CMPS 312

Read Chapter 1 & 6

Android Fundamentals

Dr. Abdelkarim Erradi

CSE@QU

Outline

1. Introduction to Android
2. Android Programming Model

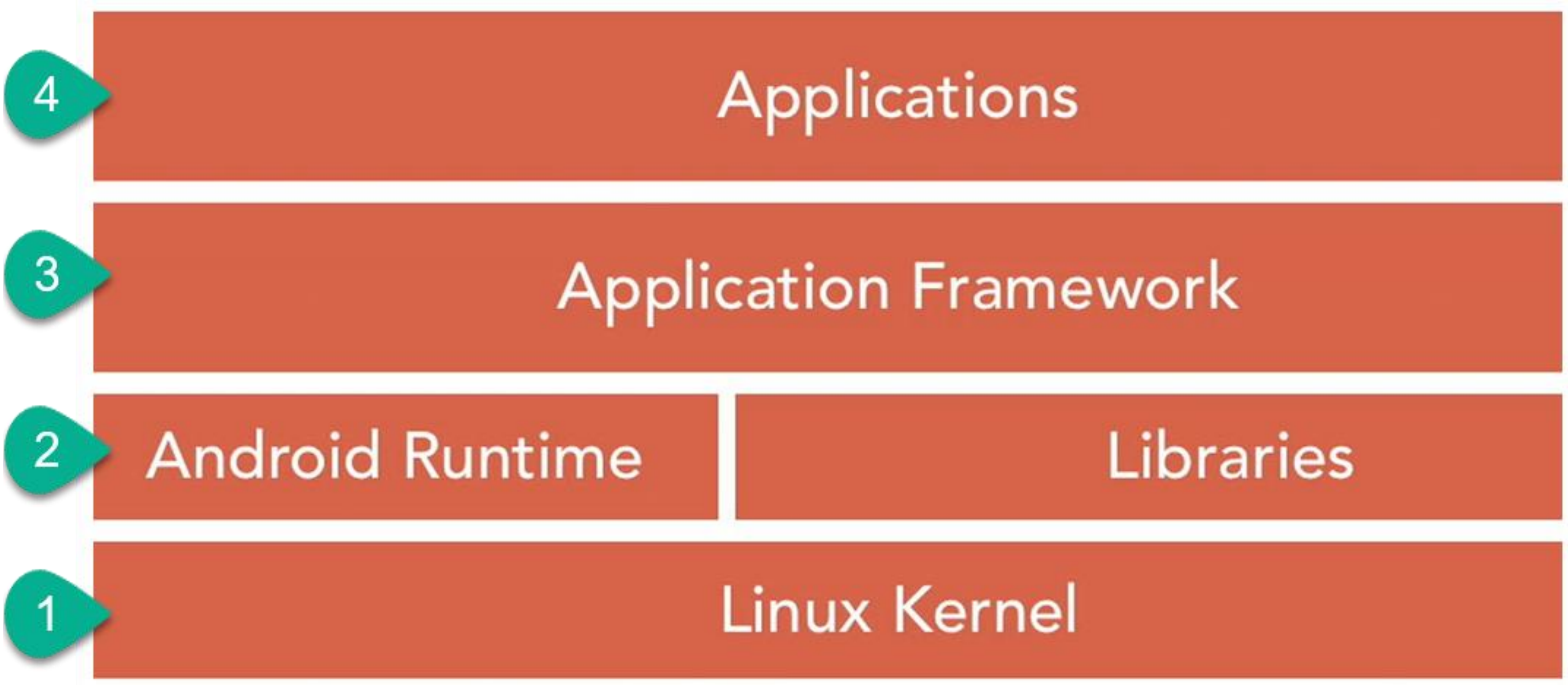
Introduction to Android

< Back

What is Android?

- Open source mobile operating system (OS) based on Linux kernel for phones, tablets, wearable
  - originally purchased by Google from Android, Inc. in 2005
- Used on over 80% of all smartphones
- 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

