MOBILE APP DEV

Phần I: Hệ thống cấu trúc Slides

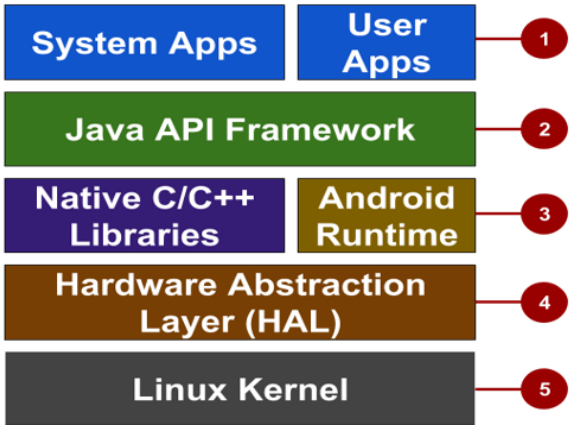
01.0: Introduction to Android

+ Android Ecosystem (Hệ sinh thái Android)

+ Android Platform Architecture (Kiến trúc nền tảng Android)

- Android stack

1. System and user apps
2. Android OS API in Java framework
3. Expose native APIs; run apps
4. Expose device hardware capabilities
5. Linux Kernel



- Android Version

+ Android App Development

- An Android App

- Một hoặc nhiều màn hình tương tác
- Viết mã bởi Java hoặc XML
- Sư dụng Android SDK
- Sử dụng thư viện Android và Android App Famework
- Thực thi bởi Android Runtime Virtual machine (ART)

- App building blocks

- Resources (Tài nguyên): layouts, images, strings, colors as XML và media files.
- Components (Thành phần): activities, services, and helper classes (lớp phụ trợ) as Java code.
- Manifest (Khai báo rõ ràng): thông tin về app để sử dụng khi chạy
- Build configuration (Xây dựng cấu hình): APK versions in Gradle config files.

01.1: Build first Android App

+ First Android app

- Prerequisites (Điều kiện tiên quyết)

- Java Programming Language
- Object-oriented programming
- XML – properties / attributes
- Using an IDE for development and debugging

- Android Studio (IDE)

- Creating first Android app

- Name main activity: MainActivity
- Name layout: activity_main

- Project folders

- **Manifests –** Mô tả app được đọc bởi Android runtime
- **Java –** Java source code packages
- **Res –** resources (XML) – layouts, strings, images, dimensions, colors,..
- **Build.gradle –** xây dựng cấu hình trong các file Gradle

- Gradle build system (những thứ chứa trong Gradle Scripts)

- project: build.gradle (Project: android - Runtime)
- module: build.gradle (Module: Application)
- settings: setting.gradle (Project Settings)

- Logging statement

```
import android.util.Log;

    // Use class name as tag

private static final String TAG = MainActivity.class.getSimpleName();

    // Show message in Android Monitor, logcat pane

    // Log.<log-level>(TAG, "Message");

Log.d(TAG, “Creating the URI…”);
```

01.1: Build first Android App: Layouts and resources for the UI

+ Views

- Các lớp con của Views là khối giao diện cơ bản

- Display text (TextView class), edit text (EditText class)
- Buttons (Button class), menus, other controls
- Scrollable (ScrollView, RecyclerView)
- Show images (ImageView)
- Group views (ConstraintLayout and LinearLayout)

- Tạo views và layouts

- XML Editor

  <TextView

  android:id="@+id/show_count"

  android:layout_width="match_parent"

  android:layout_height="wrap_content"

  android:background="@color/myBackgroundColor"

  android:text="@string/count_initial_value"

  android:textColor="@color/colorPrimary"

  android:textSize="@dimen/count_text_size"

  android:textStyle="bold"

  />

  android:<property_name>="<property_value>"
  Example: android:layout_width="match_parent"
  android:<property_name>="@<resource_type>/resource_id"
  Example: android:text="@string/button_label_next"
  android:<property_name>="@+id/view_id"
  Example: android:id="@+id/show_count"

- Java code

  In an Activity:

  TextView myText = new TextView(this);

  myText.setText("Display this text!");

- Context

  Get the context:

  Context context = getApplicationContext();

  An Activity is its own context:

  TextView myText = new TextView(this);

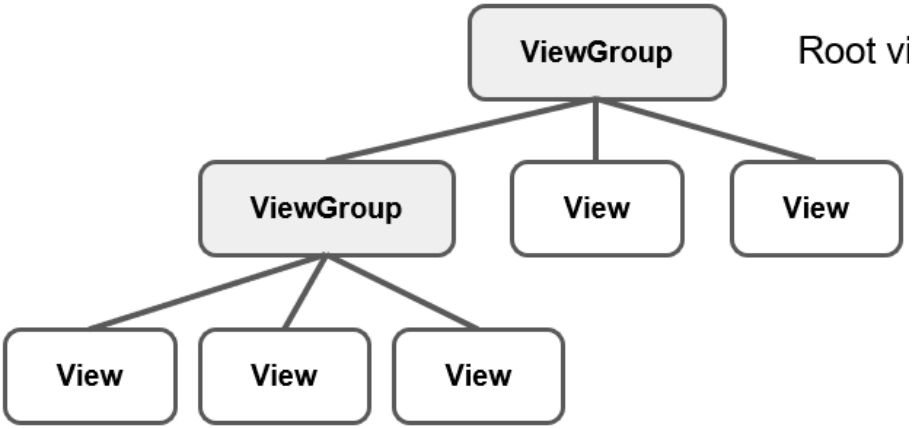
+ ViewGroup and ViewHierarchy

- Các lớp con của ViewGroup

- **ConstraintLayout:** Định vị trí các phần tử UI sử dụng những kết nối ràng buộc với những phần tử khác và các cạnh layout.
- **ScrollView:** Chứa một phần tử và cho phép cuộn.
- **RecyclerView:** Chứa danh sách các phần tử và cho phép cuộn bằng cách thêm và xóa phần tử động.

- **LinearLayout, ConstraintLayout, GridLayout, TableLayout, RelativeLaout, FrameLayout, …**

- Phân cấp viewgroups, views



- LinearLayout

- **XML**

```
<LinearLayout
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
    <Button
      ... />
    <TextView
      ... />
    <Button
      ... />
</LinearLayout
```

- **Java code**

```
LinearLayout linearL = new LinearLayout(this);
linearL.setOrientation(LinearLayout.VERTICAL);
TextView myText = new TextView(this);
myText.setText("Display this text!");
linearL.addView(myText);
setContentView(linearL);
// Set the width and height of a view:
LinearLayout.LayoutParams layoutParams =
  new Linear.LayoutParams(
     LayoutParams.MATCH_PARENT,
     LayoutParams.MATCH_CONTENT);
myView.setLayoutParams(layoutParams);
```

+ Event Handling

- Events là thứ gì đó xảy ra

- In UI: Click, tap, drag
- Device: DetectedActivity such as walking, driving, tilting
- Events are "noticed" by the Android system

- Event Handlers

- Trong một Method, Event Handler giúp kích hoạt sự kiện cụ thể để làm việc gì đó đáp ứng lại sự kiện đã xảy ra.
- Attach in XML

```
android:onClick="showToast"
```

- Implement in Java

```
public void showToast(View view) {
  String msg = "Hello Toast!";
  Toast toast = Toast.makeText(this, msg, duration);
  toast.show();
 }
}
```

- Alternative: Set click handler in Java

```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String msg = "Hello Toast!";
        Toast toast = Toast.makeText(this, msg, duration);
        toast.show();
        }
    });
```

- Resources and measurements

- Layout:
  R.layout.activity_main
  setContentView(R.layout.activity_main);
- View:
  R.id.recyclerview
  rv = (RecyclerView) findViewById(R.id.recyclerview);
- String:
  In Java: R.string.title
  In XML: android:text="@string/title"
- Density-independent Pixels (dp): for Views
- Scale-independent Pixels (sp): for text

01.3: Text và scrolling views

+ TextView

- TextView là lớp con của lớp View cho 1 hoặc nhiều dòng text.

