tháng năm trên giao diện ứng dụng Android.



# View

-- Các view hiển thị danh sách--

- Nhằm hiển thị danh sách mục thông tin.
- Có 2 loại view hiển thị danh sách: ListView, SpinnerView.



# Xử lý sự kiện khi người dùng tương tác với các View

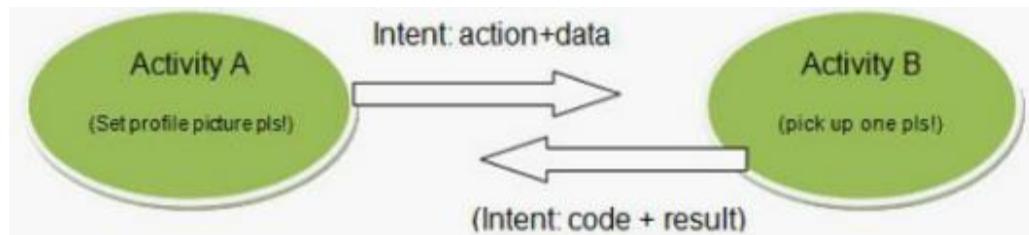
- Các view nhận được các tương tác từ người dùng sẽ xử lý các sự kiện tương ứng.
- Các sự kiện này được Android Framework quản lý bởi 1 hàng đợi (queue) hoạt động theo cơ chế FIFO (First In First Out).
- Các bước cần làm để xử lý sự kiện cho các đối tượng giao diện:
  - Đăng ký lắng nghe sự kiện (Event Listeners Registration).
  - Khai báo đối tượng nhận thông báo khi có sự kiện xuất hiện (Event Listeners).
  - Xử lý sự kiện (Event Handlers)

# ViewGroup

- ViewGroup thực ra chính là View hay nói đúng hơn thì ViewGroup chính là các widget layout được dùng để bố trí các đối tượng khác trong một màn hình giao diện.
- Các ViewGroup trong Android được định nghĩa sẵn trong lớp thư viện **android.view.ViewGroup**.
- Các ViewGroup này được chia thành các loại chính sau :
  - LinearLayout
  - AbsoluteLayout
  - TableLayout
  - RelativeLayout
  - FrameLayout
  - ScrollView
- Có thể kết hợp các loại ViewGroup khác nhau để tạo ra giao diện ứng dụng Android.

## INTENT

- Ứng dụng Android thường bao gồm nhiều Activity, mỗi Activity hoạt động độc lập với nhau và thực hiện những công việc khác nhau.
- Intent giống như người đưa thư, giúp Activity này có thể triệu gọi cũng như truyền dữ liệu cần thiết tới một Activity khác.

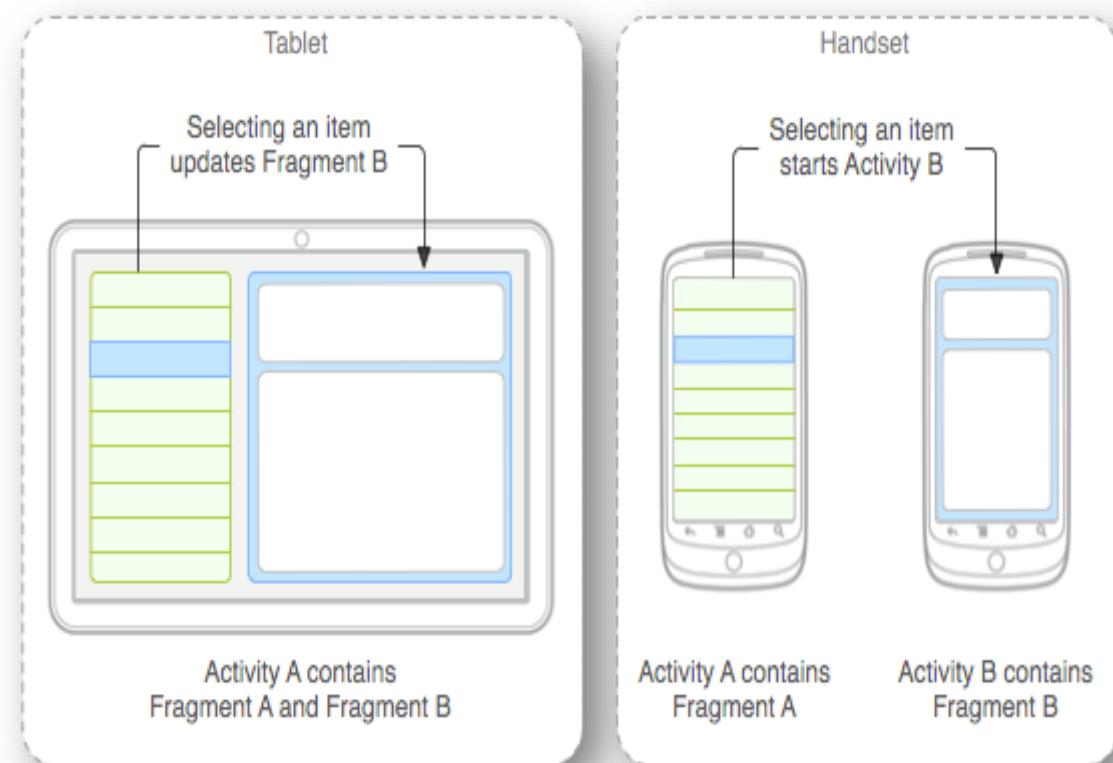


Hình 3.28. Dùng Intent để truyền dữ liệu giữa 2 Activity

# FRAGMENT

Fragment là một thành phần giao diện được nhúng vào Activity cho phép thiết kế Activity linh động.

Fragment có thể coi như **sub-activity**.



*Hình 3.29. Ví dụ kết hợp các Fragment trên một Activity  
Giao diện ứng dụng trên máy tính bảng (trái) và điện thoại di động (phải)*

❖ Phân loại Fragment:

- **ListFragment** : dùng để hiển thị danh sách mục tin.
- **DialogFragment** : fragment hiển thị dưới dạng dialog động phía trên Activity.
- **PreferenceFragment** : dùng để hiển thị phân cấp các đối tượng.

## ❖ Đặc điểm của Fragment:

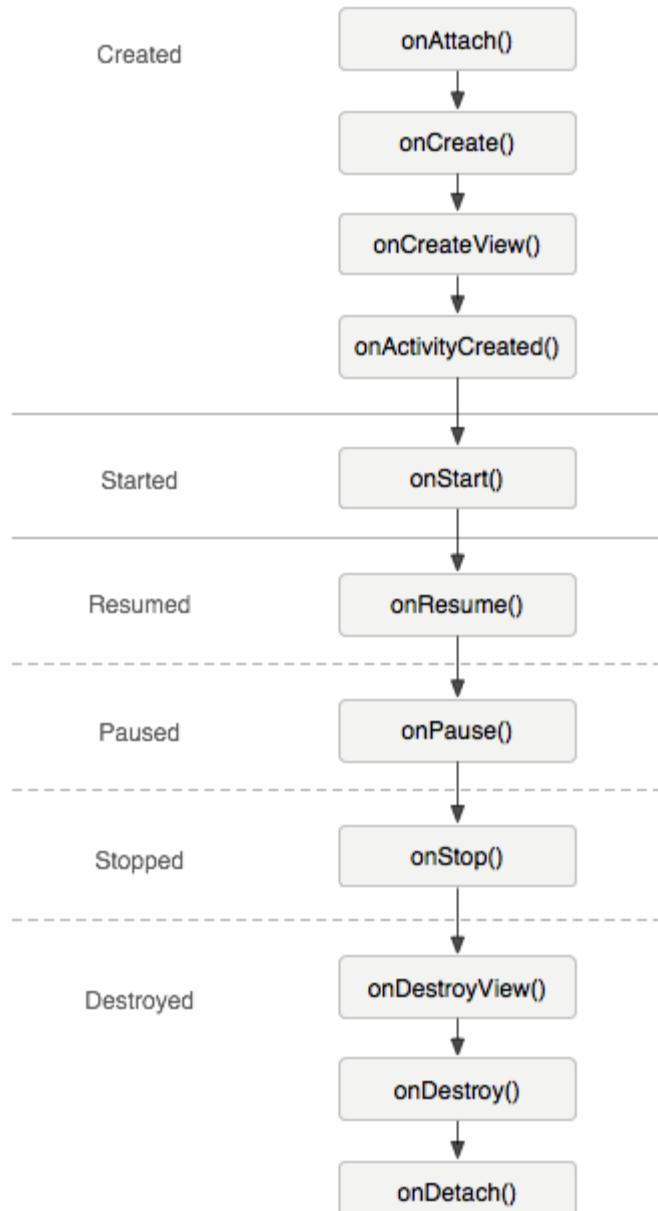
- Fragment có layout và vòng đời hoạt động riêng. Vòng đời hoạt động của Fragment phụ thuộc vào vòng đời của Activity chứa nó.
- Fragment có thể được thêm vào hoặc gỡ bỏ khỏi Activity trong khi Activity đang hoạt động.
- Một Activity có thể được hình thành từ nhiều Fragment, một Fragment có thể được sử dụng bởi nhiều Activity.
- Fragment được đưa vào từ Android API 11.

## ❖ Cách tạo và sử dụng Fragment:

- Tạo Fragment bằng cách tạo 1 lớp kế thừa từ class Fragment cơ sở.
- Tạo giao diện cho Fragment dưới dạng mã XML
- Nhúng Fragment vào Activity bằng cách khai báo thành phần <fragment> trong file giao diện của Activity.

# Vòng đời fragment

## Activity State      Fragment Callbacks



# SV chia nhóm báo cáo

Trình bày cách thức sử dụng + ví dụ về:

1. Layout.
2. Fragment, TabSelection.
3. Intent
4. TextView, EditText.
5. Button, ImageButton.
6. CheckBox, ToggleButton.
7. RadioButton, RadioGroup.
8. TimePicker, DatePicker.
9. ScrollView, ImageView.
10. ListView, Spinner.
11. AutoCompleteTextView, Gallery.
12. SlideDrawer, Menu.
13. Xử lý sự kiện khi người dùng tương tác với các View

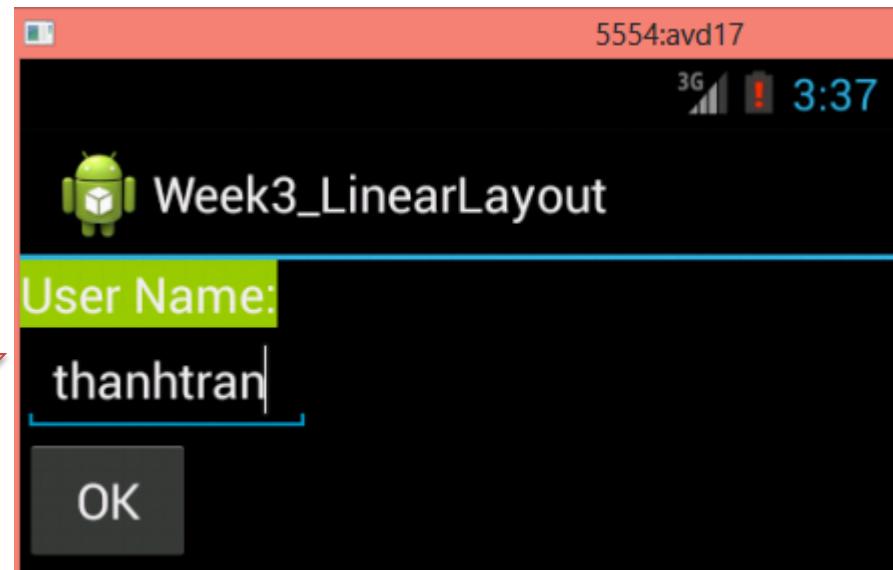
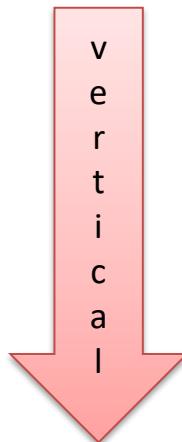
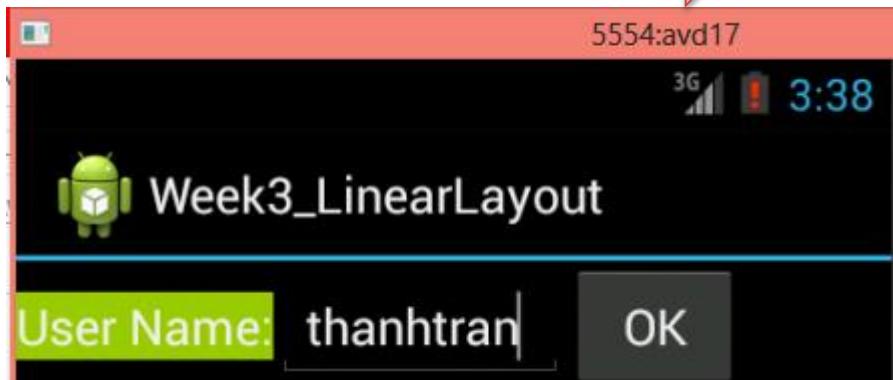
# **Bài tập 1: Layout**

# ViewGroup

- ✓ LinearLayout
- ✓ AbsoluteLayout
- ✓ TableLayout
- ✓ RelativeLayout
- ✓ FrameLayout

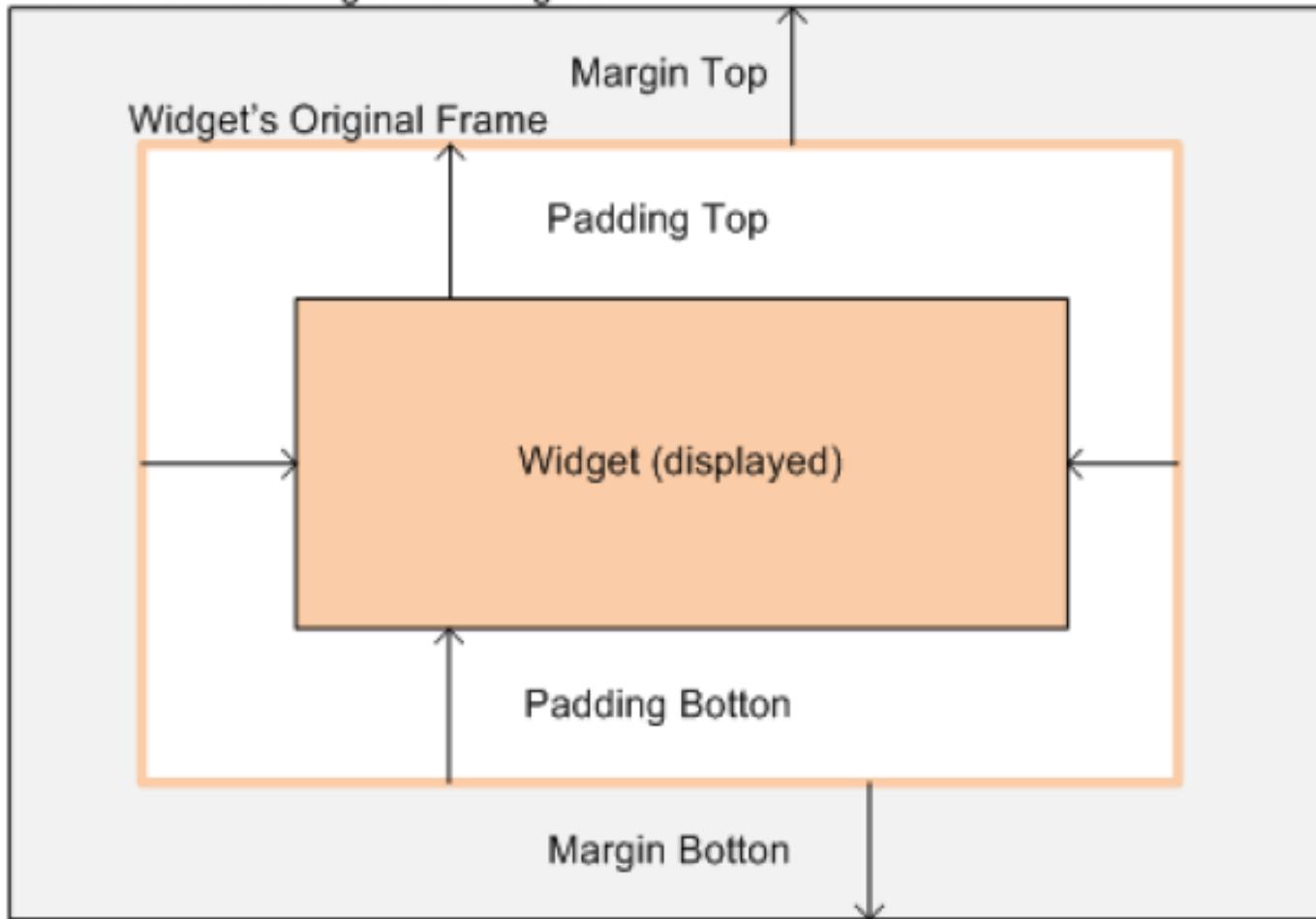
# LinearLayout

Horizontal



# LinearLayout

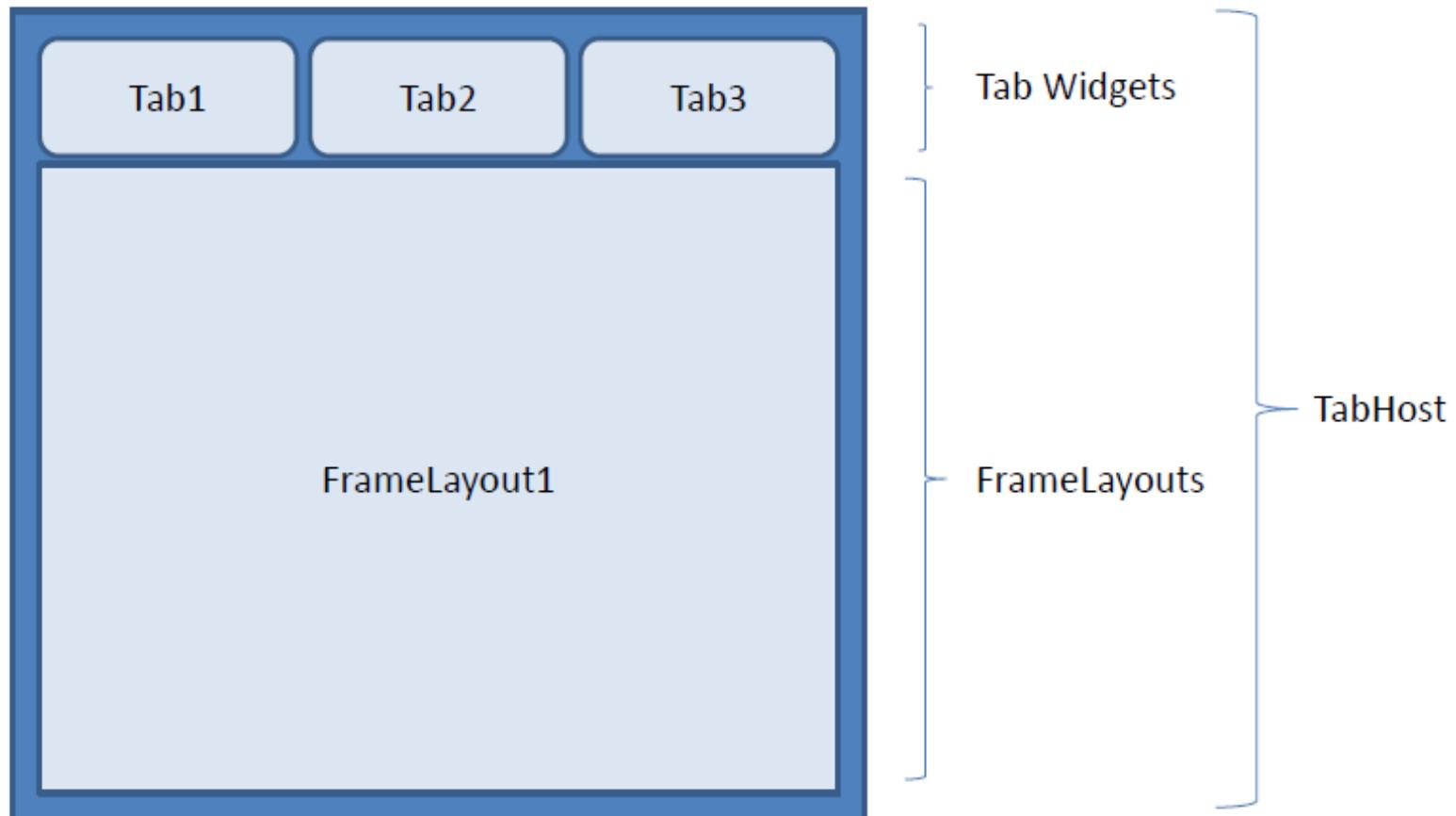
Boundaries touching other widgets



## **Bài tập 2: Fragment, TabSelection**

# Tab Selector

## Components



## **Bài tập 3: TextView, EditText**

# TextView

- Cách tạo:
  - Kéo thả TextView từ Palette.
- Sử dụng:
  - Sau khi kéo thả TextView từ Palette => trong **activity\_main.xml** xuất hiện Tab:

```
<TextView  
    android:id="@+id/textView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentTop="true"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="154dp"  
    android:text="TextView" />
```

# TextView

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
TextView tv1;  
tv1 = (TextView) findViewById(R.id.textView1);  
tv1.setText("Hello! How are you?");
```

# EditText

- Cách tạo:
  - Kéo thả EditText (hay Plain Text) từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong **activity\_main.xml** xuất hiện Tab:

```
<EditText  
    android:id="@+id/editText1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/textView1"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="36dp"  
    android:ems="10" >
```

# EditText

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
EditText ed1;  
ed1 = (EditText) findViewById(R.id.editText1);  
String t = ed1.getText().toString(); //Lấy nội dung trong  
EditText.
```

```
//Nếu muốn convert sang số nguyên (nếu dữ liệu là  
dạng số):  
int i = Integer.parseInt(ed1.getText().toString());
```

## **Bài tập 4: Button, ImageButton**

# Button

- Cách tạo:
  - Kéo thả Button từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong `activity_main.xml` xuất hiện Tab:

```
<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:text="Button" />
```

# Button

- Sử dụng:
  - Khai báo trong [MainActivity.java](#) (ví dụ trong hàm `onCreate(...)`):

```
Button bt1;
bt1 = (Button) findViewById(R.id.button1);
bt1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        tv1.setText(""+ed1.getText().toString());
    }
});
```

//Lưu ý: khi đó TextView tv1 và EditText ed1 phải có tính chất **final**.

