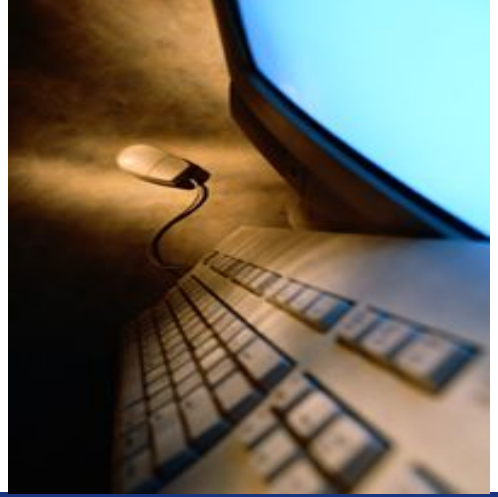
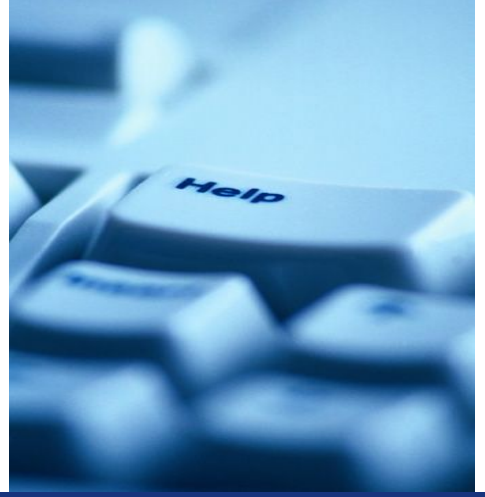
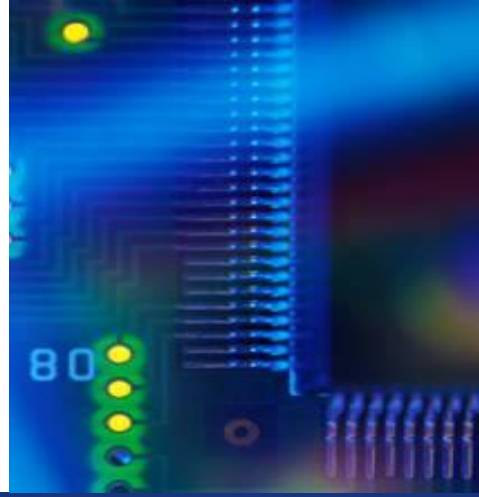
UNIVERSITAS
BUDI LUHUR



FAKULTAS TEKNOLOGI INFORMASI

Mobile Programming

[PG119/ 3 SKS]

Riskiana Wulan, MKom

About this Lesson

Lesson 2: Build your first Android app

- Your first app
- Anatomy of an Android app
- Layouts and resources in Android
- Activities

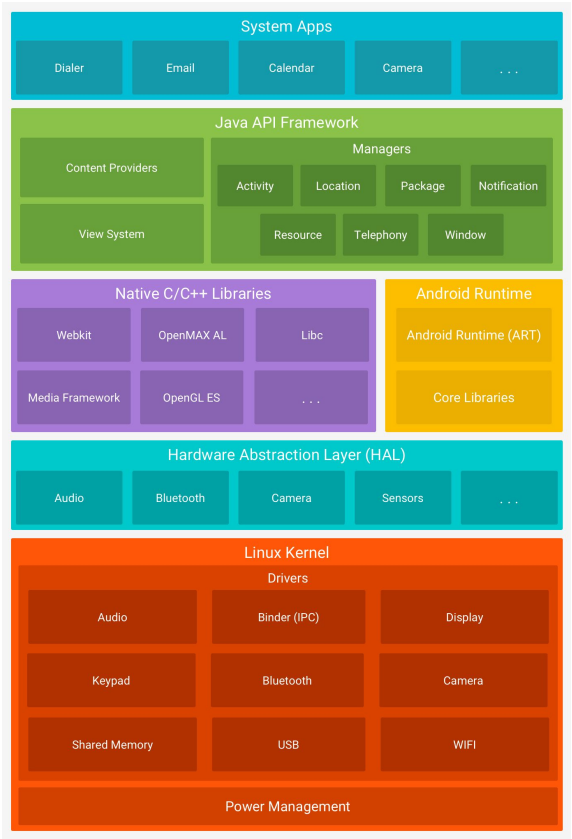

What is Android?

a mobile operating system that is based on a modified version of Linux. It was originally developed by a startup of the same name, Android, Inc.