- **XML**

```
<TextView
  android:id="@+id/article"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:autoLink="web"
  android:text="@string/article_text"/>
```

android:**text** - text to display
android:**textColor** - color of text
android:**textAppearance** - predefined style or theme
android:**textSize** - text size in sp
android:**textStyle** - normal, bold, italic, or bold|italic
android:**typeface** - normal, sans, serif, or monospace
android:**lineSpacingExtra** - extra space between lines in sp

- **Java code**

```
TextView myTextview = new TextView(this);
myTextView.setWidth(LayoutParams.MATCH_PARENT);
myTextView.setHeight(LayoutParams.WRAP_CONTENT);
myTextView.setMinLines(3);
myTextView.setText(R.string.my_story);
myTextView.append(userComment);
```

- EditView là lớp con của TextView với text có thể sửa.

+ ScrollView

- ScrollView là một lớp con của FrameLayout

- Sử dụng ***HorizontalScrollView*** cho cuộn ngang
- Sử dụng một ***RecyclerView*** cho các danh sách (lists)
  R.layout.activity_main
- **XML: one TextView**

```
<ScrollView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_below="@id/article_subheading">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    .../>
</ScrollView>
```

- **XML: view group**

```
<ScrollView ...
  <LinearLayout
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:orientation="vertical">
     <TextView
        android:id="@+id/article_subheading"
        .../>
     <TextView
        android:id="@+id/article" ... />
  </LinearLayout>
</ScrollView>
```

- **XML: image and button**

```
<ScrollView...>
   <LinearLayout...>
      <ImageView.../>
      <Button.../>
      <TextView.../>
   </LinearLayout>
</ScrollView>
```

02.1: Activities and Intents

+ Activities (high-level view)

- Activity là một thành phần ứng dụng

- Implementing Activities

- **1. Define layout in XML**

```
<?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">
  <TextView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Let's Shop for Food!" />
</RelativeLayout>
```

- **2. Define Activity Java class**

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

- **3. Connect activity with layout**

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

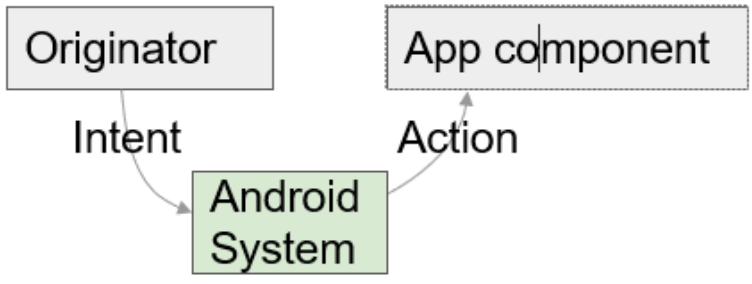
- **4. Declare activity in Android Manifest**

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

+ Intents

- **Intent** là mô tả cho một hoạt động được diễn ra

- **Intent** là một đối tượng được dùng để yêu cầu một hành động từ **app component** khác thông qua hệ thống Android.



- Intents có thể làm gì?

- Start an Activity
  - Click Butons khởi động một Activity mới cho nhập text
  - Click Share mở ra một app cho phép đăng bức ảnh
- Start an Service

    - Khởi tạo tải xuống file ở nền ứng dụng (background)
- Deliver Broadcast
    - Hệ thống thống báo cho mọi người là điện thoại đang được sạc

- Hai loại intents

- **Explicit Intent**
    - Bắt đầu một Activity cụ thể
      - Yêu cầu trả với sữa được giao bởi Nikita
      - Main activity bắt đầu ViewShoppingCart Activity
- **Implicit Intent**
    - Hỏi hệ thống để tìm Activity có thể xử lý yêu cầu (request)
      - Tìm một quán đang mở bán trà xanh
      - Click Share mở một bộ lựa chọn với danh sách cac ứng dụng

+ Starting Activities

- An explicit intent

- **Khởi tạo Intent**
    - Intent intent = new Intent(this, ActivityName.class);
- **Sử dụng Intent để bắt đầu Activity**
    - startActivity(intent);

- An implicit intent

- **Khởi tạo Intent**
    - Intent intent = new Intent(action, uri);
- **Sử dụng Intent để bắt đầu Activity**
    - startActivity(intent);
- **Show a web page**

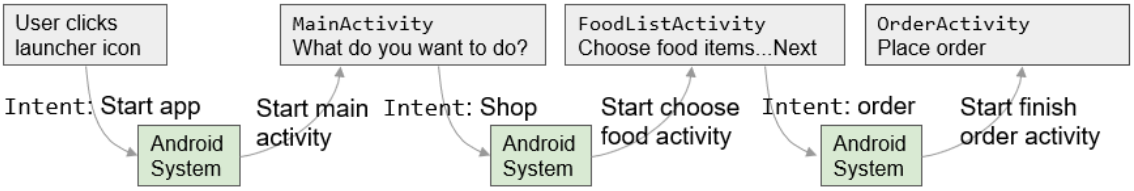
```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

- **Dial a phone number**

```
Uri uri = Uri.parse("tel:8005551234");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
startActivity(it);
```

- Cách mà Activities run

- Tất cả Activity instances được quản lý bởi Android runtime
- Được bắt đầu với “Intent”, một thông điệp tới Android runtime để chạy



+ Sending and Reveiving data

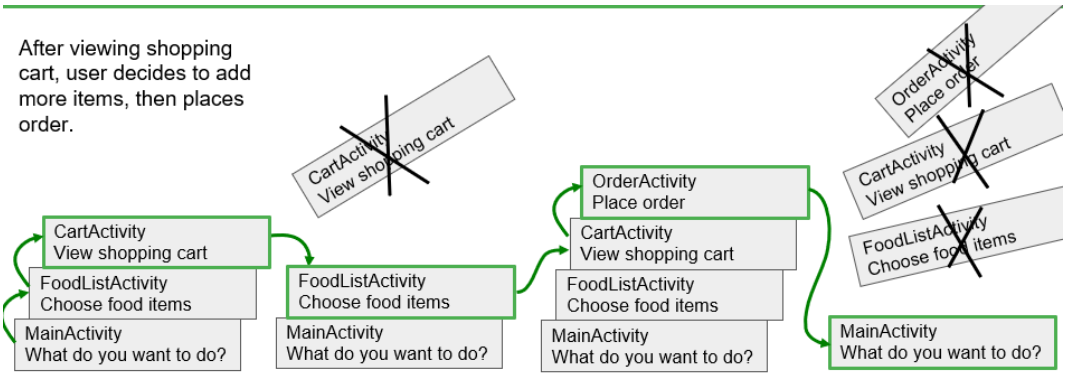
- Hai cách send data với intents

- **Data –** một đoạn thông tin chứa địa chỉ dữ liệu có thể được đại diện bởi một URI
- **Extras –** một hoặc nhiều đoạn thông tin như một lựa chọn của những cặp key-value trong một Bundle

- Gửi và nhận Dữ liệu: slides 26-37 trong PP **02.1**

+ Navigation

- Activity stack



- Set pareantACtivityName