# ImageButton

- Cách tạo:
  - Kéo thả ImageButton từ Palette => chọn Create New Icon => đặt tên cho Icon Name
- Sử dụng:
  - Sau khi kéo thả Palette => trong `activity_main.xml` xuất hiện Tab:

```
<ImageButton  
    android:id="@+id/imageButton1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/button1"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="61dp"  
    android:src="@drawable/imagebtn" />
```

# ImageButton

- Sử dụng:
  - Khai báo trong [MainActivity.java](#) (ví dụ trong hàm `onCreate(...)`):

```
ImageButton imgbt1;  
imgbt1 = (ImageButton) findViewById(R.id.button1);  
imgbt1.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // TODO Auto-generated method stub  
        tv1.setText(""+ed1.getText().toString());  
    }  
});
```

//Lưu ý: khi đó TextView tv1 và EditText ed1 phải có tính chất **final**.

## **Bài tập 5: CheckBox, ToggleButton**

# CheckBox

- Cách tạo:
  - Kéo thả CheckBox từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong **activity\_main.xml** xuất hiện Tab:

```
<CheckBox  
    android:id="@+id/checkBox1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/imageButton1"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="23dp"  
    android:text="CheckBox" />
```

# CheckBox

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
CheckBox chk = (CheckBox) findViewById(R.id.checkBox1);
```

```
if (chk.isChecked()){  
    tv1.setText("CheckBox is checked");  
}else  
    tv1.setText("CheckBox is UNchecked");
```

# ToggleButton

- Cách tạo:
  - Kéo thả ToggleButton từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong **activity\_main.xml** xuất hiện Tab:

```
<ToggleButton  
    android:id="@+id/toggleButton1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignRight="@+id/button1"  
    android:layout_below="@+id/checkBox1"  
    android:text="ToggleButton" />
```

# ToggleButton

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
ToggleButton tgbt1 = (ToggleButton) findViewById  
(R.id.toggleButton1);  
if (tgbt.isChecked()){  
    tgb1.setText(" ToggleButton is checked");  
}else  
    tgb1.setText(" ToggleButton is UNchecked");
```

## **Bài tập 6: RadioButton, RadioGroup**

# RadioGroup

- Cách tạo:
  - Kéo thả RadioGroup từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong `activity_main.xml` xuất hiện Tab:

```
<RadioGroup  
    android:id="@+id/radioGroup1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/imageButton1"  
    android:layout_below="@+id radioButton1"  
    android:layout_marginTop="16dp" >  
  
<RadioButton  
    android:id="@+id/radioButton0"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:checked="true"  
    android:text="RadioButton" />  
  
<RadioButton  
    android:id="@+id/radioButton1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="RadioButton1" />  
  
<RadioButton  
    android:id="@+id/radioButton2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="RadioButton2" />  
</RadioGroup>
```

# RadioGroup

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
RadioGroup rdgroup = (RadioGroup)findViewById(R.id.radioGroup1);
int idChecked = rdgroup.getCheckedRadioButtonId();

if (idChecked == R.id.radio0){
    tv2.setText("Radio0 is checked");
}
if (idChecked == R.id.radio1){
    tv2.setText("Radio1 is checked");
}
if (idChecked == R.id.radio2){
    tv2.setText("Radio2 is checked");
}
```

## **Bài tập 7: TimePicker, DatePicker**

# TimePicker

- Cách tạo:
  - Kéo thả TimePicker từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong **activity\_main.xml** xuất hiện Tab:

```
<TimePicker  
    android:id="@+id/timePicker1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentBottom="true"  
    android:layout_centerHorizontal="true"  
    android:layout_marginBottom="22dp" />
```

# TimePicker

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
TimePicker tpk1 = (TimePicker) findViewById(R.id.timePicker1);  
tpk1.setIs24HourView(true);
```

```
Toast.makeText(getApplicationContext(),"Time  
"+tpk1.getCurrentHour()+":"+tpk1.getCurrentMinute(),  
Toast.LENGTH_SHORT).show();
```

# DatePicker

- Cách tạo:
  - Kéo thả TimePicker từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong `activity_main.xml` xuất hiện Tab:

```
<DatePicker  
    android:id="@+id/datePicker1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentBottom="true"  
    android:layout_alignParentLeft="true" />
```

# DatePicker

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
DatePicker dpk1 = (DatePicker)findViewById(R.id.datePicker1);
Toast.makeText(getApplicationContext(), "Date:
"+dpk1.getDayOfMonth()+
"/"+dpk1.getMonth()+"/"+dpk1.getYear(),
Toast.LENGTH_SHORT).show();
```

## **Bài tập 8: ScrollView, ImageView**

# ScrollView

- Cách tạo:
  - Kéo thả ScrollView từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong `activity_main.xml` xuất hiện Tab:

```
<ScrollView  
    android:id="@+id/scrollView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" >  
  
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="200dp"  
    android:orientation="vertical" >  
  
<TextView  
    android:id="@+id/textView2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Large Text"  
    android:textAppearance="?android:attr/textAppearanceLarge" />  
<Button  
    android:id="@+id/button1"  
    style="?android:attr/buttonStyleSmall"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Button" />  
</LinearLayout>  
</ScrollView>
```

# ImageView

- Cách tạo:
  - Copy (hoặc link) file hình vào trong các thư mục drawable.
  - Kéo thả ImageView từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong `activity_main.xml` xuất hiện Tab:

```
<ImageView  
    android:id="@+id/imageView2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/button3_2"  
    android:layout_alignParentBottom="true"  
    android:src="@drawable/vd" />
```

# ImageView

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
ImageView imgview =  
(ImageView)findViewById(R.id.imageView2);
```

## **Bài tập 9: ListView, Spinner**

# ListView

- Cách tạo:
  - Kéo thả ListView từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong <file>.xml xuất hiện Tab:

```
<ListView  
    android:id="@+id/listView1"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:layout_marginTop="59dp" >  
</ListView>
```

# ListView

- Sử dụng:

- Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
String[] students = {"Nam", "Lan", "Hoa", "Hanh", "Thanh",
                     "Trung", "Son", "Dung", "Van", "Manh", "Thang"};
```

```
ListView lv = (ListView) findViewById(R.id.listView1);
```

```
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, students);
```

```
lv.setAdapter(adapter);
```

```
lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
        Toast.makeText(getApplicationContext(), "Position :" + arg2 + " - Value = " + students[arg2],
        Toast.LENGTH_SHORT).show();
    }
});
```

# Spinner

- Cách tạo:
  - Kéo thả Spinner từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong <file>.xml xuất hiện Tab:

```
<Spinner  
    android:id="@+id/spinner1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentTop="true"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="124dp" />
```

# Spinner

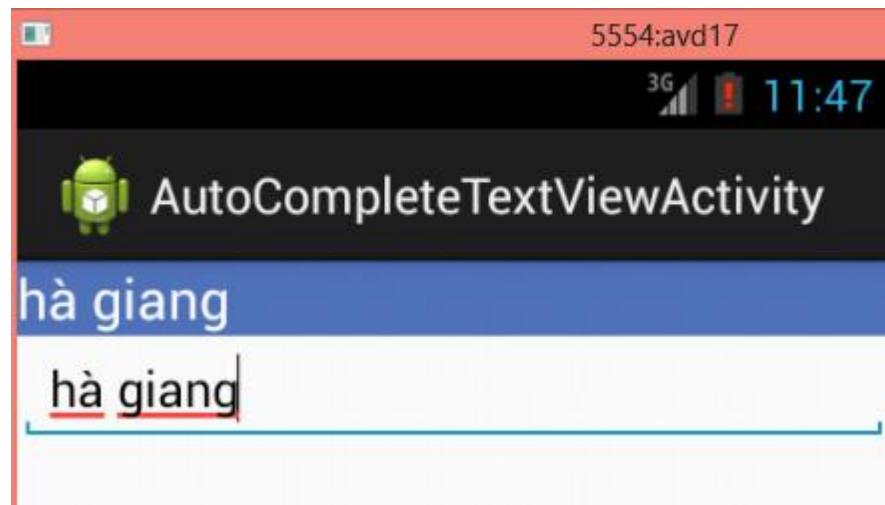
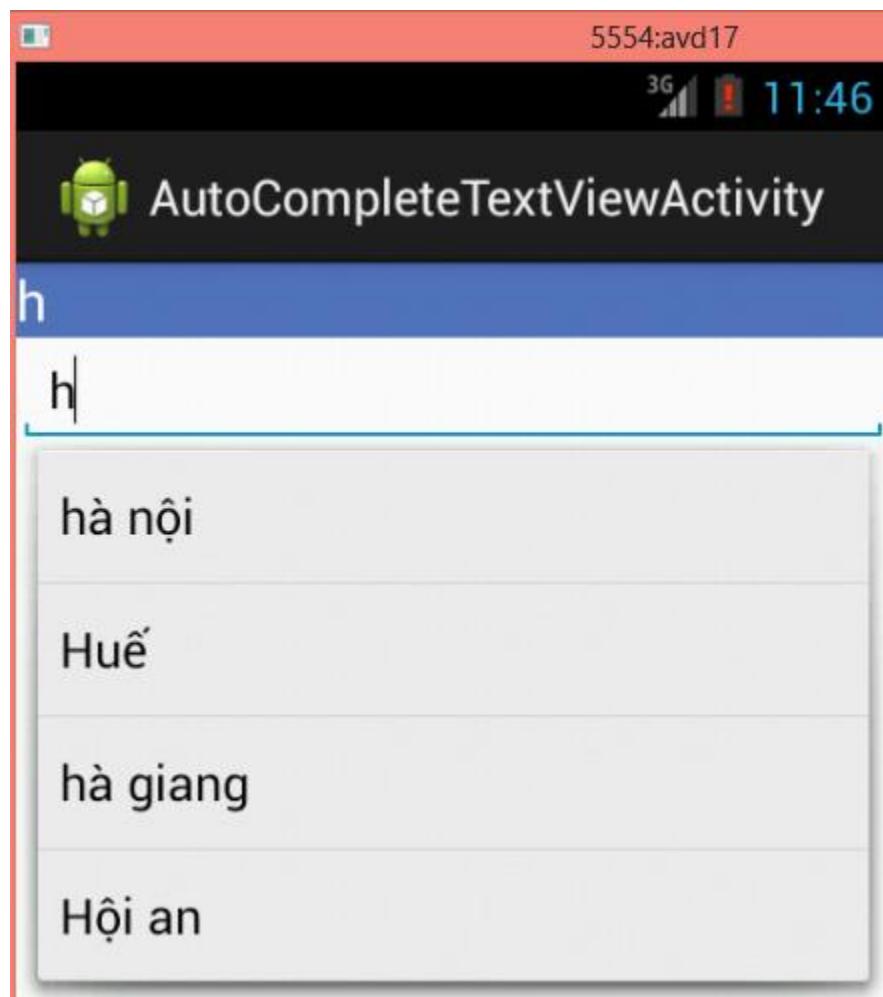
- Sử dụng:
  - Khai báo trong **MainActivity.java** (ví dụ trong hàm `onCreate(...)`):

```
String goods[] = {"TV", "Labtop", "SmartPhone", "Watch", "Book"};  
  
Spinner spin = (Spinner) findViewById(R.id.spinner1);  
  
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, goods);  
adapter.setDropDownViewResource(android.R.layout.simple_list_item_single_choice);  
  
spin.setAdapter(adapter);  
  
spin.setOnItemSelectedListener(new OnItemSelectedListener() {  
  
    @Override  
    public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {  
        Toast.makeText(getApplicationContext(), "Position: "+arg2+" - Selected Value: "+goods[arg2], Toast.LENGTH_SHORT).show();  
    }  
  
    @Override  
    public void onNothingSelected(AdapterView<?> arg0) {  
        // TODO Auto-generated method stub  
  
    }  
});
```

# **Bài tập 10:**

## **AutoCompleteTextView, Gallery**

# AutocompleteTextView



# AutoCompleteTextView

- Cách tạo:
  - Kéo thả AutoCompleteTextView và Multi.AutoCompleteTextView từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong <file>.xml xuất hiện Tab:

```
<AutoCompleteTextView  
    android:id="@+id/autoCompleteTextView1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentTop="true"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="90dp"  
    android:ems="10"  
    android:completionThreshold="1"  
    android:text="" />  
  
<Multi.AutoCompleteTextView  
    android:id="@+id/multiAutoCompleteTextView1"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/autoCompleteTextView1"  
    android:layout_centerHorizontal="true"  
    android:layout_marginTop="49dp"  
    android:ems="10"  
    android:completionThreshold="1"  
    android:text="" />
```

# AutoCompleteTextView

- Sử dụng:
  - Khai báo trong `MainActivity.java` (ví dụ trong hàm `onCreate(...)`):

```
String[] students =  
{"Nam", "Lan", "Hoa", "Hanh", "Thanh", "Trung", "Son", "Dung", "Van", "Manh", "Thang"};
```

```
AutoCompleteTextView autoComTV =  
(AutoCompleteTextView)findViewById(R.id.autoCompleteTextView1);  
autoComTV.addTextChangedListener(this);  
autoComTV.setAdapter(new  
ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, students));
```

```
MultiAutoCompleteTextView mulComTV = (MultiAutoCompleteTextView)findViewById(R.id.multi.AutoComplete.TextView1);  
mulComTV.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_dropdown_item_1line, students));  
mulComTV.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
```

# Picture Gallery



# Picture Gallery

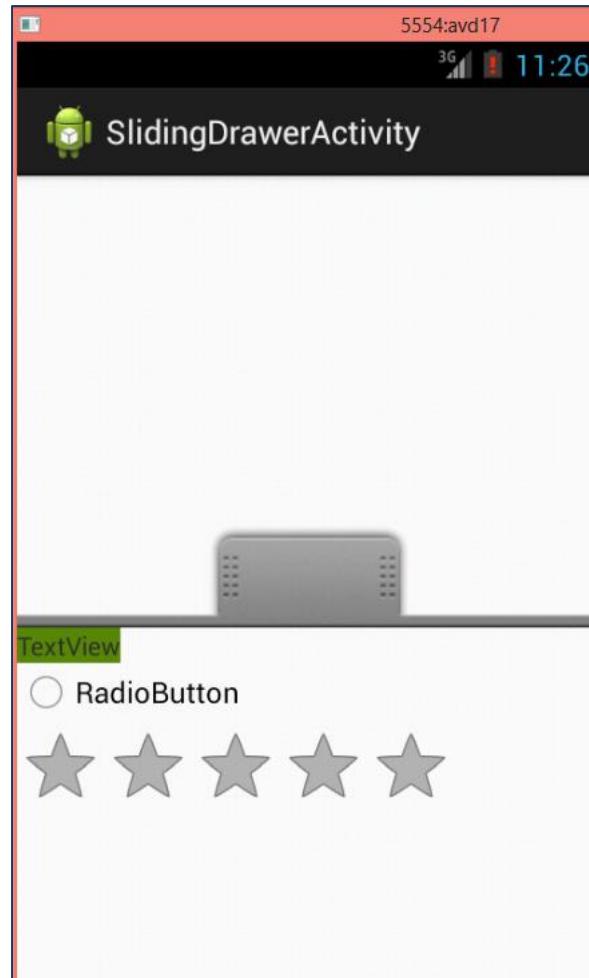
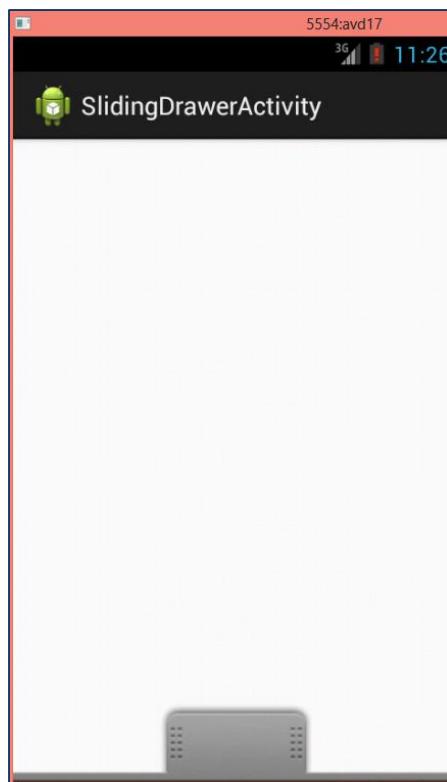
- Cách tạo:
  - Kéo thả ImageView và Gallery từ Palette.
- Sử dụng:
  - Sau khi kéo thả Palette => trong <file>.xml xuất hiện Tab:

```
<Gallery  
    android:id="@+id/Gallery01"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content" >  
</Gallery>
```

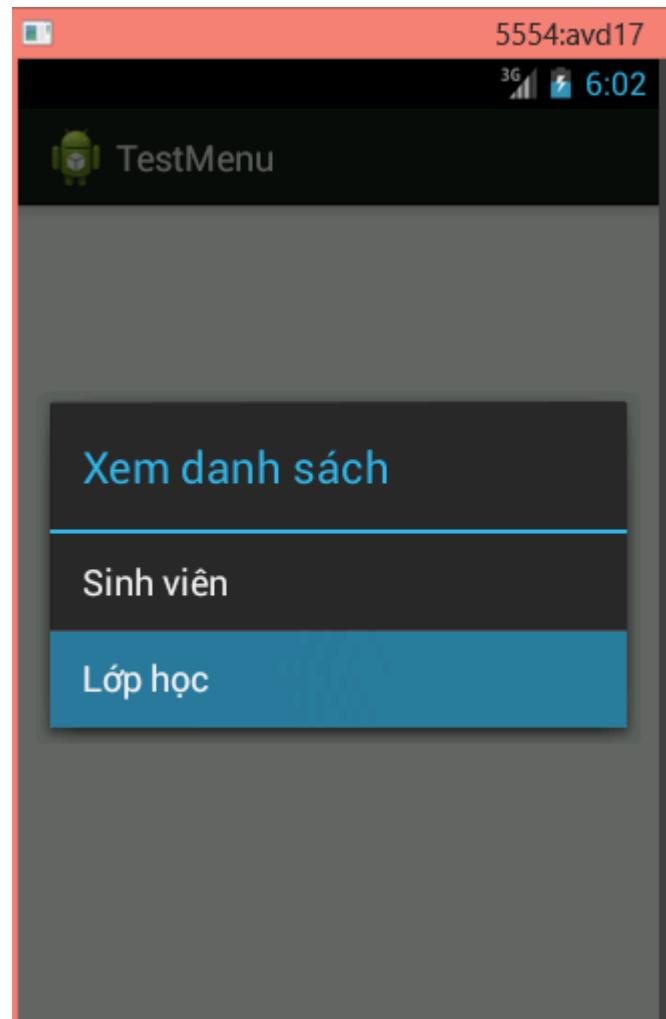
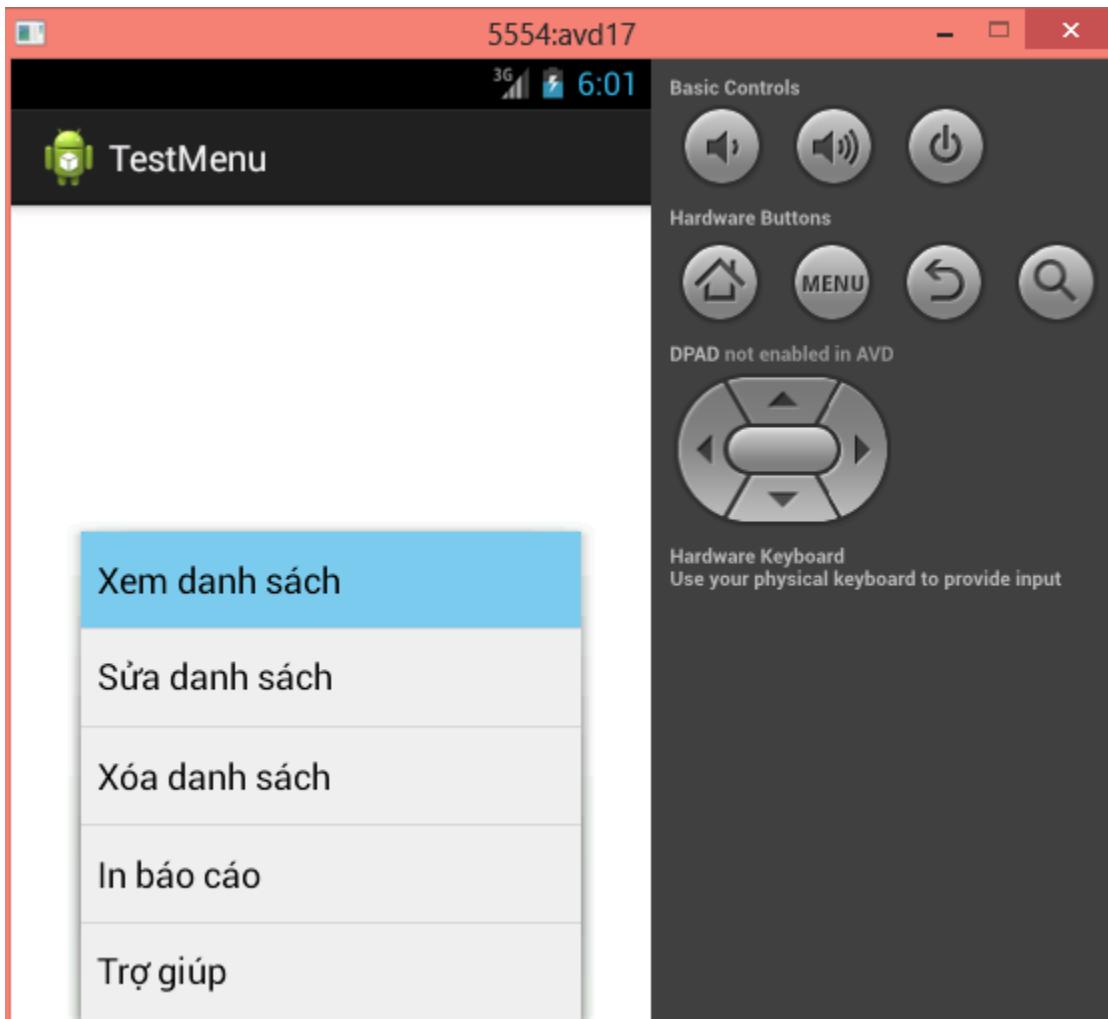
```
<ImageView  
    android:id="@+id/ImageView01"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" >  
</ImageView>
```

## **Bài tập 11: SlideDrawer, Menu**

# SlideDrawer



# Menu



# BTVN: Trình bày về quản lý dữ liệu và đưa ra ví dụ cụ thể

## 1. Quản lý dữ liệu dạng key-value

- Tạo và kết nối tới file SharedPreferences.
- Ghi dữ liệu vào SharePreferences.
- Đọc dữ liệu từ SharePreferences.

## 2. Quản lý dữ liệu từ file trong bộ nhớ thiết bị

- Đọc, ghi vào file (bộ nhớ trong).
- Đọc, ghi vào file (bộ nhớ ngoài).

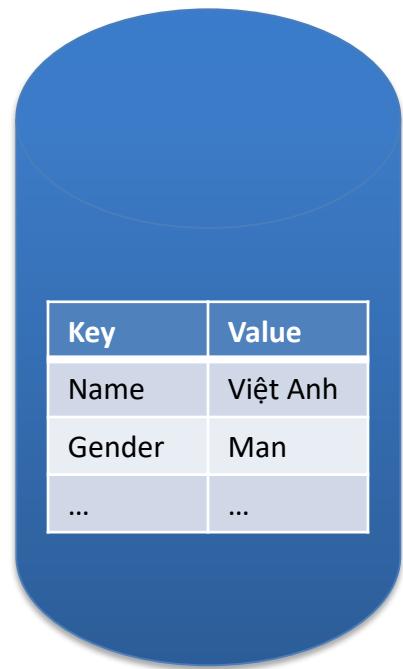
### **3. Lập trình quản lý dữ liệu**

# Các cách quản lý dữ liệu cho ứng dụng Android

1. Quản lý dữ liệu với SharedPreferences
2. Quản lý dữ liệu từ file trong bộ nhớ thiết bị
3. Quản lý dữ liệu với cơ sở dữ liệu quan hệ SQLite

# 1. Quản lý dữ liệu với SharedPreferences

- Android cung cấp tập các API SharePreferences cho phép quản lý lượng nhỏ dữ liệu đơn giản của ứng dụng thông qua cặp **từ khóa – giá trị (key-value)**.
- SharedPreferences lưu trữ dữ liệu trong 1 file .xml tự động được sinh ra.
- Thư mục lưu trữ: **data/data/<package-name>shared-prefs** trong bộ nhớ trong của thiết bị.



# SharedPreferences dùng trong trường hợp nào?

- Cách quản lý dữ liệu này phù hợp khi muốn lưu trạng thái ứng dụng
  - Thông tin người dùng
  - Thiết lập của người dùng
  - Vị trí (Location)
  - ...

**Ví dụ:** Trong ứng dụng có tùy chọn cho phép người dùng cài đặt kích cỡ chữ hiển thị. Yêu cầu khi người dùng chọn cài đặt xong và tắt ứng dụng đi thì lần sau mở lại ứng dụng vẫn hiển thị nội dung với kích cỡ chữ đã được cài đặt trước đó.

# Làm việc với SharedPreferences

- Tạo và kết nối tới file SharedPreferences
- Ghi dữ liệu vào SharedPreferences
- Đọc dữ liệu từ SharedPreferences

# Làm việc với SharedPreferences

## 1.1. Tạo và kết nối tới file SharedPreferences

- getSharedPreferences(String name, int mode)
  - Khi có nhiều file xml SharedPreferences trong bộ nhớ.
- getPreferences(int mode)
  - Khi có 1 file xml SharedPreferences.

# Làm việc với SharedPreferences

## 1.1. Tạo và kết nối tới file SharedPreferences

### Input:

name: tên file

mode:

- **MODE\_PRIVATE**: chỉ riêng ứng dụng đang dùng mới có thể truy nhập được vào file.
- **MODE\_WORLD\_READABLE**: tất cả các ứng dụng đều có thể đọc được file.
- **MODE\_WORLD\_WRITEABLE**: tất cả các ứng dụng đều có thể ghi được vào file.
- **MODE\_MULTI\_PROCESS**: các process khác nhau đều có thể thay đổi được nội dung của file.

# Làm việc với SharedPreferences

## 1.1. Tạo và kết nối tới file SharedPreferences

### Output:

Đối tượng thuộc lớp **SharedPreferences** (Dùng đối tượng này để đọc/ ghi dữ liệu vào file SharedPreferences trên thiết bị).

# Làm việc với SharedPreferences

## 1.1. Tạo và kết nối tới file SharedPreferences

- Tham chiếu tới các file SharedPreferences

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences(
    getString(R.string.preference_file_key), Context.MODE_PRIVATE);
```

- Tham chiếu tới 1 file SharedPreferences

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

# Làm việc với SharedPreferences

## 1.2. Ghi dữ liệu vào SharedPreferences

- Tạo đối tượng SharedPreferences.Editor
- Gọi phương thức như putInt(), putString()...với tham số truyền vào là giá trị dạng key-value để lưu dữ liệu vào file.

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score), newHighScore);
editor.commit();
```

# Làm việc với SharedPreferences

## 1.2. Ghi dữ liệu vào SharedPreferences

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score), newHighScore);
editor.commit();
```

- Tạo đối tượng SharedPreferences.Editor
- Gọi phương thức như putInt(), putString()...với tham số truyền vào là giá trị dạng key-value để lưu dữ liệu vào file.
- Gọi phương thức commit() của đối tượng SharedPreferences.Editor để hoàn tất việc ghi dữ liệu

# Làm việc với SharedPreferences

## 1.3. Đọc dữ liệu từ SharedPreferences

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.string.saved_high_score_default);
long highScore = sharedPref.getInt(getString(R.string.saved_high_score), defaultValue);
```

- Tham số truyền vào là **key** mà muốn lấy **value**, nếu không truyền key cho hàm thì giá trị defaultValue sẽ trả về.

## 2. Quản lý dữ liệu từ file trong bộ nhớ thiết bị

- Android cung cấp tập các API File cho phép ứng dụng đọc/ ghi lượng lớn dữ liệu vào file trong bộ nhớ thiết bị. Tập các API File được cung cấp bởi gói java.io.
- Các thiết bị Android có 2 bộ nhớ dữ liệu: bộ nhớ trong (Internal storage) và bộ nhớ ngoài như thẻ SD (External storage).

Bộ nhớ trong (Internal storage)	Bộ nhớ ngoài (External storage)
<b>Luôn sẵn sàng</b>	Không phải lúc nào cũng sẵn sàng vì người dùng có thể thêm vào hoặc gỡ bỏ bộ nhớ ngoài khỏi thiết bị.
<b>Các file lưu trữ ở bộ nhớ trong chỉ được phép truy cập bởi các ứng dụng tạo ra chúng.</b>	Các file lưu trữ ở bộ nhớ ngoài có thể được truy cập tự do.
<b>Khi gỡ bỏ ứng dụng khỏi thiết bị thì hệ thống cũng gỡ bỏ luôn những file do ứng dụng tạo ra ở bộ nhớ trong.</b>	Khi gỡ bỏ ứng dụng khỏi thiết bị thì hệ thống chỉ gỡ bỏ những file do ứng dụng tạo ra ở bộ nhớ ngoài khi trước đó ứng dụng lưu trữ dữ liệu vào đường dẫn trả về từ phương thức <code>getExternalFilesDir()</code> .
<b>Phù hợp khi không muốn người dùng hoặc ứng dụng khác truy cập vào file dữ liệu của ứng dụng.</b>	Phù hợp khi ứng dụng cho phép file dữ liệu của mình được truy cập tự do từ người dùng, ứng dụng khác hay từ máy tính.

# Làm việc với file

1. Đọc, ghi dữ liệu từ bộ nhớ trong của thiết bị
2. Đọc, ghi dữ liệu từ bộ nhớ ngoài của thiết bị

# 1. Làm việc với file thuộc bộ nhớ trong

- Mặc định ứng dụng được cấp quyền truy cập bộ nhớ trong
- Các chế độ cấp cho file thuộc bộ nhớ trong:
  - **MODE\_WORLD\_READABLE** : cho phép tất cả ứng dụng có thể đọc dữ liệu từ file.
  - **MODE\_PRIVATE** : cho phép file chỉ được truy cập bởi ứng dụng tạo ra nó.
  - **MODE\_WORLD\_WRITEABLE** : cho phép tất cả ứng dụng có thể ghi dữ liệu vào file.
  - **MODE\_APPEND**: nếu file chưa tồn tại -> Tạo mới; Nếu file đã tồn tại -> cho phép thêm nội dung vào file

# Ghi dữ liệu vào bộ nhớ trong

```
FileOutputStream fOut =
    openFileOutput("textfile.txt",
                  MODE_WORLD_READABLE);
OutputStreamWriter osw = new
    OutputStreamWriter(fOut);

//---write the string to the file---
osw.write(str);
osw.flush();
osw.close();
```

- `openFileOutput()`: mở file để ghi dữ liệu.
- `write()` của đối tượng `OutputStreamWriter`: ghi dữ liệu vào bộ nhớ trong

# Đọc dữ liệu từ bộ nhớ trong

```
FileInputStream fIn =
    openFileInput("textfile.txt");
InputStreamReader isr = new
    InputStreamReader(fIn);

char[] inputBuffer = new char[READ_BLOCK_SIZE];
String s = "";

int charRead;
while ((charRead = isr.read(inputBuffer))>0)
{
    //---convert the chars to a String---
    String readString =
        String.valueOf(inputBuffer, 0,
                      charRead);
    s += readString;

    inputBuffer = new char[READ_BLOCK_SIZE];
}
```

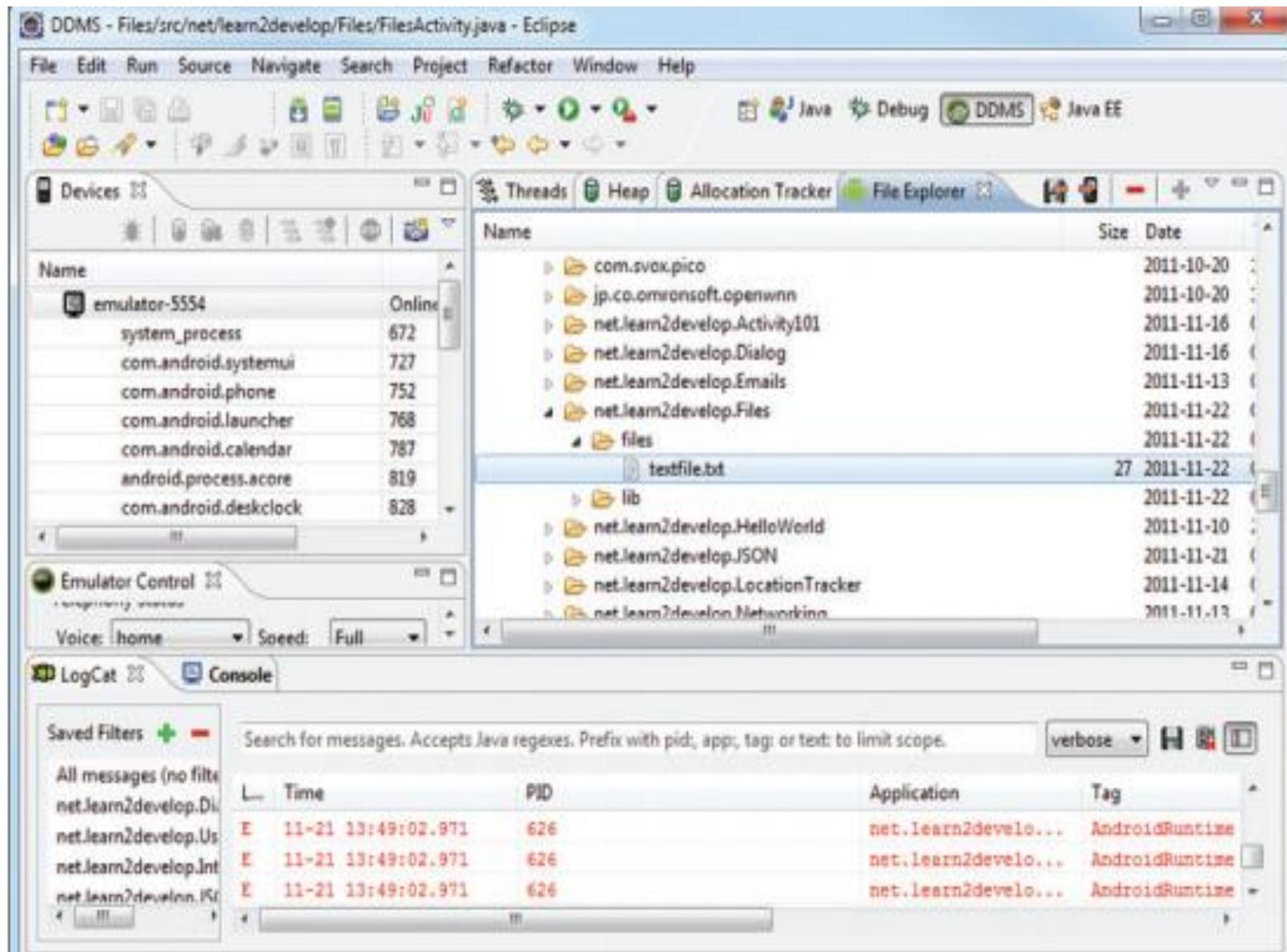
- `openFileInput()` : mở file để đọc dữ liệu.
- `read()` của đối tượng `InputStreamReader`: đọc lần lượt khối có kích thước `READ_BLOCK_SIZE` vào bộ đệm buffer.

# Xóa file lưu trong bộ nhớ trong ứng dụng

- Gọi hàm deleteFile() của đối tượng Context của ứng dụng

```
myContext.deleteFile(fileName);
```

# Đường dẫn tới file lưu trong bộ nhớ trong



## 2. Làm việc với file thuộc bộ nhớ ngoài

- Cấp quyền cho ứng dụng truy cập bộ nhớ ngoài
- Kiểm tra thiết bị có bộ nhớ ngoài hay không
- Ghi dữ liệu vào bộ nhớ ngoài
- Đọc dữ liệu từ bộ nhớ ngoài

# Cấp quyền cho ứng dụng truy cập bộ nhớ ngoài

- Trong file AndroidManifest.xml cần khai báo quyền để cho phép ứng dụng ghi dữ liệu vào bộ nhớ ngoài (**WRITE\_EXTERNAL\_STORAGE**)

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

- Khai báo quyền cho phép ứng dụng đọc dữ liệu từ bộ nhớ ngoài

```
<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

- Nếu ứng dụng được cấp quyền ghi thì mặc định được cấp quyền đọc dữ liệu từ bộ nhớ ngoài.

# Kiểm tra thiết bị có bộ nhớ ngoài hay không

- Environment.getExternalStorageState()

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

# Kiểm tra thiết bị có bộ nhớ ngoài hay không

Các giá trị trả về của hàm Environment.getExternalStorageState()

- **MEDIA\_MOUNTED**: thẻ sẵn sàng cho việc đọc ghi.
- **MEDIA\_MOUNTED\_READ\_ONLY** : thẻ chỉ cho phép đọc.
- **MEDIA\_BAD\_REMOVAL**: sdcard đã bị tháo ra không đúng cách.
- **MEDIA\_CHECKING**: sdcard đang trong quá trình kiểm tra lỗi.
- **MEDIA\_REMOVED**: sdcard đã được tháo ra.
- **MEDIA\_NOFS**: sdcard đang rỗng hoặc sử dụng định dạng không phù hợp.
- **MEDIA\_UNMOUNTABLE**: sdcard không thể sử dụng được do bị lỗi.
- **MEDIA\_UNMOUNTED**: sdard có trong máy nhưng đã được gỡ bằng lệnh.

