Lập trình di động

Thông tin học phần

- **3 TC**
- **Điểm quá trình: 50%**
  - Thái độ: 10%
  - Bài tập thường xuyên: 20%
  - Bài kiểm tra: 20%
- **Điểm thi kết thúc: 50% (**Hình thức thi: Báo cáo Project)
- ***Sách, giáo trình chính:***
- *[1]. Horton J. (2015) Android Programming for Beginners, Packt Publishing*
- ***Sách (TLTK) tham khảo:***
- [1] *https://developer.android.com/docs*
- *[2]. https://www.tutorialspoint.com/android/*
- *[3]. https://www.homeandlearn.co.uk/android/android.html*

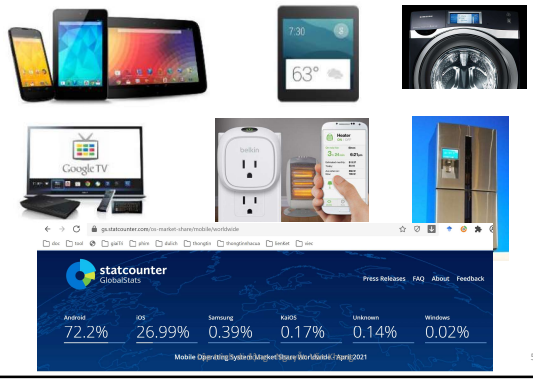
Bài 1. Tổng quan về lập trình Android

1. Tổng quan về Android.
2. Môi trường phát triển ứng dụng Android Studio.
4. Tạo ứng dụng đầu tiên.

1. Tổng quan về Android

- Android là hệ điều hành mã nguồn mở dành cho dành cho các thiết bị di động, phát triển từ Linux
- Khả năng tuỳ biến cao.
- Là hệ điều hành di động phổ biến nhất hiện nay chiếm thị phần sử dụng cao trên toàn thế giới.
- Kho ứng dụng Google Play với nhiều ứng dụng, trò chơi phong phú.
- Hỗ trợ nhiều dịch vụ như nhắn tin (SMS và MMS), trình duyệt web, lưu trữ (SQLite), kết nối (GSM, CDMA, Blue Tooth, Wi-Fi)…

Sự phổ biến của Android



Các phiên bản Android gần đây

Tên	Version	API level	Ngày phát hành
Android 11	11	30	08/09/20
Android 10	10	29	03/09/19
Pie	9	28	06/08/18
Oreo	8.1.0	27	25/10/17
Oreo	8.0.0	26	21/08/17
Nougat	7.1	25	19/10/16
Nougat	7	24	22/08/16
Marshmallow	6	23	05/10/15
Lollipop	5.1	22	09/03/15
Lollipop	5	21	20/10/14

2. Android Studio

- Android Studio là IDE (integrated development environment)
- Chức năng dò và sửa lỗi nhanh, hướng Android.
- Công cụ chỉnh sửa màn hình dạng kéo thả tiện lợi.
- Các wizard tích hợp nhằm giúp lập trình viên tạo ứng dụng từ mẫu có sẵn.
- Tích hợp Google Cloud Platform, dễ dàng tích hợp với Google Cloud Messaging và App Engine của Google.

Tải về và cài đặt Android studio

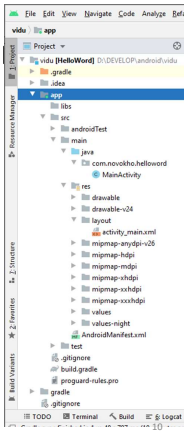
- SV Tự tìm nơi tải chính thức và cài đặt
- https://developer.android.com/studio

Xây dựng ứng dụng đầu tiên

- https://developer.android.com/docs
- Chọn ***Build your fist app*** và làm theo hướng dẫn

Một số thành phần dự á

- Thư mục app/scr: chứa mã nguồn
  - app/scr/main/java: chưa code java
  - app/scr/main/res: Tài nguyên: (giao diện, tranh ảnh, văn bản, tham số…)
- Tập tin AndroidManifest.xml: chứa thông tin cài đặt ứng dụng
- Tập tin Gradle: chứa cấu hình dự án. Chỉ được phép thay đổi app/build.gradle



AndroidManifest

- Lưu trữ thông tin tên gói ứng dụng, tồn tại duy nhất một tên gói trong mỗi ứng dụng.
- Ví dụ: com.htsi.myfirstapp
- Cho biết ứng dụng sử dụng các thành phần nào, mỗi thành phần được khai trong một cặp thẻ.
- Ví dụ: <activity>…..</activity>
- Định nghĩa tiến trình quản lý các thành phần ứng dụng.
- Định nghĩa ác quyền sử dụng API và truy xuất ứng dụng khác.
- Quy định các yêu cầu khi được ứng dụng khác truy xuất
- Khai báo cấp độ API tối thiểu xây dựng ứng dụng.
- Khai báo các thư viện liên quan.

Bài tập

- Tạo dự án helloworld
- Xem nội dung các file java, AndroidManifest, app/build.gradle
- Tạo một điện thoại ảo (mục AVD) để chạy
- Chạy bằng điện thoại Android:
  - Developper (tùy chọn nhà phát triển)→Enable USB debugging
  - Thực hiện chạy trên điện thoại
- Chỉnh sửa dự án bằng cách thêm ô chứa tên. Thay đổi màu sắc, font chữ.

Bài 2. Các thành phần cơ bản

1. Activity
2. Toast
3. Layout & View
4. Controls

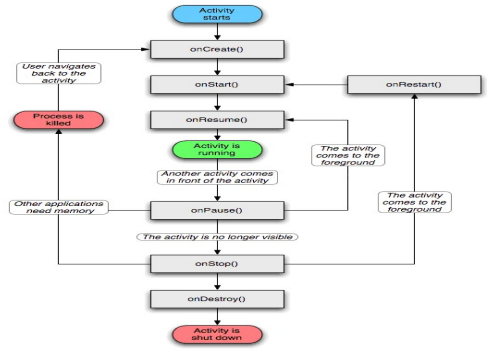
1. Activity

- Trong ứng dụng Android, Activity đóng vai trò là một màn hình, nơi người dùng có thể tương tác với ứng dụng, ví dụ: chụp hình, xem bản đồ, gửi email…
- Một ứng dụng có thể có một hoặc nhiều Activity, Activity được khởi chạy đầu tiên khi ứng dụng hoạt động được gọi là "MainActivity"
- Activity có thể hiển thị ở chế độ toàn màn hình, hoặc ở dạng cửa sổ với một kích thước nhất định.
- Các Activity có thể gọi đến các Activity khác, Activity được gọi sẽ nhận được tương tác ở thời điểm đó.

