Android Development Lab

- Course no.: 61985

- Lecturer: Daniel Bouenos
- Email: dbouenos@braude.ac.il

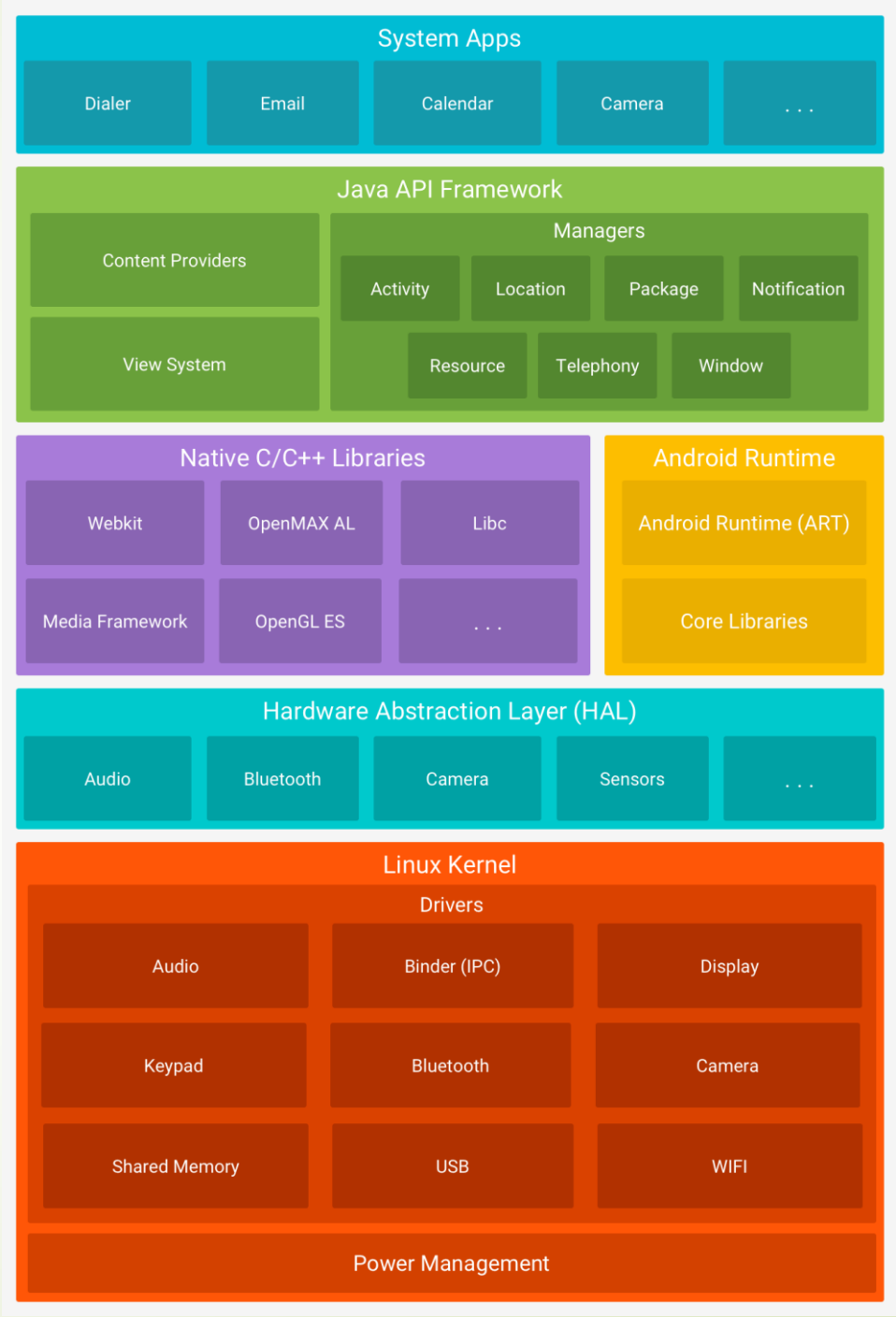
Course attendance & Assignments

- The attendance is not mandatory but will be registered
- Weekly lab exercises will be handed out
  - Minimum 10 submissions
  - Group of 2 students
- Final course project at the end of the semester (group of 2 students)
- Final course grade will be calculated 50% for the weekly exercises + 50% for the final project