# 2 loại file trong bộ nhớ ngoài

- Các file lưu trữ trong bộ nhớ ngoài do ứng dụng tạo ra thuộc 1 trong 2 kiểu sau:
  - **Public files**: mọi ứng dụng đều có thể truy cập (vd: file download, file ảnh chụp,...) => vẫn **tồn tại** khi gỡ ứng dụng.
    - `getExternalStorageDirectory()` trả về đường dẫn của public file trong bộ nhớ ngoài.
  - **Private files**: chỉ được sử dụng bởi ứng dụng tạo ra file đó => **bị xóa** khi gỡ ứng dụng.
    - `getExternalFilesDir()` trả về đường dẫn của private file trong bộ nhớ ngoài.

# Ghi dữ liệu vào bộ nhớ ngoài

```
//---SD Card Storage---
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File (sdCard.getAbsolutePath() + "/MyFiles");
directory.mkdirs();
File file = new File(directory, "textfile.txt");
FileOutputStream fOut = new FileOutputStream(file);

/*
FileOutputStream fOut =
    openFileOutput("textfile.txt",
                  MODE_WORLD_READABLE);
*/

OutputStreamWriter osw = new
    OutputStreamWriter(fOut);

//---write the string to the file---
osw.write(str);
osw.flush(); osw.close();
```

# Đọc file dữ liệu từ bộ nhớ ngoài

```
//---SD Storage---
File sdCard = Environment.getExternalStorageDirectory();
File directory = new File (sdCard.getAbsolutePath() + "/MyFiles");
File file = new File(directory, "textfile.txt");
FileInputStream fIn = new FileInputStream(file);
InputStreamReader isr = new InputStreamReader(fIn);

/*
FileInputStream fIn =
    openFileInput("textfile.txt");
InputStreamReader isr = new
    InputStreamReader(fIn);
*/

char[] inputBuffer = new char[READ BLOCK SIZE];
String s = "";

int charRead;
while ((charRead = isr.read(inputBuffer)) >0)
{
    //---convert the chars to a String---
    String readString =
        String.valueOf(inputBuffer, 0,
                      charRead);
    s += readString;

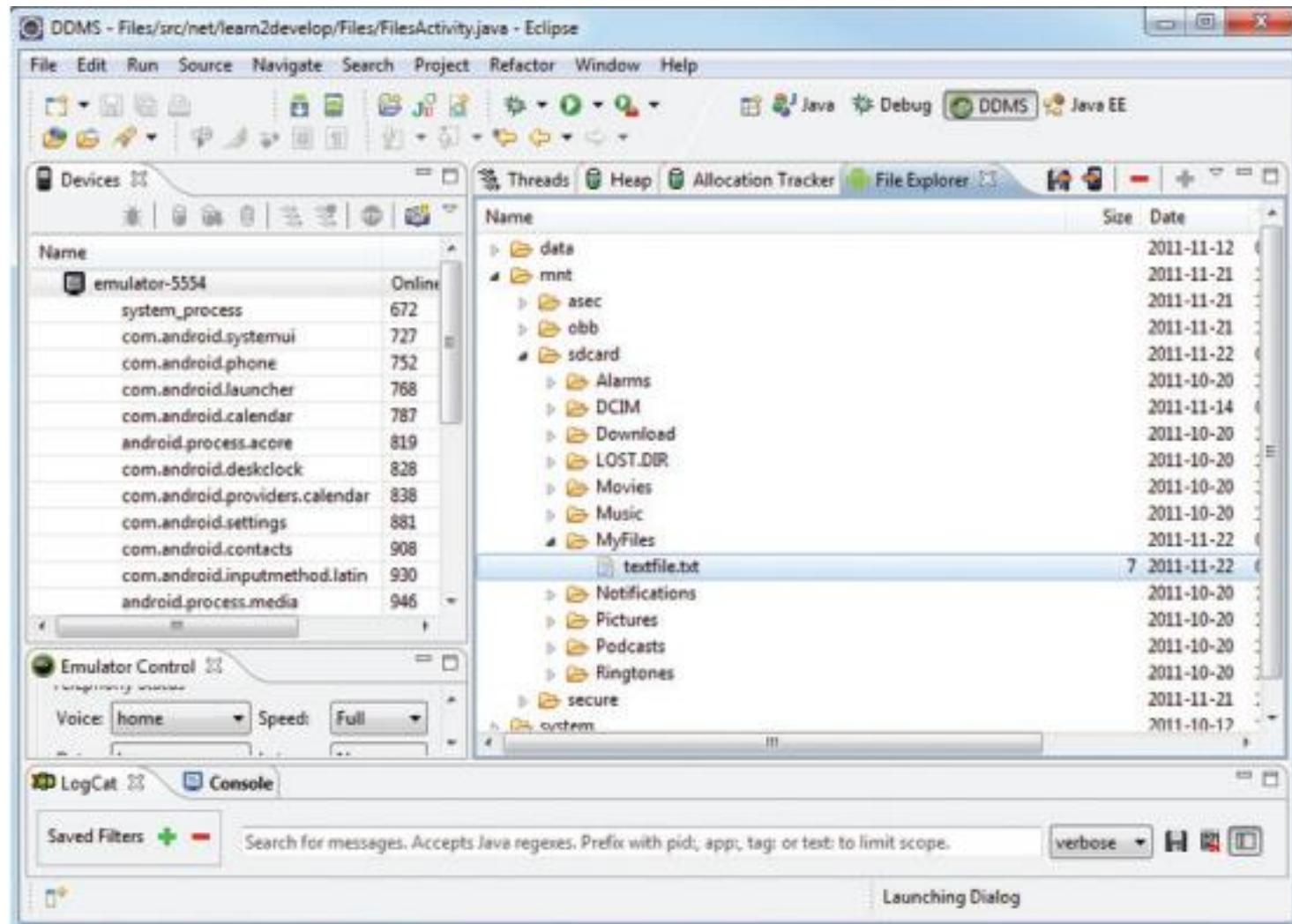
    inputBuffer = new char[READ BLOCK SIZE];
}
```

# Xóa file lưu ở bộ nhớ ngoài

- Gọi hàm delete() của đối tượng File

```
myFile.delete();
```

# Đường dẫn tới file lưu trong bộ nhớ ngoài SD của thiết bị



### 3. Quản lý dữ liệu với CSDL quan hệ

- Android cung cấp hệ quản trị cơ sở dữ liệu SQLite để quản lý cơ sở dữ liệu quan hệ của ứng dụng.
- Cơ sở dữ liệu quan hệ SQLite tạo ra cho của ứng dụng này thì chỉ được truy cập bởi chính nó mà không cho phép ứng dụng khác truy cập vào, cơ sở dữ liệu này được lưu trữ trong thư mục `/data/data/<package_name>/databases` của hệ thống
  - Sử dụng DDMS -> File Explorer, vào thư mục trên sẽ thấy file cơ sở dữ liệu của ứng dụng.

# Làm việc với CSDL Sqlite (cách 1)

- Định nghĩa lược đồ CSDL
- Tạo cơ sở dữ liệu. Tạo bảng, xóa bảng trong CSDL
- Đọc dữ liệu từ CSDL
- Ghi (thêm) dữ liệu vào trong CSDL
- Xóa dữ liệu trong CSDL
- Cập nhật dữ liệu trong CSDL

# Định nghĩa lược đồ CSDL

- Lược đồ cơ sở dữ liệu là khai báo cách tổ chức cơ sở dữ liệu.  
Lược đồ cơ sở dữ liệu được khai báo thông qua lớp chứa định nghĩa các thành phần của cơ sở dữ liệu

```
public final class FeedReaderContract {  
    // To prevent someone from accidentally instantiating the contract class,  
    // give it an empty constructor.  
    public FeedReaderContract() {}  
  
    /* Inner class that defines the table contents */  
    public static abstract class FeedEntry implements BaseColumns {  
        public static final String TABLE_NAME = "entry";  
        public static final String COLUMN_NAME_ENTRY_ID = "entryid";  
        public static final String COLUMN_NAME_TITLE = "title";  
        public static final String COLUMN_NAME_SUBTITLE = "subtitle";  
        ...  
    }  
}
```

# Tạo cơ sở dữ liệu

```
SQLiteDatabase database=null;  
database=openOrCreateDatabase("mydata.db",  
    SQLiteDatabase.CREATE_IF_NECESSARY, null);
```

## Trong đó:

openOrCreateDatabase (String name, int mode,  
SQLiteDatabase.CursorFactory factory)

- Name: tên của CSDL.
- Mode: Chế độ mở file (MODE\_PRIVATE,  
MODE\_WORLD\_READABLE,  
MODE\_WORLD\_WRITEABLE...)
- Factory: lớp tùy chọn được gọi để thiết lập con trỏ khi  
truy vấn.

# Tạo bảng trong CSDL

- Khai báo câu lệnh để tạo và xóa bảng trong CSDL

```
private static final String TEXT_TYPE = " TEXT";
private static final String COMMA_SEP = ",";
private static final String SQL_CREATE_ENTRIES =
    "CREATE TABLE " + FeedEntry.TABLE_NAME + " (" +
    FeedEntry._ID + " INTEGER PRIMARY KEY," +
    FeedEntry.COLUMN_NAME_ENTRY_ID + TEXT_TYPE + COMMA_SEP +
    FeedEntry.COLUMN_NAME_TITLE + TEXT_TYPE + COMMA_SEP +
    ... // Any other options for the CREATE command
    " )";

private static final String SQL_DELETE_ENTRIES =
    "DROP TABLE IF EXISTS " + FeedEntry.TABLE_NAME;
```

- Tạo bảng

database.execSQL(SQL\_CREATE\_ENTRIES); //database là đối tượng của lớp SQLiteDatabase đã tạo ra ở phần tạo CSDL

- Xóa bảng

database.execSQL(SQL\_DELETE\_ENTRIES);

# Thêm dữ liệu vào CSDL

- Sử dụng phương thức **insert()** của đối tượng **SQLiteDatabase**

**ContentValues[(key1,value1), (key2,value2)...]**

**insert (String table, String nullColumnHack, ContentValues values)**

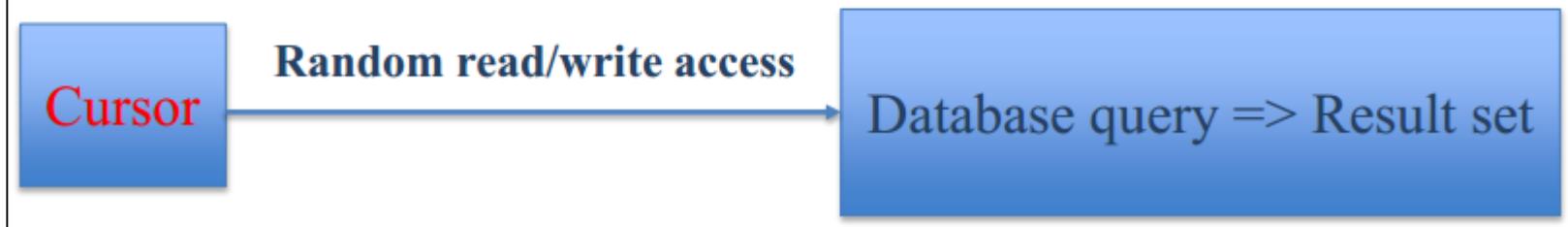
```
// Gets the data repository in write mode
SQLiteDatabase db = mDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_ENTRY_ID, id);
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_CONTENT, content);

// Insert the new row, returning the primary key value of the new row
long newRowId;
newRowId = db.insert(
    FeedEntry.TABLE_NAME,
    FeedEntry.COLUMN_NAME_NULLABLE,
    values);
```

# Đọc (truy vấn) dữ liệu từ CSDL

- Sử dụng phương thức query() của đối tượng SQLiteDatabase
  - public `Cursor query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)`
  - public `Cursor rawQuery (String sql, String[] selectionArgs)`



# Ví dụ truy vấn dữ liệu từ CSDL (1)

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();  
  
// Define a projection that specifies which columns from the database  
// you will actually use after this query.  
String[] projection = {  
    FeedEntry._ID,  
    FeedEntry.COLUMN_NAME_TITLE,  
    FeedEntry.COLUMN_NAME_UPDATED,  
    ...  
};  
  
// How you want the results sorted in the resulting Cursor  
String sortOrder =  
    FeedEntry.COLUMN_NAME_UPDATED + " DESC";  
  
Cursor c = db.query(  
    FeedEntry.TABLE_NAME, // The table to query  
    projection,           // The columns to return  
    selection,            // The columns for the WHERE clause  
    selectionArgs,         // The values for the WHERE clause  
    null,                 // don't group the rows  
    null,                 // don't filter by row groups  
    sortOrder             // The sort order  
);
```

## Ví dụ truy vấn dữ liệu từ CSDL (2)

```
Cursor c=database.query("tbllop",
    null, null, null, null, null, null);
c.moveToFirst();
String data="";
while(c.isAfterLast()==false)
{
    data+=c.getString(0)+"-"++
        c.getString(1)+"-"++
        c.getString(2);
    data+="\n";
    c.moveToNext();
}
Toast.makeText(this, data, Toast.LENGTH_LONG).show();
c.close();
```

# Xóa dữ liệu trong CSDL

- Sử dụng phương thức delete() của đối tượng SQLiteDatabase để xóa các dòng dữ liệu từ 1 bảng trong CSDL

```
delete(String table, String whereClause, String[] whereArgs)
```

```
// Define 'where' part of query.  
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";  
// Specify arguments in placeholder order.  
String[] selectionArgs = { String.valueOf(rowId) };  
// Issue SQL statement.  
db.delete(table_name, selection, selectionArgs);
```

# Cập nhật dữ liệu trong CSDL

- Sử dụng phương thức update() của đối tượng SQLiteDatabase để cập nhật dữ liệu đã có trong CSDL

update(String table, ContentValues values, String  
whereClause, String[] whereArgs)

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// New value for one column
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);

// Which row to update, based on the ID
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
String[] selectionArgs = { String.valueOf(rowId) };

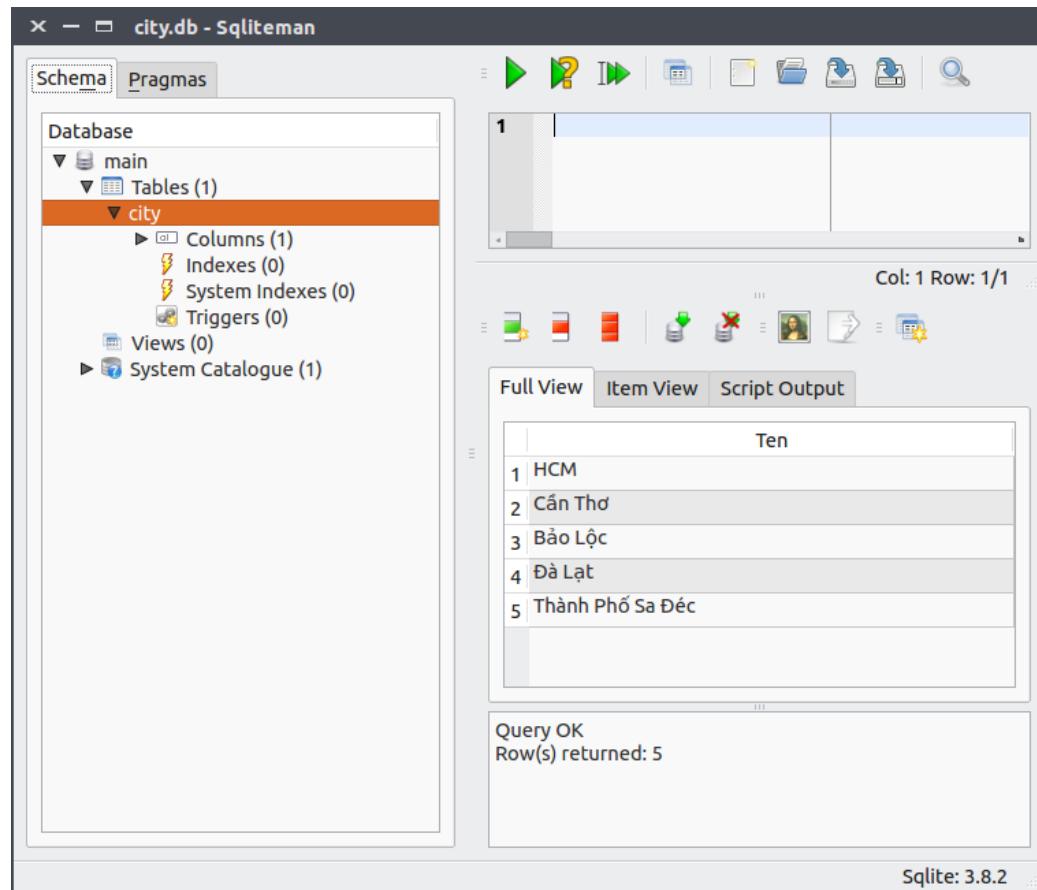
int count = db.update(
    FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values,
    selection,
    selectionArgs);
```

# Làm việc với CSDL Sqlite (cách 2)

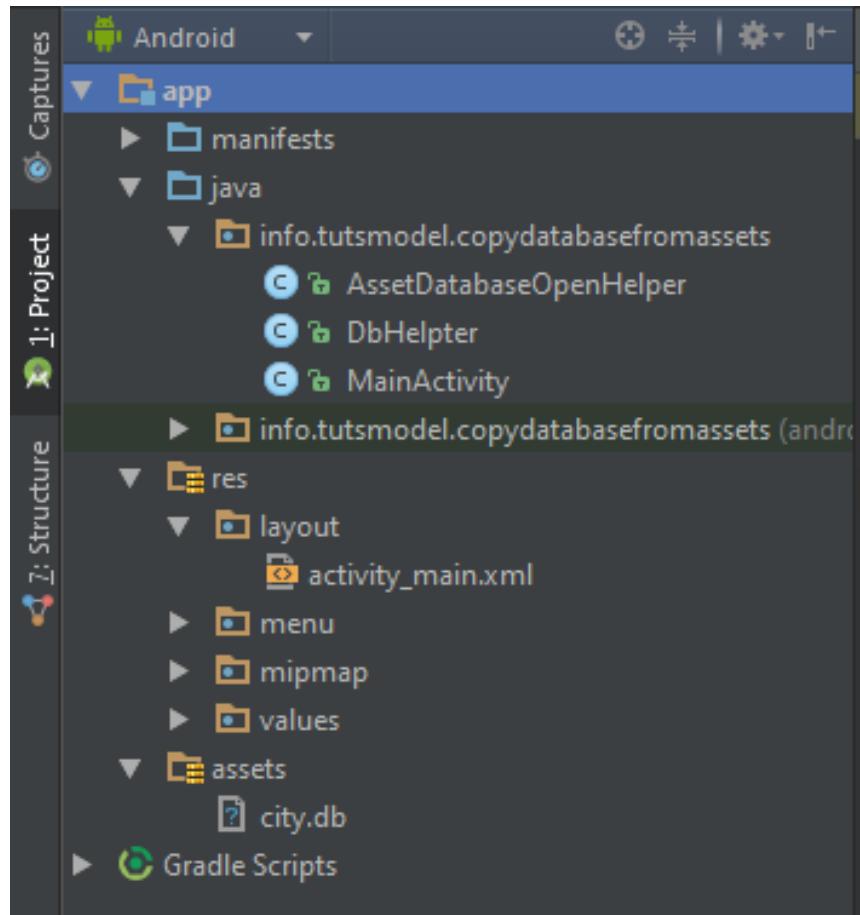
- Tạo cơ sở dữ liệu với công cụ thay vì định nghĩa thủ công lược đồ CSDL
- Copy CSDL đã tạo từ thư mục Assets vào bộ nhớ thiết bị
- Cấp quyền đọc, ghi file vào bộ nhớ thiết bị ở AndroidManifest.xml
- Đọc, thêm, sửa, xóa dữ liệu trong ứng dụng như cách 1

# Tạo CSDL Sqlite với công cụ

- Sqlite Expert
- Navicat
- ...