In 2005, as part of its strategy to enter the mobile space, Google purchased Android, Inc. and took over its development work (as well as its development team).

Src: Beginning Android Programming with Android Studio, by Jerome DiMarzio

Platform Architecture



1. **System Apps:** basic functionalities and are available to users out of the box
2. **Java API Framework:** the building blocks for Android application development.
3. **Native C/C++ Libraries**
4. **Android Runtime**
   a. **Android Runtime (ART)**: Executes Android applications. ART is a replacement for Dalvik and provides improved performance and memory management.
   b. **Core Libraries**: Provide essential classes and functions for app development.
5. **Hardware Abstraction Layer (HAL)**
6. **Linux Kernel**
   a. **Drivers**: Control hardware components like audio, display, USB, Bluetooth, and camera.
   b. **Binder (IPC)**: Handles inter-process communication.
   c. **Power Management**: Manages the power usage and state of the device.

https://developer.android.com/guide/platform

Application Components

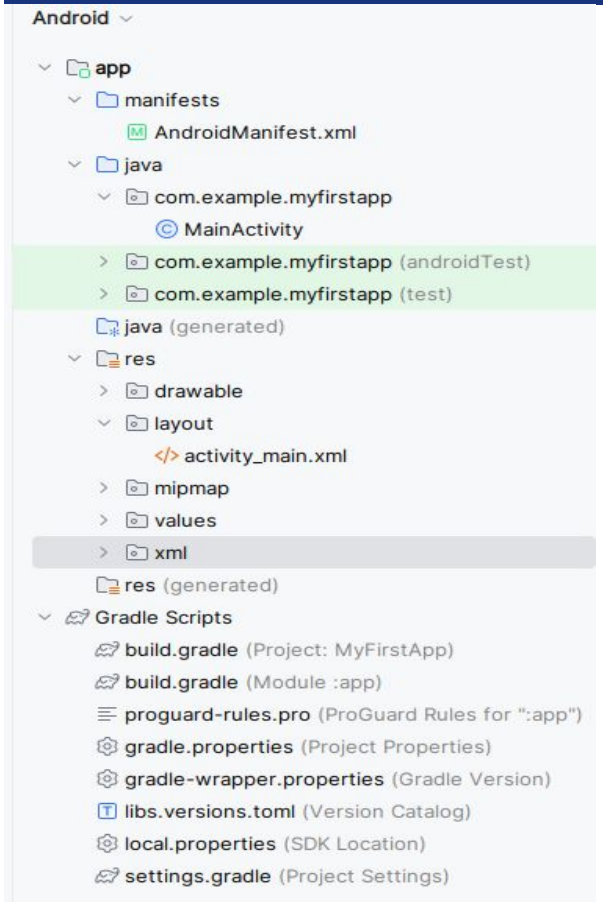
- Activities
- Services
- Broadcast receivers
- Content providers

Anatomy of an Android App project

Anatomy of a basic app project

- Activity
- Resources (layout files, images, audio files, themes, and colors)
- Gradle files

Android app project structure



- **`app`** - stores source code, tests, and resources for your app
- **`androidTest`** - test code that's specific to Android
- **test** - local unit tests that will execute on your computer
- **`AndroidManifest.xml`** - declares essential information for your app
- **res** : contains all the resources in its subdirectories: an image resource, two layout resources, a mipmap/ directory for launcher icons, and a string resource file
- **`gradle` scripts**- controls how your application builds, tests, and deploys itself