Quản lý trạng thái Activity

- Activity bao gồm ba trạng thái:
  - Resumed: đang trong trạng thái nhận tương tác.
  - Paused: không thể tương tác nhưng vẫn được thấy bởi người dùng.
  - Stopped: thực hiện chạy ở chế độ ngầm.
- Thực hiện gọi các hàm quản lý trạng thái;
  - onStart
  - onRestart
  - onCreate
  - onPause
  - onResume
  - onStop
  - onDestroys

Quản lý trạng thái Activity



Bài tập

- Thử các sự kiện sau của Activity
  - onStart
  - onPause
  - onResume
  - onStop
- Ví dụ:

(HD:Trong code Avtivity,

R-click→Generate→Override Methods, chọn phương thức,vd: onPause, code sinh ra:

```
@Override
protected void onPause() {
    super.onPause();
    Log.i("ACT", "Pause");
}
```


Sử dụng Toast

- **Android Toast** cho phép tạo một thông báo nhỏ, nó xuất hiện gần phía cuối màn hình (theo mặc định) và tự động biến mất khi hết thời gian.
- Ví dụ sử dụng:

```
Toast toast =  Toast.makeText(MainActivity.this,
        "This is a message!", Toast.LENGTH_SHORT);
toast.setGravity(Gravity.CENTER, 20, 30); //vị trí xuất hiện
toast.show();
```

Bài tập

- Thêm **Button** vào layout chính và thực hiện gọi **Toast** để hiển thị thông báo.
- Hints: Tìm hướng dẫn Android button và android toast

Thêm vào trong phương thức **OnCreate** của Activity:

```
Button button = findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
Toast toast =  Toast.makeText(MainActivity.this,
        "This is a message!", Toast.LENGTH_SHORT);
toast.show();
}
});
```


Layout

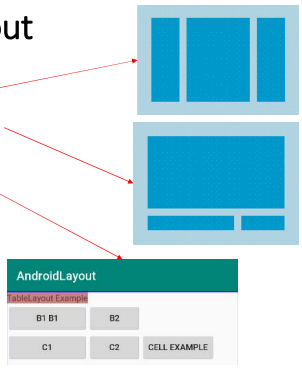
- Layout tạo bố cục giao diện, bố trí các *View*
- *View* là các đối tượng trên giao diện để xem, tương tác, như Button hay TextView
- Layout file: res/layout/*.xml

  vd: res/layout /activity_main.xml
- Tải lên từ trong Activity:

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


Các loại layout

- LinerLayout
- RelativeLayout
- TableLayout
- AbsoluteLayout
- ConstraintLayout



LinerLayout

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

    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/btn_test2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

match_parent
fill_parent
wrap_content

Horizontal
vertical

RelativeLayout

- Các view bên trong có ví trí tương đối trên dưới trái phải so với Layout hoặc so với view khác.
- Một số thuộc tính view bên trong RelativeLayout:

android:layout_alignParentTop="true/false"

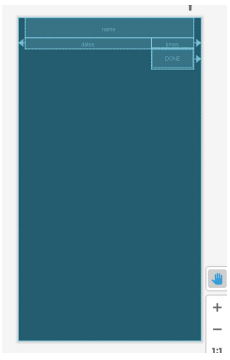
android:layout_centerVertical="true/false"

android:layout_below=id_view_khac

android:layout_toRightOf=id_view_khac

Ví dụ RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="16dp"
   android:paddingRight="16dp" >
   <EditText
      android:id="@+id/name"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="@string/reminder" />
   <Spinner
      android:id="@+id/dates"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/name"
      android:layout_alignParentLeft="true"
      android:layout_toLeftOf="@+id/times" />
   <Spinner
      android:id="@id/times"
      android:layout_width="96dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/name"
      android:layout_alignParentRight="true" />
   <Button
      android:layout_width="96dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/times"
      android:layout_alignParentRight="true"
      android:text="@string/done" />
</RelativeLayout>
```



ConstraintLayout

- Các đối tượng trong ConstraintLayout được xác định vị trí theo các ràng buộc vào các đối tượng khác theo 4 hướng.
- VD:

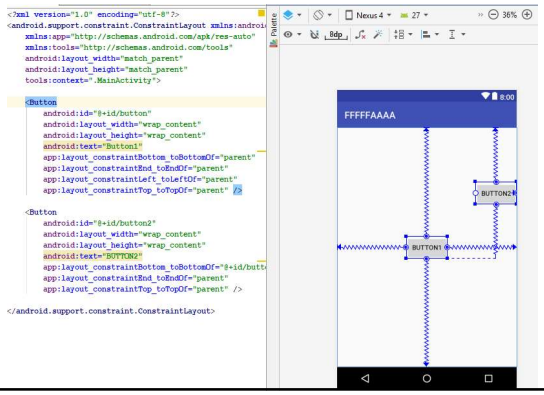
app:layout_constraintBottom_toBottomOf="parent”

Ràng buộc bên dưới theo bên dưới của đối tượng chứa nó

Các ràng buộc trong ConstraintLayout

Ràng buộc	Ý nghĩa ràng buộc
layout_constraintLeft_toLeftOf	Rang buộc cạnh trái của phần tử tới phần tử chỉ ra trong giá trị (gán ID)
layout_constraintLeft_toRightOf	Bên trái với bên phải của phần tử chỉ ra
layout_constraintRight_toLeftOf	Bên phải với bên trái
layout_constraintRight_toRightOf	Phải với phải
layout_constraintTop_toTopOf	Cạnh trên với cạnh trên
layout_constraintTop_toBottomOf	Cạnh trên nối với cạnh dưới
layout_constraintBottom_toTopOf	Dưới với trên
layout_constraintBottom_toBottomOf	Dưới với dưới
layout_constraintBaseline_toBaselineOf	Trùng Baseline
layout_constraintStart_toEndOf	Bắt đầu - Kết húc
layout_constraintStart_toStartOf	Bắt đầu - Bắt đầu
layout_constraintEnd_toStartOf	Cuối với bắt đầu
layout_constraintEnd_toEndOf	Cuối với cuối

Ví dụ ConstraintLayout



UI Control

Khái niệm

UI Control là các thành phần (view) có tính tương tác trong giao diện UI của ứng dụng như Button, EditText, CheckBox…

Các điều khiển cơ bản trình bày trong bài học:

- TextView
- EditText
- Botton
- CheckBox
- RadioGroup, RadioButton

TextView

- TextView là một view rất phổ biến chuyên để hiển thị các đoạn văn bản lên màn hình ứng dụng
- Thuộc tính cơ bản
  - android:id
  - android:capitalize
  - android:editable
  - android:fontFamily
  - android:gravity
  - android:inputType
  - android:password
  - android:text
  - android:textSize

TextView

- Thiết lập nội dung hiển thị:
  - Trong Java code:
    - textView.setText(“Đối tượng TextView”);
  - Trong XML:
    - Android:text=“Đối tượng TextView”

EditText

- EditText là một lớp con của TextView và bao gồm khả năng chỉnh sửa.
- Thuộc tính cơ bản:
  - android:id
  - android:autoText
  - android:background
  - android:onClick
  - android:text
  - android:ems
- Bài tập TH: Chèn EditText, button

EditText

- Thiết lập nội dung hiển thị:
  - Trong Java code:
    edittext.setText("Đối tượng TextView");
  - Trong XML:
    - Android:text="Đối tượng TextView"
- Lấy nội dung:
  String text = editText.getText().toString();

Button

- Thuộc tính cơ bản
  - android:id
  - android:drawableRight
  - android:onClick
  - android:text
- Bắt sự kiện click trong Java-Code:

```
button.setOnclickListener(new OnclickLisntener(){
@Override
   pubblic void onClick(View v){
         Log.i("HTSI", "onClick");
}
});
```

- Bắt sự kiện click trong XML
  Android:onClick="tenPhuongThuc"

Radio button

- Thuộc tính cơ bản
  - android:id
  - android:checked
  - android:text

RadioGroup

- RadioGroup cung cấp khả năng lựa chọn chỉ một RadioButton từ bộ này
- Thuộc tính cơ bản
- android:id
- android:checkedButton
- android:onClick
- android:background

Ví dụ RadioGroup và RadioButton

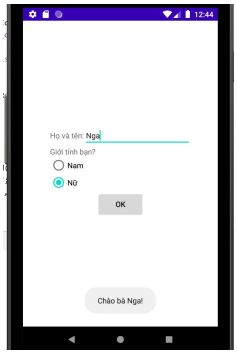
```
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/rad_gioitinh">
    <RadioButton
        android:id="@+id/rad_nam"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nam" />
    <RadioButton
        android:id="@+id/radioButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nữ" />
</RadioGroup>
```

CheckBox

- CheckBox cho phép chọn hoặc bỏ chọn.
- Thuộc tính cơ bản
  - android:id
  - android:background
  - android:checked
  - android:text

Bài tập

- Tạo form nhập đơn giản như hình.
- Bấm nút OK thì hiển thị câu chào ở dưới.



Hướng dẫn BT

- Tạo ứng dụng với Activity rỗng
- Có thể sửa file layout lại như dưới:

File layout phần 1

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
        <TextView
            android:id="@+id/txt_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Họ và tên:"
            android:layout_alignParentTop="true"
            android:layout_alignParentLeft="true"/>
```


File layout phần 2

```
<EditText
          android:id="@+id/edt_name"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_alignBaseline="@id/txt_name"
          android:layout_toRightOf="@id/txt_name"
          android:ems="12"
          android:textSize="14sp" />
      <TextView
          android:id="@+id/txt_sex"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Giới tính bạn?"
          android:layout_below="@id/edt_name"/>
      <RadioGroup
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:id="@+id/rdg_sex"
          android:layout_below="@id/txt_sex">
```


File layout phần 3

```
        <RadioButton
              android:id="@+id/rad_man"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Nam"
              android:checked="true"/>
          <RadioButton
              android:id="@+id/rad_women"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Nữ" />
      </RadioGroup>
      <Button
          android:id="@+id/btn_ok"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text=" OK "
          android:layout_below="@id/rdg_sex"
          android:layout_centerHorizontal="true"  />
    </RelativeLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Hướng dẫn BT

- Code trong Activity →onCreate

```
RadioGroup radioSexGroup=(RadioGroup)findViewById(R.id.rdg_sex);
Button btnDisplay=(Button)findViewById(R.id.btn_ok);
btnDisplay.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int selectedId=radioSexGroup.getCheckedRadioButtonId();
        String hello="Chào ";
        if(selectedId==R.id.rad_man)
            hello+="ông ";
        else
            hello+="bà ";
        EditText edt_name=(EditText)findViewById(R.id.edt_name);
        hello+=edt_name.getText().toString()+"!";

Toast.makeText(MainActivity.this,hello,Toast.LENGTH_LONG).show();
    }
});
```


Hardcoding

- Lưu trữ những cấu hình hoặc là dữ liệu đầu vào (đường dẫn file, remote host name hay một đoạn văn bản ở ngôn ngữ cụ thể nào đó) ở trong mã nguồn ứng dụng thay vì lưu chúng ở trong những file cấu hình.
- Ví dụ:
  - Trong layout:

    ```
    android:text="Giới tính bạn?"
    ```

  - Trong java

    ```
    String server="hopto.com.vn";
    ```

- Tác hại
  - Chương trình chỉ hoạt động tốt trong một môi trường cụ thể.
  - Khó bảo trì.

✍ Trong tài liệu học, hardcode tạm sử dụng để code ngắn dễ thấy. Khi làm đồ án, sv cần tập tránh hardcode

Sử dụng tài nguyên string.xml

- Vị trí: res/values/strings.xml

```
<resources>
    <string
      name="app_name">testControl</string>
    <string name="full_name">Họ và tên:</string>
    <string name="your_sex">Giới tính của
bạn?</string>
</resources>
```

- Trong layout:

```
<TextView
    android:id="@+id/txt_name"
    android:text="@string/full_name"
…
/>
```

- Trong code

```
String lab_name = getString(R.string.  full_name);
```


Bài tập

- Làm chương trình với một câu hỏi 4 lựa chọnNút bấm “Kết thúc” hiển thị ra “Bạn trả lời đúng” hoặc “Bạn trả lời sai”.
- Câu hỏi sv tự đặt và chỉ một đáp án đúng.

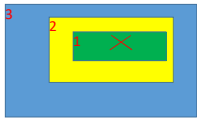
Bài 3

Bài 3. Quản lý sự kiện (Event handling)

- Sự kiện - Event là sự thay đổi (có một hành động, thay đổi dữ liệu…) trong môi trường mà chương trình có thể nhận biết.
- Các sự kiện phổ biến: tương tác người dùng (ấn nút, chạm màn hình, gõ ký tự…)

Cơ chế sự kiện tương tác

- Nhiều sự kiện→ hàng đợi, sự kiến đến trước được xử lý trước.
- Nhiều view lồng nhau: ưu tiên xử lý bởi nhỏ đối tượng nhỏ nhất



Các khái niệm

- Event Listener
  - Là một Interface trong lớp View
  - Chứa một phương thức callback kích hoạt khi có tương tác.
- Event Listener Registration – Event Registration: đăng ký để nhằm thực thi Event Handler khi có sự kiện xãy ra
- Event Handler – Phương thức xử lý sự kiện.

Một số Event Listener & Event Handler

Event Listener Name	Event Handler	Sự kiện
OnClickListener()	onClick()	Click hoặc chạm (touche) trên các view như button, text, image vv.
OnLongClickListener()	onLongClick()	Click hoặc chạm (touche) trên các view như button, text, image vv. trong một hoặc nhiều giây.
OnFocusChangeListener()	onFocusChange()	Sự kiện phát sinh khi view mất focus.
OnFocusChangeListener()	onKey()	Người dùng focus trên view và nhấn một phím bàn phím thật.
OnTouchListener()	onTouch()	Người dùng chạm trên màn hình
TextWatcher	onTextChanged()	Khi nội dung EditText vừa thay đổi
	beforeTextChanged()	Trước khi nội dung EditText thay đổi
	afterTextChanged()	Gọi sau klhi EditText thay đổi

Các cách đăng ký sự kiện phổ biến

- Bắt sự kiện trong Layout (Handle event in Layout)
- Bắt sự kiện bằng lớp nặc danh (Inline anonymous listener).
- Kế thừa Interface OnClickListener (Implements OnClickListener Interface)

- Xem thêm: https://thangcoder.com/lap-trinh-android/hoc-lap-trinh-android-nang-cao/6-cach-xu-ly-su-kien-trong-lap-trinh-android

Bắt sự kiện trong file Layout

- Đơn giản, hạn chế khả năng tái sử dụng của layout
- Trong Layout:

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="button 1"
    android:onClick="button1click"/>
```

- Trong Activity:

```
public void button1click(View view) {
    Toast.makeText(MainActivity.this, "Button1
clicked", Toast.LENGTH_SHORT).show();
}
```


Bắt sự kiện bằng lớp nặc danh

- Hay được dung
- Trong phương thức onCreate của Activity:

```
Button button1=(Button)findViewById(R.id.button1);

button1.setOnClickListener(
  new View.OnClickListener() {
  @Override
    public void onClick(View view) {
       Toast.makeText(MainActivity.this,
         "Button1 clicked",
         Toast.LENGTH_SHORT).show();
       }
  }
);
```

Kế thừa Interface OnClickListener

- Thích hợp khi có nhiều đối tượng cùng bắt một loại sự kiện

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

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

        });
        Button button1=(Button)findViewById(R.id.button1);
        button1.setOnClickListener(this);
    }

    @Override
    public void onClick(View view) {
        if(view.getId()==R.id.button1){
            Toast.makeText(MainActivity.this, "Button1
         clicked", Toast.LENGTH_SHORT).show();
        }
    }

