



Android Fundamentals

Dr. Abdelkarim Erradi
CSE@QU

Outline

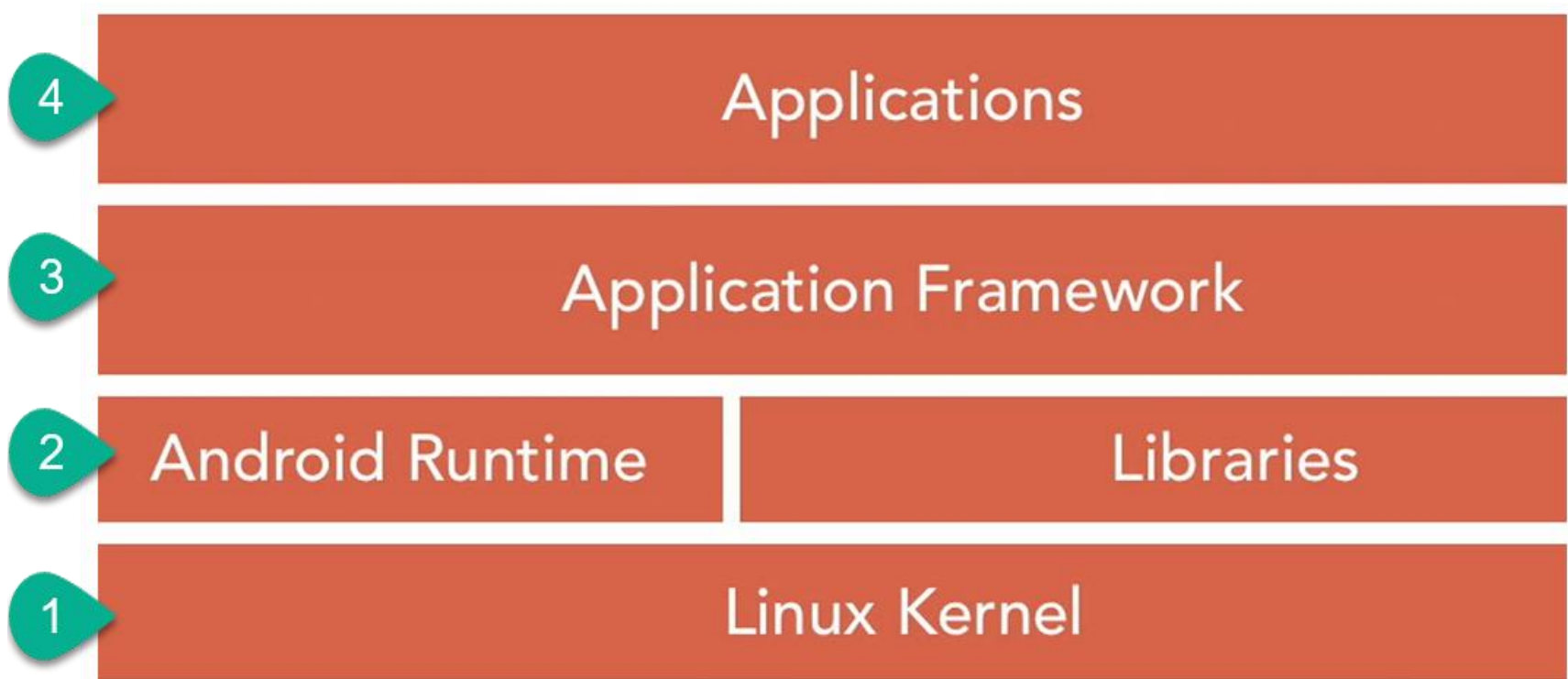
1. Introduction to Android
2. Android Programming Model

Introduction to Android

What is Android?

- Open source mobile operating system (OS) based on [Linux kernel](#)
 - originally purchased by Google from Android, Inc. in 2005
- Used on [over 80%](#) of all smartphones
- Powers devices such as watches, TVs, and cars
- The #1 OS worldwide
 - Over 2.5 billion active Android devices worldwide
 - Over 2 Million Android apps in Google Play store
- Highly customizable for devices by vendors

Android Software Stack



1. Linux Kernel: interacts and manages hardware
2. Expose native APIs; run apps
3. Java API exposing Android OS features
4. System and user apps