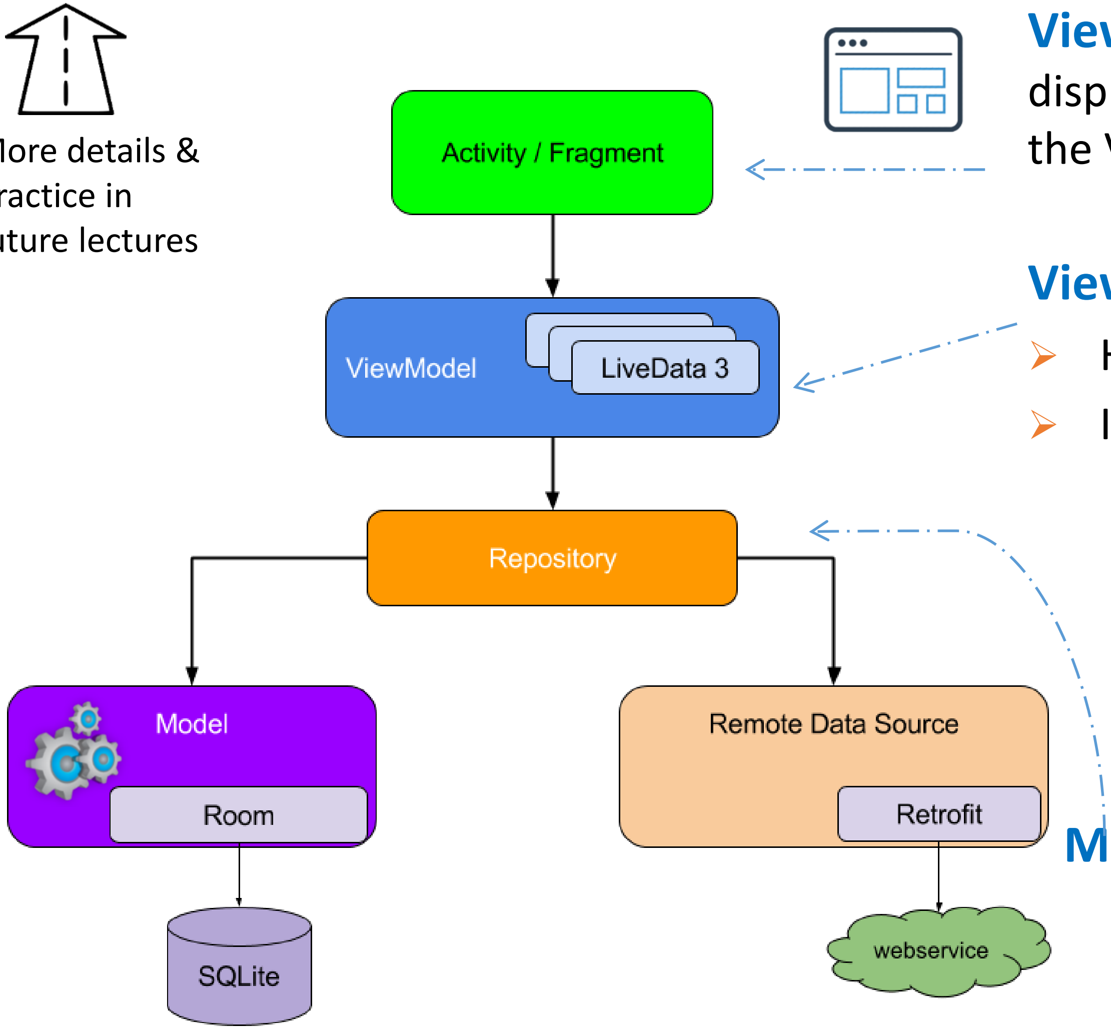
1. Optimized **Linux Kernel** manages core services such as device hardware drivers, process and memory management, and power management
   - Acts as an abstraction layer between the hardware and the rest of the software stack
2. **Android runtime (ART)** = Virtual Machine to run Apps
   - Every App 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

More details & practice in future lectures



View = UI to get input from the user or display output. It forwards UI events to the ViewModel

ViewModel

- Holds data needed for the UI
- Implements UI logic
  - Handles events raised by the UI
  - Instructs the repository to perform actions based on user input
  - Passes the results to the View to display the output

Model

- Implements business logic / computation and manages the application data either in a Local SQLite Database (using **Room** library) or a Remote Web API (using **Retrofit** library)

Advantages of MVVM

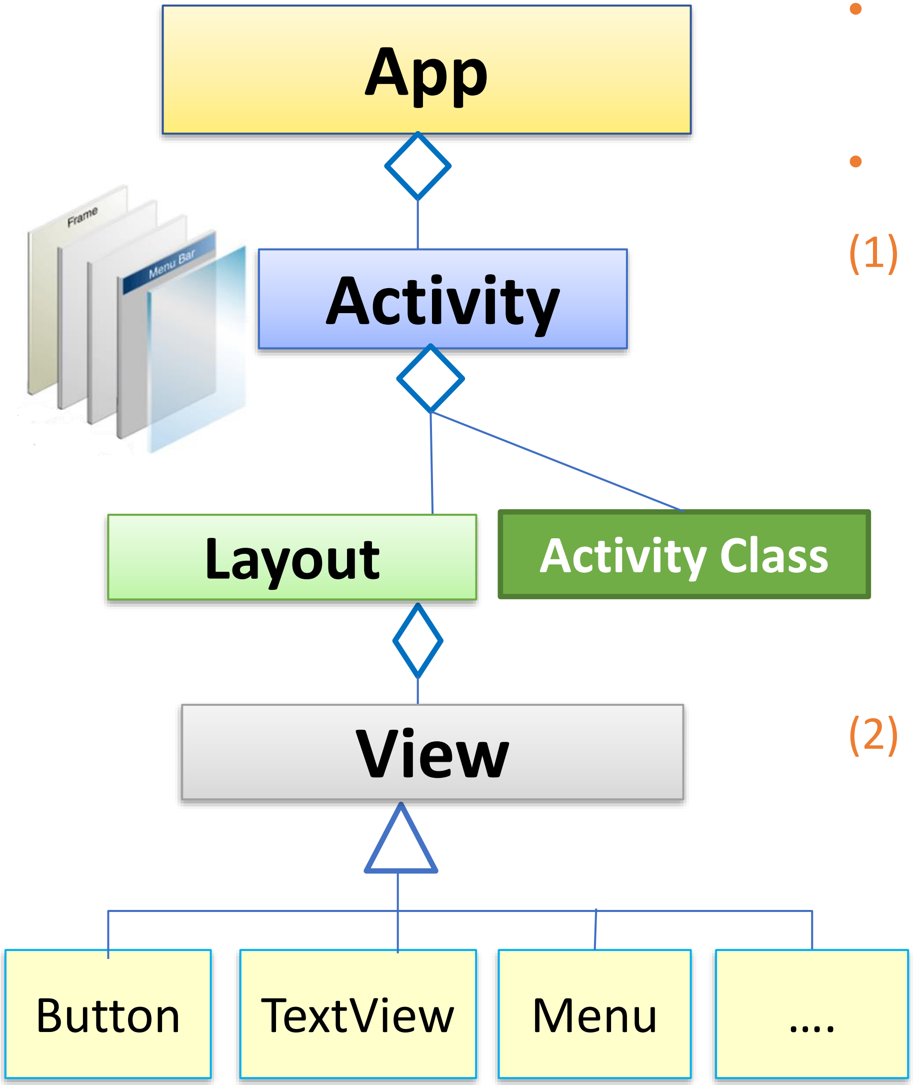
- ***Separation of concerns***
  - View, ViewModel, and Model are **separate components**
    - Computation is not intermixed with UI. Consequently, code is cleaner, flexible and easier to understand and change.
    - Allow changing a component without significantly disturbing the others (e.g., UI can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