```
<activity
  android:name=".ShowDinnerActivity"
  android:parentActivityName=".MainActivity" >
</activity>
```

02.2: Activity lifecycle and state

+ Activities lifecycle

- Một vòng đời của Activity

- Created (not visible yet)
- Started (visible)
- Resume (visible)
- Paused(partially invisible)
- Stopped (hidden)
- Destroyed (gone from memory)

+ Activities lifecycle callbacks

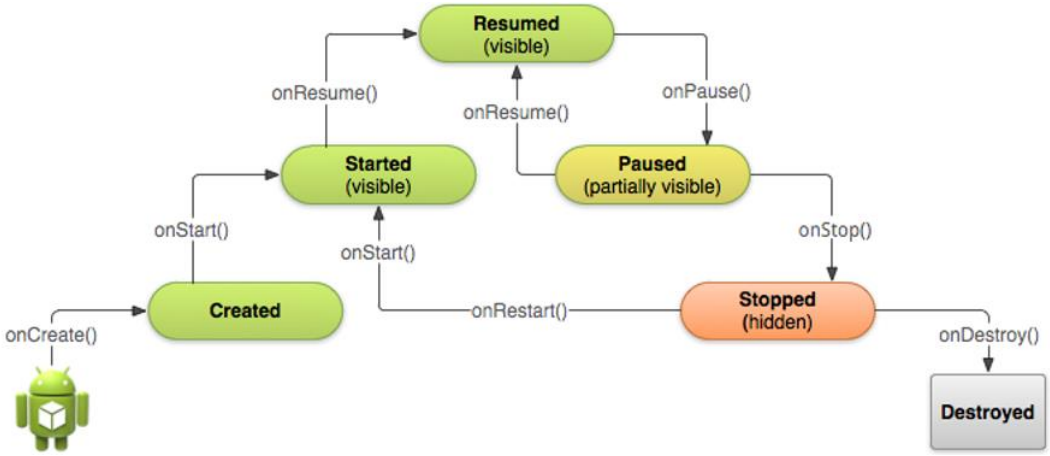
onCreate(Bundle savedInstanceState) - static initialization
onStart() - when Activity (screen) is becoming visible
onRestart() - called if Activity was stopped (calls onStart())

onResume() - start to interact with user

onPause() - about to resume PREVIOUS Activity

onStop() - no longer visible, but still exists and all state info preserved

onDestroy() - final call before Android system destroys Activity



+ Activities instance state

- Thông tin **State** được tạo ra trong khi Activity đang run, như là một counter, user text, animation progression.

- **State** bị mất đi khi thiết bị bị xoay, thay đổi ngôn ngữ, ấn vào nút Back hoặc hệ thống clear bộ nhớ.

+ Saving and restoring Activity state

- **Hệ thống chỉ lưu lại:**
  - Trạng thái của views với ID duy nhất (android:id) như là text được nhập trong EditText
  - Intent đã khởi động activity và dữ liệu ở dạng Extras
- **Saving instance state**
  - Thực thi ***onSaveInstanceState()*** trong Activity

```
@Override
public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    // Add information for saving HelloToast counter
    // to the to the outState bundle
    outState.putString("count",
            String.valueOf(mShowCount.getText()));
}
```

- **Restoring in onCreate()**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mShowCount = findViewById(R.id.show_count);
    if (savedInstanceState != null) {
```

```
        String count = savedInstanceState.getString("count");
        if (mShowCount != null)
            mShowCount.setText(count);
    }
}
@Override
public void onRestoreInstanceState (Bundle mySavedState) {
    super.onRestoreInstanceState(mySavedState);
    if (mySavedState != null) {
        String count = mySavedState.getString("count");
        if (count != null)
            mShowCount.setText(count);
    }
}
```

02.3: Implicit Intents

+ Overview

- Android Runtime giữ một danh sách Apps đã đăng ký
- App Chooser cho phép người dùng chọn Handler

+ Sending an implicit Intent

- **Tạo một Intent cho một hoạt động**

```
Intent intent = new Intent(Intent.ACTION_CALL_BUTTON);
```

- **Khởi động một Activity**

```
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

- **Ví dụ Gửi một implicit intent với data URI**

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:8005551234"));
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

- **Hiển thị một web page**

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

- **Quay một số điện thoại**

```
Uri uri = Uri.parse("tel:8005551234");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
```

```
startActivity(it);
```

- **Gửi một imlplicit intent với extras**

```
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
String query = edittext.getText().toString();
intent.putExtra(SearchManager.QUERY, query));
if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
}
```

- **Tạo một Intent cho một hoạt động**

```
Intent intent = new Intent(Intent.ACTION_CREATE_DOCUMENT);
```

- **Đặt mime type và phân loại cho thông tin được thêm**

```
intent.setType("application/pdf"); // set MIME type
intent.addCategory(Intent.CATEGORY_OPENABLE);
if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(myIntent,
        ACTIVITY_REQUEST_CREATE_FILE);
}
```

+ Receiving an implicit Intent

- **Đăng ký App để nhận Intent**

Khai báo một hoặc nhiều Intent filters choActivity trong Manifest.xml Filter thông báo khả năng của Activity để chấp nhận implicit Intent Filter đặt điều kiện trên Intent mà Activity chấp nhận

- **Intent filter trong AndroidManifest.xml**

```
<activity android:name="ShareActivity">
 <intent-filter>
  <action android:name="android.intent.action.SEND"/>
  <category android:name="android.intent.category.DEFAULT"/>
  <data android:mimeType="text/plain"/>
 </intent-filter>
</activity>
```

- **Intent filters**
  - **action** – khớp với một hoặc nhiều hành động cố định
    - android.intent.action.VIEW - matches any Intent with ACTION_VIEW
    - android.intent.action.SEND - matches any Intent with ACTION_SEND
  - **category** – thông tin thêm vào **(list of categories)**
    - android.intent.category.BROWSABLE - can be started by web browser

    - android.intent.category.LAUNCHER - Show activity as launcher icon
  - **data -** Filter on data URIs, MIME type
    - android:scheme="https" - require URIs to be https protocol
    - android:host="developer.android.com" - only accept an Intent from specified hosts
    - android:mimeType="text/plain" - limit the acceptable types of documents

+ Note

- Một Activity có thể có nhiều Filters

- Một filter có thể có nhiều actions và data

04.2: Input control

+ Overview

- Ví dụ của input controls

- Freedom text and numbers: EditText (using keyboard)
- Providung choices: CheckBox, RadioButton, Spinner
- Swiching on/off: Toggle, Switch
- Choosing value in range of value: SeekBar

+ View focus

- Focus được chỉ định bởi

- User tapping a View
- App hướng dẫn người dùng từ một text input control đến sử dụng tiếp **Return**, **Tab**, or **arrow keys**
- Calling requestFocus() trên bất kì View được focusable

- Clickable so với focusable

- **Clickable** – View có thể phản hồi khi được click hoặc tap
- **Focusable** – View có thể thu lấy focus để chấp nhận input

- Guiding focus

- **Specify ordering in XML**

  android:id="@+id/top"
  android:focusable="true"
  android:nextFocusDown="@+id/bottom"

- Cài đặt focus rõ ràng

Use methods of the **View** class to set focus

- **setFocusable()** sets whether a view can have focus
- **requestFocus()** gives focus to a specific view
- **setOnFocusChangeListeners()** sets listener for when view gains or loses focus
- **onFocusChanged()** called when focus on a view changes

+ Freedom text and numbers

Slide 17-24 PP 02.1

+ Providung choices

Slide 25-31 PP 02.1

04.3: Menu and pickers

+ Overview

- Types of Menus

- App bar with options menu
- Context menu
- Contextual action bar
- Popup menu

- Dialogs and pickers

- Alert dialog (Hội thoại cảnh báo)
- Date picker (Bảng chọn ngày)
- Time picker (Bảng chọn giờ)

+ App Bar with Options Menu

+ Adding Options Menu

- Thêm Menu item attributes

```
<item
      android:id="@+id/action_favorites"
      android:icon="@drawable/ic_favorite"
      android:orderInCategory="30"
      android:title="@string/action_favorites"
      app:showAsAction="ifRoom" />

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.action_settings:
      showSettings();
      return true;
```

```
            case R.id.action_favorites:
              showFavorites();
              return true;
            default:
                return super.onOptionsItemSelected(item);
            }
        }
```

+ Contextual menus

- Floating context menu – long-press on a View
  - User can modify View or use it in some fashion
  - User performs action on one View at a time
- Contextual action mode – temporary action bar in place of or underneath app bar
  - Action items affect the selected View element(s)
  - User can perform action on multiple View elements at once

+ Floating context menu

1. Create XML menu resource file and assign appearance and position attributes
2. Register View using registerForContextMenu()
3. Implement onCreateContextMenu() in Activity to inflate menu
4. Implement onContextItemSelected() to handle menu item clicks
5. Create method to perform action for each context menu item

+ Contextual Action Bar

- Action mode có một lifecycle

- Start it with **startActionMode()**, for example, in the listener
- **ActionMode.Callback** interface provides lifecycle methods you override:
  - **onCreateActionMode(ActionMode, Menu)** once on initial creation
  - **onPrepareActionMode(ActionMode, Menu)** after creation and any time ActionMode is invalidated
  - **onActionItemClicked(ActionMode, MenuItem)** any time contextual action button is clicked
  - **onDestroyActionMode(ActionMode)** when action mode is closed

- Tạo Action mode

Slide 30-36 PP 04.3

+ Popup Menu

+ Dialogs

+ Pickers

04.4: User navigation

+ Back Naviagtion

+ Hierarchical Navigation

- **Parent screen** - Screen that enables navigation down to child screens, such as home screen and main Activity
- **Collection sibling** - Screen enabling navigation to a collection of child screens, such as a list of headlines
- **Section sibling** - Screen with content, such as a story

+ Descendant Navigation

+ Navigation Drawer

1. Icon in app bar
2. Header
3. Menu items

- Implement navigation drawer

1. Implement com.android.support:design:28.0.0
2. Implement com.android.support:support-core-utils:28.0.0
3. Use DrawerLayout for the MainActivity
4. setDisplayHomeAsUpEnable and setHomeButtonEnable for the support action bar
5. Create an ActionBarDrawerToggle object and link it with the DrawerLayout
    a. onPostCreate(Bundle) -> sync the toggle state
    b. onConfigurationChanged
    c. onOptionsItemSelected

+ Ancestral Navigation

- Khai báo AndroidManifest