Android Software Stack

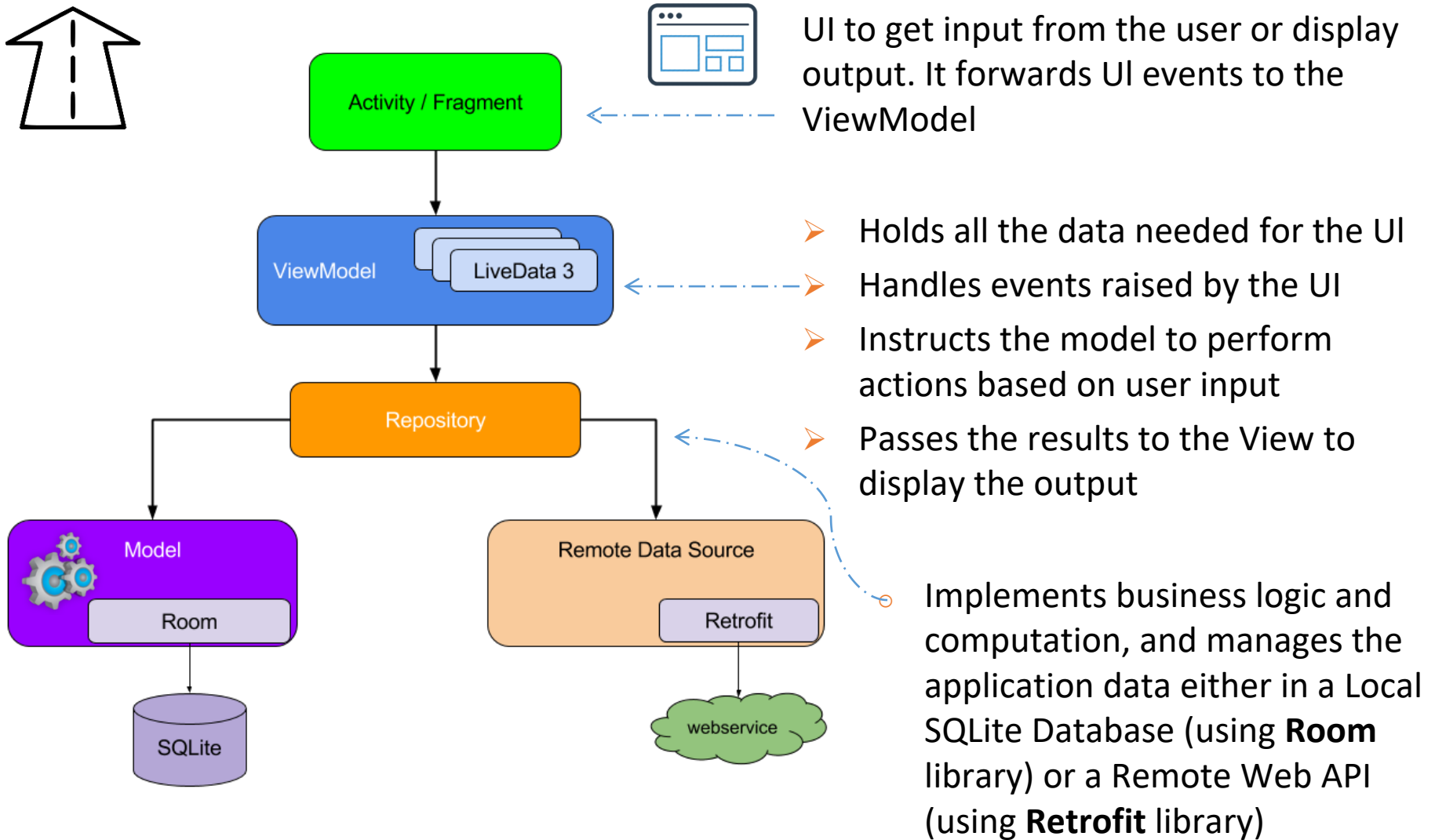
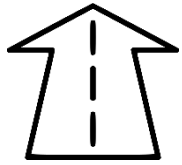
1. Optimized **Linux Kernel** manages core services such as device hardware drivers, process and memory management, and power management
 - Acts as an abstraction layer between the hardware and the rest of the software stack
2. **Android runtime (ART)** = Virtual Machine to run Apps
 - Every App runs in its own process in its own instance of the Android Runtime
 - Expose native APIs and OS Core Libraries including 2D/3D graphics, SQLite database, encryption ...
3. **Application Framework**: Java APIs (Application Programming Interfaces) make Android OS features available to Apps

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



Model-View-ViewModel (MVVM) Architecture

IMPORTANT



Advantages of MVVM



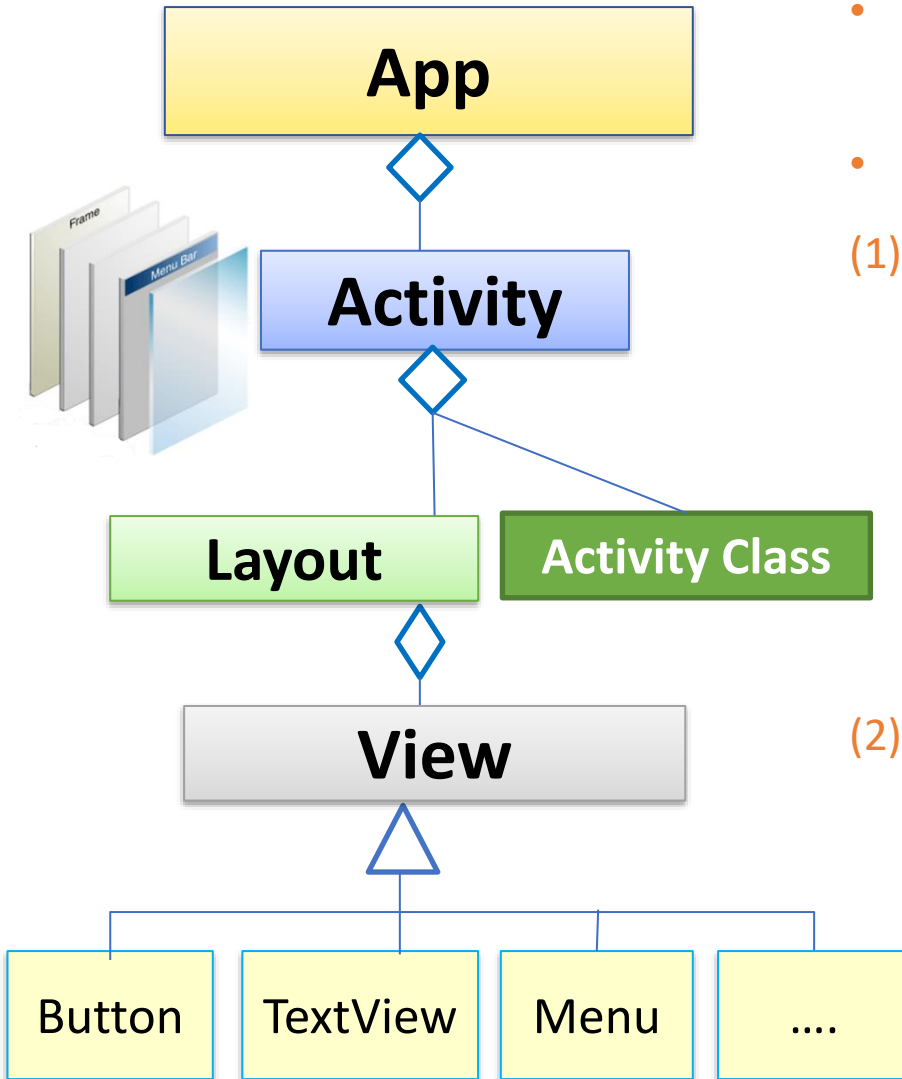
- ***Separation of concerns***
 - View, ViewModel, and Model are **separate components**
 - Computation is not intermixed with Presentation. Consequently, code is cleaner, flexible and easier to understand and change.
 - Allow changing a component without significantly disturbing the others (e.g., UI can be completely changed without touching the model)
 - => Easier maintenance