Android Programming Model

< Back

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

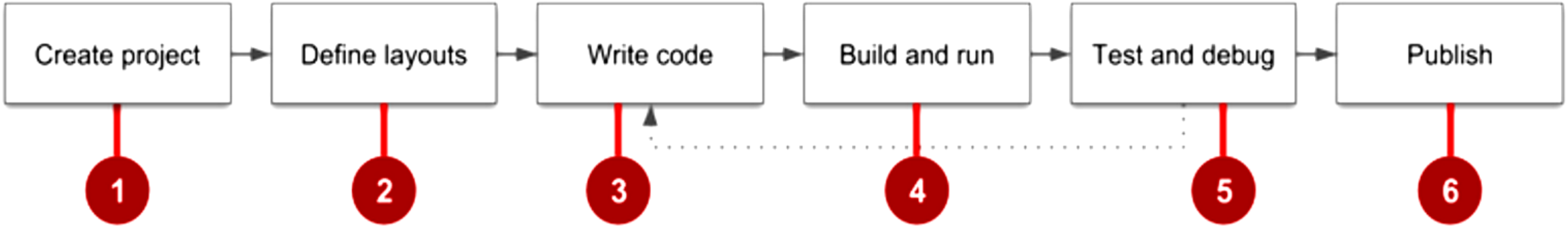
- **Activity** is a screen that displays a UI to allow the user to do something such order groceries, send email …
  - Has layout (.xml) file & Activity class
  - This allows a **clear separation** between the UI and the app logic
- Connecting activity with the layout is done in the **onCreate** method
- 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

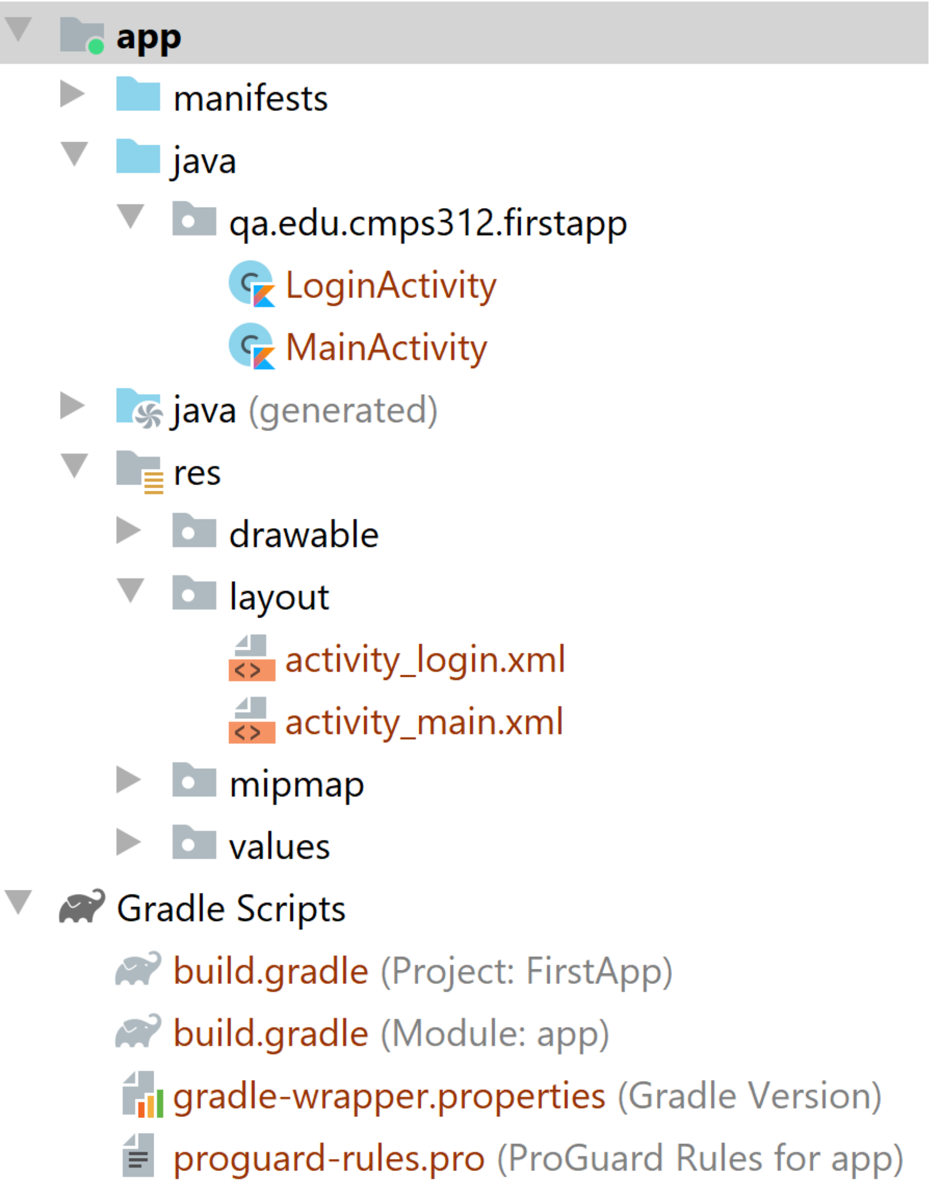
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
          super.onCreate(savedInstanceState)
          setContentView(R.layout.activity_main)