```
<activity android:name=".OrderActivity"
  android:label="@string/title_activity_order"
  android:parentActivityName="com.example.android.
                optionsmenuorderactivity.MainActivity">
  <meta-data
    android:name="android.support.PARENT_ACTIVITY"
```

```
            android:value=".MainActivity"/>
        </activity>
```

+ Lateral Navigation (Điều hướng các bên)

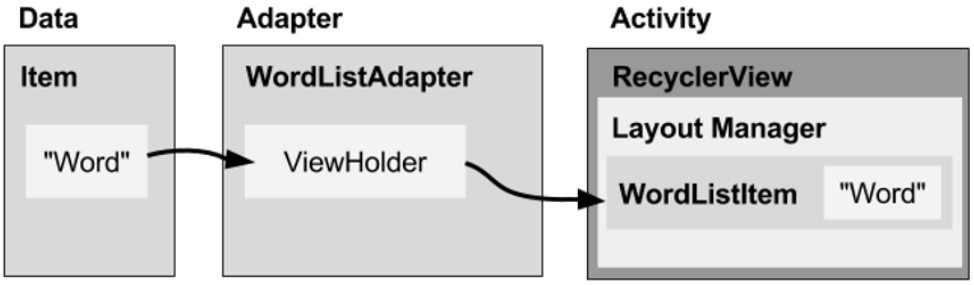
1. Define the tab layout using **TabLayout**
2. Implement a Fragment and its layout for each tab
3. Implement a PagerAdapter from **FragmentPagerAdapter** or **FragmentStatePagerAdapter**
4. Create an instance of the tab layout
5. Use PagerAdapter to manage screens (each screen is a Fragment)
6. Set a listener to determine which tab is tapped

04.5: RecyclerView

+ Overview

- **RecyclerView** là một vùng chứa có thể cuộn cho lượng dữ liệu lớn
- **Lợi ích:**
  - Sử dụng và tái sử dụng số lượng giới hạn của View elements
  - Cập nhật dữ liệu thay đổi nhanh chóng

+ RecyclerView Components



- **Data**
- **RecyclerView** scrolling list for list items—**RecyclerView**
- **Layout** for one item of data—XML file
- **Layout manage**r handles the organization of UI components in a View - **Recyclerview.LayoutManager**
- **Adapter** connects data to the RecyclerView - **RecyclerView.Adapter**
- **ViewHolder** has view information for displaying one item - **RecyclerView.ViewHolder**

+ Implementing RecyclerView

1. Add **RecyclerView** dependency to build.gradle if needed

```
dependencies {
  ...
  compile 'com.android.support:recyclerview-v7:26.1.0'
  ... }
```

2. Add **RecyclerView** to layout

<**android.support.v7.widget.RecyclerView**

android:id="@+id/recyclerview"

android:layout_width="match_parent"

android:layout_height="match_parent">

</android.support.v7.widget.RecyclerView>

3. **Create XML layout for item**

<LinearLayout …>

<TextView

android:id="@+id/word"

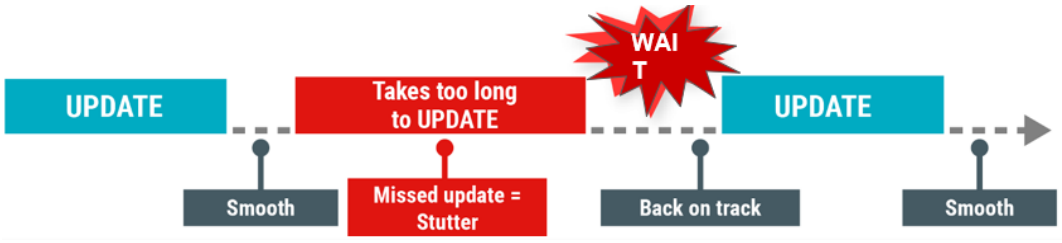
style="@style/word_title" />

</LinearLayout>

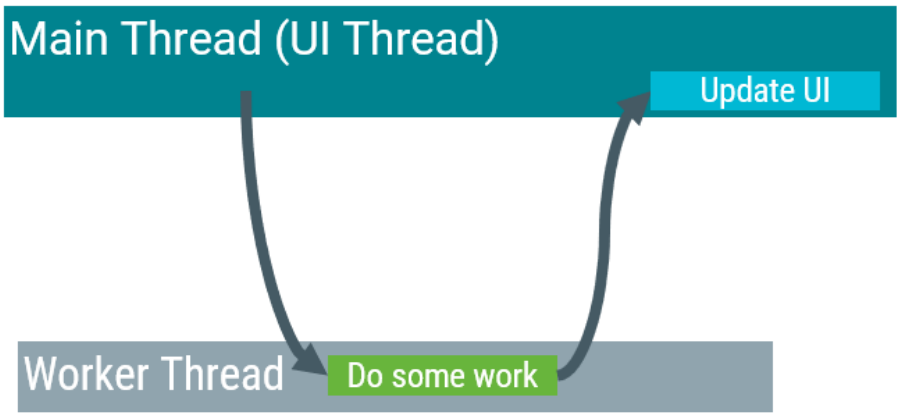
4. Extend **RecyclerView.Adapter**
5. Extend **RecyclerView.ViewHolder**
6. In **Activity onCreate(),** create **RecyclerView** with adapter and layout manager

07.1: AsyncTask và AsyncTaskLoader

+ Threads



- Background threads

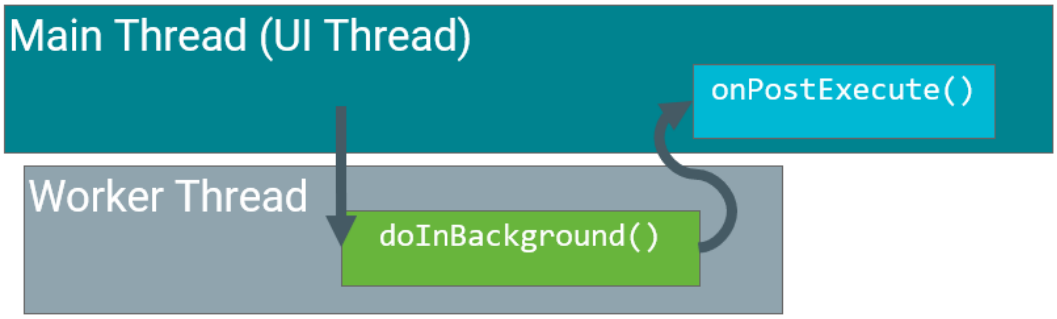


- Two rules for Android threads

- Do not block the UI thread
- Do not access the Android UI toolkit from outside the UI thread

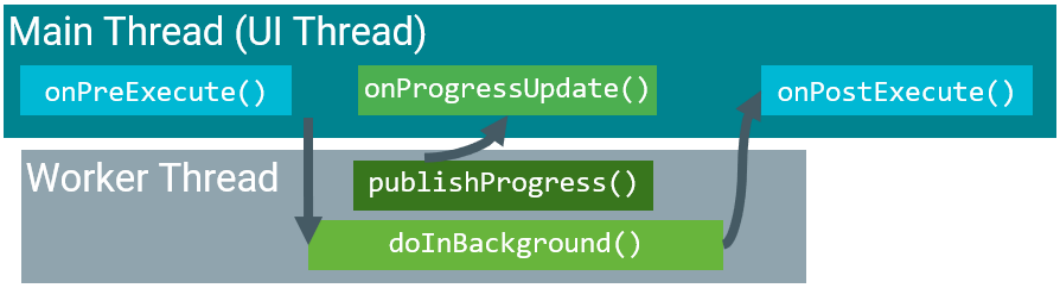
+ AsyncTask

- Override two methods



- doInBackground()—runs on a background thread
  - All the work to happen in the background
- onPostExecute()—runs on main thread when work done
  - Process results
  - Publish results to the UI

- Helper methods



- onPreExecute()
  - Runs on the main thread
  - Sets up the task
- onProgressUpdate()
  - Runs on the main thread
  - receives calls from publishProgress() from background thread

- Override two methods

1. Subclass AsyncTask
2. Provide data type sent to doInBackground()
3. Provide data type of progress units for onProgressUpdate()
4. Provide data type of result for onPostExecute()