Android Programming Model

Android Programming Model

IMPORTANT



- App is composed of one or more **screens** (called **Activity**)
- An activity has:
 - (1) a **Layout** that define its appearance (how it **looks like**)
 - Layout acts as a **container** for UI Components (called **View**)
 - It decides the size and positions of views placed in it
 - (2) Activity Kotlin class that provides the data to the UI and handles events
 - UI Components **raise Events** when the user interacts with them (such as a Clicked event is raised when a button is pressed).
 - In the activity class we define **Event Handlers** to respond to the UI events

Activity

- **Activity** is a screen that displays a UI to allow the user to do something such order groceries, send email ...
 - Has layout (.xml) file & Activity class
 - This allows a **clear separation** between the UI and the app logic
- Connecting activity with the layout is done in the **onCreate** method
- Can start other activities in the same or other apps
- Has a lifecycle: created, started, paused, resumed, stopped, and destroyed
- Listeners have code to handle events:
 - User interaction events such press a button or enters text in a text view
 - External events such as receiving a notification or screen rotation

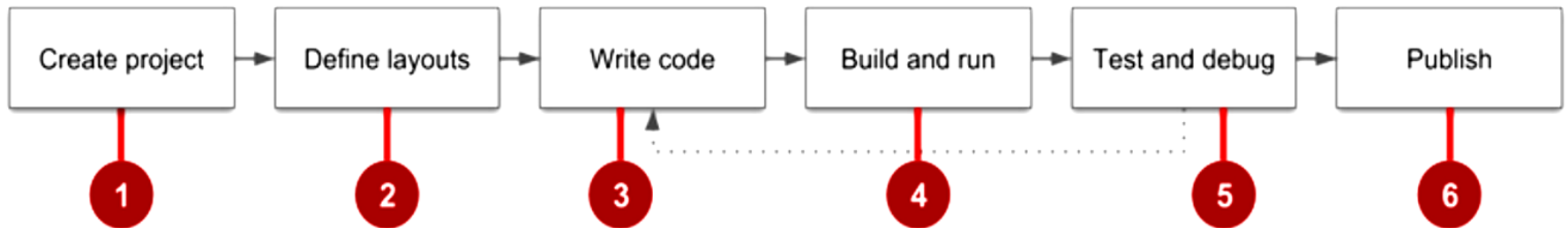
Example

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
        changeColorBtn.setOnClickListener {  
            greetingTv.setTextColor(getRandomColor())  
        }  
    }  
}
```

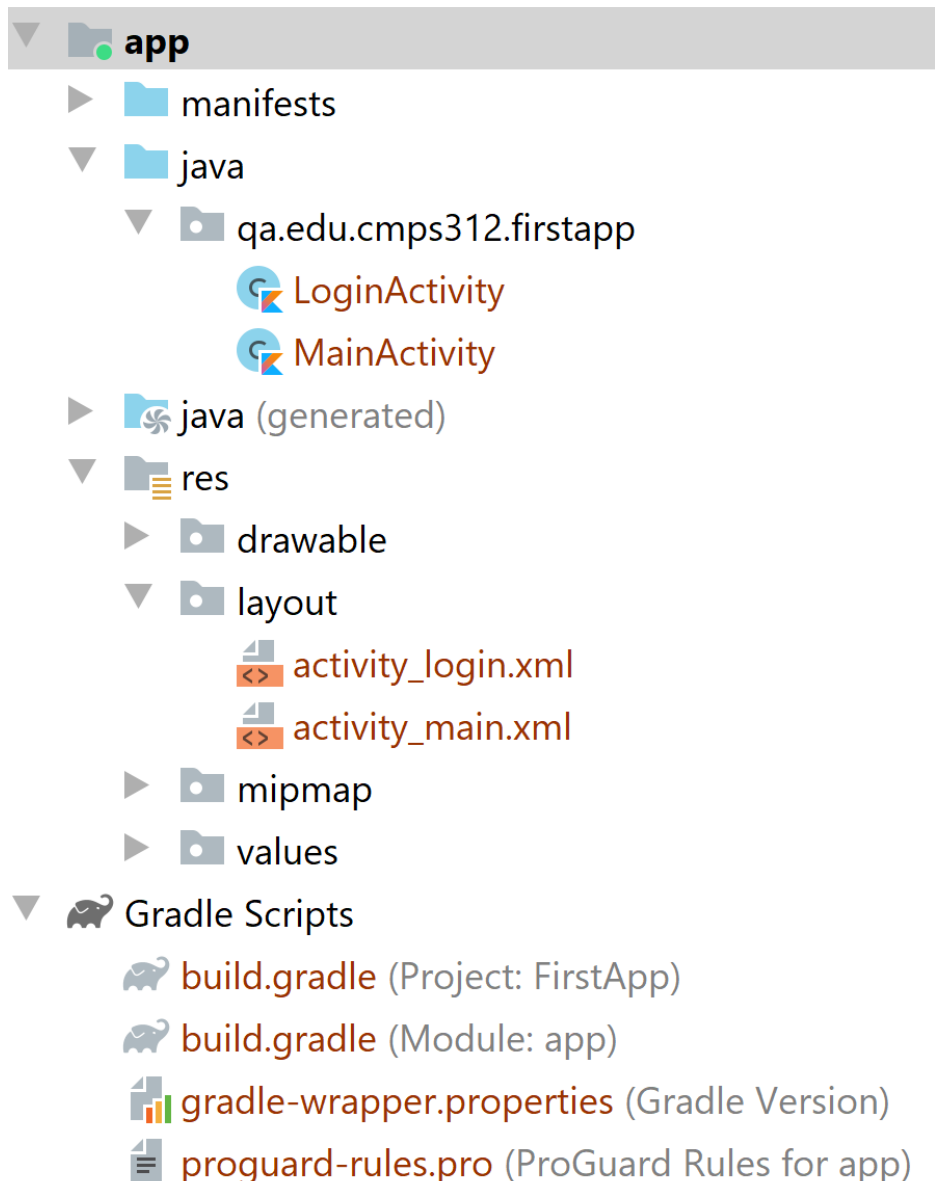