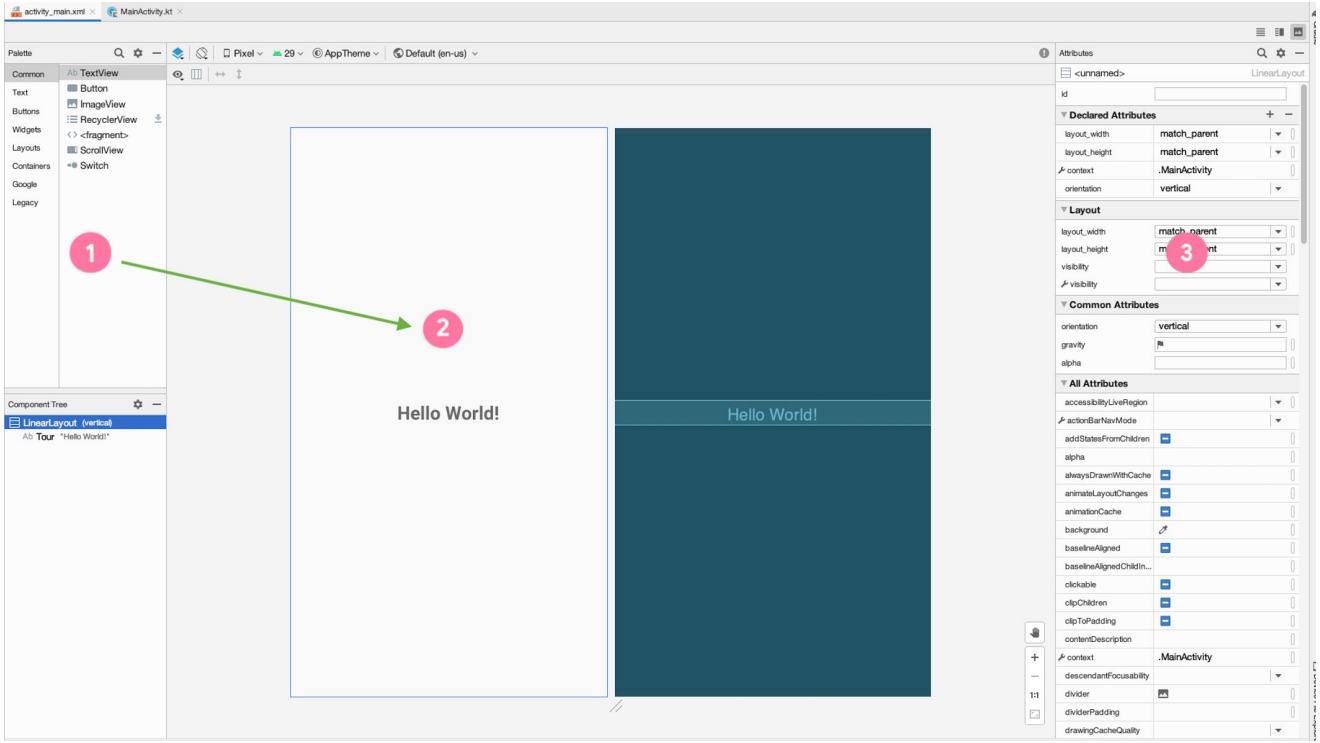
Layouts and resources in Android

Views

- **Views are the** user interface building blocks in Android
  - Bounded by a rectangular area on the screen
  - Responsible for drawing and event handling
  - Examples: TextView, ImageView, Button

- Can be grouped to form more complex user interfaces

Layout Editor




XML Layouts

You can also edit your layout in XML.