```
private class MyAsyncTask
    extends AsyncTask<String, Integer, Bitmap> {...}
```

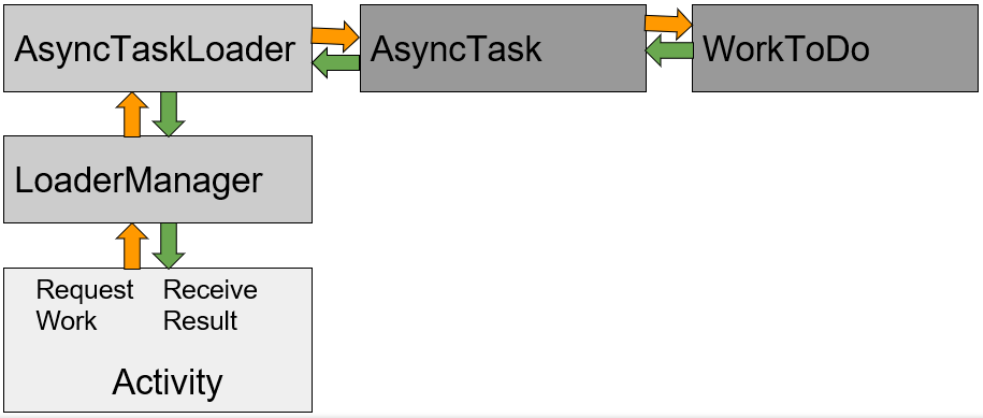
doInBackground() onProgressUpdate() onPostExecute()

- String—could be query, URI for filename
- Integer—percentage completed, steps done
- Bitmap—an image to be displayed

+ Loaders

- Cung cấp tải dữ liệu không đồng bộ
- Kết nối lại với Activity sau khi thay đổi cấu hình
- Có thể theo dõi các thay đổi trong nguồn dữ liệu và cung cấp dữ liệu mới
- Các lệnh gọi lại được triển khai trong Activity
- Nhiều loại Loaders có sẵn

+ Implementing AsyncTaskLoader



+ AsyncTask => AsyncTaskLoader

```
doInBackground()  ⟶  loadInBackground()
onPostExecute()   ⟶  onLoadFinished()
```

1. Subclass **AsyncTaskLoader**
2. Implement constructor
3. loadInBackground()
4. onStartLoading()

07.2: Internet connection

+ Permissions in AndroidManifest

- **Internet**

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

- **Check Network state**

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

+ Manage Network Connection

- Getting Network information

- **ConnectivityManager**
  - Answers queries about the state of network connectivity
  - Notifies applications when network connectivity changes
- **NetworkInfo**
  - Describes status of a network interface of a given type
  - Mobile or Wi-Fi

+ Worker Thread

- Use Worker Thread

- **AsyncTask -** very short task, or no result returned to UI
- **AsyncTaskLoader -** for longer tasks, returns result to UI
- **Background Service -** later chapter

- Background work

1. Create URI
2. Make HTTP Connection
3. Download Data

+ Create URI

```
//Constants for Parameters
final String BASE_URL =
   "https://www.googleapis.com/books/v1/volumes?";
final String QUERY_PARAM = "q";
final String MAX_RESULTS = "maxResults";
final String PRINT_TYPE = "printType";

Uri builtURI = Uri.parse(BASE_URL).buildUpon()
     .appendQueryParameter(QUERY_PARAM, "pride+prejudice")
     .appendQueryParameter(MAX_RESULTS, "10")
     .appendQueryParameter(PRINT_TYPE, "books")
     .build();
URL requestURL = new URL(builtURI.toString());
```

+ HTTP Client Connection

- Create a HttpURLConnection

```
HttpURLConnection conn =
  (HttpURLConnection) requestURL.openConnection();
```

- Configure connection

```
conn.setReadTimeout(10000 /* milliseconds */);
conn.setConnectTimeout(15000 /* milliseconds */);
conn.setRequestMethod("GET");
```

```
conn.setDoInput(true);
```

- **Connect and get Response**

```
conn.connect();
int response = conn.getResponseCode();

InputStream is = conn.getInputStream();
String contentAsString = convertIsToString(is, len);
return contentAsString;
```

- **Close Connection and Stream**

```
} finally {
        conn.disconnect();
    if (is != null) {
       is.close();
    }
}
```

+ Convert Response to String

- **Reader**

```
public String convertIsToString(InputStream stream, int len)
   throws IOException, UnsupportedEncodingException {

   Reader reader = null;
   reader = new InputStreamReader(stream, "UTF-8");
   char[] buffer = new char[len];
   reader.read(buffer);
   return new String(buffer);
}
```

- **BufferedReader**

```
StringBuilder builder = new StringBuilder();
BufferedReader reader =
   new BufferedReader(new InputStreamReader(inputStream));
String line;
while ((line = reader.readLine()) != null) {
  builder.append(line + "\n");
}
if (builder.length() == 0) {
  return null;
}
resultString = builder.toString();
```

+ HTTP Client Connection Libraries

- **Volley**

```
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";
```

```
StringRequest stringRequest = new
StringRequest(Request.Method.GET, url,
        new Response.Listener<String>() {
   @Override
   public void onResponse(String response) {
      // Do something with response
   }
}, new Response.ErrorListener() {
   @Override
   public void onErrorResponse(VolleyError error) {}
});
queue.add(stringRequest);
```

- **OkHttp**

```
OkHttpClient client = new OkHttpClient();
Request request = new Request.Builder()
   .url("http://publicobject.com/helloworld.txt").build();
client.newCall(request).enqueue(new Callback() {
   @Override
   public void onResponse(Call call, final Response response)
      throws IOException {
      try {
         String responseData = response.body().string();
         JSONObject json = new JSONObject(responseData);
         final String owner = json.getString("name");
      } catch (JSONException e) {}
   }
});
```

+ Parse Results

- Triển khai phương thức nhận và xử lý kết quả (onPostExecute ())
- Phản hồi thường là JSON hoặc XML
- Phân tích kết quả bằng cách sử dụng các lớp trợ giúp

  JSONObject, JSONArray

  XMLPullParser - phân tích cú pháp XML

- **JSON basics**

```
{
 "population":1,252,000,000,
 "country":"India",
 "cities":["New Delhi","Mumbai","Kolkata","Chennai"]
}
```

- **JSONOjects basics**

```
JSONObject jsonObject = new JSONObject(response);
String nameOfCountry = (String) jsonObject.get("country");
```

```
long population = (Long) jsonObject.get("population");
JSONArray listOfCities = (JSONArray) jsonObject.get("cities");
Iterator<String> iterator = listOfCities.iterator();
while (iterator.hasNext()) {
   // do something
}
```

- **JSON example**

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}

//Get "onclick" value of the 3rd item in the "menuitem" array
JSONObject data = new JSONObject(responseString);
JSONArray menuItemArray =
      data.getJSONArray("menuitem");
JSONObject thirdItem =
      menuItemArray.getJSONObject(2);
String onClick = thirdItem.getString("onclick");
```

07.3: Broadcasts

+ Broadcasts

- **Broadcasts** là những thông điệp được gửi bởi Android system và những Android app khác khi một event of interest xảy ra.
- **Broadcasts** được gói trong một **Intent** object.