Connects
activity
with layout



Development Process



Project structure



AndroidManifest.xml

- app config and settings (e.g., list app activities and required permissions)

java/...

- Kotlin source code

res/... = resource files (*many are XML*)

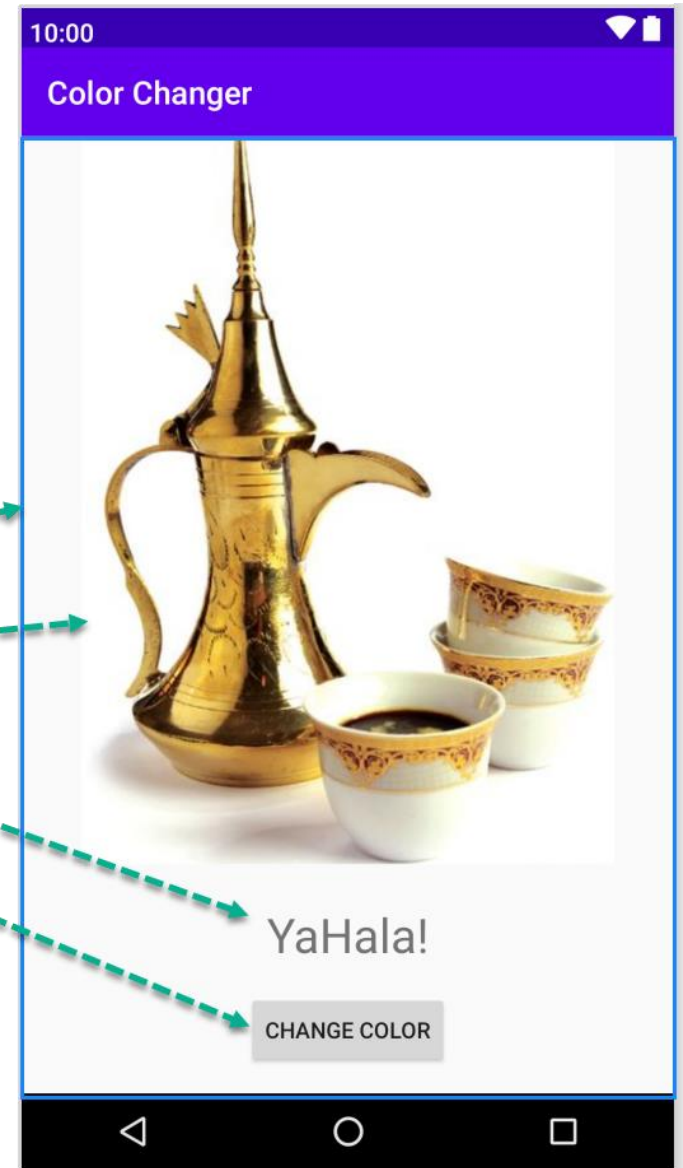
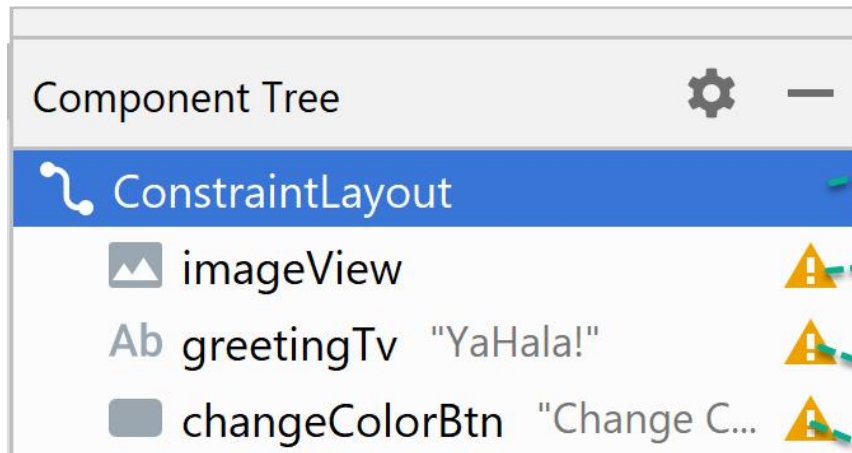
- drawable/ = images
- layout/ = GUI layouts
- menu/ = app menu options
- values/ = constant values
- strings/ = localization data
- styles/ = appearance styling

Gradle

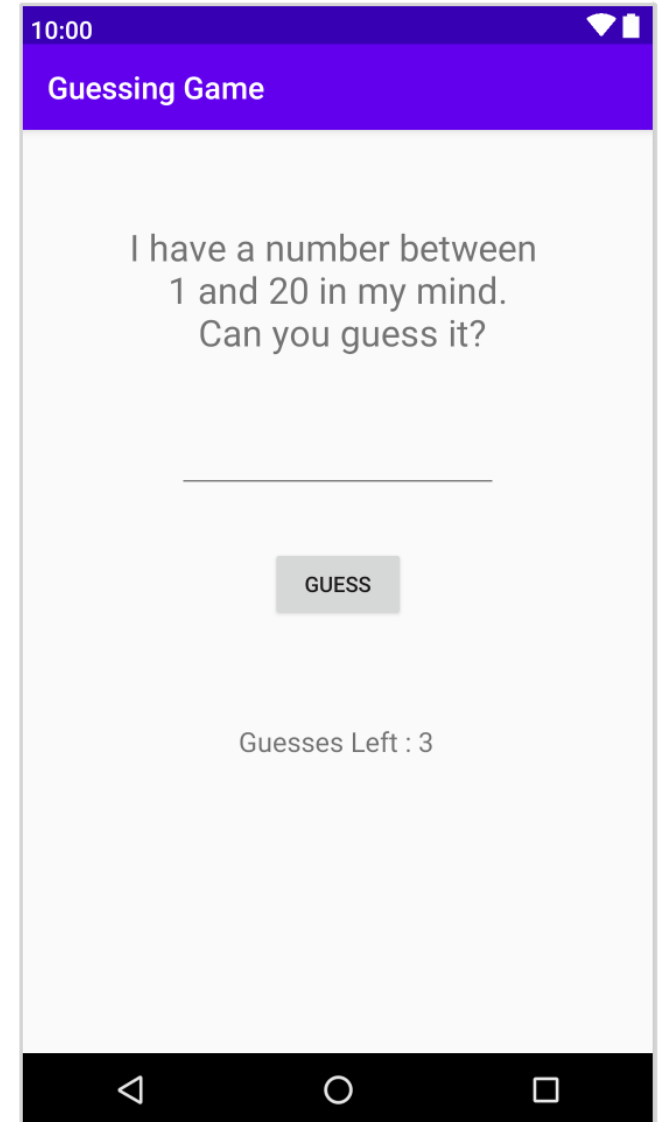
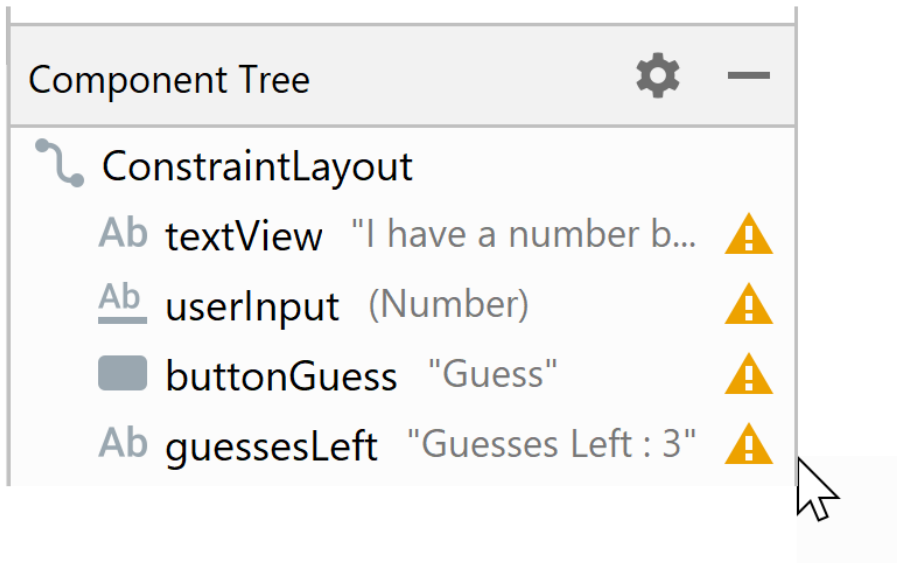
- a build/compile management system
- **build.gradle** = main build config file

App 1 - Color Changer

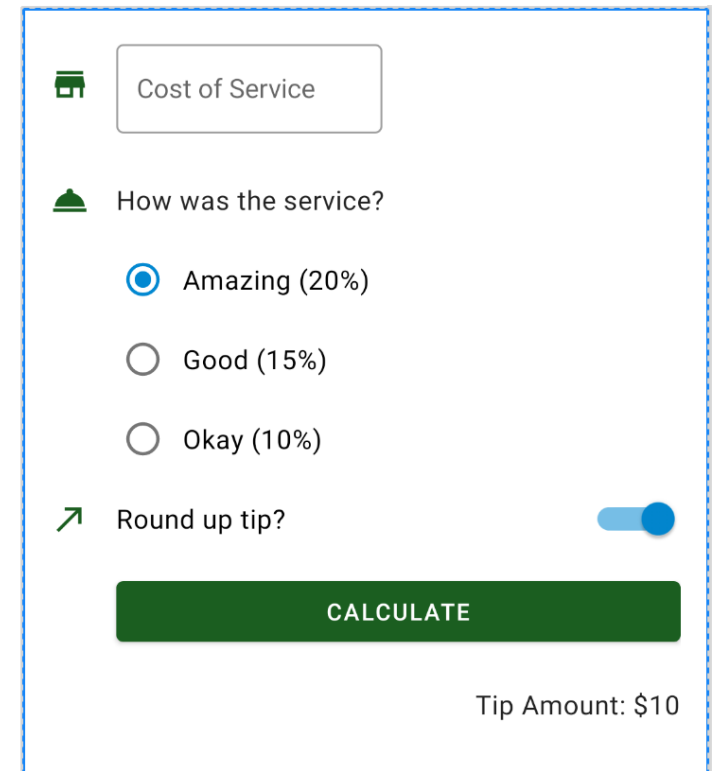
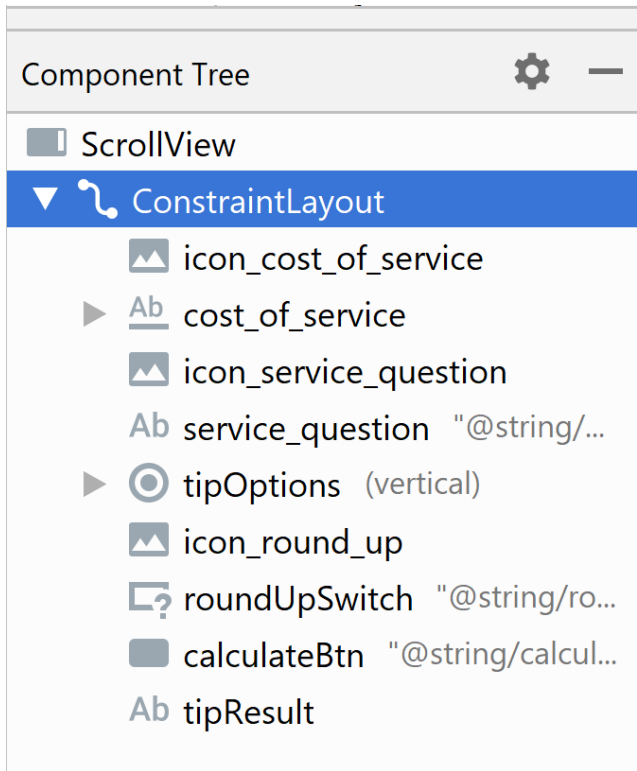