# Copy CSDL ngoài vào thư mục assets của project



# Copy CSDL từ assets vào bộ nhớ trong

```
public class AssetDatabaseOpenHelper {
    // tên file database trong thư mục assets
    private static final String DB_NAME = "city.db";
    private Context context;

    public AssetDatabaseOpenHelper(Context context) {
        this.context = context;
    }

    public SQLiteDatabase StoreDatabase() {
        ///data/data/{package của bạn}/databases
        String path = "/data/data/info.tutsmodel.copydatabasefromassets";
        File pathDb = new File(path);
        try {
            // kiểm tra nếu có thư mục databases thì sẽ tạo
            if (!pathDb.exists()){
                pathDb.mkdir();
            }
            // kiểm tra file đó chưa tồn tại thì copy vào
            // điều kiện này tránh trường hợp mỗi lần mở ứng dụng sẽ
            if (!new File(path + "/" + DB_NAME).exists()) {
                copy(path);
            }
        } catch (IOException e) {
            Log.d("IOException", e.getMessage());
        }
        Log.i("DB",context.getDatabasePath(DB_NAME).getPath());
        return SQLiteDatabase.openDatabase(context.getDatabasePath(DB_NAME), null, 0);
    }

    /**
     * Thực hiện copy file
     * @param path Đường dẫn thư mục đến
     * @throws IOException
     */
    private void copy(String path) throws IOException {
        // code
    }
}
```

# Cấp quyền đọc, ghi file vào bộ nhớ thiết bị

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="info.tutsmodel.copydatabasefromassets" >
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

# **BTVN**

## **1/ Tạo ứng dụng cho phép cài đặt ngôn ngữ**

- Lần đầu tiên cho phép cài đặt ngôn ngữ (English, Tiếng Việt)
- Lần tiếp theo không hỏi lại và hiển thị nội dung theo ngôn ngữ đã cài đặt trước đó

## **2/ Tạo ứng dụng quản lý sách có các chức năng**

- Đăng nhập, đăng ký
- Xem danh sách chủng loại sách
- Thêm, sửa, xóa sách
- Tìm kiếm sách theo tên, chủng loại

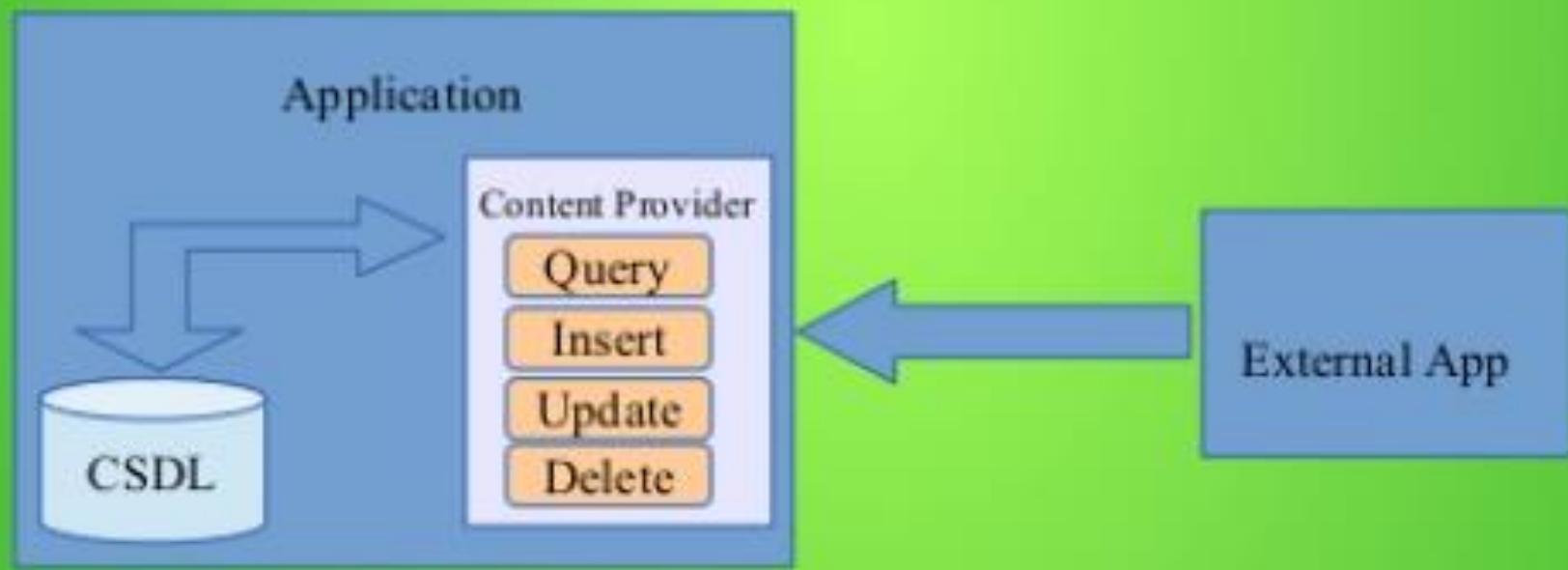
# Content provider

- BTVN cá nhân:
- Tìm hiểu về Content provider: cơ chế, demo

# Giới thiệu Content Provider

- CSDL SQLite chỉ dùng trong ứng dụng. Để chia sẻ dữ liệu này cho các ứng dụng khác cần dùng Content Provider
- Content Provider là 1 cơ chế cho phép truy xuất (truy vấn, thêm, sửa, xóa) dữ liệu (tập tin, csdl) của 1 ứng dụng từ 1 ứng dụng khác
- Ví dụ:
  - Truy xuất lịch sử các cuộc gọi
  - Truy xuất bookmark của web browser
  - Truy xuất sổ địa chỉ
  - Truy xuất vào lịch

STT	Tên	Mô tả
1	query	Truy vấn dữ liệu, trả kết quả về qua Cursor.
2	insert	Thêm dữ liệu
3	update	Cập nhật dữ liệu
4	delete	Xoá dữ liệu



# Hiện thực Content Provider

1. Định nghĩa data model
2. Định nghĩa URI (Uniform Resource Identifier)
3. Đăng ký provider vào AndroidManifest
4. Hiện thực các phương thức trừu tượng

# 1. Định nghĩa data model

- Sqlite database
- Extend ContentProvider class

```
public class MyProvider extends ContentProvider {  
    ...}
```

## 2. Định nghĩa URI

content://com.example.transportationprovider/trains/122

VD:

- content://media/internal/images
- content://contacts/people
- content://contacts/people/45

### 3. Đăng ký provider vào AndroidManifest

```
<provider  
    android:authorities="com.example.contentprovider"  
    android:name="com.example.contentprovider.MyProvider" >  
</provider>
```

## 4. Hiện thực các phương thức trừu tượng

- onCreate() //tạo đối tượng SqliteOpenDatabase
- insert(uri,...)
- query(uri,...)
- update(uri,...)
- delete(uri,...)

## ▪ Đọc dữ liệu từ các content provider có sẵn

```
String [] projection=new String[]{Calls.DATE,Calls.NUMBER,Calls.DURATION};  
Cursor c=getContentResolver().query(CallLog.Calls.CONTENT_URI,projection,  
        Calls.DURATION+"<?",new String[]{"30"},Calls.DATE +" Asc");  
c.moveToFirst();  
String s="";  
while(c.isAfterLast()==false){  
    for(int i=0;i<c.getColumnCount();i++){  
        s+=c.getString(i)+" - ";}  
    s+="\n";  
    c.moveToNext();}  
c.close();  
Toast.makeText(this, s, Toast.LENGTH_LONG).show();  
}
```

- **Đọc dữ liệu contacts từ các content provider vào listview**

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
```

```
Uri uri=Uri.parse("content://contacts/people");
ArrayList<String> list=new ArrayList<String>();
Cursor c1=getContentResolver().query(uri, null, null, null, null);
c1.moveToFirst();
while(c1.isAfterLast()==false){
    String s="";
    String idColumnName=ContactsContract.Contacts._ID;
    int idIndex=c1.getColumnIndex(idColumnName);
    s+=c1.getString(idIndex)+" - ";
    String nameColumnName=ContactsContract.Contacts.DISPLAY_NAME;
    int nameIndex=c1.getColumnIndex(nameColumnName);
    s+=c1.getString(nameIndex);
    c1.moveToNext();
    list.add(s);}
c1.close();
```



# BTVN

- Xây dựng ứng dụng lấy danh sách liên hệ được truy xuất từ ứng dụng danh bạ có sẵn trên máy.

## **4. Lập trình kết nối mạng**

# Các kết nối mạng mà Android hỗ trợ

- Wifi.
- Bluetooth.
- 2G (GSM).
- 2.5G (GPRS).
- 3G.
- 4G (LTE, LTE-A,...).
- ...

Android cung cấp lớp `ConnectivityManager` cho phép ứng dụng quản lý kết nối mạng.

# 4.1. Quản lý kết nối wifi

## Cấp quyền cho ứng dụng trong AndroidManifest.xml

- Cấp quyền truy cập các thông tin về mạng wifi  
`<uses-permission  
    android:name="android.permission.ACCESS_WIFI_STATE" />`
- Cấp quyền thay đổi trạng thái kết nối của wifi  
`<uses-permission  
    android:name="android.permission.CHANGE_WIFI_STATE"/>`

# Quản lý WiFi

- class **android.net.wifi.WifiManager**

```
WifiManager wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
```

- Cấu hình kết nối wifi
- Giám sát và chỉnh sửa cài đặt cho kết nối wifi đã có
- Quản lý kết nối wifi hiện thời
- Quét điểm truy cập wifi xung quanh thiết bị

# Kích hoạt hoặc hủy bỏ kết nối WiFi

wifi.setWifiEnabled(boolean);

# Lấy thông tin kết nối wifi hiện thời

```
WifiInfo info = wifi.getConnectionInfo();
if(info.getBSSID() != null){
    int strength = WifiManager.calculateSignalLevel(info.getRssi(), 5);
    int speed = info.getLinkSpeed();
    String units = WifiInfo.LINK_SPEED_UNITS;
    String ssid = info.getSSID();
    Toast.makeText(this, "ssid=" + ssid + ";strength=" + strength + ";speed=" +
        speed + " " + units , Toast.LENGTH_LONG).show();
}
```

# Quyết những điểm truy cập wifi xung quanh thiết bị

```
registerReceiver(new BroadcastReceiver(){
    public void onReceive(Context context, Intent intent) {
        List<ScanResult> results = wifi.getScanResults();
        ScanResult bestSignal = null;
        for(ScanResult result: results){
            if(bestSignal == null ||
               WifiManager.compareSignalLevel(bestSignal.level, result.level) < 0)
                bestSignal = result;
        }
    }, new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
    wifi.startScan();
}
```

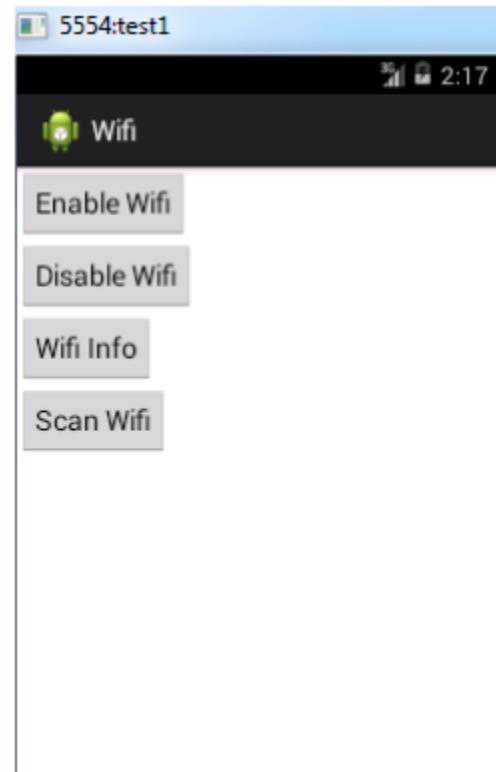
# Quản lý thông tin cấu hình wifi

- Lấy về thông tin cấu hình của mạng wifi mà thiết bị kết nối tới qua đối tượng WifiManager
- Thông tin cấu hình mạng được lưu trữ trong đối tượng WifiConfiguration gồm:
  - BSSID của điểm truy cập wifi.
  - SSID của một mạng cụ thể.
  - networkId : định danh mạng.
  - priority: mức độ ưu tiên của cấu hình mạng.
  - status: trạng thái hiện thời của mạng được kết nối tới nhận 1 trong số trạng thái sau: WifiConfiguration.Status.ENABLED,WifiConfiguration.Status.DISABLED, WifiConfiguration.Status.CURRENT.

```
// Get a list of available configurations
List<WifiConfiguration> configurations = wifi.getConfiguredNetworks();
// Get the network ID for the first one.
if (configurations.size() > 0) {
    int netID = configurations.get(0).networkId; // Enable that network.
    boolean disableAllOthers = true;
    wifi.enableNetwork(netID, disableAllOthers);
}
```

# BTVN cá nhân

- Tạo ứng dụng Android thực hiện các chức năng:
  - Liệt kê danh sách các điểm truy cập wifi quyết được theo dạng:  
SSID:..... – Strenght:..... – Speed: .....



## 4.2. Quản lý kết nối Bluetooth

- Bluetooth là giao thức hỗ trợ cho thiết bị ngang hàng (peer to peer) ở khoảng cách gần nhau có thể kết nối không dây được với nhau với băng thông thấp.

# Quản lý kết nối Bluetooth

- class **android.bluetooth**
- Cho phép ứng dụng kiểm tra, bật tắt, giám sát, tìm kiếm, truyền dữ liệu tới các thiết bị khác có hỗ trợ Bluetooth trong phạm vi cho phép

# Android.bluetooth

Lớp/ giao diện	Miêu tả
<a href="#"><u>BluetoothAdapter</u></a>	Cho phép thăm dò, truy vấn, lắng nghe kết nối bluetooth.
<a href="#"><u>BluetoothDevice</u></a>	Cho phép gửi yêu cầu, truy vấn thông tin của các thiết bị được kết nối tới thông qua bluetooth.
<a href="#"><u>BluetoothSocket</u></a>	<p>Để 2 thiết bị Android có thể kết nối với nhau thông qua bluetooth thì 1 thiết bị phải đóng vai trò là thiết bị nguồn gửi yêu cầu kết nối và 1 thiết bị đóng vai trò là thiết bị đích lắng nghe yêu cầu kết nối gửi tới thông qua bluetooth.</p> <p>Lớp này cho phép mở socket yêu cầu kết nối tới thiết bị đích và nhận hồi đáp trả về. Sau khi quá trình kết nối giữa thiết bị nguồn và đích được thiết lập, dữ liệu được truyền qua 2 thiết bị thông qua InputStream, OutputStream.</p>
<a href="#"><u>BluetoothServerSocket</u></a>	Lớp này cho phép mở socket lắng nghe kết nối và hồi đáp lại thiết bị nguồn thông qua kết nối bluetooth.

<u>BluetoothClass</u>	Cho biết thông tin đặc tả và khả năng của thiết bị hỗ trợ bluetooth.
<u>BluetoothProfile</u>	Cho biết thông tin đặc tả giao tiếp bluetooth giữa 2 thiết bị.
<u>BluetoothHeadset</u>	Cho phép bluetooth headset sử dụng được với điện thoại di động.
<u>BluetoothA2dp</u>	Định nghĩa chất lượng audio giữa 2 thiết bị kết nối với nhau thông qua bluetooth.
<u>BluetoothHealth</u>	Cho biết thông tin tình trạng của cổng điều khiển dịch vụ kết nối bluetooth.
<u>BluetoothHealthCallback</u>	Là lớp trừu tượng thực thi các phương thức callback từ BluetoothHealth. Ứng dụng khai báo 1 lớp kế thừa từ lớp này sẽ cài đặt các phương thức callback để nhận thông tin cập nhật về những thay đổi trạng thái đăng ký và trạng thái kênh kết nối bluetooth trong ứng dụng.
<u>BluetoothHealthAppConfiguration</u>	Cho biết cấu hình ứng dụng được phát triển bởi bên thứ 3 đăng ký giao tiếp với thiết bị khác thông qua kết nối bluetooth.
<u>BluetoothProfile.ServiceListener</u>	Dùng để thông báo khi thiết bị thiết lập hay ngắt kết nối với các dịch vụ thông qua kết nối bluetooth.

# Cấp quyền cho phép ứng dụng làm việc với Bluetooth

- Cho phép ứng dụng sử dụng các tính năng liên quan đến bluetooth  
`<uses-permission android:name="android.permission.BLUETOOTH"/>`
- Cho phép ứng dụng tìm kiếm và kết nối bluetooth  
`<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>`

# Kiểm tra thiết bị có hỗ trợ bluetooth hay không

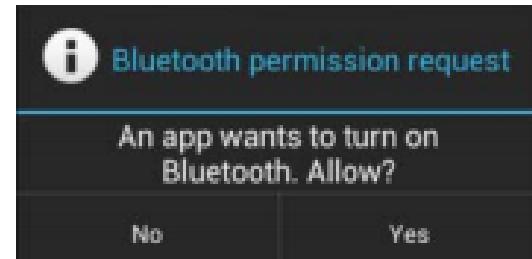
```
public class MainActivity extends Activity {  
    BluetoothAdapter bluetoothAdapter;  
    private boolean BluetoothAvailable() {  
        if (bluetoothAdapter == null) return false;  
        else return true; }  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();  
        Toast.makeText(this, "Bluetooth available: " + BluetoothAvailable(),  
            Toast.LENGTH_LONG).show(); }  
}
```

# Bật/tắt bluetooth trên thiết bị

```
private void turnOn(){
    if (!bluetoothAdapter.isEnabled()) {
        Intent enableBtIntent = new
            Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBtIntent, 0); }}

private void setDiscoverable(){
    if (!bluetoothAdapter.isDiscovering()) {
        Intent enableBtIntent = new
            Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
        startActivityForResult(enableBtIntent, 0); }}

private void turnOff(){
    bluetoothAdapter.disable(); }}
```

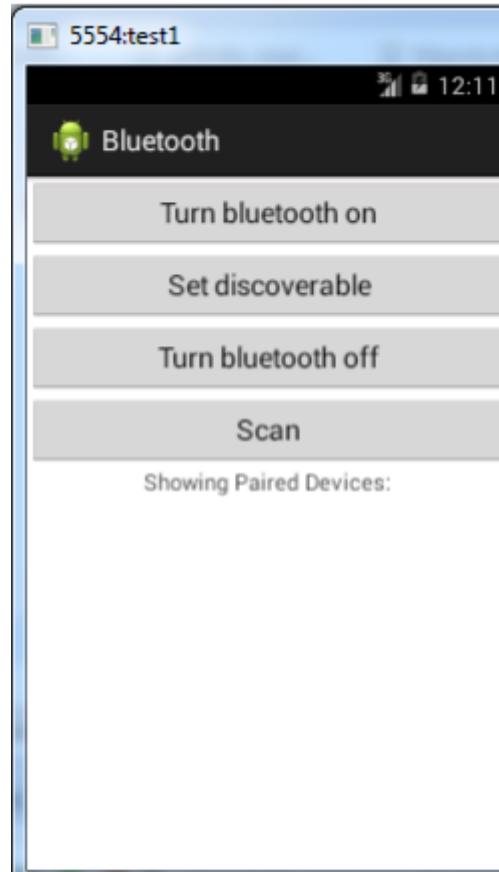


# Scan các cặp thiết bị có kết nối bluetooth

```
if (bluetoothAdapter.isEnabled()) {  
    TextView txtShowDevice = (TextView) findViewById(R.id.txtShowDevice);  
    txtShowDevice.append("\nPaired Devices are:");  
    Set<BluetoothDevice> devices = bluetoothAdapter.getBondedDevices();  
    for (BluetoothDevice device : devices) {  
        txtShowDevice.append("\n Device: " + device.getName());  
    }  
}
```

# BTVN cá nhân

- Tạo ứng dụng Android scan các thiết bị có kết nối bluetooth và hiển thị danh sách các thiết bị đó



## **5. Lập trình chức năng cuộc gọi**

# APIs Telephony

- Android cung cấp 1 tập API telephony cho phép ứng dụng thực hiện, xử lý cuộc gọi và giám sát những thông tin liên quan tới cuộc gọi ngay trong ứng dụng.

# Lập trình chức năng cuộc gọi

1. Cấp quyền cho ứng dụng sử dụng tính năng cuộc gọi
2. Kiểm tra thiết bị có hỗ trợ các phần cứng cho tính năng cuộc gọi
3. Khởi tạo cuộc gọi đi từ ứng dụng
4. Truy xuất những thông tin liên quan tới cuộc gọi
5. Giám sát quá trình thay đổi trạng thái cuộc gọi
6. Theo dõi sự thay đổi vị trí thiết bị
7. Theo dõi sự thay đổi dịch vụ mạng
8. Giám sát các thông tin liên quan tới quá trình truyền dữ liệu di động
9. Chặn cuộc gọi đi
10. Tự động trả lời cuộc gọi đến
11. Hiển thị lịch sử cuộc gọi

# 1/ Cấp quyền cho ứng dụng sử dụng tính năng cuộc gọi

- AndroidManifest.xml