VD: android.intent.action.HEATSET_PLUG

- **Các loại Broadcasts:**
  - **System broadcast:** được gửi bởi Android system khi một system event diễn ra, có thể ảnh hưởng đến app.
    - **ACTION_BOOT_COMPLETED**
    - **ACTION_POWER_CONNECTED**
  - **Custom broadcast:** là broadcasts app send outs, tương tự như Android system.
    - Một broadcast giúp app(s) biết đang dowload data, và data đó sẵn sàng để sử dụng.

+ Send a custom broadcast

- **Local broadcast** là broadcast gửi đến chính app đó.
- Không lo về vấn đề security
- Ví dụ:
  - To get an instance of **LocalBroadcastManager**.
  - Call **sendBroadcast()** on the instance.

    LocalBroadcastManager.getInstance(this)
    .sendBroadcast(**customBroadcastIntent**);
- **Custum broadcasts**
  - Sender và Receiver phải đồng ý tên duy nhất cho intent (action name).
  - Định nghĩa trong activity và broadcast receiver.
- Ví dụ:

private static final String ACTION_CUSTOM_BROADCAST =
"com.example.android.powerreceiver.**ACTION_CUSTOM_BROADCAST**";
customBroadcastIntent = new Intent( );

+ Broadcasts Receivers

- **Broadcast receivers** là **app components**
- Chúng đăng ký cho đa dạng system broadcast và/hoặc custom broadcast
- Chúng thông báo (thông qua **Intent**)
  - Với hệ thống, khi một system events diễn ra mà app đăng ký
  - Với app khác, bao gồm cả của bạn nếu app của bạn đăng ký cho custom event.
- **Đăng ký broadcast receiver:** 2 cách
  - Static receivers
    - Đăng ký trong AndroidManifest.xml, và cũng được gọi là Manifest-declared receivers.
  - Dynamic receivers
    - Đăng ký sử dụng app or activities' context in your Java files, và cũng được gọi là Context-registered receivers.
- **Nhận một system broadcast**

+ Thực hiện Broad Receivers

- **Tạo một Broadcast Receiver**
  - Lớp con của **BroadcastReceiver** và ghi đè phương thức **onReceive()**
  - Đăng ký broadcast receiver và chỉ định các intent-filter
    - Statically, in the **Manifest**.
    - Dynamically, with **registerReceiver()**.
- **Intent-filters**
  - **Intent-filters** chỉ định những loại intent một broadcast receiver có thể nhận được. Chúng lọc intents đến dựa vào giá trị **Intent** như **action.**
  - Để thêm một **intent-filter**:

    - Với **AndroidManifest.xml**, sử dụng thẻ **<intent-filter>**
    - Với Java file, sử dụng **IntentFilter** object.

- Lớp con một broadcast receiver

- **File > New > Other > BroadcastReceiver**
- **Ví dụ:**

```
public class CustomReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    // This method is called when the BroadcastReceiver
    //  is receiving an Intent broadcast.
    throw new UnsupportedOperationException("Not yet
implemented");
  }
}
```

- Thực hiện **onReceiver()**

- **Ví dụ:**

```
@Override
public void onReceive(Context context, Intent intent) {
  String intentAction = intent.getAction();
  switch (intentAction){
    case Intent.ACTION_POWER_CONNECTED:
      break;
    case Intent.ACTION_POWER_DISCONNECTED:
      break;
  }
}
```

- Đăng ký **statically** trong **AndroidManifest.xml**

- <receiver> **element inside** <application> tag.
- **Ví dụ:**

```
<receiver
  android:name=".CustomReceiver"
  android:enabled="true"
  android:exported="true">
  <intent-filter>
    <action
android:name="android.intent.action.BOOT_COMPLETED"/>
  </intent-filter>
</receiver>
```

- Đăng ký **dynamically**

- Register your receiver in **onCreate()** or **onResume()**.

```
// Register the receiver using the activity context.
  this.registerReceiver(mReceiver, filter);
```

- Unregister in **onDestroy()** or **onPause()**.

// **Unregister** the receiver
this.**unregisterReceiver**(**mReceiver**);

- Đăng ký một **Local broadcast receiver**
  - Đăng ký dynamically, bởi vì đăng ký tĩnh trong manifest là không thể
  - Đăng ký một **local broadcast**
    - Get an instance of **LocalBroadcastManager.**
    - Call **registerReceiver().**
  - Ví dụ:

LocalBroadcastManager.getInstance(this).registerReceiver (mReceiver,
new IntentFilter(CustomReceiver.**ACTION_CUSTOM_BROADCAST**));

- Hủy đăng ký một **Local broadcast receiver**
  - Hủy đăng ký một **local broadcast**
    - Get an instance of **LocalBroadcastManager.**
    - Call **LocalBroadcastManager.unregisterReceiver().**
  - Ví dụ:

LocalBroadcastManager.getInstance(**this**).unregisterReceiver(**mReceiver**);

07.4: Notifications

+ Notification

- **Notifications** là những thông điệp được hiển thị ngoài app UI bình thường.
- Một vài ví dụ notification:
  - Notification drawer
  - App icon badge (notification dot)

+ Notification channels

- Sử dụng để tạo channel người dùng tùy chỉnh cho mỗi loại notification được hiển thị.
- Nhiều notification có thể được nhóm lại thành một channel.
- Đặt notification behaviour như âm thanh, đèn, …
- Notification channel xuất hiện như Categories dưới App notifications trong Device Settings.

+ Creating a Notification channel

- **Notification channel instance** được tạo bởi **NotificationChannel** constructor.
- Bạn phải chỉ định:
  - Một ID duy nhất trong your package.
  - Tên User hiển thị của channel.
  - Mức độ quan trọng của channel.
    - IMPORTANCE_MIN(0) đến IMPORTANCE_HIGH(4)
    - MIN – LOW – DEFAULT – HIGH
- Ví dụ:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
   NotificationChannel notificationChannel =
      new NotificationChannel(CHANNEL_ID, “Mascot Notification”,
      NotificationManager.IMPORTANCE_DEFAULT);
      setPriority(NotificationCompat.PRIORITY_HIGH);
}
```

+ Creating Notifications

- **Notification** được tạo bởi sử dụng lớp **NotificationCompat.Builder.**
- Chuyển app context and notification channel ID cho contructor.
- Ví dụ:

```
NotificationCompat.Builder mBuilder = new
      NotificationCompat.Builder(this, CHANNEL_ID);
```

- Cài đặt nội dung notification
  - Một icon nhỏ, cài đặt bởi **setSmallIcon().**
  - Một tiêu đề, cài đặt bởi **setContentTitle().**
  - Nội dung, cài đặt bởi **setContextText().**
- Ví dụ:

```
NotificationCompat.Builder mBuilder =
   new NotificationCompat.Builder(this, CHANNEL_ID)
      .setSmallIcon(R.drawable.android_icon)
      .setContentTitle("You've been notified!")
      .setContentText("This is your notification text.");
```

+ Tap action and Action buttons

- Thêm **notification tap action**
  - Tất cả **Notifications** phải phản hồi khi được khi tap, thường là launch app.
  - Đặt nội dung intent sử dụng phương thức **setContentIntent().**
  - Chuyền **Intent** được gói trong một **PendingIntent** object.
- **Notification action buttons**
  - Thêm action button, chuyền một **PendingIntent** tới giao thức **addAction().**
  - Một **PendingIntent** là một mô tả của Intent and target action để thực hiện với nó.
    - ***PendingIntent.getActivity()***
    - ***PendingIntent.getBroadcast()***
    - ***PendingIntent.getService()***
- Ví dụ:

```
// Create intent
      Intent notificationIntent =
            new Intent(this, MainActivity.class);
// Create PendingIntent
```

```
PendingIntent notificationPendingIntent =
   PendingIntent.getActivity(
        this,           // Application context
        NOTIFICATION_ID,        // Request code
        notificationIntent,          // Intent to be delivered
     PendingIntent.FLAG_UPDATE_CURRENT); //PendingIntent flag
// Add to notification builder
        setContentIntent(notificationPendingIntent);
// Add action buttons
        .addAction(R.drawable.ic_color_lens_black_24dp,
        "R.string.label",
        notificationPendingIntent);
// Setting styles
        mNotifyBuilder.setStyle(new NotificationCompat.BigPictureStyle()
            .bigPicture(myBitmapImage)
            .setBigContentTitle("Notification!"));
// BigImage: sử dụng class phụ trợ:
        NotificationCompat.BigPictureStyle
```

+ Delivering Notifications

- Delivering notifications

- Sử dụng lớp **NotificationManager** để gửi notifications.
  - Create an instance of **NotificationManager**.
  - Call **notify()** to deliver the notification.

- Instantiate NotificationManager

- Gọi **getSystemService(),** chuyền constant NOTIFICATION_SERVICE

```
mNotifyManager = (NotificationManager)
   getSystemService(NOTIFICATION_SERVICE);
```

- Gửi notification

- Gọi **notify()** để gửi notification, chuyền trong 2 giá trị sau:
  - Một **notification ID**, cái mà được dùng để cập nhật hoặc hủy notification.
  - **NotificationCompat** object bạn đã tạo ra sử dụng **NotificationCompat.Builder** object.