App that contains Text reading “YaHala!”, an Image and a **Button** that randomly changes text’s color with every click



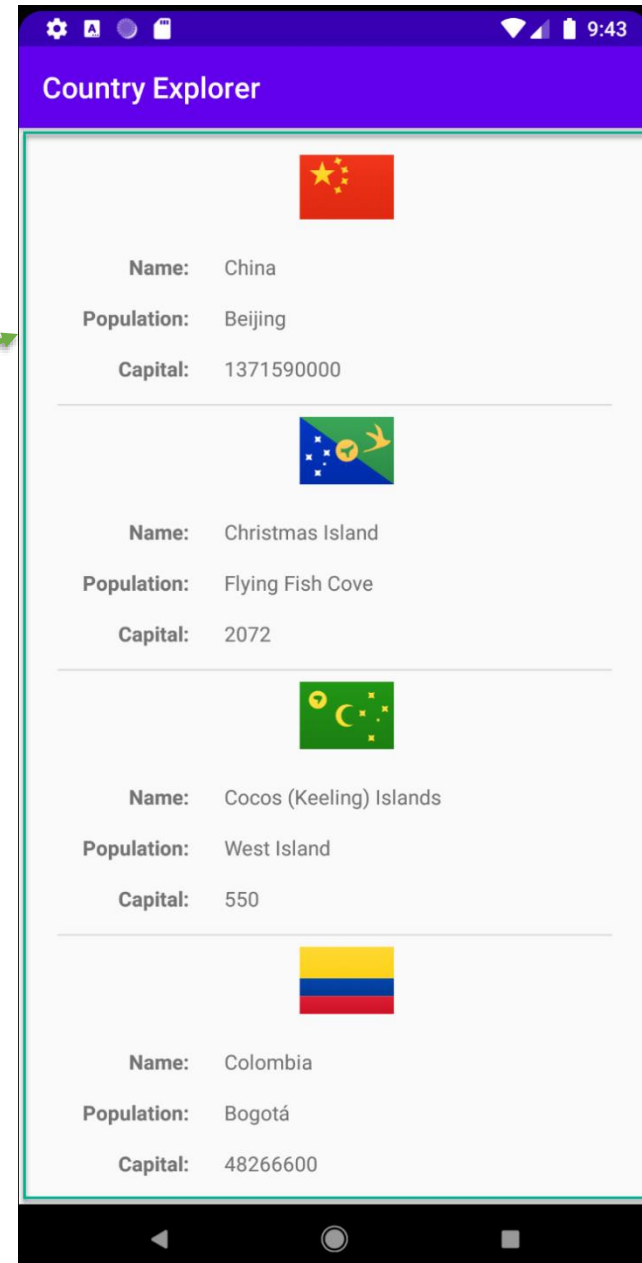
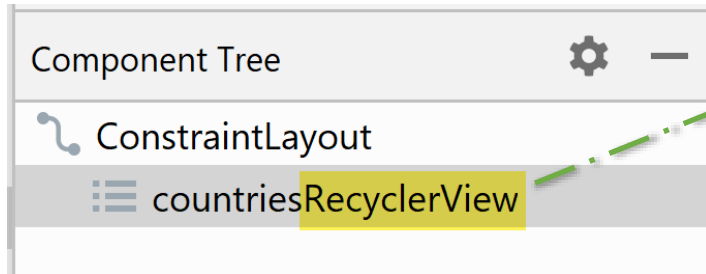
App 2 – Guessing Game



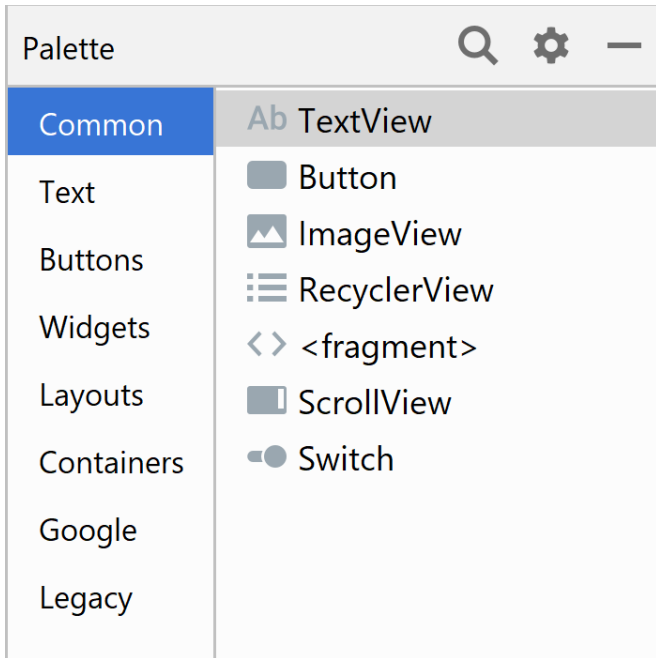
App 3 – Tips Calculator



App 4 – Country Explorer



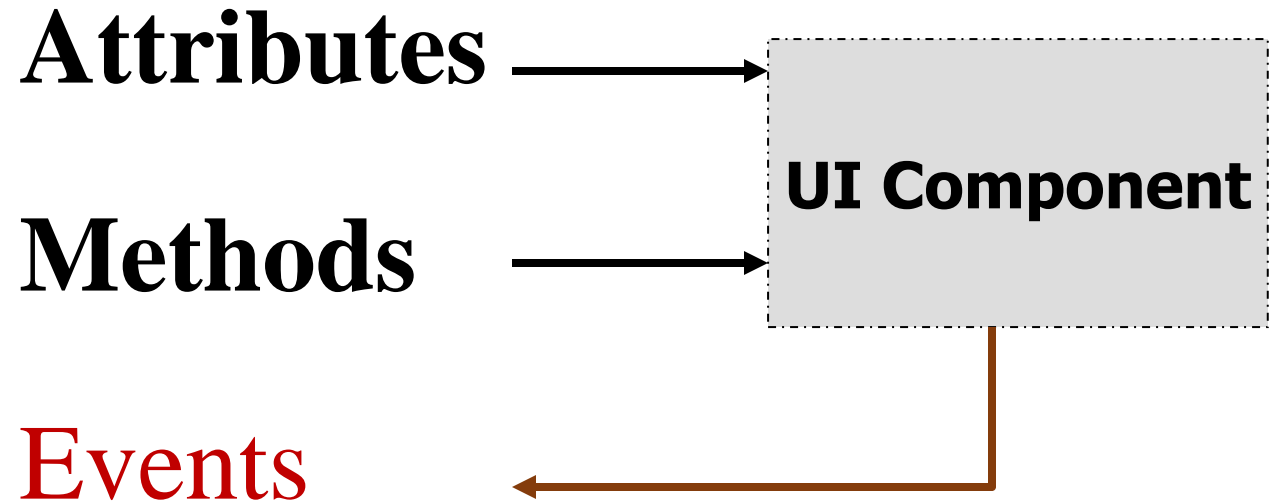
What Makes up Android UI?



- **UI components**
 - Set of pre-built UI components that can be composed to create a GUI
 - e.g. button, TextView, Menu, List, etc.
- **Layout containers**
 - Control placement/positioning of components in the Activity

UI Component

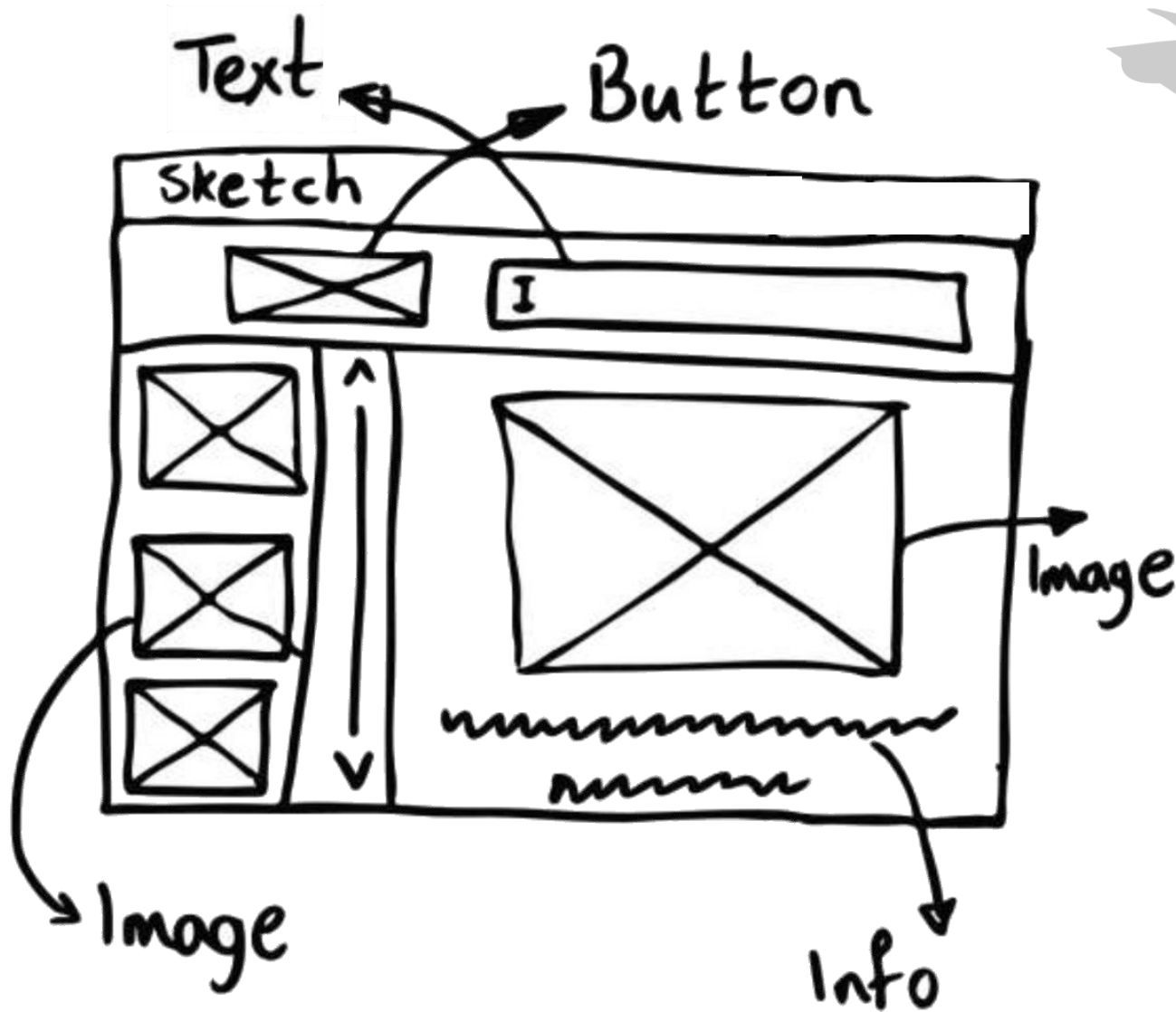
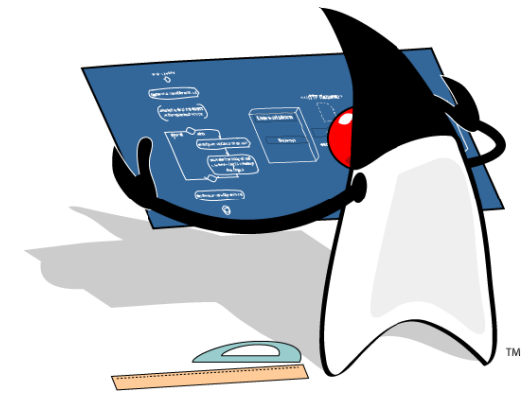
- UI component is a class that has:



Steps to creating a GUI Interface

1. Design it on paper (sketch)
 - Decide what information to present to user and what input they should supply
 - Decide the UI components and the layout on paper
2. Create a layout and add UI components to it using the Layout Editor
 - Use the Layout Editor to group and arrange components
3. Add event handlers to respond to the user actions
 - Do something when the user presses a button, selects an item from list, change text of input field, etc.

UI Sketch - Example

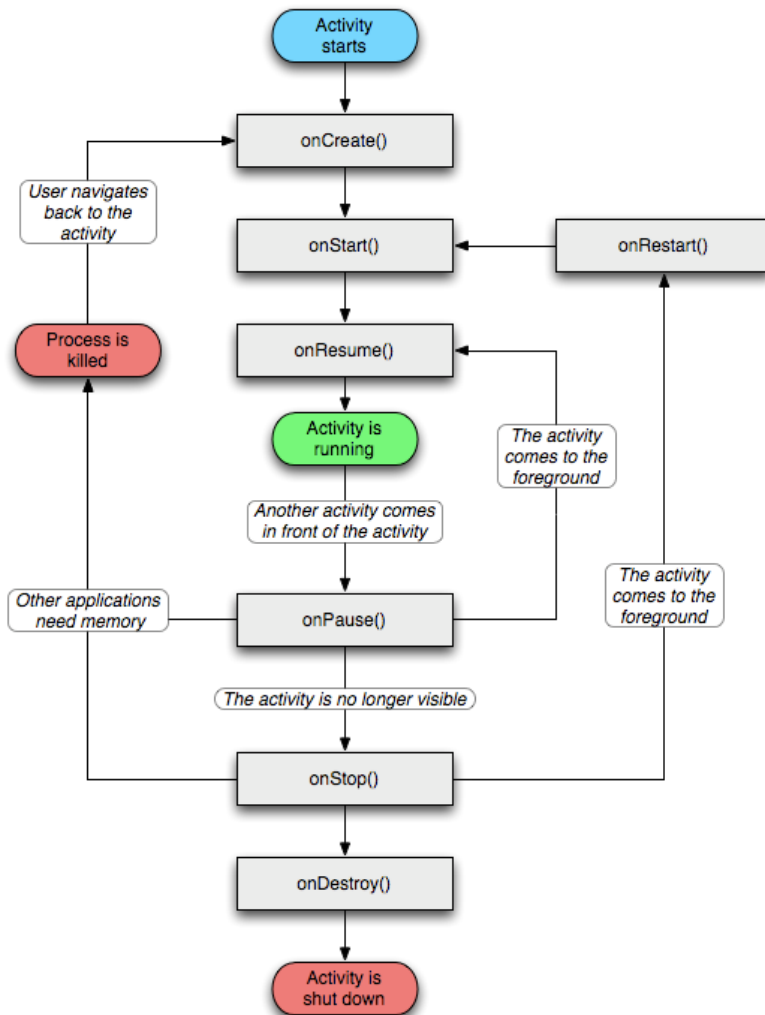


You may design different layouts per screen size

Activity Lifecycle

An activity has essentially **four states**:

- If an activity is in the foreground of the screen (at the top of the stack), it is **active**
- If an activity has lost focus but is still visible (e.g., beneath a dialog box), it is **paused**. A paused activity is completely alive but can be killed by the system in case of low memory.
- If an activity is completely obscured by another activity, it is **stopped**. It still retains all state and member information but can be **destroyed** by the system when memory is needed.
- If an activity is paused or stopped, it maybe killed. When it is displayed, it must be completely **restarted** and restored to its previous state.



Resources

- Android Kotlin Fundamentals Course
 - <https://developer.android.com/courses/kotlin-android-fundamentals/overview>
 - <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
- Android Dev Guide
 - <https://developer.android.com/guide/>