```
<uses-feature android:name="android.hardware.telephony"  
    android:required="true"/>
```

## 2/ Kiểm tra thiết bị có hỗ trợ các phần cứng cho tính năng cuộc gọi

```
PackageManager pm = getPackageManager();
```

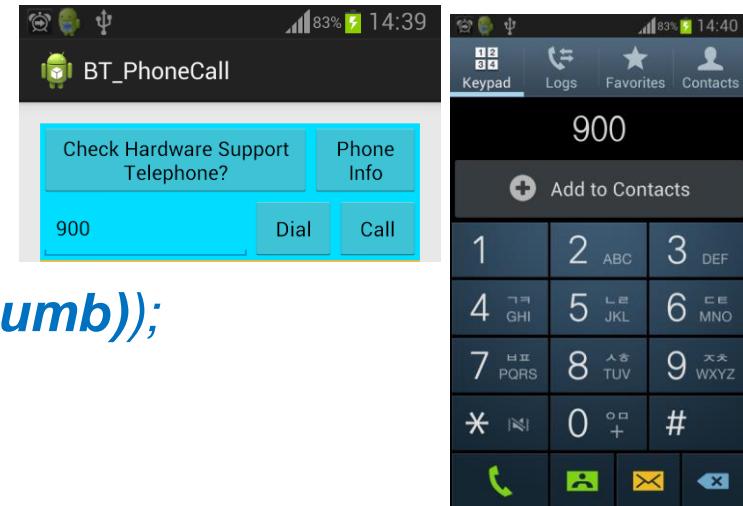
```
boolean telephonySupported = pm.hasSystemFeature(PackageManager.FEATURE_TELEPHONY);  
boolean gsmSupported = pm.hasSystemFeature(PackageManager.FEATURE_TELEPHONY_GSM);  
boolean cdmaSupported = pm.hasSystemFeature(PackageManager.FEATURE_TELEPHONY_CDMA);
```

# 3/ Khởi tạo cuộc gọi đi từ ứng dụng

## Native Dialer:

```
String numb = phonenum.getText().toString();
```

```
Intent i = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:"+numb));  
startActivity(i);
```



## Directly Call:

```
Require? => <uses-permission android:name="android.permission.CALL_PHONE"/>
```

```
Intent i = new Intent(Intent.ACTION_CALL, Uri.parse("tel:"+numb));  
startActivity(i);
```



## 4/ Truy xuất những thông tin liên quan tới cuộc gọi

- Thông tin liên quan tới cuộc gọi: các thông tin về thiết bị, mạng lưới, SIM, trạng thái cuộc gọi.
- **Cấp quyền cho ứng dụng đọc các thông tin liên quan tới cuộc gọi (AndroidManifest.xml)**

```
<uses-permission  
    android:name="android.permission.READ_PHONE_STATE"/>
```

- **Tạo đối tượng TelephonyManager chứa các thông tin liên quan tới cuộc gọi**

```
String srvcName = Context.TELEPHONY_SERVICE;  
TelephonyManager telephonyManager =  
(TelephonyManager) getSystemService(srvcName);
```

# Lấy về thông tin của thiết bị liên quan tới cuộc gọi (1)

- **Lấy về thông tin loại mạng thiết bị kết nối tới (GSM / CDMA / SIP)**

```
String phoneTypeStr = "unknown";
int phoneType = telephonyManager.getPhoneType();
switch (phoneType) {
    case (TelephonyManager.PHONE_TYPE_CDMA):
        phoneTypeStr = "CDMA";
        break;
    case (TelephonyManager.PHONE_TYPE_GSM) :
        phoneTypeStr = "GSM";
        break;
    case (TelephonyManager.PHONE_TYPE_SIP):
        phoneTypeStr = "SIP";
        break;
    case (TelephonyManager.PHONE_TYPE_NONE):
        phoneTypeStr = "None";
        break;
    default: break;
```

# Lấy về thông tin của thiết bị liên quan tới cuộc gọi (2)

- **Lấy về thông tin định danh thiết bị (IMEI / MEID), phiên bản phần mềm, số điện thoại di động đang dùng cho thiết bị**

```
// -- These require READ_PHONE_STATE uses-permission --
// Read the IMEI for GSM or MEID for CDMA
String deviceId = telephonyManager.getDeviceId();
// Read the software version on the phone (note -- not the SDK version)
String softwareVersion = telephonyManager.getDeviceSoftwareVersion();
// Get the phone's number (if available)
String phoneNumber = telephonyManager.getLine1Number();
```

# Lấy về thông tin của thiết bị liên quan tới cuộc gọi (3)

- **Lấy về thông tin mạng mà thiết bị kết nối tới**

```
// Get connected network country ISO code  
String networkCountry = telephonyManager.getNetworkCountryIso();  
  
// Get String the connected network operator ID (MCC + MNC)  
networkOperatorId = telephonyManager.getNetworkOperator();  
  
// Get String the connected network operator name  
networkName = telephonyManager.getNetworkOperatorName();  
  
// Get the type of network you are connected to  
int networkType = telephonyManager.getNetworkType();  
  
  
switch (networkType) {  
    case (TelephonyManager.NETWORK_TYPE_1xRTT) : [... do something ...]  
    break;  
    case (TelephonyManager.NETWORK_TYPE_CDMA) : [... do something ...]  
    break;  
    case (TelephonyManager.NETWORK_TYPE_EDGE) : [... do something ...]  
    break;  
    case (TelephonyManager.NETWORK_TYPE_EHRPD) : [... do something ...]  
    break;  
    case (TelephonyManager.NETWORK_TYPE_EVDO_0) : [... do something ...]  
}
```

# Lấy về thông tin của thiết bị liên quan tới cuộc gọi (4)

- **Lấy về thông tin SIM**

```
int simState = telephonyManager.getSimState();
switch (simState) {
    case (TelephonyManager.SIM_STATE_ABSENT): break;
    case (TelephonyManager.SIM_STATE_NETWORK_LOCKED): break;
    case (TelephonyManager.SIM_STATE_PIN_REQUIRED): break;
    case (TelephonyManager.SIM_STATE_PUK_REQUIRED): break;
    case (TelephonyManager.SIM_STATE_UNKNOWN): break;
    case (TelephonyManager.SIM_STATE_READY): {
        // Get the SIM country ISO code
        String simCountry = telephonyManager.getSimCountryIso();
        // Get the operator code of the active SIM (MCC + MNC)
        String simOperatorCode = telephonyManager.getSimOperator();
        // Get the name of the SIM operator
        String simOperatorName = telephonyManager.getSimOperatorName();
        // -- Requires READ_PHONE_STATE uses-permission --
        // Get the SIM's serial number
        String simSerial = telephonyManager.getSimSerialNumber();
        break;
    }
    default: break;
}
```

# Lấy về thông tin của thiết bị liên quan tới cuộc gọi (5)

- Đọc thông tin trạng thái kết nối và truyền dữ liệu cuộc gọi

```
int dataActivity = telephonyManager.getDataActivity();
int dataState = telephonyManager.getDataState();
switch (dataActivity) {
    case TelephonyManager.DATA_ACTIVITY_IN : break;
    case TelephonyManager.DATA_ACTIVITY_OUT : break;
    case TelephonyManager.DATA_ACTIVITY_INOUT : break;
    case TelephonyManager.DATA_ACTIVITY_NONE : break;
}
switch (dataState) {
    case TelephonyManager.DATA_CONNECTED : break;
    case TelephonyManager.DATA_CONNECTING : break;
    case TelephonyManager.DATA_DISCONNECTED : break;
    case TelephonyManager.DATA_SUSPENDED : break;
}
```

# 5/ Giám sát quá trình thay đổi trạng thái cuộc gọi (1)

- **Giám sát sự thay đổi trạng thái cuộc gọi đến khi ứng dụng đang chạy** (AndroidManifest.xml)

```
<uses-permission  
    android:name="android.permission.READ_PHONE_STATE"/>
```

- **Giám sát các trạng thái liên quan tới cuộc gọi** (như : khi có cuộc gọi đến, vị trí thiết bị thay đổi, dịch vụ thay đổi, các thông tin liên quan tới quá trình truyền dữ liệu)

```
telephonyManager.listen(phoneStateListener,  
    PhoneStateListener.LISTEN_CALL_FORWARDING_INDICATOR |  
    PhoneStateListener.LISTEN_CALL_STATE |  
    PhoneStateListener.LISTEN_CELL_LOCATION |  
    PhoneStateListener.LISTEN_DATA_ACTIVITY |  
    PhoneStateListener.LISTEN_DATA_CONNECTION_STATE |  
    PhoneStateListener.LISTEN_MESSAGE_WAITING_INDICATOR |  
    PhoneStateListener.LISTEN_SERVICE_STATE |  
    PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
```

# 5/ Giám sát quá trình thay đổi trạng thái cuộc gọi (2)

- **Giám sát sự thay đổi trạng thái cuộc gọi đến khi ứng dụng đang chạy**

```
PhoneStateListener callStateListener = new PhoneStateListener() {  
    public void onCallStateChanged(int state, String incomingNumber) {  
        String callStateStr = "Unknown";  
        switch (state) {  
            case TelephonyManager.CALL_STATE_IDLE :  
                callStateStr = "idle"; break;  
            case TelephonyManager.CALL_STATE_OFFHOOK :  
                callStateStr = "offhook"; break;  
            case TelephonyManager.CALL_STATE_RINGING :  
                callStateStr = "ringing. Incoming number is: "  
                + incomingNumber;  
                break;  
            default : break;  
        }  
        Toast.makeText(MyActivity.this, callStateStr, Toast.LENGTH_LONG).show();  
    }  
};  
telephonyManager.listen(callStateListener,PhoneStateListener.LISTEN_CALL_STATE);
```

# 5/ Giám sát quá trình thay đổi trạng thái cuộc gọi (3)

- Giám sát sự thay đổi trạng thái cuộc gọi đến ở chế độ nền khi ứng dụng không đang chạy

- Cấp quyền READ\_PHONE\_STATE trong AndroidManifest.xml

```
<uses-permission  
    android:name="android.permission.READ_PHONE_STATE"/>
```

- Đăng ký Intent Receiver trong <application> để lắng nghe Broadcast Intent:

```
<receiver android:name="PhoneStateChangedReceiver">  
    <intent-filter>  
        <action android:name="android.intent.action.PHONE_STATE"></action>  
    </intent-filter>  
    </receiver>
```

# 5/ Giám sát quá trình thay đổi trạng thái cuộc gọi (3)

- **Giám sát sự thay đổi trạng thái cuộc gọi đến ở chế độ nền khi ứng dụng không đang chạy**
  - Tạo ra 1 lớp Java kế thừa từ lớp cơ sở BroadcastReceiver (PhoneStateReceiver.java)
    - Cài đặt inner class cài đặt thực thi lớp PhoneStateListener
    - Cài đặt nạp chồng phương thức **onReceive()** của lớp BroadcastReceiver
      - Gọi phương thức **listen()** của lớp TelephonyManager để giám sát sự thay đổi trạng thái cuộc gọi.

# 6/ Theo dõi sự thay đổi vị trí thiết bị (1)

- Cấp quyền cho phép ứng dụng giám sát sự thay đổi trạng thái cuộc gọi khi vị trí thiết bị thay đổi

```
<uses-permission  
    android:name="android.permission.ACCESS_COARSE_LOCATION  
    "/>
```

# 6/ Theo dõi sự thay đổi vị trí thiết bị (2)

- **Lấy về thông báo khi vị trí thiết bị thay đổi trong ứng dụng**

```
PhoneStateListener cellLocationListener = new PhoneStateListener() {  
    public void onCellLocationChanged(CellLocation location) {  
        if (location instanceof GsmCellLocation) {  
            GsmCellLocation gsmLocation = (GsmCellLocation)location;  
            Toast.makeText(getApplicationContext(),  
                String.valueOf(gsmLocation.getCid()),  
                Toast.LENGTH_LONG).show();  
        }  
        else if (location instanceof CdmaCellLocation) {  
            CdmaCellLocation cdmaLocation = (CdmaCellLocation)location;  
            StringBuilder sb = new StringBuilder();  
            sb.append(cdmaLocation.getBaseStationId());  
            sb.append("\n@");  
            sb.append(cdmaLocation.getBaseStationLatitude());  
            sb.append(cdmaLocation.getBaseStationLongitude());  
            Toast.makeText(getApplicationContext(),  
                sb.toString(),  
                Toast.LENGTH_LONG).show();  
        }  
    };  
    telephonyManager.listen(cellLocationListener,  
        PhoneStateListener.LISTEN_CELL_LOCATION);  
};
```

## 7/ Theo dõi sự thay đổi dịch vụ mạng (1)

```
PhoneStateListener serviceStateListener = new
PhoneStateListener() {
public void onServiceStateChanged(ServiceState serviceState)
{
    if (serviceState.getState() == ServiceState.STATE_IN_SERVICE) {
        String toastText = "Operator: " +
        serviceState.getOperatorAlphaLong();
        Toast.makeText(MyActivity.this, toastText, Toast.LENGTH_SHORT);
    }
}
};

telephonyManager.listen(serviceStateListener,
PhoneStateListener.LISTEN_SERVICE_STATE);
```

# 7/ Theo dõi sự thay đổi dịch vụ mạng (2)

Trong đó:

- **onServiceStateChanged ()** có tham số truyền vào là đối tượng ServiceState chứa các thông tin chi tiết về trạng thái dịch vụ hiện thời.
- **getState()** của đối tượng ServiceState trả về trạng thái dịch vụ hiện thời nhận 1 trong các giá trị sau:
  - **STATE\_IN\_SERVICE**: cho biết dịch vụ mạng đang hoạt động bình thường.
  - **STATE\_EMERGENCY\_ONLY**: cho biết dịch vụ mạng chỉ được cung cấp cho những cuộc gọi khẩn cấp.
  - **STATE\_OUT\_OF\_SERVICE**: cho biết không có dịch vụ mạng nào đang sẵn sàng.
  - **STATE\_POWER\_OFF**: cho biết điện thoại đang không kết nối mạng nào (thường khi điện thoại ở chế độ máy bay).
- Danh sách các phương thức có dạng **getOperator\*()** trả về thông tin chi tiết nhà mạng cung cấp dịch vụ mạng tới thiết bị.

## 8/ Giám sát các thông tin liên quan tới quá trình truyền dữ liệu di động (1)

- Để giám sát quá trình kết nối cũng như chuyển tiếp dữ liệu qua mạng di động, Android cung cấp trình quản lý kết nối (Connectivity Manager).
- Android cung cấp lớp PhoneStateListener với 2 sự kiện xử lý kết nối dữ liệu của thiết bị:
  - **onDataActivity()** để theo dõi hoạt động truyền dữ liệu
  - **onDataConnectionStateChanged()** để thông báo trạng thái kết nối dữ liệu thay đổi.

# 8/ Giám sát các thông tin liên quan tới quá trình truyền dữ liệu di động (2)

```
PhoneStateListener dataStateListener = new PhoneStateListener() {  
    public void onDataActivity(int direction) {  
        String dataActivityStr = "None";  
        switch (direction) {  
            case TelephonyManager.DATA_ACTIVITY_IN :  
                dataActivityStr = "Downloading"; break;  
            case TelephonyManager.DATA_ACTIVITY_OUT :  
                dataActivityStr = "Uploading"; break;  
            case TelephonyManager.DATA_ACTIVITY_INOUT :  
                dataActivityStr = "Uploading/Downloading"; break;  
            case TelephonyManager.DATA_ACTIVITY_NONE :  
                dataActivityStr = "No Activity"; break;  
        }  
        Toast.makeText(MyActivity.this, "Data Activity is " + dataActivityStr, Toast.LENGTH_LONG).show();  
    }  
};  
telephonyManager.listen(dataStateListener,  
PhoneStateListener.LISTEN_DATA_ACTIVITY);
```

# 8/ Giám sát các thông tin liên quan tới quá trình truyền dữ liệu di động (2)

```
PhoneStateListener dataStateListener = new PhoneStateListener() {  
    public void onDataConnectionStateChanged(int state) {  
        String dataStateStr = "Unknown";  
        switch (state) {  
            case TelephonyManager.DATA_CONNECTED :  
                dataStateStr = "Connected"; break;  
            case TelephonyManager.DATA_CONNECTING :  
                dataStateStr = "Connecting"; break;  
            case TelephonyManager.DATA_DISCONNECTED :  
                dataStateStr = "Disconnected"; break;  
            case TelephonyManager.DATA_SUSPENDED :  
                dataStateStr = "Suspended"; break;  
        }  
        Toast.makeText(MyActivity.this, "Data Connectivity is " + dataStateStr, Toast.LENGTH_LONG).show();  
    }  
};  
telephonyManager.listen(dataStateListener,  
PhoneStateListener.LISTEN_DATA_CONNECTION_STATE);
```

# 9/ Chặn cuộc gọi đi (1)

- Cấp quyền **PROCESS\_OUTGOING\_CALLS** cho ứng dụng xử lý cuộc gọi đi và khai báo **receiver** trong **AndroidManifest.xml** như sau:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.blockoutgoingcall"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <uses-permission
        android:name="android.permission.PROCESS_OUTGOING_CALLS"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
                />
            </intent-filter>
        </activity>

        <receiver android:name=".OutgoingCallsReceiver" >
            <intent-filter android:priority="0">
                <action android:name=
                    "android.intent.action.NEW_OUTGOING_CALL" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

# 9/ Chặn cuộc gọi đi (2)

- Tạo 1 lớp Java kế thừa từ lớp **BroadcastReceiver** trong đó cài đặt nạp chồng phương thức **onReceive()**, phương thức này được gọi tới khi điện thoại phát 1 cuộc gọi đi.

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class OutgoingCallsReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String outgoingNumber =
        intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER).toString();
        if (outgoingNumber.contentEquals("1234567")) {
            setResultData(null);
            Toast.makeText(context, "This call is not allowed!", Toast.LENGTH_LONG).show();
        }
    }
}
```

# 10/ Tự động trả lời cuộc gọi đến (1)

- Cấp quyền cho ứng dụng trong AndroidManifest.xml

```
manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.autocalls"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <uses-permission
        android:name="android.permission.READ_PHONE_STATE" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".IncomingCallsReceiver" >
            <intent-filter>
                <action
                    android:name="android.intent.action.PHONE_STATE" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

# 10/ Tự động trả lời cuộc gọi đến (2)

- Tạo 1 lớp Java kế thừa từ lớp BroadcastReceiver trong đó cài đặt nạp chồng phương thức **onReceive()**, phương thức này được gọi tới khi có 1 cuộc gọi đến.

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.TelephonyManager;
import android.view.KeyEvent;

public class IncomingCallsReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (!intent.getAction().equals(
                "android.intent.action.PHONE_STATE")) return;
        String extraState =
                intent.getStringExtra(TelephonyManager.EXTRA_STATE);

        if (extraState.equals(TelephonyManager.EXTRA_STATE_RINGING)) {
            String incomingNumber =
                    intent.getStringExtra(TelephonyManager.EXTRA_INCOM-
ING_NUMBER);

            if (incomingNumber.contentEquals("1234567")) {
                //---answer the call---
                Intent i = new Intent(Intent.ACTION_MEDIA_BUTTON);
                i.putExtra(Intent.EXTRA_KEY_EVENT,
                        new KeyEvent(KeyEvent.ACTION_UP,
                                KeyEvent.KEYCODE_HEADSETHOOK));
                context.sendOrderedBroadcast(i, null);
            }
        }
        return;
    }
}
```

# 11/ Hiển thị lịch sử cuộc gọi

- Cấp quyền READ\_CONTACTS cho ứng dụng trong AndroidManifest.xml như sau:

```
<uses-permission  
    android:name="android.permission.READ_CONTACTS"/>
```

- Hiển thị thông tin lịch sử cuộc gọi sắp xếp theo ngày tháng (ID, số điện thoại, thời gian phát sinh cuộc gọi, loại cuộc gọi) sử dụng đối tượng Cursor

# BTVN

1. Khởi tạo cuộc gọi đi từ ứng dụng
2. Truy xuất những thông tin liên quan tới cuộc gọi
3. Giám sát quá trình thay đổi trạng thái cuộc gọi
4. Giám sát các thông tin liên quan tới quá trình truyền dữ liệu di động
5. Chặn cuộc gọi đi
6. Tự động trả lời cuộc gọi đến
7. Hiển thị lịch sử cuộc gọi

## **6. Lập trình chức năng tin nhắn ngắn**

# SMS

- SMS (Short Messaging Service) là công nghệ gửi tin nhắn ngắn giữa các điện thoại di động.
- Tin nhắn ngắn tồn tại dưới 2 dạng:
  - Dạng văn bản người dùng đọc được
  - Dạng dữ liệu (như dạng nhị phân) dùng để truyền tín hiệu trao đổi giữa các ứng dụng.