...
```

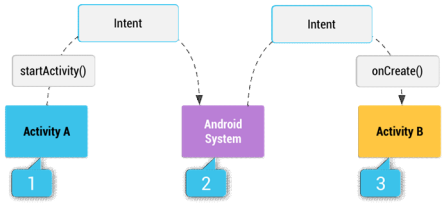

Lab

- Viết ứng dụng Android nhập vào 2 số (sử dụng EditText), người dùng chọn các nút (button) tương ứng với các chức năng cộng, trừ, nhân, chia thì kết quả sẽ được hiển thị (sử dụng TextView) tương ứng với từng phép tính.
- Yêu đăng ký sự kiện click theo 3 cach khác nhau

Bài 4. Intent và Intent Filter

- **Intent được sử dụng để truyền tải thông điệp, yêu cầu một hành động xử lý từ thành phần được gọi.**
- **Intent được sử dụng trong ba trường hợp chính:**
  - Khởi động Activity thông qua phương thức startActivity.
  - Khởi động Service thông qua phương thức startService.
  - Chuyển thông điệp đến BroadcastReceiver thông qua phương thức sendBroadcast.

- **Intent được chia làm hai dạng:**
  - Explicit Intent: chỉ định rõ thành phần xử lý thông qua tên lớp, thường được dùng để gọi đến các thành phần trong cùng ứng dụng.
  - Implicit Intent: không chỉ định rõ thành phần xử lý, thay vào đó bổ sung các thuộc tính như: mô tả hành động, dạng dữ liệu…



Explicit Intent

- Khai báo:

  Intent intent = new Intent(this, <Component>);

- Ví dụ: khởi động Activity có tên SecondActivity từ MainActivity

```
Intent exintent=new
Intent(MainActivity.this,
   SecondActivity.class);
exintent.putExtra("HELLO", "Chào bạn");
startActivity(exintent);
```


Implicit Intent

- Implicit Intents chỉ rõ hành động cần được thực hiện và dữ liệu cho hành động.
- Hệ thống sẽ tìm kiếm ứng dụng phù hợp (đã đăng ký)
- Ví dụ

```
Intent imintent=new
Intent(Intent.ACTION_DIAL,
     Uri.parse("tel:0394999775"));
startActivity(imintent);
```

```
Intent
intent=new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http:7/www.g
oogle.com"));
startActivity(intent);
```

Truyền dữ liệu

- Truyền đơn giản: dùng
- `intent.putExtra("Key","Value");`
- Ví dụ

```
Intent exintent=new
Intent(MainActivity.this,
   SecondActivity.class);
exintent.putExtra("HELLO", "Chào bạn");
startActivity(exintent);
```


Truyền dữ liệu

- Nhận dữ liệu dung các phương thức getBooleanExtra(), getStringExtra(), getIntExtra().. của lớp Intent.
- Ví dụ

```
Intent intent = this.getIntent();
String msg==intent.getStringExtra("HELLO"));
```


Intent Filter

- Intent Filter là thành phần giúp cho hệ thống Android biết được ứng dụng của bạn có thể làm được những gì.
- Ví dụ:
- Khi bạn mở một file text, hệ thống đưa ra một hộp thoại với tên các ứng dụng có thể mở file này như: officesuite, quickoffice, note, ...

→Các ứng dụng trên đã được cài đặt IntentFilter để báo cho android biết "tôi có thể thực hiện công việc đó".

- Tất cả các Activity, Service và Broadcast Receiver (Service và Broadcast Receiver sẽ học ở những bài tiếp theo) đều sử dụng IntentFilter để thông báo cho hệ thống biết các dạng Implicit Intent mà nó có thể xử lý.
- Intent Filter là bộ lọc chỉ cho intent mà nó hiểu được phép đi qua nó.

Intent Filter

- Khai báo trong AndroidManifest.xml, trong thẻ Activity, Service…
- Trong thẻ **<intent-filter>**, có chứa 3 thẻ khác:
- **<action>** mô tả hành động sẽ thực hiện.
- **<category>** để android.intent.category.LAUNCHER nếu muốn nhận một implicit intent
- **<data>** thẻ này nói rõ hơn cho hệ thống biết là chỉ nhận những hành động có dữ liệu liệu cụ thể.

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.SENDTO" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="sms" />
    </intent-filter>
</activity>
```


Bổ sung control Spinner

- Tạo <Spinner> trong layout
- Tạo string-array trong 1 file xml trong res/values
- Trong Activity:
- Lấy đối tượng spinner theo id

```
ArrayAdapter adapter =
ArrayAdapter.createFromResource(this,
id_cua_string-array,
android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

- Bắt sự kiện chọn:
  - `spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {…});`
- Lấy vị trí chọn trị:

```
spinner.getSelectedItemPosition()
```


Bài 5. Service

- Ứng dụng Android có 4 loại thành phần chính gọi là app component
- Mỗi app component một là điểm mà hệ thống hoặc người dùng truy cập vào ứng dụng.
- Các loại app component:
  - Activity
  - Service
  - Broadcast receiver
  - Content provider

Service là gì?

- Service là một app component chạy background của hệ thống
- Thường dung thực hiện các tác vụ dài hạn
- Không tương tác với người dùng
- Có thể hoạt động ngay cả khi ứng dụng gọi nó bị hủy bỏ.
- Ví dụ:
  - Ứng dụng báo thức
  - Ứng dụng chơi nhạc

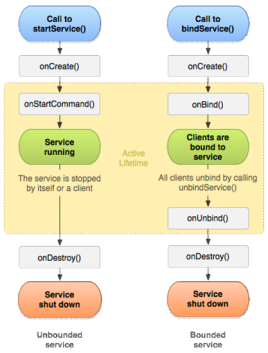
Unbound Service

- Unbound Service (không ràng buộc)
  - Được khởi chạy bằng phương thức **startService()**
  - Chạy vô thời hạn và phải hủy bỏ bằng cách gọi **stopSelf()** hoặc **stopService()**

Bound Service

- Bắt đầu khi một component (client) gọi phương thức **bindService()**
- Cho phép các component tương tác với Service theo dạng Client - Service thông qua Ibinder
- Client có thể ngắt kết nối bằng cách gọi unbindService()
- Nhiều Clients có thể liên kết với cùng một Service và khi tất cả chúng không còn liên kết, hệ thống sẽ tự động hủy Service

Vòng đời Service



Một số phương thức callback

- onCreate() - Hệ thống gọi hàm này khi lần đầu Service chạy trước khi chạy hàm onStartCommand() và onBind(). Nếu Service chạy rồi thì hàm này không được gọi nữa
- onStartCommand() - Hệ thống sẽ gọi hàm này khi một component hoặc một Activity yêu cầu bắt đầu Service bằng cách gọi startService()
- onBind() - Hệ thống sẽ gọi hàm này khi một component muốn ràng buộc với Service bằng cách gọi bindService(). Khi cài đặt hàm này bạn phải cung cấp interface giữa Client và Service bằng cách trả về IBinder
- onUnBind() - Ngắt tất cả các kết nối từ Client với Service bằng cách gọi unbindService()
- onReBind() - Cho phép Client kết nối lại với Service khi nó đã gọi onUnBind() trước đó
- onDestroy() - Service không hoạt động trong một thời gian dài và sẽ bị hủy bỏ để giải phóng tài nguyên

Dùng service

- Tạo lớp thừa kế lớp Service, implement các phương thức callback
- Khai báo trong file manifest

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService"
/>
      ...
  </application>
</manifest>
```