          changeColorBtn.setOnClickListener {
              greetingTv.setTextColor(getRandomColor())
          }
    }
```

Connects activity with layout

Development Process



Project structure

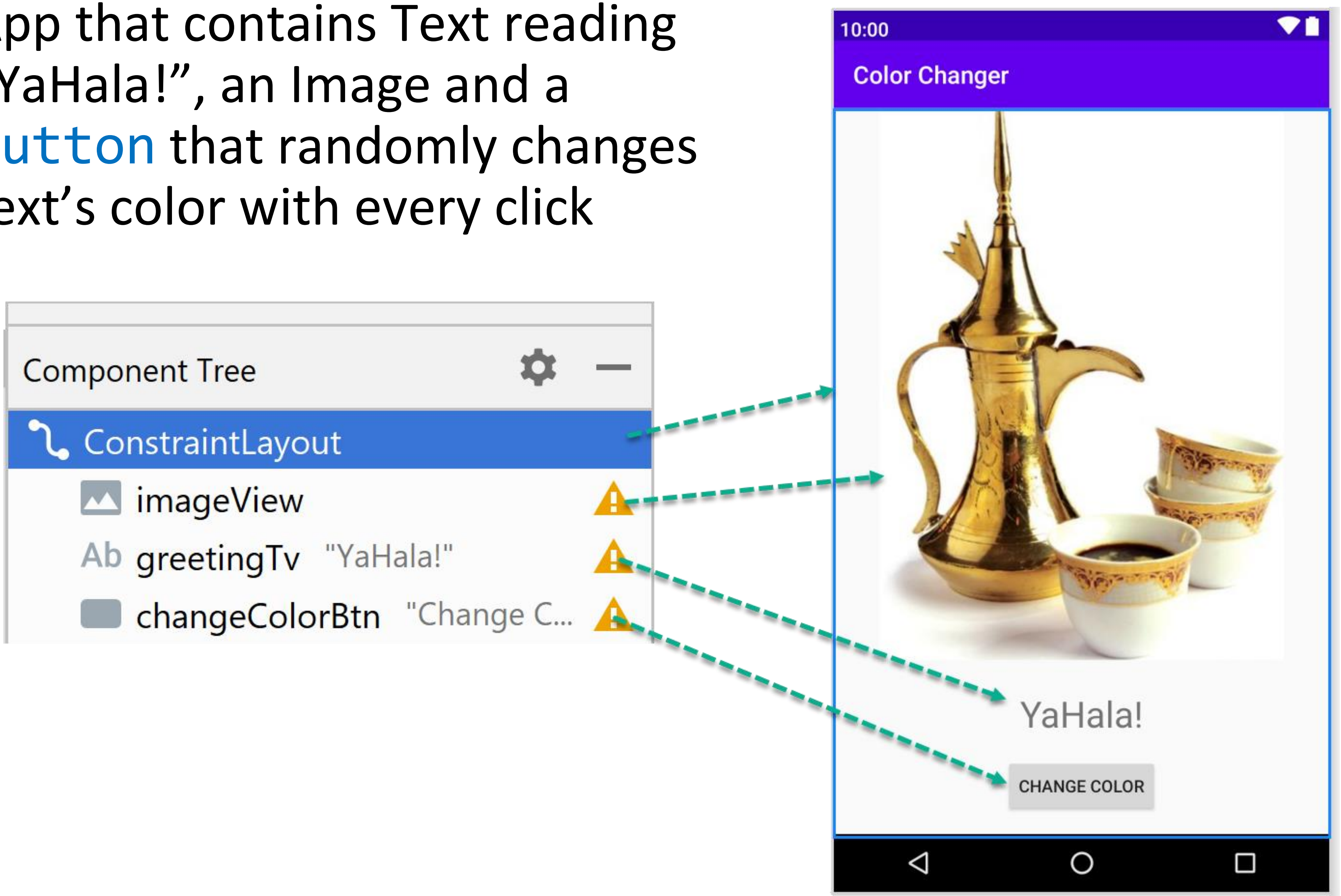