- **Android uses XML to specify the layout of user interfaces (including View attributes)**

- **Each View in XML corresponds to a class in Java that controls how that View functions**

XML for a TextView

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"/>
```

Hello World!


Size of a View

- **wrap_content**

```
android:layout_width="wrap_content"
```

- **match_parent**

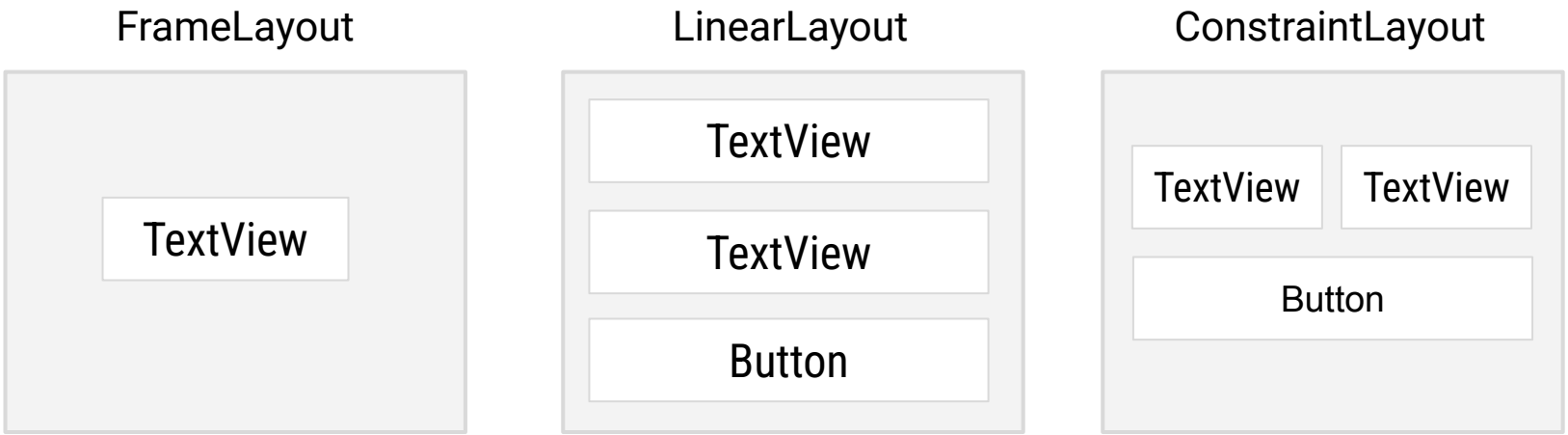
```
android:layout_width="match_parent"
```

- **Fixed value (use dp units)**

```
android:layout_width="48dp"
```

ViewGroups

A `ViewGroup` is a container that determines how views are displayed.



The ViewGroup is the parent and the views inside it are its children.

FrameLayout example

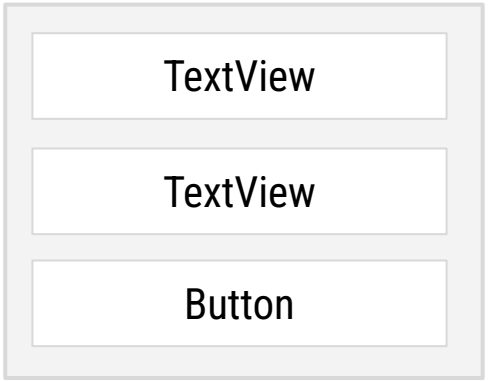
A `FrameLayout` generally holds a single child `View`.