Let's Start…

- **The Android Architecture**
  - based of Linux kernel open source (versions 4.4-4.9)
  - The HAL layer provides a generic interface to the various hardware devices drivers
  - Android Run-Time (ART) layer – this is actually the VM layer that execute our Java/Kotlin compiled code (DEX files)
  - The C/C++ libraries provide some extensions module to the underlying Linux OS
  - The Java API framework provides all the necessary SDK capabilities to the Android developer to build his own app
  - The System Apps layers is the "Application layer" that brings pre-built apps that the dev can utilize in their own apps

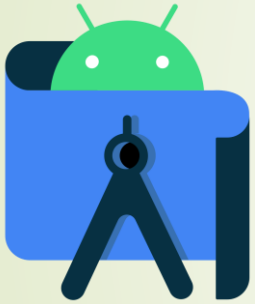
See details in this link…



Android Versions and API levels

- Up to version 9.0, the released versions were named after sweets (like marshmelo, oreo, etc.)
- Each version has a name, a number and an API level
- The API level refers to the internal changes in the SDK (not necessarily visible to the end user)
- The latest version is 11.0 with API leve 30
- See details in this link…

The IDE and the Emulator (AVD)



Development Environment

+ Available IDE's: Android Studio, IntelliJ, NetBeans,…

+ in our course, we will use "Android Studio"

+ Can be installed from here…

The emulator

Debug and test your app on your Windows desktop

It is installed and configured within the IDE, using the AVD manager

Points to discuss & demonstrate

1. Creating a new project
2. Project folders structure/views
3. The Gradle system
4. Separating the project resources
5. Manifest file
6. Creating & configuring the Emulator
7. Attaching a real Android device
8. Running & Debugging our code
9. Logging our code