Tạo một unbound service

- Trong lớp MyService, implement onStartCommand để thục hiện công việc
- Khởi chạy (từ activity):

```
intent = new Intent(this, MyService.class);
startService(intent);
```


Implement Bound service cơ bản

```
public class MyBoundService extends Service {
   private MyBinder mBinder mBinder=new MyBinder();
 public MyBoundService() {
 }
 @Override
 public IBinder onBind(Intent intent) {
    return mBinder;
 }
 public class MyBinder extends Binder {
    public MyBoundService getService() {
       return MyBoundService.this;
    }
 }
  //Các phương thức để client sử dụng
 }
```


Implement cơ bản activity client của unbound service

```
public class BoundServiceActivity extends AppCompatActivity {
    private MyBoundService mService;
    private boolean mBound=false;
    private ServiceConnection mConnection = new ServiceConnection() {
    public void onServiceConnected(ComponentName className, IBinder service) {
MyBoundService.MyBinder binder = (MyBoundService.MyBinder) service;
        mService = binder.getService();
        mBound = true;
      }
      @Override
      public void onServiceDisconnected(ComponentName componentName) {
        mBound = false;
      }
    };
    @Override
    protected void onStart() {
      super.onStart();
      Intent intent = new Intent(this, MyBoundService.class);
      bindService(intent, mConnection, Context.BIND_AUTO_CREATE);

    }
    @Override
    protected void onStop() {
      super.onStop();
      if(mBound) {
        unbindService(mConnection);
        mBound=false;
      }
    }
}
```

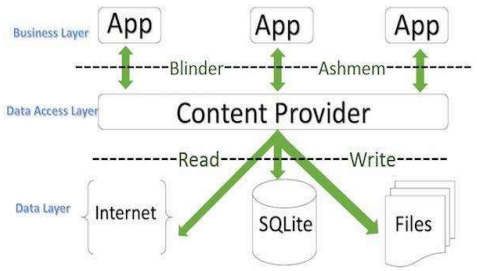

Bài 6. Content Provider

- Giới thiệu Content Provider
- Content URI trong Android
- Tạo Content Provider
- Sử dụng Content Provider

Giới thiệu Content Provider

- Là một thành phần của Android có nhiệm vụ cung cấp dữ liệu từ một ứng dụng này cho các ứng dụng khác theo yêu cầu.
- Đóng gói dữ liệu và cung cấp cho các ứng dụng thông qua interface **ContentResolver**
- Hữu dụng trong việc chia sẽ dữ liệu giữa các ứng dụng. Ví dụ danh sách địa chỉ được dùng bởi nhiều ứng dụng nên cần được lưu trữ trong một Content Provider.
- Giống với một cơ sở dữ liệu, việc truy vấn, chính sửa, thêm hay xóa các nội dung thông qua các phương thức insert(), update(), delete(), query() của ContentResolver.

Giới thiệu Content Provider



Content URI trong Android

- URI (Uniform Resource Identifier): chuỗi ký tự được sử dụng để định danh tên hoặc tài nguyên trên internet. Dạng phổ biến: URL
- Content URI là một URI định danh dữ liệu trong một Content Provider, có định dạng như sau:
- <prefix>://<authority>/<data_type>/<id>
- <prefix> luôn được thiết lập là **content**
- <authority> chỉ định tên cụ thể của Content Provider (ví dụ: Contacts, Browser, funix.prm.lab,...)
- <data_type> chỉ rõ kiểu dữ liệu
- <id> chỉ rõ một record
- Ví dụ:
  - Contact ở vị trí thứ 5 trong danh bạ: **content://contacts/people/5**
  - Mail trong inbox: **content://sms/inbox**

Tạo Content Provider

- Tạo một lớp con của lớp kế thừa Content Provider
- Định nghĩa Content Provider URI
- Khai báo Content Provider trong manifest bằng thẻ **provider**
- Implement lớp ContentProvider và các phương thức được yêu cầu.

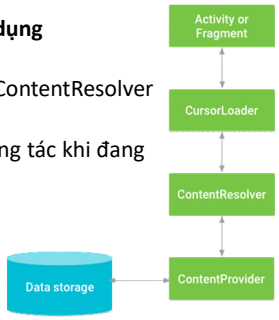
Sử dụng Content Provider

- Yêu cầu quyền
  - Khai báo trong Manifest
  - Yêu cầu quyền trong runtime
- Xác định URI
- Sử dụng ContentResolver

Đọc dữ liệu ContentProvider từ UI

Mô hình phổ biến sử dụng CursorLoader:

- CursorLoader sẽ gọi ContentResolver chạy background.
- Cho phép UI vẫn tương tác khi đang truy vấn



Ví dụ CursorLoader

```
private void showContacts() {
   Uri uri= ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
   CursorLoader loader=new CursorLoader(this,uri, null, null,null,null);
   Cursor cursor=loader.loadInBackground();

   cursor.moveToFirst();
   while(! cursor.isAfterLast())
   {
      String idName=ContactsContract.Contacts.DISPLAY_NAME;
      int colNameIndex=cursor.getColumnIndex(idName);
      String name=cursor.getString(colNameIndex);


      String idPhone=ContactsContract.CommonDataKinds.Phone.NUMBER;
      int colPhoneIndex=cursor.getColumnIndex(idPhone);
      String phone="";
      if(colPhoneIndex>=0) phone= cursor.getString(colPhoneIndex);

      Log.i("CONTACTS", "Tên:"+name +", SĐT:"+phone);
      cursor.moveToNext();
   }
   cursor.close();
}
```


Khai báo quyền

- Khi ứng dụng muốn truy cập tài nguyên ngoài ứng dụng, cần phải yêu cầu quyền
- Khai báo trong AndroidManifest.xml bằng thẻ **uses-permission**
- Ví dụ

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.novokho.getcontacts">
<uses-permission
android:name="android.permission.READ_CONTACTS"/>
<application
…
```

Khai báo quyền

- Một số quyền khác:
  - WRITE_CONTACTS
  - INTERNET
  - ACCESS_NETWORK_STATE
  - BLUETOOTH
  - CAMERA
  - CALL_PHONE
  - BATTERY_STATS

Tham khảo thêm:
https://developer.android.com/reference/android/Manifest.permission

Bài 7. SQLite trong android

SQLite là gì?

- SQLite là một cơ sở dữ liệu SQL mã nguồn mở
- Mặc định đã được tích hợp trên thiết bị Android.
- Không cần phải thiết lập bất kỳ loại kết nối nào cho nó như JDBC, ODBC,

Ưu điểm SQlite

- Tin cậy, các hoạt động transaction trong csdl được thực hiện trọn vẹn, không gây lỗi khi xảy ra sự cố phần cứng
- Tuân theo chuẩn SQL92 (chỉ có một vài đặc điểm không hỗ trợ)
- Không cần cài đặt cấu hình
- Kích thước chương trình gọn nhẹ
- Thực hiện các thao tác đơn giản nhanh hơn các hệ thống csdl client/server khác
- Không cần phầm mềm phụ trợ
- Phầm mềm mã nguồn mở

Các đối tượng chính

- **SQLiteOpenHelper**: đối tượng dùng để tạo, nâng cấp và đóng mở kết nối CSDL
- **SQLiteDatabase** - đối tượng dùng để thực thi các câu lệnh SQL trên một CSDL

SQLiteOpenHelper

Khởi tạo thông dụng:
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)

- **Context** là một lớp trừu tượng chứa thông tin môi trường ứng dụng, cung cấp các phương thức để có thể tương tác với hệ điều hành. Activity, Service… là các lớp con của Context
- **name:** tên CSDL
- **factory**: dung để tạo đối tượng cursor, thường để null.
- **version**: phiên bản của CSDL

SQLiteOpenHelper

Khi khởi tạo đối tượng này, cần implement các phương thức callback:

- **onCreate()**: framework nếu có yêu cầu truy cập mà chưa khởi tạo database. Ở đây sẽ viết code khởi tạo database.
- **onUpgrade()**: được gọi khi ứng phiên bản database tăng. Dùng để cập nhật database hoặc khởi tạo lại thông qua **onCreate()**

SQLiteOpenHelper

Phương thức lấy đối tượng CSDL:

- SQLiteDatabase getReadableDatabase(): Lấy đối tượng CSDL dạng chỉ đọc
- SQLiteDatabase getWritableDatabase(): lấy đối tượng CSDL có thể đọc/ghi

VD Xây dựng lớp kế thừa SQLiteOpenHelper