```
<FrameLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <TextView
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       android:text="Hello World!"/>
</FrameLayout>
```

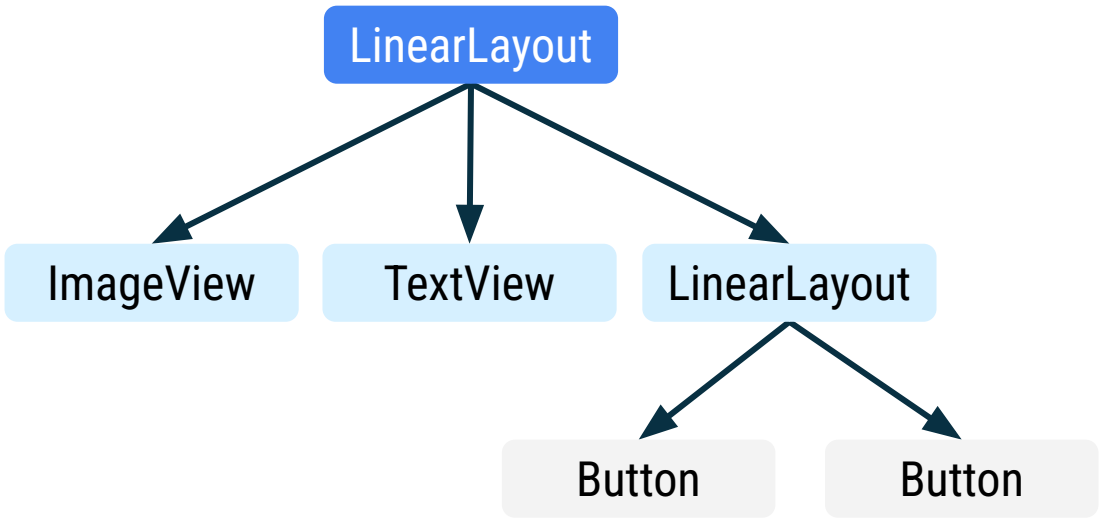
LinearLayout example

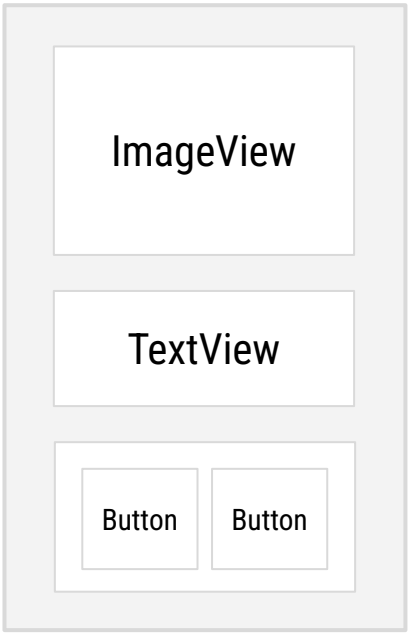
- Aligns child views in a row or column
- Set `android:orientation` to `horizontal` or `vertical`

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



View hierarchy






App resources

Static content or additional files that your code uses

- **Layout files**
- **Images**
- **Audio files**
- **User interface strings**
- **App icon**

Common resource directories

Add resources to your app by including them in the appropriate resource directory under the parent `res` folder.

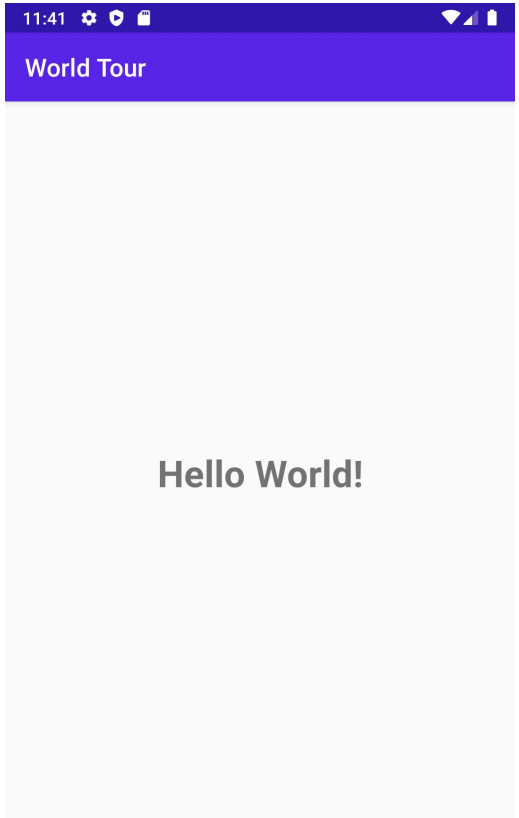
```
main
├── java
└── res
    ├── drawable
    ├── layout
    ├── mipmap
    └── values
```

Activities

https://developer.android.com/guide/components/activities/

What's an Activity?

- An Activity is a means for the user to accomplish one main goal.
- An Android app is composed of one or more activities.
- An *activity* is the entry point for interacting with the user. It represents a single screen with a user interface.

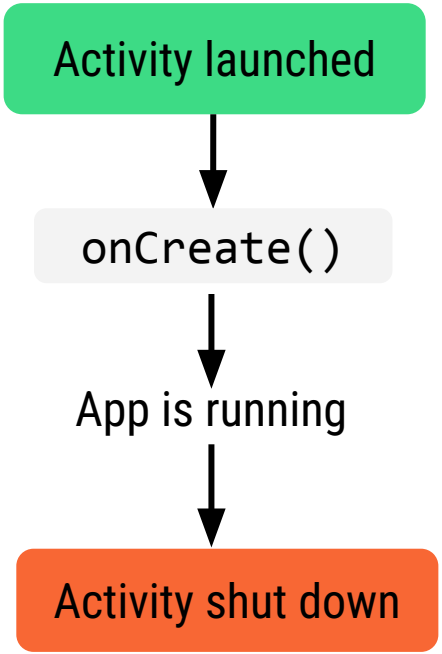


MainActivity.java