```
mNotifyManager.notify(NOTIFICATION_ID,  myNotification);
```

+ Managing Notifications

- Update Notifications

- Cập nhật thông báo bằng cách thay đổi và thêm một số nội dung của thông báo.
- Đưa ra thông báo với các thông số được cập nhật bằng trình tạo.
- Cuộc gọi **notify()** chuyển trong cùng một ID thông báo.
  - Nếu thông báo trước đó vẫn hiển thị, hãy cập nhật hệ thống.

- Nếu thông báo trước đó đã bị loại bỏ, thông báo mới sẽ được gửi.

- Canceling Notifications

Thông báo vẫn hiển thị cho đến khi:

- Người dùng loại bỏ nó bằng cách vuốt hoặc sử dụng "Clear All".
- Gọi **setAutoCancel()** khi tạo thông báo, xóa thông báo khỏi thanh trạng thái khi người dùng nhấp vào.
- App gọi **cancel()** hoặc **cancelALL()** trên **NotificationManager**. mNotifyManager.cancel (**NOTIFICATION_ID**);

- Summary Steps

- getSystemService(**NOTIFICATION_SERVICE**) -> Manager
- new **NotificationChannel** and set its properties
- **Manager** -> **CreateNotificationChannel**
- **New NotificationCompat.Builder** (set style if necessary)
- **Builder.build()** -> **Manager.Notify**

07.5: Services

+ Services

- **Services** rất phức tạp
- **Service** là một app component mà có thể chạy lâu dài dưới nền ứng dụng và không cung cấp giao diện người dùng.
- Có vài cách để cấu hình một service
- Slide chỉ có những thông tin giới thiệu

+ Characteristics của services:

- Bắt đầu với một **Intent**.
- Có thể tiếp tục chạy khi người dùng chuyển đổi ứng dụng.
- Vòng đời - mà bạn phải quản lý.
- Các ứng dụng khác có thể sử dụng dịch vụ - quản lý quyền.
- Chạy trong chuỗi chính của quá trình lưu trữ của nó.

+ Form of services:

- Started

- Started with **startService().**
- Runs indefinitely until it stops itself.
- Usually does not update the UI.

- Bound

- Offers a client-server interface that allows components to interact with the service.
- Clients send requests and get results.
- Started with bindService().
- Ends when all clients unbind.

+ Services and threads

- Dù services được chia từ UI, chúng có thể chạy ở main thread như mặc định (trừ **IntentService**).
- Giảm tải công việc đòi hỏi nhiều CPU xuống một luồng riêng biệt trong service.
- Sử dụng **broadcast receiver** để update the app.

+ Foregroud and Background services

- **Foreground**: Chạy trong nền nhưng yêu cầu người dùng chủ động biết rằng nó tồn tại - ví dụ: máy nghe nhạc sử dụng dịch vụ âm nhạc
    - Ưu tiên cao hơn các dịch vụ nền vì người dùng sẽ nhận thấy sự vắng mặt của nó - không có khả năng bị giết bởi hệ thống
    - Phải cung cấp thông báo mà người dùng không thể loại bỏ khi dịch vụ đang chạy.
- **Background**
    - Một **foreground app**, có thể tạo và chạy cả **foreground and background services.**
    - Khi một ứng dụng chuyển sang **background**, hệ thống sẽ dừng các **foreground services** của ứng dụng đó.
    - Phương thức **startService()** hiện ném **IllegalStateException** nếu một ứng dụng đang nhắm mục tiêu API 26.
    - Những hạn chế này không ảnh hưởng đến các dịch vụ nền trước hoặc các dịch vụ ràng buộc.

+ Creating and Stopping a service

- **Creating**
    - <service android:name=".ExampleService" />
    - Manage permissions.
    - Subclass **IntentService** or **Service** class.
    - Implement lifecycle methods.
    - Start service from **Activity**.
    - Make sure service is stoppable.
- **Stopping**
    - A **started service** must manage its own lifecycle
    - If not stopped, will keep running and consuming resources
    - The service must stop itself by calling **stopSelf()**
    - Another component can stop it by calling **stopService()**
    - **Bound service** is destroyed when all clients unbound
    - **IntentService** is destroyed after **onHandleIntent()** returns

09.1: Shared Preferences

+ Shared Preferences

- Đọc và ghi một lượng nhỏ dữ liệu nguyên thủy dưới dạng các cặp **key/value** vào một tệp trên bộ nhớ của thiết bị
- Lớp **SharedPreference** cung cấp các API để đọc, ghi và quản lý dữ liệu này.
- Lưu dữ liệu bằng **onPause()** khôi phục bằng **onCreate()**

+ Shared Preferences vs. Saved Instance State

+ Creating Shared Preferences

- Chỉ cần một tệp **Shared Preferences** cho mỗi app.
- Đặt tên nó bằng tên gói ứng dụng của bạn – duy nhất và dễ kết hợp với ứng dụng.
- Đối số **MODE** cho **getSharedPreferences()** là để tương thích ngược - chỉ sử dụng **MODE_PRIVATE** để bảo mật
- Ví dụ:

```
private String sharedPrefFile = "com.example.android.hellosharedprefs";
mPreferences = getSharedPreferences(sharedPrefFile, MODE_PRIVATE);
```

+ Saving Shared Preferences

- **SharedPreferences.Editor** interface.
- Quan tâm đến tất cả các hoạt động tệp
- Đặt phương thức ghi đè nếu key tồn tại
- **apply()** lưu không đồng bộ và an toàn
- Ví dụ:

```
@Override
protected void onPause() {
  super.onPause();
  SharedPreferences.Editor preferencesEditor =
     mPreferences.edit();
  preferencesEditor.putInt("count", mCount);
  preferencesEditor.putInt("color", mCurrentColor);
  preferencesEditor.apply();
}
```

+ Restoring Shared Preferences

- Khôi phục trong **onCreate()** trong **Activity**.
- Các phương thức **Get** có hai đối số - **key/value** mặc định nếu không tìm thấy key.
- Sử dụng đối số mặc định để bạn không phải kiểm tra xem tùy chọn có tồn tại trong tệp hay không.
- Ví dụ:

```
mPreferences = getSharedPreferences(sharedPrefFile, MODE_PRIVATE);
```

```
if (savedInstanceState != null) {
   mCount = mPreferences.getInt("count", 1);
   mShowCount.setText(String.format("%s", mCount));
   mCurrentColor = mPreferences.getInt("color", mCurrentColor);
   mShowCount.setBackgroundColor(mCurrentColor);
   mNewText = mPreferences.getString("text", "");
} else { … }
```

+ Clearing

- Gọi phương thức **clear()** bằng **SharedPreferences.Editor** và lưu thay đổi.
- Phối hợp gọi **put** và **clear.** Tuy nhiên, khi **apply()**, **clear()** luôn done trước, không phụ thuộc vào thứ tự.
- Ví dụ:

```
SharedPreferences.Editor preferencesEditor = mPreferences.edit();
preferencesEditor.clear();
preferencesEditor.apply();
```

+ Listening to Changes

- Listening to changes
  - Thực thi interface **SharedPreference.OnSharedPreferenceChangeListener.**
  - Đăng ký listener với **registerOnSharedPreferenceChangeListener()**
  - Đăng ký hoặc hủy đăng ký với **onResume()** và **onPause()**.
  - Thực thi bằng gọi lại **onSharedPreferenceChanged().**