```
app
  manifests
  java
    qa.edu.cmps312.firstapp
      LoginActivity
      MainActivity
  java (generated)
  res
    drawable
    layout
      activity_login.xml
      activity_main.xml
    mipmap
    values
Gradle Scripts
  build.gradle (Project: FirstApp)
  build.gradle (Module: app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for app)
```

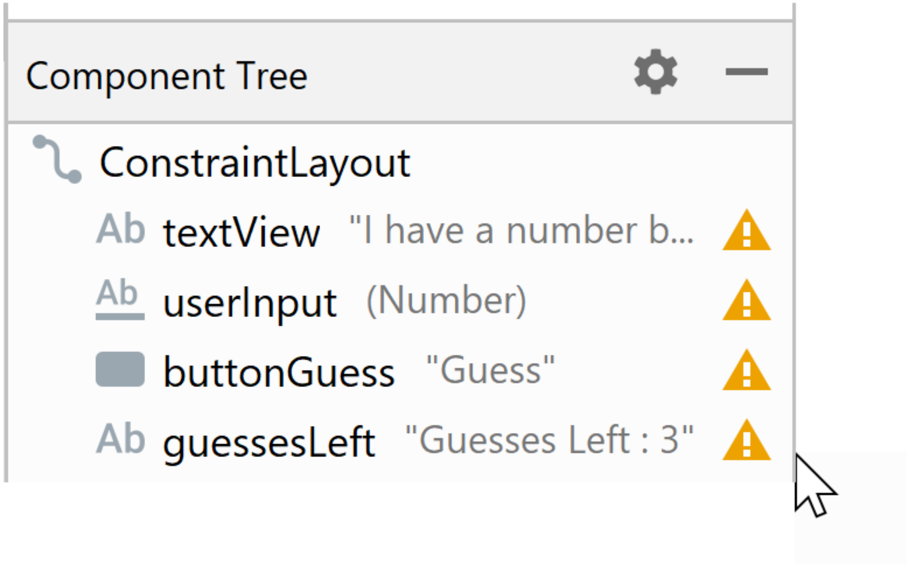
- **AndroidManifest.xml**
  - app config and settings (e.g., list app activities and required permissions)
- **java/...**
  - Kotlin source code
- **res/...** = resource files *(many are **XML**)*
  - drawable/ = images
  - layout/ = GUI layouts
  - menu/ = app menu options
  - values/ = constant values