```
public class MyDBHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "schoolManager";
    private static final int DATABASE_VERSION = 1;
    private static final String TABLE_NAME = "students";

    private static final String KEY_ID = "id";
    private static final String KEY_NAME = "name";
    private static final String KEY_ADDRESS = "address";
    private static final String KEY_PHONE_NUMBER = "phone_number";

    public MyDBHelper(Context context) {
        super(context, DATABASE_NAME, factory: null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String create_students_table = String.format(
                "CREATE TABLE %s(%s INTEGER PRIMARY KEY, %s TEXT, %s TEXT, %s TEXT)",
                TABLE_NAME, KEY_ID, KEY_NAME, KEY_ADDRESS, KEY_PHONE_NUMBER);
        db.execSQL(create_students_table);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        String drop_students_table = String.format("DROP TABLE IF EXISTS %s", TABLE_NAME);
        db.execSQL(drop_students_table);

        onCreate(db);
    }
```


VD Xây dựng lớp kế thừa SQLiteOpenHelper

- Sử dụng trong Activity:
  - MyDBHelper dbhp=new MyDBHelper(this);
  - SQLiteDatabase db=dbhp. getWritableDatabase();
  - Sử dụng db. query(), d. insert()…
- Hoặc có thể viết các phương thức cập nhật trong MyDBHelper (xem lab).

Truy vấn DL với SQLiteDatabase

public Cursor query (String table,
String[] columns,
String selection,
String[] selectionArgs,
String groupBy,
String having,
String orderBy)

String table: tên của bảng cần truy vấn.
String[] columns: danh sách các cột sẽ trả về dữ liệu.
String selection: chứa các điều kiện truy vấn.
String[] selectionArgs: danh sách các tham số phụ cho câu điều kiện.
String[] groupBy: gom nhóm các cột kết quả.
String[] having: bộ lọc theo điều kiện.
String[] orderBy: sắp xếp theo mảng cột được chỉ định.

Truy vấn DL với SQLiteDatabase

- Lưu ý **selection** giống như điều kiện WHERE của SQL (không ghi chữ "WHERE"). Những vị trí cần tham số thì để dấu ?. Các dấu hỏi này sẽ được lần lượt thay thế bằng các giá trị trong **selectionArgs**
- Ví dụ:

SQLiteDatabase db= getReadableDatabase()
Cursor cursor=db.query("STUDENTS",
new String[]{"id", "name"},
"name=? OR name=?",
new String[]{"Tùng", "Cúc"}, null, null, null);

Truy vấn DL với SQLiteDatabase

Có thể dung rawQuery để tuy vấn bằng câu SQL truyền tham số qua dấu ?

public Cursor rawQuery (String sql, String[] selectionArgs)

- Ví dụ:

Cursor cursor=db.rawQuery("SELECT id, name FROM STUDENTS WHERE name=? OR name=?", new String[]{"Tùng", "Cúc");

Truy vấn DL với SQLiteDatabase

- Lưu ý **selection** giống như điều kiện WHERE của SQL (không ghi chữ "WHERE"). Những vị trí cần tham số thì để dấu ?. Các dấu hỏi này sẽ được lần lượt thay thế bằng các giá trị trong **selectionArgs**
- Ví dụ:

```
SQLiteDatabase db= getReadableDatabase()
Cursor cursor=db.query("STUDENTS",
  new String[]{"id", "name"},
 "name=? OR name=?",
  new String[]{"Tùng", "Cúc"}, null, null, null);
```


Truy vấn DL với SQLiteDatabase

Chèn thêm một bản ghi:

```
public long insert (String table,
        String nullColumnHack,
        ContentValues values)
```

- **table**: tên bảng
- **nullColumnHack**: tên cột đặt giá trị null nếu values rỗng. Thường để **null**.
- **values**: bộ giá trị từng cặp tên trường - giá trị
- Giá trị trả về là ID của dòng tạo ra hoặc -1 nếu gặp lỗi.

- ContentValues: đối tượng cho phép xác định key/value. được sử dụng để chèn và cập nhật các mục cơ sở dữ liệu

Truy vấn DL với SQLiteDatabase

Ví dụ Insert

```
public void doInsertRecord()
{
    ContentValues values=new ContentValues();
    values.put("malop", "DHTH7C");
    values.put("tenlop", "Đại học 7C");
    values.put("siso", 30);
    String msg="";
    if(database.insert("tbllop", null, values)==-1){
        msg="Failed to insert record";
    }
    else{
        msg="insert record is successful";
    }
    Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
}
```

Truy vấn DL với SQLiteDatabase

Thay đổi dữ liệu:

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

- Trả về: số dòng cập nhật

```
public void updateLopName(String malop,String new_tenlop)
{
    ContentValues values=new ContentValues();
    values.put("tenlop", new_tenlop);
    int ret=database.update("tbllop", values,
            "malop=?", new String[]{malop});
    if(ret==0){
        //failed;
    }
    else{
        //ok
    }
}
```


Truy vấn DL với SQLiteDatabase

Xóa dữ liệu:

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

Trả về: số dòng xóa

– Muốn xóa toàn bộ dữ liệu trong bảng thì ta truyền null vào 2 đối số cuối:

```
database.delete("tbllop", null, null);
```

– Muốn xóa theo 1 mã nào đó:

```
String malop="DHTH7C";
database.delete("tbllop",
        "malop=?",
        new String[]{malop});
```


Truy vấn DL với SQLiteDatabase

Thực thi một câu SQL không có kết quả trả về:

```
public void execSQL (String sql)
```

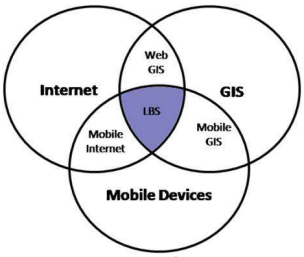
Đối tượng Cursor

- Interface cung cấp quyền truy cập đọc-ghi ngẫu nhiên vào tập kết quả do truy vấn cơ sở dữ liệu trả về.
- Một số phương thức tiêu biểu:

```
int     getCount()
boolean moveToFirst()
boolean moveToNext()
boolean isAfterLast()
int  getColumnIndex(String columnName)
String  getString(int columnIndex)
int  getInt(int columnIndex)
float   getFloat(int columnIndex)
```


Bài 8. Android Location

- Location Based Service (LBS) là hệ thống định vị và giám sát đối tượng dựa vào sự kết hợp giữa hệ thống thông tin địa lý GIS, hệ thống định vị toàn cầu GPS và hệ thống viễn thông Telecom để theo dõi và giám sát vị trí của đối tượng.



LBS Trong Android

1. Google maps
   - Tạo Maps activity
   - Đăng ký API key
2. Lấy vị trí
   - Yêu cầu quyền

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

   - Lây ví trí cuối:

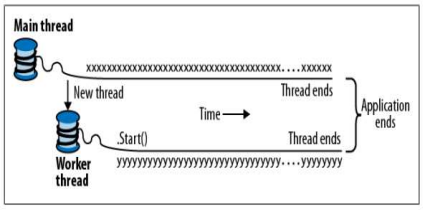
LocationManager. getLastKnownLocation

   - Yêu cầu cập nhật vị trí

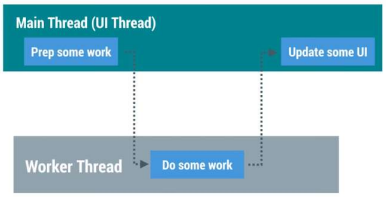
LocationManager .requestLocationUpdates

Bài 9. AsyncTask

1. Multithreading
   - Tương tác chạy song song
   - Khi ứng dụng chạy→Main thread hay gọi là UI thread.
   - Có thể tạo ra các Thread khác để làm các tác vụ song song



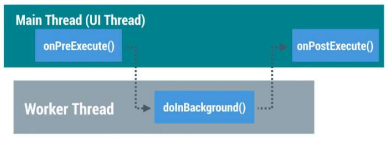
Multithreading trong Android



- Có thể sử dụng Lớp Thread (hoặc Runable)
  - Xử lý nằm trong phương thức run
  - Gọi phương thức start() để chạy Thread
- Khi cập nhật lên UI thread→ gây ra lỗi→Sử dụng message gửi thông tin về.

AsyncTask

- Là một lớp trừu tượng trong Android xử lý tách biệt và đồng thời Background Thread và UI Thread
- Có thể dễ dàng cập nhật lên UI thread



AsyncTask - các phương thức trừu tượng

- **onPreExecute()** - Được chạy đầu tiên khi task được bắt đầu. Phương thức này làm việc trên UI Thread.
- **onPostExecute()** - Phương thức được gọi ngay sau khi phương thức doInBackground() kết thúc. Đối số của phương thức này chính là kết quả trả về từ phương thức **doInBackground()**. Phương thức này làm việc trên UI Thread.
- **doInBackground()** - Phương thức chính để xử lý nhiệm vụ. Phương thức này chạy trên Background Thread, chúng ta không được cập nhật giá trị lên UI trong phương thức này.
- **publishProgress()** – Dùng để sử dụng trong doInBackground(). Nó gọi **onProgressUpdate()** để cập nhật giá trị lên trên UI Thread.
- **onProgressUpdate()** – Nơi implement cập nhập giá trị lên UI, nó được thực thi nếu phương thức **publishProgress()** được gọi.

AsyncTask

- Khi tạo AsynTask, chúng ta sẽ tạo một lớp thừ kế AsynTask.
- Chúng ta cần truyền vào 3 kiểu đối số tương ứng với Param, Progress, Result
- Param - Kiểu đối số được truyền vào cho doInBackground()
- Progress - Kiểu đối số dùng để cập nhập giá trị trong quá trình doInBackground() chạy. Giá trị này được truyền vào phương thức onProgressUpdate()
- Result - Kiểu trả về của phương thức doInBackground() và là kiểu đối số của phương thức onPostExecute()

AsyncTask

```
private class DownloadFilesTask extends AsyncTask<URL, Integer, Long> {
    protected void onPreExecute() {
    }
    protected Long doInBackground(URL... urls) {
      int count = urls.length;
      long totalSize = 0;
      for (int i = 0; i < count; i++) {
         totalSize += Downloader.downloadFile(urls[i]);
         publishProgress((int) ((i / (float) count) * 100));
      }
      return totalSize;
    }

    protected void onProgressUpdate(Integer... progress) {
      setProgressPercent(progress[0]);
    }

    protected void onPostExecute(Long result) {
      showDialog("Downloaded " + result + " bytes");
    }
}
```

Sử dụng:

```
new DownloadFilesTask().execute(url1, url2, url3);
```


Ưu nhược điểm AsyncTask

- Ưu điểm
  - Đơn giản, hiểu quả và nhanh chóng
  - Dễ kiểm soát được quá trình thực hiện
  - Phù hợp với các tác vụ cần cập nhật UI
- Nhược điểm:
  - Chỉ nên dùng với tác vụ nhỏ lẻ
  - Chỉ nên khởi tạo từ Main thread

Mở rộng lab

- Thay lại chương trình tìm số hoàn hảo: số nguyên bằng tổng các ước số
- Ví dụ 6=1+2+3
- doInBackground nhận tham số N, in ra các số hoàn hảo <=N
- Để in 1 số HH, dùng onProgressUpdate với tham số là số cần in.

```
boolean hoanHao(int n){
   int sum=0,i;
   for(i=1; i<=n/2; i++){
      if(n %i==0) sum+=i;
   }
   return (sum==n);
}
```


Bài 10. Kết nối mạng

1. Khai báo quyền

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.novokho.testnetwork">
   <uses-permission
android:name="android.permission.INTERNET" />
   <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE“
…
```

2. Kiểm tra kết nối mạng

Android cung cấp lớp ConnectivityManager. Lớp này sẽ giúp kiểm tra kết nối mạng thông qua phương thức getSystemService():

```
ConnectivityManager cm =
        (ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
isConnected = activeNetwork != null &&  activeNetwork.isConnected();
```


3. Thực hiện thao tác trên mạng

- Lớp HtttpURLConnection và URL

```
String link = "https://funix.edu.vn";
URL url = new URL(link);
```

- Sau đó gọi openConnection() của lớp URL và nhận lại một đối tượng HttpURLConnection

```
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.connect();
```

- lấy mã HTML từ website thông qua InputStream và BufferedReader

```
InputStream is = conn.getInputStream();
BufferedReader reader =new BufferedReader(new InputStreamReader(is, charsetName: "UTF-8"));
String webPage = "",data="";

while ((data = reader.readLine()) != null){
    webPage += data + "\n";
}
```


2. Thực hiện thao tác trên mạng

- Lấy mã HTML từ website thông qua InputStream và BufferedReader

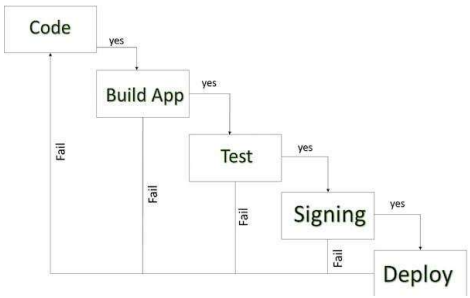
```
InputStream is = conn.getInputStream();
BufferedReader reader =new BufferedReader(new InputStreamReader(is, charsetName: "UTF-8"));
String webPage = "",data="";

while ((data = reader.readLine()) != null){
    webPage += data + "\n";
}
```