### 3.6 Lập trình chức năng nhắn tin ngắn SMS

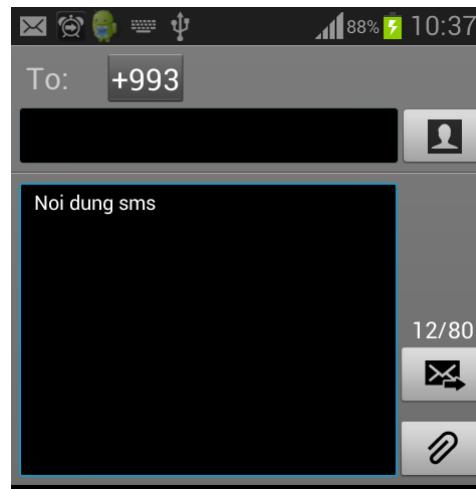
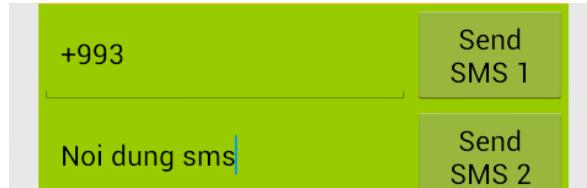
- Làm thế nào để gửi tin nhắn từ thông qua ứng dụng có sẵn (build-in Messaging).
- Làm thế nào để gửi tin nhắn từ chính ứng dụng.
- Làm thế nào để giám sát trạng thái tin nhắn.
- ...

### 3.6 Lập trình chức năng nhắn tin ngắn SMS

Gửi tin nhắn thông qua ứng dụng có sẵn (build-in Messaging).

– How? =>

```
Intent smsIntent = new Intent(Intent.ACTION_SENDTO, Uri.parse("sms:55512345"));
smsIntent.putExtra("sms_body", "Press send to send me");
startActivity(smsIntent);
```



# 3.6 Lập trình chức năng nhắn tin ngắn SMS

Gửi tin nhắn thông qua ứng dụng có sẵn (build-in Messaging).

– How? =>

Or

*Intent i = new*

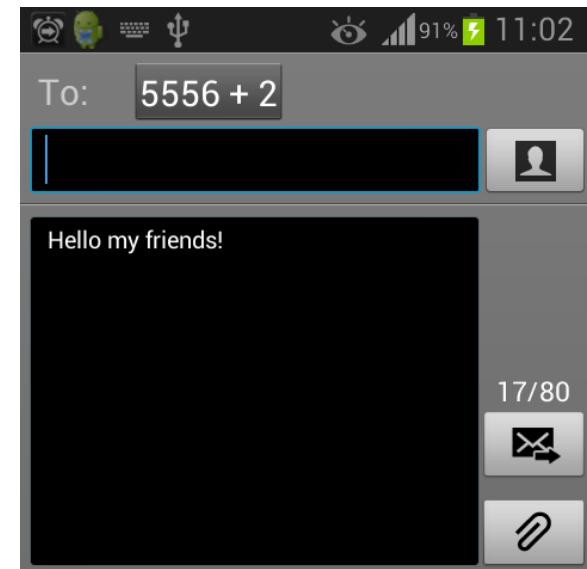
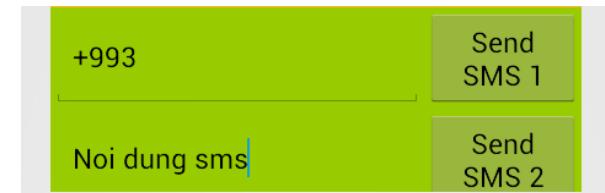
*Intent(android.content.Intent.ACTION\_VIEW);*

*i.putExtra("address", "5556; 5558; 5560");*

*i.putExtra("sms\_body", "Hello my friends!");*

*i.setType("vnd.android-dir/mms-sms");*

*startActivity(i);*



### 3.6 Lập trình chức năng nhắn tin ngắn SMS

Gửi tin nhắn từ chính ứng dụng (không sử dụng build-in Messaging).

- Require? =>

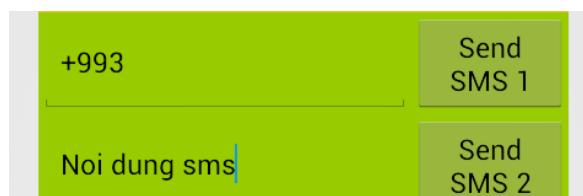
```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

- How? =>

```
SmsManager smsManager = SmsManager.getDefault();
```

```
sms.sendTextMessage(destinationAddress, scAddress, text, sentIntent, deliveryIntent);
```

Ví dụ: `smsManager.sendTextMessage(smsSendNumber.getText().toString(), null, smsContent.getText().toString(), sentPI, deliveredPI);`



## 3.6 Lập trình chức năng nhắn tin ngắn SMS

Gửi tin nhắn từ chính ứng dụng (không sử dụng build-in Messaging).

```
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(destinationAddress, scAddress, text, sentIntent, deliveryIntent);
```

Trong đó:

- **destinationAddress** — Số điện thoại cần gửi.
- **scAddress** — Số Service Center; sử dụng null cho default SMSC.
- **text** — Nội dung của tin nhắn.
- **sentIntent** — tham số **Pending intent** này cho biết trạng thái gửi tin từ bên gửi (có thể nhận giá trị null).
- **deliveryIntent** — tham số **Pending intent** cho biết trạng thái nhận tin nhắn bởi bên nhận (có thể nhận giá trị null).

# 3.6 Lập trình chức năng nhắn tin ngắn SMS

Giám sát trạng thái tin nhắn:

Để giám sát trạng thái tin nhắn gửi đi cần đăng ký **Broadcast Receivers** lắng nghe các sự kiện với giá trị đặc tả được truyền vào tham số Pending Intents của phương thức **sendTextMessage()**.

Tham số **Pending Intent đầu tiên (sendIntent)** cho biết trạng thái tin nhắn gửi đi từ bên gửi là thành công hay thất bại thông qua 1 trong các mã đặc tả sau:

- **Activity.RESULT\_OK**: cho biết tin nhắn gửi thành công.
- **SmsManager.RESULT\_ERROR\_GENERIC\_FAILURE**: cho biết quá trình truyền tin nhắn lỗi.

### 3.6 Lập trình chức năng nhắn tin ngắn SMS

Giám sát trạng thái tin nhắn:

- **SmsManager.RESULT\_ERROR\_RADIO\_OFF:** cho biết tin nhắn không truyền đi do không có kết nối mạng.
- **SmsManager.RESULT\_ERROR\_NULL\_PDU:** lỗi PDU (protocol description unit).
- **SmsManager.RESULT\_ERROR\_NO\_SERVICE:** cho biết dịch vụ mạng hiện thời không sẵn sàng.

## 3.6 Lập trình chức năng nhắn tin ngắn SMS

Giám sát trạng thái tin nhắn:

Tham số **Pending Intent thứ 2 (deliveryIntent)** cho biết trạng thái tin nhắn gửi đến bên nhận là thành công hay thất bại thông qua 1 trong 2 đặc tả sau:

- **Activity.RESULT\_OK**: cho biết bên nhận đã nhận được tin nhắn.
- **Activity.RESULT\_CANCELED**: cho biết bên nhận không nhận được tin nhắn.

# BTVN

- Xây dựng ứng dụng nhắn tin sms sử dụng
  - build-in Messaging
  - Không sử dụng build-in Messaging

# BTVN

## 1) Xây dựng ứng dụng :

- a. Lấy thông tin danh bạ trên máy hiển thị lên danh sách.
- b. Chọn 1 người dùng trong danh sách để :
  - Gọi điện trực tiếp từ ứng dụng
  - Gửi tin nhắn Không sử dụng build-in Messaging, có kèm giám sát trạng thái gửi nhận tin nhắn.

## **6. Lập trình xử lý đa phương tiện**

# Định dạng đa phương tiện

Image	Audio	Video	Giao thức
JPEG	AAC LC/LTP HE-AACv1	H.263 H.264 AVC	RTSP (RTP, SDP)
PNG	(AAC+) HE-AACv2	MPEG-4 SP	
WEBP(định dạng này bắt đầu được hỗ trợ từ phiên bản Android 4.0)	(AAC+ nâng cao) AMR-NB AMR-WB MP3 MIDI	VP8 (định dạng này bắt đầu được hỗ trợ từ phiên bản Android 2.3.3)	HTTP/HTTPS progressive streaming
GIF BMP	Ogg Vorbis PCM/WAVE		HTTP/HTTPS live streaming (giao thức này này bắt đầu được hỗ trợ từ phiên bản Android 3.0)
	FLAC (định dạng này bắt đầu được hỗ trợ từ phiên bản Android 3.1)		

# **6. Lập trình xử lý đa phương tiện**

## **6.1. Hiển thị hình ảnh**

# ImageView

```
<ImageView
```

```
    android:id="@+id/imgView"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/ic_launcher" />
```



```
image.setImageResource(R.drawable.flag);
```

# Gallery

- Hiện thị hình ảnh cuộn theo chiều ngang



# Gallery

- Sử dụng Gallery

```
Integer[] imageIDs = {R.drawable.flag,R.drawable.flag1,R.drawable.flag2,  
    R.drawable.flag3,R.drawable.flag4,R.drawable.flag5};  
Gallery gallery = (Gallery) findViewById(R.id.gallery1);  
gallery.setAdapter(new ImageAdapter(this));  
gallery.setOnItemClickListener(new OnItemClickListener() {  
    public void onItemClick(AdapterView<?> parent, View v, int position,  
        long id){  
        ImageView imageView = (ImageView)  
            findViewById(R.id.image1);  
        imageView.setImageResource(imageIDs[position]);  
    }  
});
```

# Gallery

- Khai báo Adapter

```
public class ImageAdapter extends BaseAdapter {  
    private Context context;  
    public ImageAdapter(Context c){context = c;}  
    public int getCount() {return imageIDs.length;}  
    public Object getItem(int position) {return position;}  
    public long getItemId(int position) {return position;}  
    public View getView(int position, View convertView, ViewGroup parent)  
    {  
        ImageView i= new ImageView(context);  
        i.setImageResource(imageIDs[position]);  
        i.setScaleType(ImageView.ScaleType.FIT_CENTER);  
        i.setLayoutParams(new Gallery.LayoutParams(200, 150));  
        return imageView;}}}
```

# ImageSwitcher

## Hiển thị hình ảnh kèm hiệu ứng

```
public class MainActivity extends Activity implements ViewFactory {  
    public View makeView() {  
        ImageView iView = new ImageView(this);  
        iView.setLayoutParams(new ImageSwitcher.LayoutParams  
            (LayoutParams.MATCH_PARENT,LayoutParams.MATCH_PARENT));  
        iView.setBackgroundColor(0xFF000000);  
        return iView;  
    }  
  
    imgSwitcher.setFactory(this);  
    imgSwitcher.setInAnimation(AnimationUtils.LoadAnimation(this,  
        android.R.anim.slide_in_left));  
    imgSwitcher.setOutAnimation(AnimationUtils.LoadAnimation(this,  
        android.R.anim.slide_out_right));
```

# GridView

Hiển thị hình ảnh dưới dạng lưới

```
gridView.setNumColumns(3);
gridView.setAdapter(new ImageAdapter(this));
public View getView(int position, View convertView, ViewGroup
parent) {
    ImageView imageView;
    if (convertView == null) {
        imageView = new ImageView(context);
        imageView.setLayoutParams(new
            GridView.LayoutParams(85, 85));
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);
        imageView.setPadding(5, 5, 5, 5);
    } else {
        imageView = (ImageView) convertView;
        imageView.setImageResource(imageIDs[position]);
    }
    return imageView;
}
```

# BTVN

1. **Tạo ứng dụng xem ảnh cho phép:** xem danh sách hình ảnh dạng lưới (grid). Chọn 1 ảnh để xem chi tiết (phóng to) lên ImageView, đồng thời hiển thị Gallery danh sách hình ảnh cuộn theo chiều ngang phía dưới cho chọn xem chi tiết.

## **6. Lập trình xử lý đa phương tiện**

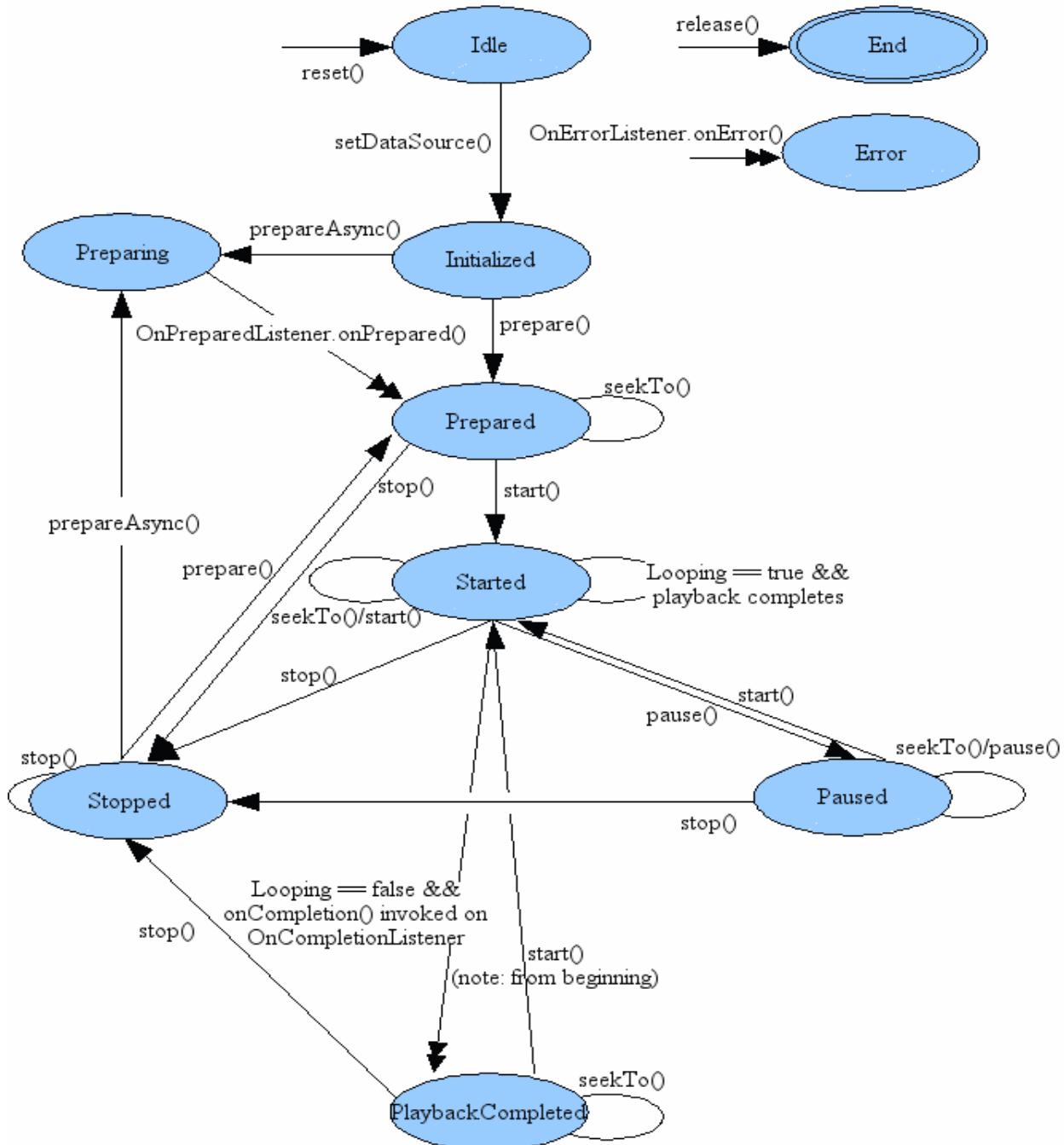
### **6.2. Lập trình xử lý audio, video**

# MediaPlayer

- Lớp cung cấp 1 tập hợp các API cho phép phát các file audio, video được lưu trữ trong:
  - Chính ứng dụng.
  - Trên hệ thống file của thiết bị.
  - Truyền tải trực tiếp từ trên mạng (streamming audio/ video).

# Quản lý phát audio và video thông qua MediaPlayer

- 1/ Khởi tạo đối tượng MediaPlayer
- 2/ Chuẩn bị phát audio/video
- 3/ Bắt đầu phát audio/video
- 4/ Tạm dừng hoặc ngừng phát audio/video
- 5/ Hoàn thành phát audio/video



# Phát audio trực tiếp từ mạng

```
<uses-permission android:name="android.permission.INTERNET" />
```

# Chuẩn bị phát audio

- File audio có thể được xác định thông qua:
  - Vị trí trực tiếp lưu trong project (thư mục res/raw)
  - URI của file trên hệ thống ([file://schema](#))
  - URI trực tuyến lưu file (URL)
  - URI Content Provider.
- Chuẩn bị audio: Cách 1

```
// Load an audio resource from a package resource.  
MediaPlayer resourcePlayer = MediaPlayer.create(this, R.raw.my_audio);  
  
// Load an audio resource from a local file.  
MediaPlayer filePlayer = MediaPlayer.create(this,  
    Uri.parse("file:///sdcard/localfile.mp3"));  
  
// Load an audio resource from an online resource.  
MediaPlayer urlPlayer = MediaPlayer.create(this,  
    Uri.parse("http://site.com/audio/audio.mp3"));  
  
// Load an audio resource from a Content Provider.  
MediaPlayer contentPlayer = MediaPlayer.create(this,  
    Settings.System.DEFAULT_RINGTONE_URI);
```

# Chuẩn bị phát audio

- File audio có thể được xác định thông qua:
  - Vị trí trực tiếp lưu trong project (thư mục res/raw)
  - URI của file trên hệ thống ([file://schema](#))
  - URI trực tuyến lưu file (URL)
  - URI Content Provider.
- Chuẩn bị audio: Cách 2

```
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setDataSource("/sdcard/mydopetunes.mp3");
mediaPlayer.prepare();
```

# Xử lý audio

- Phát audio

```
mediaPlayer.start();
```

- Tạm dừng

```
if(mediaPlayer != null){  
    mediaPlayer.pause();  
    length = mediaPlayer.getCurrentPosition();}
```

- Tiếp tục phát

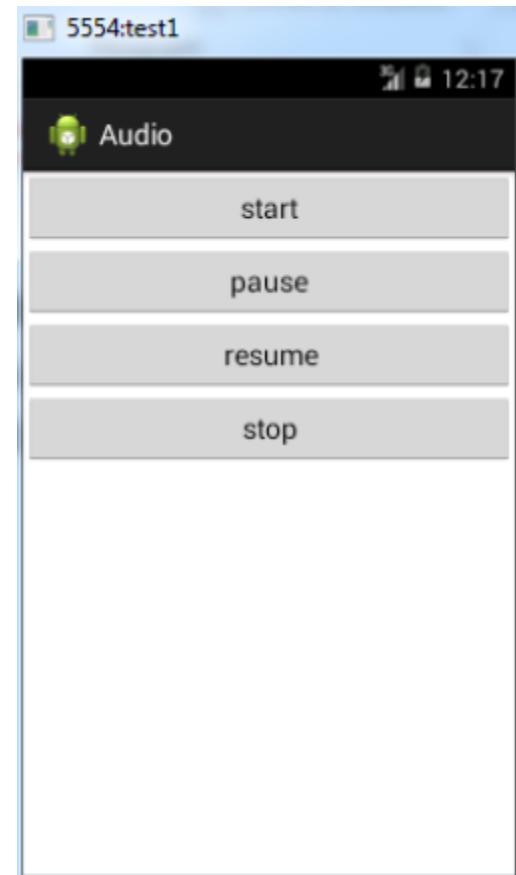
```
if(mediaPlayer != null && length>0){  
    mediaPlayer.seekTo(length);  
    mediaPlayer.start();}
```

- Dừng phát audio

```
if(mediaPlayer != null){  
    mediaPlayer.stop();  
    mediaPlayer.release();  
    mediaPlayer = null;}
```

# Bài tập

- Tạo ứng dụng xử lý audio gồm các tính năng
  - Start audio
  - Pause
  - Resume
  - Stop audio



# Chuẩn bị phát video

- Để phát video cần có giao diện hiển thị video. Giao diện hiển thị video được tạo ra theo 1 trong 2 cách:
  - Sử dụng lớp VideoView do Androi cung cấp;
  - Tự tạo ra giao diện phát video với SurfaceView.