  - 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



Component Tree

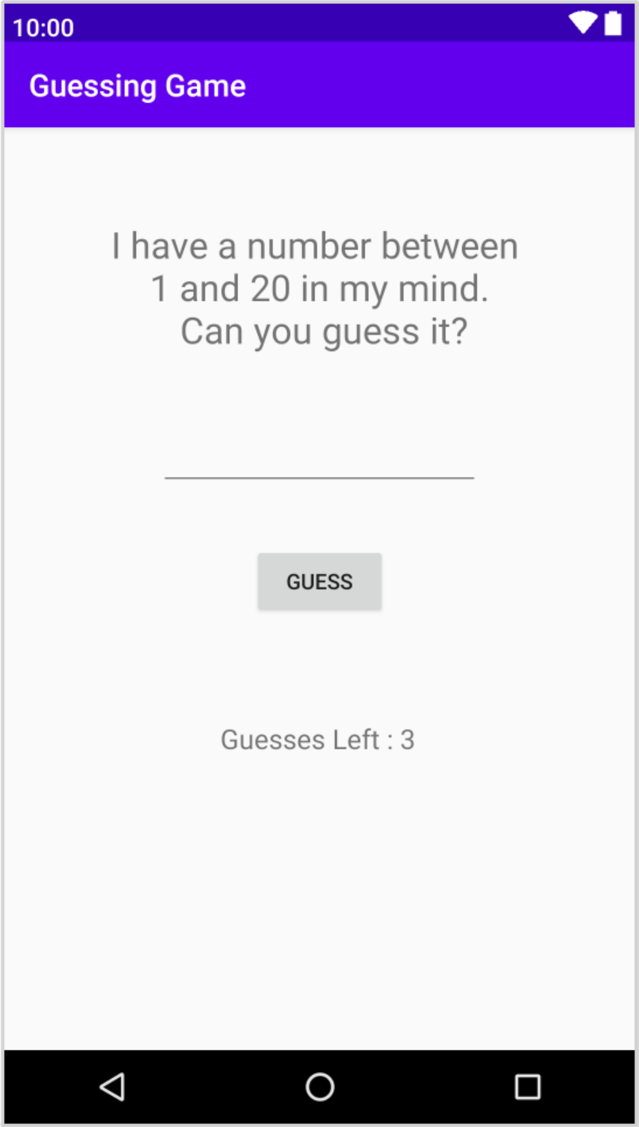
ConstraintLayout

Ab textView "I have a number b...

Ab userInput (Number)

buttonGuess "Guess"

Ab guessesLeft "Guesses Left : 3"



10:00

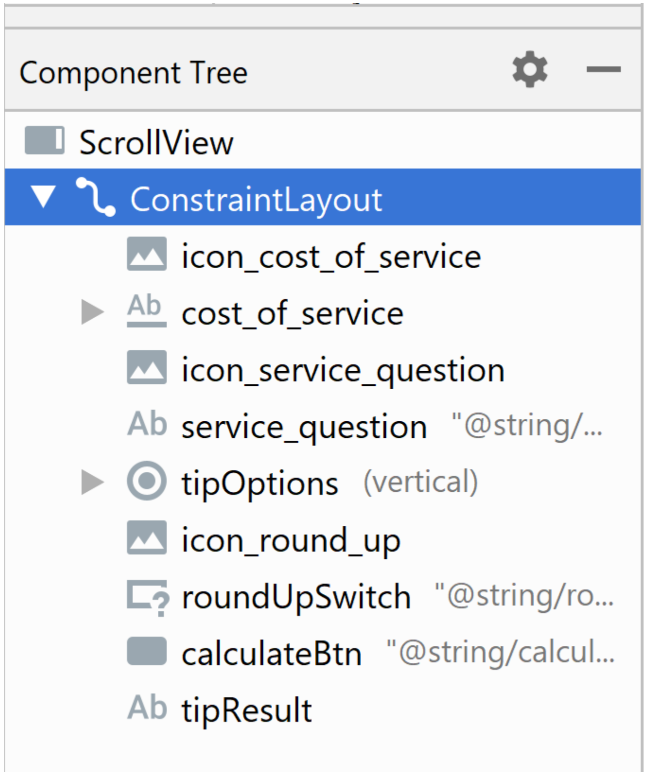
Guessing Game

I have a number between
1 and 20 in my mind.
Can you guess it?

GUESS

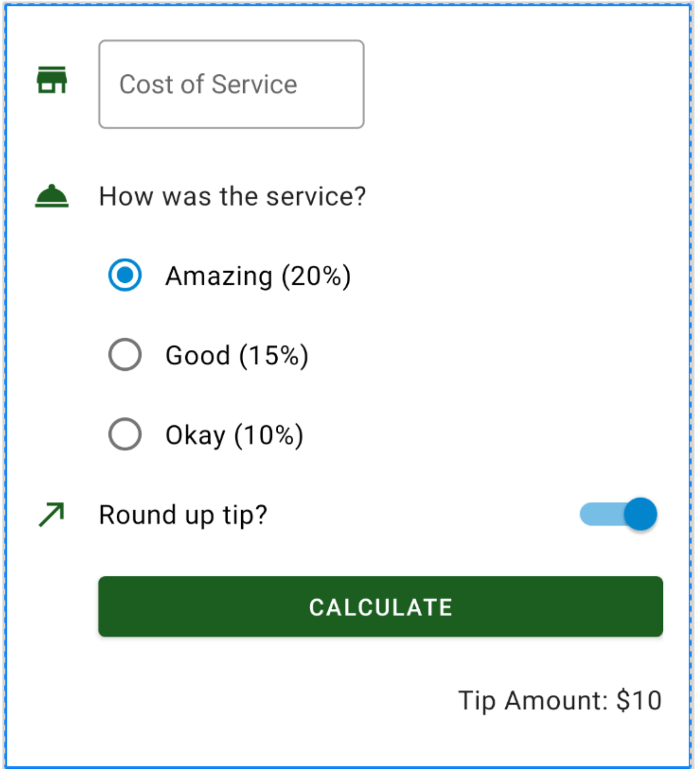
Guesses Left : 3

App 3 – Tips Calculator



Component Tree

- ScrollView
  - ConstraintLayout
    - icon_cost_of_service
    - cost_of_service
    - icon_service_question
    - service_question "@string/...
    - tipOptions (vertical)
    - icon_round_up
    - roundUpSwitch "@string/ro...