Bài 10. Phát hành các ứng dụng Android

1. Vòng đời phát triển một ứng dụng Android
2. Export ứng dụng Android
3. Đăng ký Google Play

1. Vòng đời phát triển một ứng dụng Android



Phần phối app

- Thông qua Google Play
- Gửi trực tiếp hoặc cho download từ website

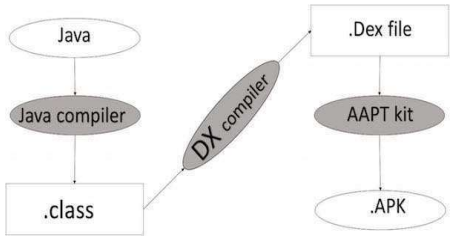
Kiểm tra trước khi phân phối

Công việc	Mô tả
Regression Tesing	Trước khi bạn Publish ứng dụng của mình bạn phải chắc chắn nó thỏa mãn các yêu cầu về chất lượng trên tất cả các thiết bị Android.
Application Rating	Khi publish ứng dụng trên Google Play, bạn phải chỉ rõ độ tuổi cho phép phù hợp với ứng dụng. Hiện tại có 4 mức độ tuổi: • (a) Everyone - Tất cả các độ tuổi • (b) Low maturity - Trưởng thành thấp • (c) Medium maturity - Trưởng thành trung bình • (d) High maturity - Trưởng thành
Targeted Regions	Google Play cho phép người dụng ở quốc gia và vùng lãnh thổ nào có thể tải và sử dụng ứng dụng của bạn.
Application Size	Kích thước một file APK có thể tải lên Google Play là 50Mb. Nếu vượt quá kích thước đó, bạn có thể sử dụng APK Expansion Files, được google host miễn phí và tự động tải về thiết bị.

Kiểm tra trước khi phân phối

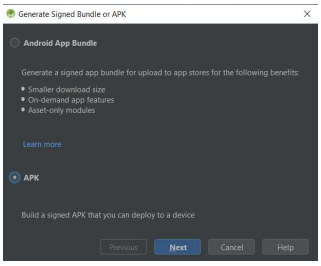
Công việc	Mô tả
SDK and Screen Compatibilily	Hãy chắc chắn ứng dụng của bạn có thể chạy tốt trên các phiên bản Platform Android và các kích cỡ màn hình khác nhau.
Application Pricing	Quyết định ứng dụng của bạn miễn phí hay thu phí. Việc này rất quan trọng, bởi vì nếu miễn phí thì sẽ mãi mãi là miễn phí và không được chuyển đổi thành bản thu phí.
Promotional Content	Để tốt hơn cho việc marketing, bạn nên chuẩn bị các hình ảnh chất lượng cao chứa nội dung về ứng dụng. Sau khi publish, những hình ảnh này xuất hiện trên trang chi tiết về ứng dụng trên store.
Build and Upload release-ready APK	File APK sẽ được tải lên Developer Console và phân phối tới người dùng
Finalize Application Detail	Google Play cung cấp cho bạn nhiều lựa chọn để quảng bá ứng dụng của mình trên trang chi tiết từ hình ảnh, màu sắc, ảnh chụp từ ứng dụng, video, mô tả và hướng dẫn bằng ngôn ngữ địa phương, link tới ứng dụng, ... Bạn có thể trang trí trang ứng dụng của mình và cung cấp thông tin đến mức chi tiết nhất có thể.

2. Export ứng dụng Android



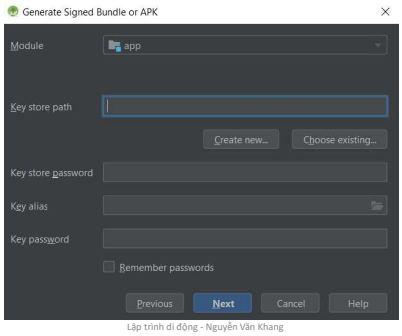
Export ứng dụng trên AndroidStudio

- **Bước 1:** Chọn Build -> Generate Signed APK
- **Bước 2**: Chọn định dạng file dịch của app



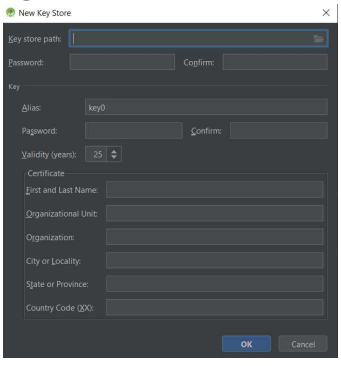
Export ứng dụng trên AndroidStudio

- **Bước 3:** Chọn key/ tạo key



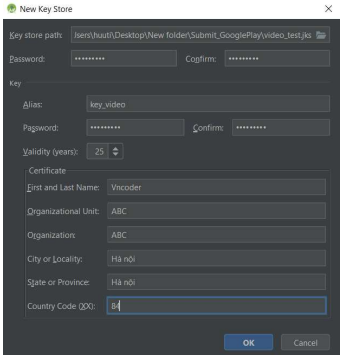
Export ứng dụng trên AndroidStudio

- **Tạo key:** bấm vào Create key → giao diện sau:



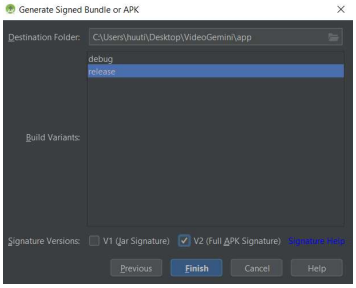
Export ứng dụng trên AndroidStudio

- **Tạo key:** bấm vào Create key → giao diện sau:



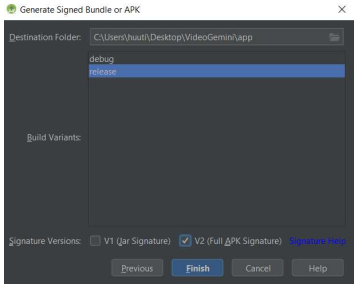
Export ứng dụng trên AndroidStudio

- **Bước 4:** Chọn loại file dịch release:



Export ứng dụng trên AndroidStudio

- **Bước 4:** Chọn đường dẫn file dịch và loại file dịch *release*:
- Bấm **Finish**

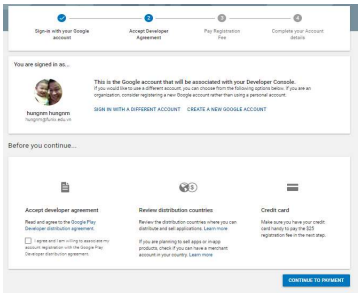


3. Đăng ký Google Play

- Vào Google Play Marketplace.
- Bạn đăng nhập bằng tài khoản Google

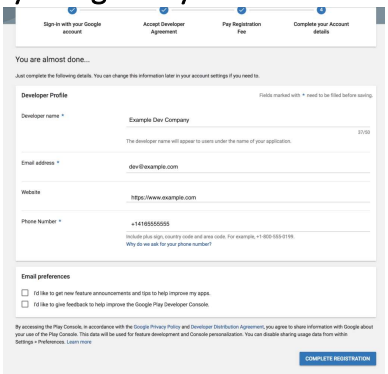
3. Đăng ký Google Play

- Tick vào ô bên dưới "Accept Developer Agreement" và bấm **Continue to payment** .



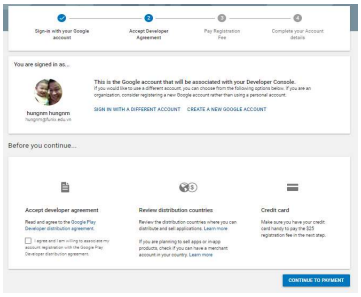
3. Đăng ký Google Play

- Thanh toán bằng thẻ VISA
- Điền thông tin và hoàn thành



3. Đăng ký Google Play

- Tick vào ô bên dưới "Accept Developer Agreement" và bấm **Continue to payment** .



Lab

- Xuất bản một project
- Đưa ứng dụng lên Google play store

Tham khảo hướng dẫn https://appteng.net/huong-dan/huong-dan-cac-buoc-xuat-ban-ung-dung-android/