```
// Interface and callback
public class SettingsActivity extends AppCompatActivity
   implements OnSharedPreferenceChangeListener { ...
   public void onSharedPreferenceChanged(
      SharedPreferences sharedPreferences, String key) {
      if (key.equals(MY_KEY)) {
         // Do something
      }
   }
}
// Creating and registering listener
SharedPreferences.OnSharedPreferenceChangeListener listener =
   new SharedPreferences.OnSharedPreferenceChangeListener() {
  public void onSharedPreferenceChanged(
    SharedPreferences prefs, String key) {
      // Implement listener here
  }
};
prefs.registerOnSharedPreferenceChangeListener(listener);
```

- Cần một STRONG references to the listener
    - Khi đăng ký trình **listener**, trình quản lý ưu tiên không lưu trữ một tham chiếu mạnh mẽ đến trình **listener**.
    - Bạn phải lưu trữ một tham chiếu mạnh mẽ cho **listener**, nếu không nó sẽ dễ bị thu gom.
    - Giữ một tham chiếu đến **listener** trong dữ liệu cá thể của một đối tượng sẽ tồn tại miễn là bạn cần **listener.**

09.2: App settings

+ App settings

- User có thể đặt features và behaviour của app.
- Những giá trị ít khi thay đổi và liên quan đến hầu hết người dùng.
- Nếu giá trị thường xuyên thanh đổi, sử dụng **menu** hoặc **nav drawer**
    - Thực thi interface **SharedPreference.OnSharedPreferenceChangeListener.**
    - Đăng ký listener với **registerOnSharedPreferenceChangeListener()**

+ Setting screens

- Số lượng tùy chọn có thể dự đoán, quản lý được
- 7 trở xuống: sắp xếp theo mức độ ưu tiên với quan trọng nhất ở trên cùng
- 7-15 cài đặt: nhóm các cài đặt liên quan trong bộ chia phần
- 16 trở lên: nhóm thành các màn hình được mở từ Settings screen chính.

+ Preferences Screen

- Sử dụng **Preferences** objects thay vì **View** object trong **Setting screens**.
- Ví dụ:

```
<PreferenceScreen>
  <PreferenceCategory
    android:title="Flight Preferences">
        <CheckBoxPreference
              android:title="Wake for meals"
        ... />
           <EditTextPreference
              android:title="Favorite city"
              android:key="fav_city"
            .../>
          </PreferenceCategory>
</PreferenceScreen>
```

- Mỗi **Preference** phải có **key**

+ Some types of Preferenes

- **SwitchPreference**

```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
  <SwitchPreference
    android:defaultValue="true"
    android:title="@string/pref_title_social"
    android:key="switch"
    android:summary="@string/pref_sum_social" />
</PreferenceScreen>
```

- **EditTextPreference**

```
<EditTextPreference
   android:capitalize="words"
   android:inputType="textCapWords"
   android:key="user_display_name"
   android:maxLines="1"
   android:defaultValue="@string/pref_default_display_name"
   android:title="@string/pref_title_display_name" />
```

- **ListPreference**

```
<ListPreference
  android:defaultValue="-1"
  android:key="add_friends_key"
  android:entries="@array/pref_example_list_titles"
  android:entryValues="@array/pref_example_list_values"
  android:title="@string/pref_title_add_friends_to_messages" />
```

- **Preferences class/ subclass/ classes for grouping**

+ Implement settings

- Setting UI sử dụng fragments
  - Sử dụng một **Activity** với một **Fragment** để hiển thị **Settings screen**
  - Sử dụng lớp con của **Activity** và **Fragment** chuyên biệt để xử lý việc lưu lại settings.
- **Steps to implementing Settings**

For AppCompatActivity with PreferenceFragmentCompat

- Create the preferences screen
- Create an Activity for the settings
- Create a Fragment for the settings
- Add the preferenceTheme to the AppTheme
- Add code to invoke Settings UI

- Ví dụ:

```
//Setting Activity
public class MySettingsActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
```

```
        getSupportFragmentManager().beginTransaction()
            .replace(android.R.id.content, new MySettingsFragment())
            .commit();
    }
}
//Setting Fragment
public class MySettingsFragment extends PreferenceFragmentCompat {
    @Override
    public void onCreatePreferences(Bundle savedInstanceState,
                                    String rootKey) {
        setPreferencesFromResource(R.xml.preferences, rootKey);
    }
}
// Add PreferenceTheme to app's theme
If using PreferenceFragmentCompat, set preferenceTheme in styles.xml:
<style name="AppTheme" parent="...">
    ...
    <item name="preferenceTheme">
        @style/PreferenceThemeOverlay
    </item>
    …
</style>
```

- Invoke Settigns UI

Gửi Intent để bắt đầu Settings Activity:

- Từ Options menu, cập nhật **onOptionItemsSelected()**
- Từ **Navigation drawer**, cập nhật **onItemClick()** trên **OnItemClickListener** được cấp cho **setOnItemClickListener**

+ Default settings

- Set default values

- Sử dụng **android:defaultValue** in **Preference view** in **xml**

```
<EditTextPreference
    android:defaultValue="London"
    … />
```

- Ở **onCreate()** của **MainActivity**, lưu lại **default values**

```
PreferenceManager.setDefaultValues(
        this, R.xml.preferences, false);
```

+ Save and retrieve settings

- Sử dụng **key** như chỉ định trong **preference view** in **xml**

- Sử dụng **android:defaultValue** in **Preference view** in **xml**

```
SharedPreferences sharedPref =
    PreferenceManager.getDefaultSharedPreferences(this);
```

```
String destinationPref =
   sharedPref.getString("fav_city", "Jamaica");
```

- Trong **preference definition** in **xml**

```
<EditTextPreference
  android:defaultValue="London"
 android:key="fav_city" />
```

- Trong code, get **fav_city** setting:

```
String destinationPref =
sharedPref.getString("fav_city", "Jamaica");
```

+ Respond to changes in settings

- Listen for changes to settings
  - Định nghĩa **setOnPreferenceChangeListener()** ở in **onCreatePreferences()** trong **Settings Fragment**.

//onCreatePreferences

```
@Override
public void onCreatePreferences(Bundle savedInstanceState,
                                String rootKey) {
  setPreferencesFromResource(R.xml.preferences, rootKey);
  ListPreference colorPref =
            (ListPreference) findPreference("color_pref");
  colorPref.setOnPreferenceChangeListener(
   // see next slide
   // ...);
}
```

//onPreferenceChangeListener()

Example: change background color when setting changes

```
colorPref.setOnPreferenceChangeListener(
   new Preference.OnPreferenceChangeListener(){
      @Override
      public boolean onPreferenceChange(
        Preference preference, Object newValue){
          setMyBackgroundColor(newValue);
          return true;
      }
});
```

+ Summaries for settings

// Set summary example

```
EditTextPreference cityPref = (EditTextPreference)
                        findPreference("fav_city");
cityPref.setOnPreferenceChangeListener(
 new Preference.OnPreferenceChangeListener(){
```

```
    @Override
    public boolean onPreferenceChange(Preference pref, Object
value){
      String city = value.toString();
      pref.setSummary("Your favorite city is " + city);
      return true;
    }
  });
```

+ Activity template

- Cài đặt phức tạp
- Tương thích ngược
- Tùy chỉnh cài đặt được điền trước
- Bố cục thích ứng cho điện thoại và máy tính bảng

7.1: Location

+ Overview

+ Setting up Google Play services

+ Location permissions

+ Get device Location

+ Geocoding and reverse geocoding

+ Creating a LocationRequest object

+ Requesting location updates

9.1: Map

+ Map API

+ API setup

+ Displaying map in layout

+ Configuring initial map state

+ Setting the map type

+ Setting map styles and lite mode

+ Adding interactivity to maps

+ Moving the camera and view

+ Markers

+ Businesses and points of interest

+ UI controls

+ Overlays and shapes

+ Street view

Phần II: Trắc nghiệm

Phần III: Tự luận