    - calculateBtn "@string/calcul...
    - tipResult



Cost of Service

How was the service?

● Amazing (20%)

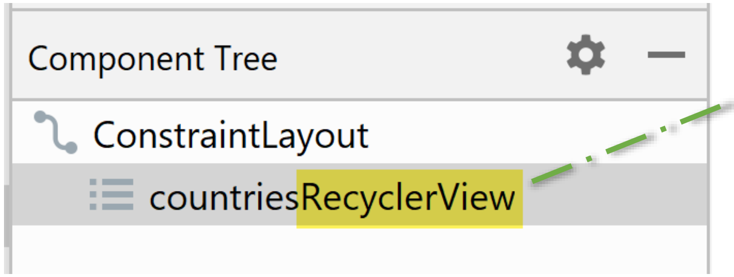
○ Good (15%)

○ Okay (10%)

Round up tip? [✓]

Tip Amount: $10

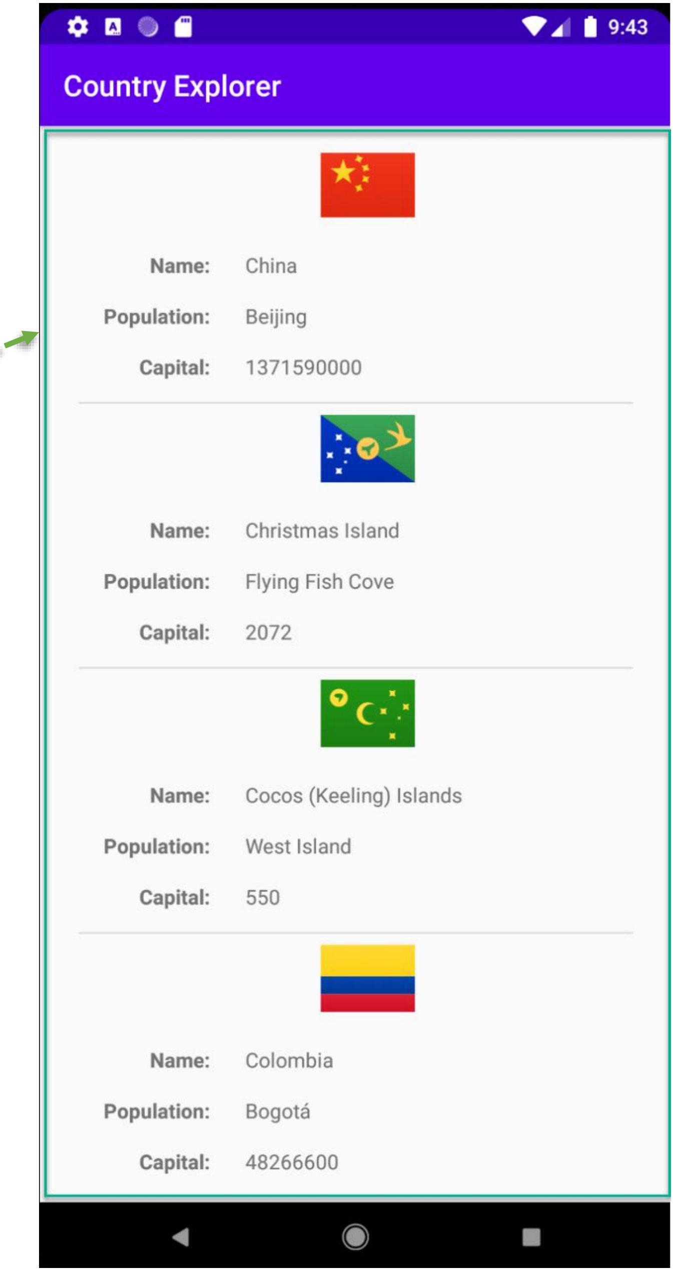
App 4 – Country Explorer



Component Tree

ConstraintLayout

countriesRecyclerView



9:43

Country Explorer

Name: China

Population: Beijing

Capital: 1371590000

Name: Christmas Island

Population: Flying Fish Cove

Capital: 2072

Name: Cocos (Keeling) Islands

Population: West Island

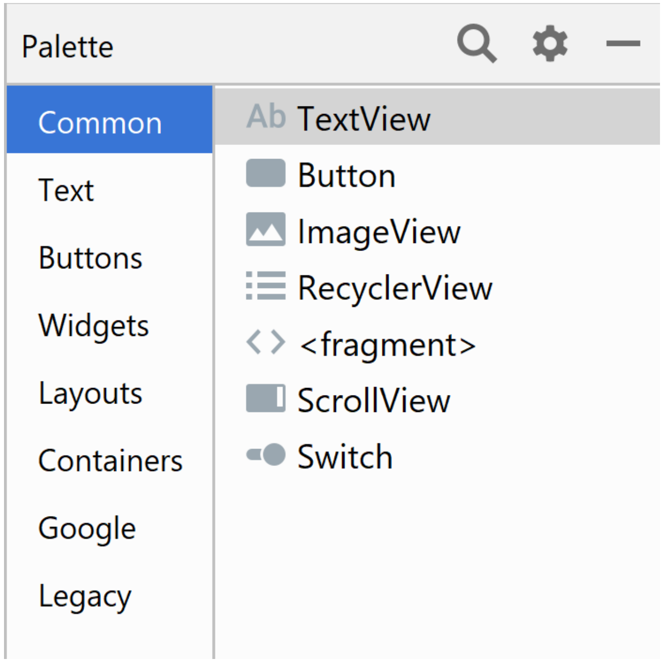
Capital: 550

Name: Colombia

Population: Bogotá

Capital: 48266600

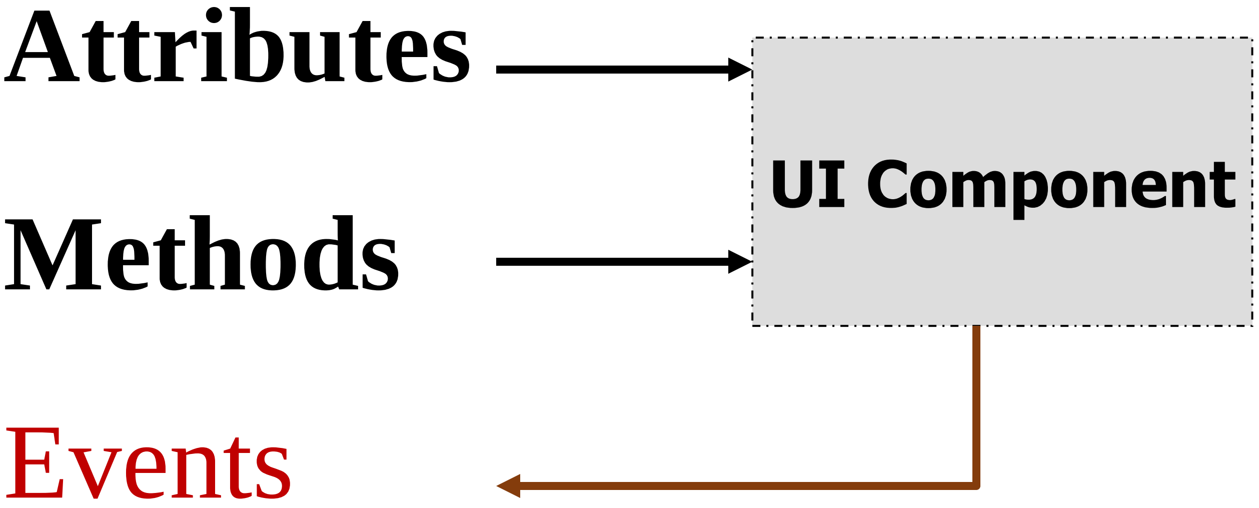