```
<VideoView android:id="@+id/vView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" >
VideoView vView =(VideoView)findViewById(R.id.vView);
MediaController mediaController= new MediaController(this);
vView.setMediaController(mediaController);
String path = "android.resource://" + getPackageName() + "/" + R.raw.mz;
vView.setVideoURI(Uri.parse(path));
vView.start();
```

# Xử lý video

- Tạm dừng

```
if(vView != null){  
    vView.pause();  
    length = mediaPlayer.getCurrentPosition();}
```

- Tiếp tục phát

```
if(vView != null && length>0){  
    vView.seekTo(length);  
    vView.start();}
```

- Dừng phát audio

```
if(vView != null){  
    vView.stopPlayback();}
```

# Bài tập

- Tạo giao diện xử lý video, sử dụng Seekbar

```
<SeekBar android:id="@+id/seekBar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    public void onStopTrackingTouch(SeekBar seekBar) {}
    public void onStartTrackingTouch(SeekBar seekBar) {}
    public void onProgressChanged(SeekBar seekBar, int progress,
        boolean fromUser) {
        vView.seekTo(progress);
        vView.start();}});
vView.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    public void onPrepared(MediaPlayer mp) {
        seekBar.setMax(vView.getDuration());}});
```



## **8. Lập trình điều khiển các thiết bị phần cứng tích hợp trên đầu cuối di động**

1. Camera
2. Sensors

## **8. Lập trình điều khiển các thiết bị phần cứng tích hợp trên đầu cuối di động**

- 1. Camera**
- 2. Sensors**

# Gọi ứng dụng Camera có sẵn

```
Intent i = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
Intent i = new Intent(android.provider.MediaStore.ACTION_VIDEO_CAPTURE);
startActivityForResult(i,0);
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    ...
}
```

# Chụp ảnh từ ứng dụng Camera có sẵn

```
private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
private Uri fileUri;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // create Intent to take a picture and return control to the calling application
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE); // create a file to save the image
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri); // set the image file name

    // start the image capture Intent
    startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
}
```

- **MediaStore.EXTRA\_OUTPUT:** thiết lập tên và vị trí lưu trữ file với fileUri truyền vào. Nếu thông tin này không được thiết lập thì file video sẽ được lưu vào vị trí mặc định với tên mặc định.

# Quay video từ ứng dụng Camera có sẵn

```
private static final int CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE = 200;
private Uri fileUri;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    //create new Intent
    Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);

    fileUri = getOutputMediaFileUri(MEDIA_TYPE_VIDEO); // create a file to save the vi
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri); // set the image file name

    intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1); // set the video image quality

    // start the Video Capture Intent
    startActivityForResult(intent, CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE);
}
```

# Quay video từ ứng dụng Camera có sẵn

- **MediaStore.EXTRA\_OUTPUT**: thiết lập tên và vị trí lưu trữ file với fileUri truyền vào. Nếu thông tin này không được thiết lập thì file video sẽ được lưu vào vị trí mặc định với tên mặc định.
- **MediaStore.EXTRA\_VIDEO\_QUALITY**: thiết lập chất lượng ảnh.Với giá trị là 0 thì chất lượng và kích thước ảnh thấp nhất, giá trị là 1 thì chất lượng và kích thước ảnh cao nhất.
- **MediaStore.EXTRA\_DURATION\_LIMIT**: thiết lập độ dài giới hạn của đoạn video theo giây.
- **MediaStore.EXTRA\_SIZE\_LIMIT**: thiết lập kích thước dung lượng giới hạn của file video theo byte.

# APIs

- **Camera:** cung cấp những API điều khiển camera. Lớp này dùng để xây dựng ứng dụng có khả năng chụp ảnh, video.
- **SurfaceView:** sử dụng để hiển thị màn hình camera trong ứng dụng.
- **MediaRecorder:** sử dụng để ghi video từ camera.

# Cấp quyền cho ứng dụng

- Cấp quyền cho ứng dụng sử dụng các loại camera trên thiết bị:

```
<uses-permission android:name="android.permission.CAMERA" />
```

- Cấp quyền cho ứng dụng sử dụng các đặc tính camera trên thiết bị:

```
<uses-feature android:name="android.hardware.camera" />
```

- Cấp quyền lưu trữ ảnh/ video trên bộ nhớ ngoài (SD card) cho ứng dụng:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- Cấp quyền thu audio cho ứng dụng:

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

- Cấp quyền cho phép ứng dụng có khả năng chụp ảnh kèm theo ghi lại thông tin vị trí GPS:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

# Cấp quyền cho ứng dụng

- **Sử dụng các đặc tính cụ thể**
  - **android.hardware.camera.autofocus**: sử dụng tính năng **autofocus** của thiết bị camera trong ứng dụng.
  - **android.hardware.camera.flash**: sử dụng tính năng **flash** của thiết bị camera trong ứng dụng.
  - **android.hardware.camera.front**: sử dụng **camera trước** của thiết bị camera trong ứng dụng.
  - **android.hardware.camera.any**: sử dụng **ít nhất 1 camera** trên thiết bị hoặc camera kết nối với thiết bị.
  - **android.hardware.camera.external**: sử dụng **camera ngoài** nếu nó được kết nối tới.

# Kiểm tra camera có tồn tại trên thiết bị không?

```
PackageManager pm = getPackageManager();
if(pm.hasSystemFeature(PackageManager.FEATURE_CAMERA)) {
    //exist
}
else{
    //not exist
}
```

- Kiểm tra số lượng camera:

```
int count = Camera.getNumberOfCameras();
```

- Truy cập camera:

```
Camera mCamera = Camera.open();
```

```
CameraPreview mCameraPreview = new CameraPreview(this, mCamera);  
FrameLayout frPreview = (FrameLayout) findViewById(R.id.frPreview);  
frPreview.addView(mCameraPreview);
```

- Tạo lớp xem trước ảnh/video từ camera

```
public class CameraPreview extends SurfaceView implements SurfaceHolder.Callback {  
    private SurfaceHolder mHolder; private Camera mCamera;  
    public CameraPreview(Context context, Camera camera) {  
        super(context); mCamera = camera;  
        mHolder = getHolder(); mHolder.addCallback(this);}  
    public void surfaceCreated(SurfaceHolder arg0) {  
        try {mCamera.setPreviewDisplay(arg0);  
            mCamera.startPreview();}  
        catch (IOException e) {e.printStackTrace();}  
    }  
    public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {}  
    public void surfaceDestroyed(SurfaceHolder holder) {}  
}
```

# Xây dựng giao diện hiển thị và điều khiển camera (1)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <FrameLayout
        android:id="@+id/camera_preview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        />

    <Button
        android:id="@+id/button_capture"
        android:text="Capture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        />
</LinearLayout>
```

# Xây dựng giao diện hiển thị và điều khiển camera (2)

- Khai báo việc hiển thị activity theo chiều ngang

```
<activity android:name=".CameraActivity"
    android:label="@string/app_name"

    android:screenOrientation="landscape">
    <!-- configure this activity to use landscape orientation -->

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

# Xây dựng giao diện hiển thị và điều khiển camera (3)

- Triệu gọi lớp xem trước ảnh/video từ camera

```
public class CameraActivity extends Activity {

    private Camera mCamera;
    private CameraPreview mPreview;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Create an instance of Camera
        mCamera = getCameraInstance();

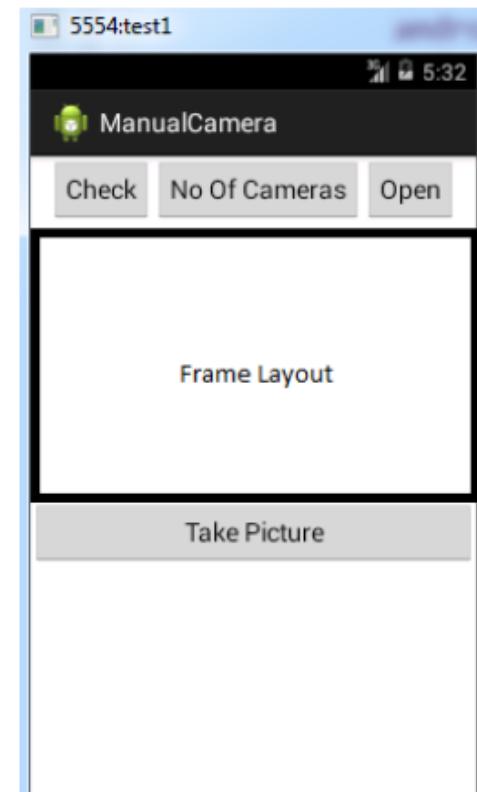
        // Create our Preview view and set it as the content of our activity.
        mPreview = new CameraPreview(this, mCamera);
        FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
        preview.addView(mPreview);
    }
}
```

# Xử lý sự kiện chụp ảnh

```
mCamera.takePicture(null, null, mPictureCallback);
PictureCallback mPictureCallback = new PictureCallback(){
    public void onPictureTaken(byte[] data, Camera camera) {
        File filepath = Environment.getExternalStorageDirectory();
        File dir = new File(filepath.getAbsolutePath() + "/Temp");
        if (!dir.exists()){
            if (!dir.mkdirs()) {showMsg("failed to create directory");return;}}
        String s= new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
        File mediaFile = new File(dir.getPath() + File.separator + "IMG_" + s + ".jpg");
        FileOutputStream fos = new FileOutputStream(mediaFile);
        fos.write(data);
        fos.close();
        mCamera.release();}}
```

# Bài tập

- Tạo giao diện điều khiển camera gồm các chức năng
  - Kiểm tra camera có tồn tại trên thiết bị
  - Hiển thị số lượng camera
  - Truy cập camera
  - Chụp và lưu ảnh



## **8. Lập trình điều khiển các thiết bị phần cứng tích hợp trên đầu cuối di động**

- 1. Camera**
- 2. Sensors**

# Cảm biến (sensor)

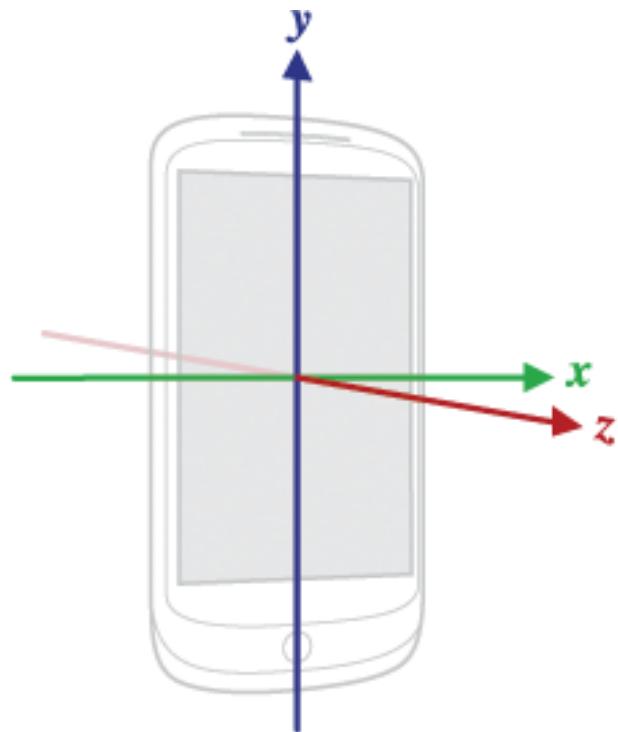
- Hầu hết các thiết bị Android đều gắn sẵn cảm biến để theo dõi chuyển động, hướng, môi trường xung quanh thiết bị.
- Các cảm biến này có khả năng **thu thập dữ liệu thô** với độ chính xác cao.
- Việc khai thác các thông tin thu được từ cảm biến gắn trên thiết bị cho phép lập trình viên tạo ra các ứng dụng đa dạng.
- **Ví dụ:** Ứng dụng thời tiết thu thập thông tin từ *cảm biến nhiệt độ* và *cảm biến độ ẩm* để tính toán và đưa ra thông báo về thời tiết tại một địa điểm nào đó.

# Android hỗ trợ các loại cảm biến

- **Loại cảm biến chuyển động (Motion sensors):** đây là các cảm biến đo lực gia tốc và lực quay dọc theo 3 trục tọa độ.
  - Cảm biến gia tốc (accelerometer sensor)
  - Cảm biến trọng lực (gravity sensor)
  - Con quay hồi chuyển (gyroscopes sensor)
  - Cảm biến vector quay (rotational vector sensor).
- **Loại cảm biến môi trường (Environment sensors):** đây là các cảm biến đo các thông số môi trường như nhiệt độ, độ ẩm, ánh sáng
  - Cảm biến đo khí áp (barometer sensor)
  - Cảm biến trắc quang (photometer sensor)
  - Cảm biến nhiệt kế (thermometer sensor).
- **Loại cảm biến vị trí (Position sensors):** đây là các cảm biến đo vị trí vật lý của thiết bị.
  - Cảm biến định hướng (orientation sensor)
  - Cảm biến từ kế (magnetometer sensor).

Cảm biến	Miêu tả	Sử dụng
TYPE_ACCELEROMETER	Đo lực gia tốc của thiết bị theo 3 hướng vật lý (x,y,z), bao gồm cả lực hấp dẫn. Đơn vị m/s <sup>2</sup>	Phát hiện chuyển động của thiết bị (lắc, nghiêng...)
TYPE_AMBIENT_TEMPERATURE	Đo nhiệt độ môi trường xung quanh thiết bị với đơn vị °C	Giám sát nhiệt độ không khí.
TYPE_GRAVITY	Đo trọng lực thiết bị theo cả 3 trục vật lý (x,y,z). Đơn vị m/s <sup>2</sup>	Phát hiện chuyển động của thiết bị (lắc, nghiêng...)
TYPE_GYROSCOPE	Đo tốc độ quay của thiết bị theo 3 trục vật lý (x,y,z). Đơn vị rad/s.	Phát hiện chuyển động quay của thiết bị.
TYPE_LIGHT	Đo mức độ chiếu sáng của môi trường xung quanh thiết bị. Kiểm soát độ sáng màn hình thiết bị. Đơn vị lx.	
TYPE_LINEAR_ACCELERATION	Đo lực gia tốc của thiết bị theo 3 hướng vật lý (x,y,z), không bao gồm cả lực hấp dẫn. Đơn vị m/s <sup>2</sup>	Theo dõi gia tốc dọc theo 1 trục.
TYPE_MAGNETIC_FIELD	Đo từ trường xung quanh thiết bị theo cả 3 trục vật lý (x,y,z). Tạo la bàn. Đơn vị µT.	
TYPE_ORIENTATION	Đo tốc độ quay của thiết bị theo 3 trục vật lý (x,y,z).	Xác định vị trí thiết bị.
TYPE_PRESSURE	Đo áp suất không khí xung quanh thiết bị. Đơn vị hPa hoặc mbar.	Giám sát sự thay đổi áp suất không khí xung quanh thiết bị.
TYPE_PROXIMITY	Đo khoảng cách của 1 đối tượng với thiết bị. Đơn vị: cm.	Xác định vị trí điện thoại trong khi có cuộc gọi.
TYPE_RELATIVE_HUMIDITY	Đo độ ẩm tương đối của môi trường xung quanh thiết bị. Đơn vị %.	Giám sát độ ẩm tương đối xung quanh thiết bị.
TYPE_ROTATION_VECTOR	Đo hướng thiết bị thông qua vector chuyển động quay của thiết bị.	Phát hiện chuyển động quay của thiết bị.
TYPE_TEMPERATURE	Đo nhiệt độ của thiết bị. Đơn vị °C	Giám sát nhiệt độ thiết bị.

# Hệ tọa độ cảm biến



# Android sensor API

- **android.hardware.SensorManager**
  - Tạo thể hiện của cảm biến
  - Truy cập, đăng ký/ hủy bỏ sự kiện, thu thập thông tin định hướng các cảm biến trên thiết bị
  - Báo cáo, thiết lập và hiệu chỉnh các thông số cảm biến.
- **android.hardware.Sensor**
  - Tạo ra thể hiện của 1 cảm biến cụ thể.
  - Xác định khả năng của cảm biến.
- **android.hardware.SensorEvent**
  - Được sử dụng để tạo ra đối tượng sự kiện cảm biến nhằm cung cấp các thông tin về sự kiện cảm biến. Đối tượng sự kiện cảm biến cung cấp các thông tin như: dữ liệu cảm biến thô, loại cảm biến tạo ra sự kiện, tính chính xác của dữ liệu và các mốc thời gian xuất hiện sự kiện.
- **SensorEventListener:**
  - Giao tiếp này cung cấp 2 phương thức callback cho phép nhận thông báo từ sự kiện cảm biến khi giá trị cảm biến thay đổi hoặc khi độ chính xác cảm biến thay đổi.

# Android sensor API

- Thông thường ứng dụng sử dụng các API cảm biến để thực hiện 2 nhiệm vụ:
  1. Xác định các cảm biến trên thiết bị và khả năng của các cảm biến đó
  2. Theo dõi các sự kiện cảm biến nhằm thu thập dữ liệu cảm biến.

# Xác định các cảm biến trên thiết bị

```
private SensorManager mSensorManager;  
...  
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

- Lấy về danh sách các cảm biến có trên thiết bị

```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

- Kiểm tra 1 loại cảm biến có tồn tại trên thiết bị không?

```
private SensorManager mSensorManager;  
...  
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);  
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null) {  
    // Success! There's a magnetometer.  
}  
else {  
    // Failure! No magnetometer.  
}
```

# Xác định khả năng của các cảm biến trên thiết bị

Lớp Sensor cung cấp 1 số phương thức cho phép xác định khả năng của các cảm biến trên thiết bị:

- `getResolution()` lấy về độ phân giải cảm biến
- `getMaximumRange()` lấy về phạm vi tối đa đo lường của cảm biến,
- `getPower()` lấy về yêu cầu nguồn cho cảm biến
- ...

# Theo dõi các sự kiện cảm biến

- Cài đặt hai phương thức callback của giao tiếp SensorEventListener:
  - `onAccuracyChanged()`
  - `onSensorChanged()`.
- Hai phương thức này được hệ thống gọi bất cứ khi nào giá trị cảm biến hoặc độ chính xác cảm biến thay đổi.

```
SensorManager sm = (SensorManager)getSystemService(SENSOR_SERVICE);

List list = sm.getSensorList(Sensor.TYPE_ACCELEROMETER);

SensorEventListener sel = new SensorEventListener(){

    public void onAccuracyChanged(Sensor sensor, int accuracy) {}

    public void onSensorChanged(SensorEvent event) {
        float[] values = event.values;
        textView1.setText("x: "+values[0]+"\ny: "+values[1]+"\nz: "+values[2]); } }

sm.registerListener(sel, (Sensor) list.get(0), SensorManager.SENSOR_DELAY_NORMAL);

sm.unregisterListener(sel);
```

# BTVN

1. Xác định các cảm biến trên thiết bị và khả năng của các cảm biến đó
2. Theo dõi các sự kiện cảm biến nhằm thu thập dữ liệu cảm biến.