```java
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable( $this$enableEdgeToEdge: this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}
```

How an Activity runs




Activity Life Cycle

Lifecycle methods control how activities are created, resumed, paused, or destroyed.

- **onCreate()**: Called when activity is first created.
- **onStart()**: Called just before the activity becomes visible.
- **onResume()**: Called when the activity starts interacting with the user.
- **onPause()**: Called when the activity loses focus but is still visible.
- **onStop()**: Called when the activity is no longer visible.
- **onDestroy()**: Called when the activity is finishing or destroyed by the system.

Intents

https://developer.android.com/reference/android/content/Intent


Intents

- **Intents** are messaging objects that allow activities and services to communicate.
- Two types of intents:
  - **Explicit Intent**: Specifies the exact activity to start (e.g., MainActivity).
  - **Implicit Intent**: Allows Android to choose which app to handle the action (e.g., opening a web page).
- Intents are used for **launching activities**, **starting services**, and **broadcasting messages**.

Explicit intent

```
val intent =
Intent(this,SecondActivity::class.java)
startActivity(intent)
```

Implicit intent

```
val intent = Intent(Intent.ACTION_VIEW)
intent.data =
Uri.parse("https://www.example.com")
startActivity(intent)
```

Gradle: Building an Android app


What is Gradle?

- Builds automation system
- Manages the build cycle via a series of tasks (for example, compiles Java sources, runs tests, installs app to device)
- Determines the proper order of tasks to run
- Manages dependencies between projects and third-party libraries


Gradle build file

- Declare plugins
- Define Android properties
- Handle dependencies
- Connect to repositories


Plugins

Provide libraries and infrastructure needed by your app

```
apply plugin: 'com.android.application'

apply plugin: 'kotlin-android'

apply plugin: 'kotlin-android-extensions'
```


Android configuration

```
android {
    namespace 'com.example.myapplication'
    compileSdk 34

    defaultConfig {
        applicationId "com.example.myapplication"
        minSdk 25
        targetSdk 34
        versionCode 1
        versionName "1.0"
```


Dependencies

```
dependencies {

    implementation libs.appcompat
    implementation libs.material
    implementation libs.activity
    implementation libs.constraintlayout
    testImplementation libs.junit
    androidTestImplementation libs.ext.junit
    androidTestImplementation libs.espresso.core
}
```


Repositories

```
repositories {
    google()
    mavenCentral()
}
```

Common Gradle tasks

- Clean
- Tasks
- InstallDebug


Java Code Style

Here you can download and install the java codestyle.
https://github.com/android10/java-code-styles

- On Unix, run the install.sh script. Windows users should use install.bat instead.
- Restart Android Studio if it's running.
- Open Android Studio, Settings -> Code Styles, change the code style for the project to the one you want.