What Makes up Android UI?



- **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:



Attributes

Methods

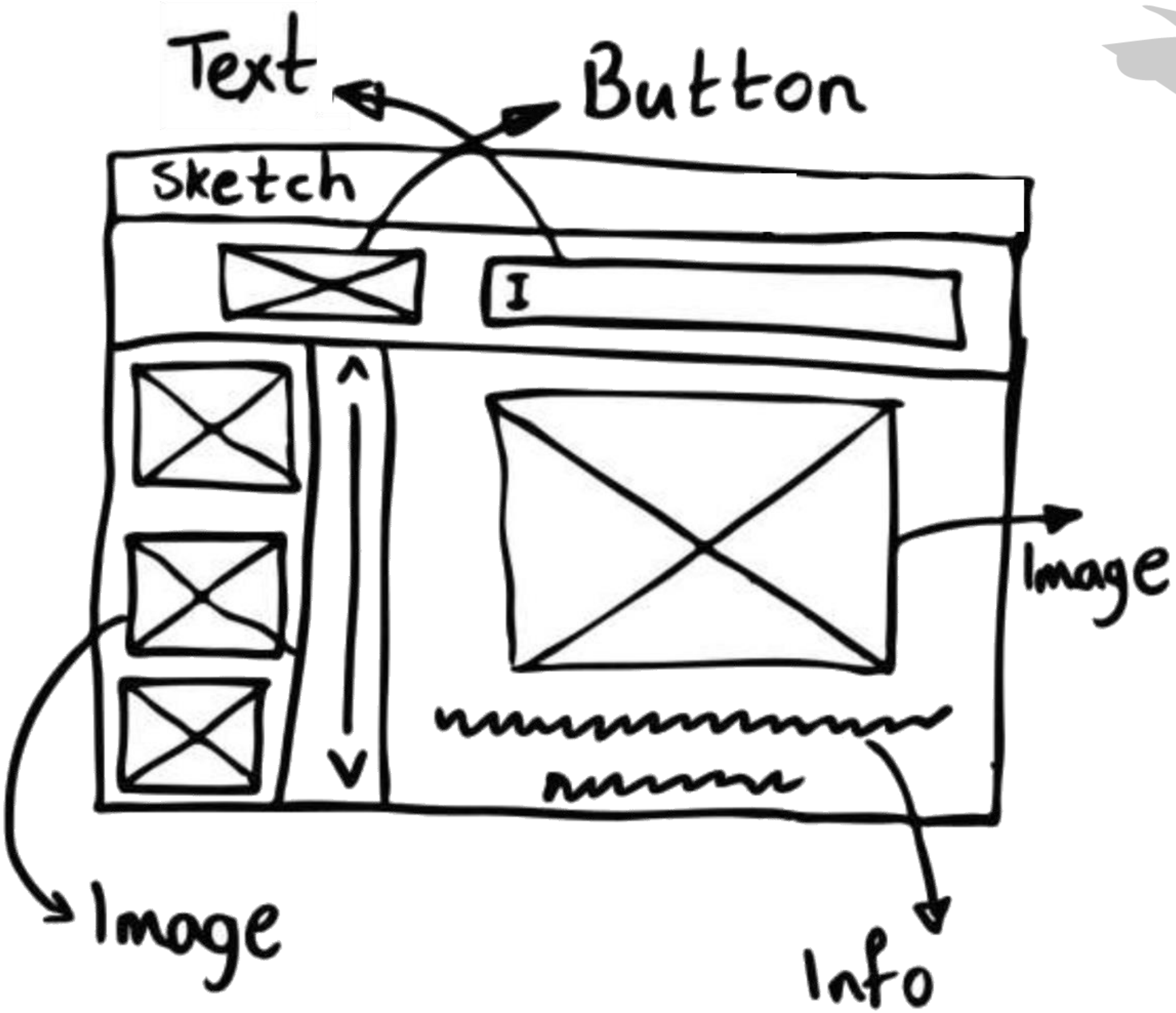
Events

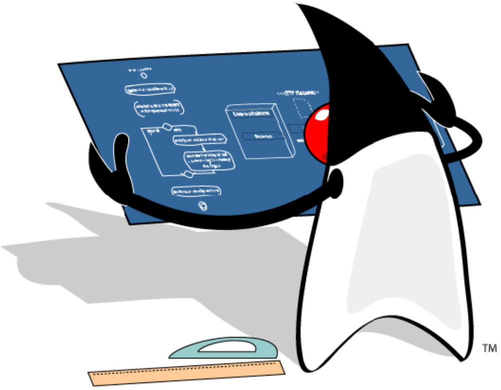
UI Component

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.

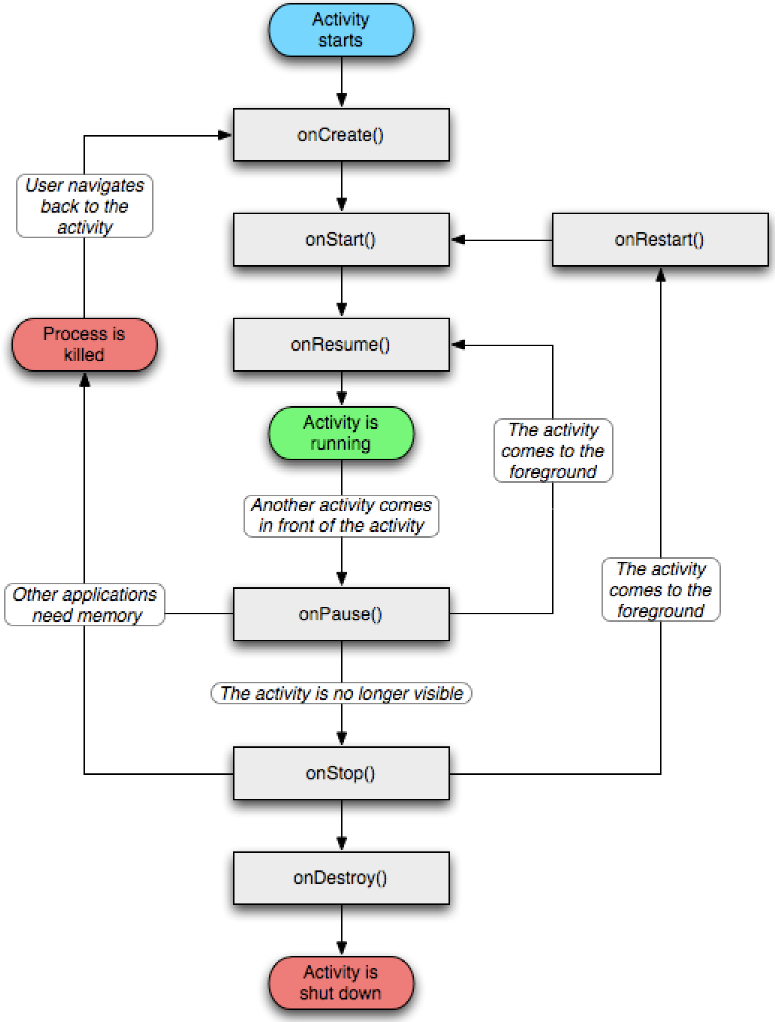
UI Sketch - Example





You may design different layouts per screen size

Activity Lifecycle


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

An activity has essentially **four states**:

- If an activity 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://codelabs.developers.google.com/android-kotlin-fundamentals/
- Android Dev Guide
  - https://developer.android.com/guide/