Creating a new project

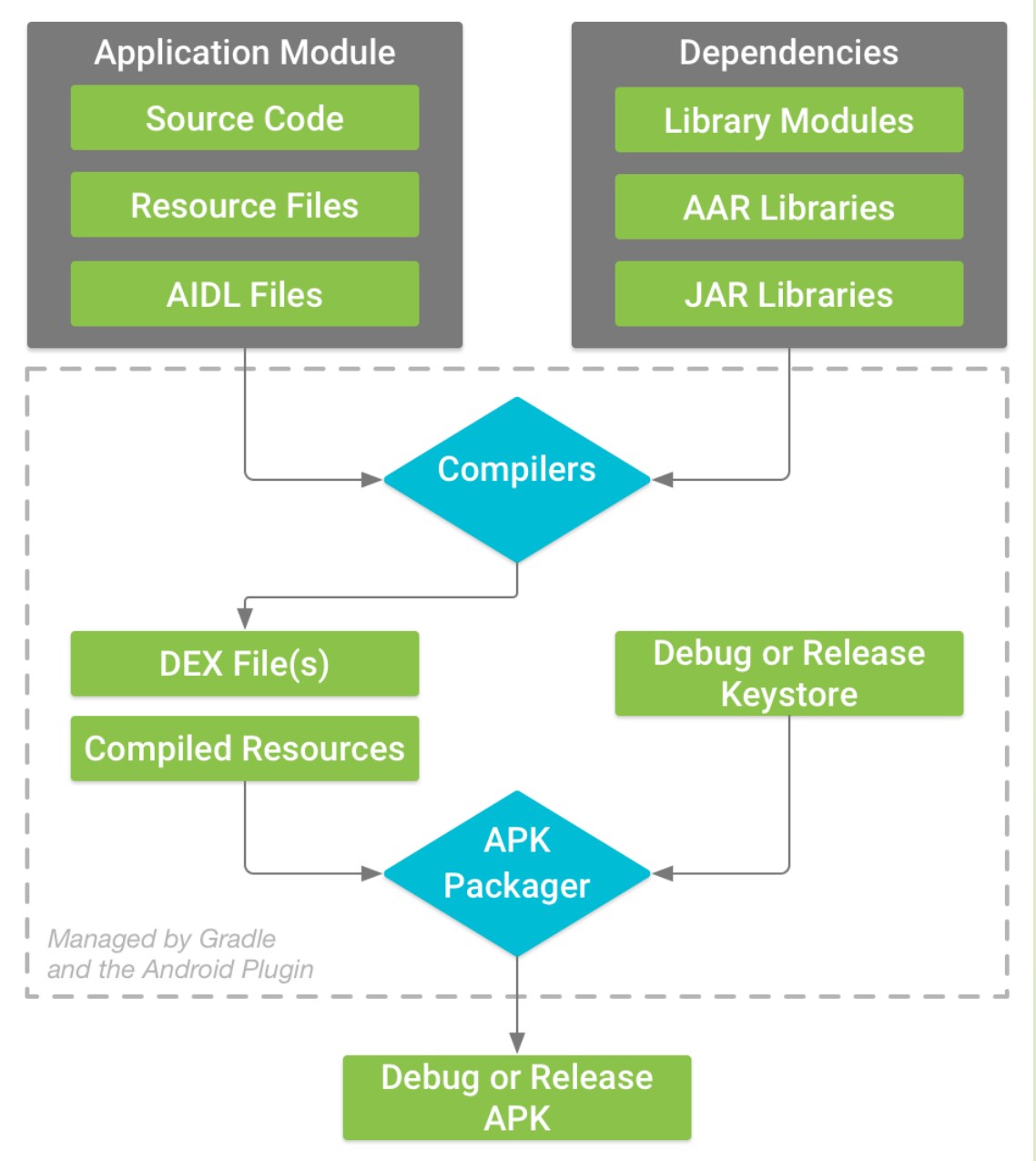
Let's live demonstrate…

Project folders structure/views

Let's live demonstrate…

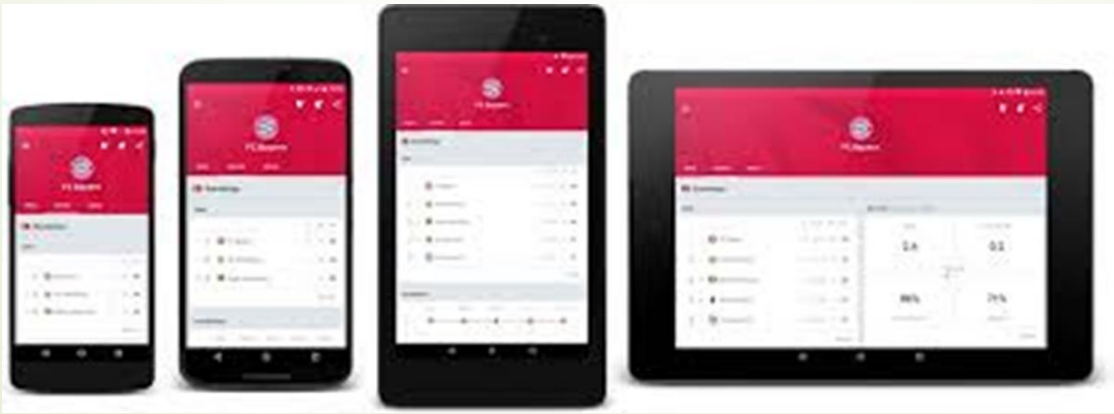
The Gradle system

- Orchestrate the build process of your whole app modules (local and remote) to create the APK file
- You set the list of modules that your project needs by adding their references to the 'dependencies' section of the gradle file
- Let's demonstrate…
- See details in this link…



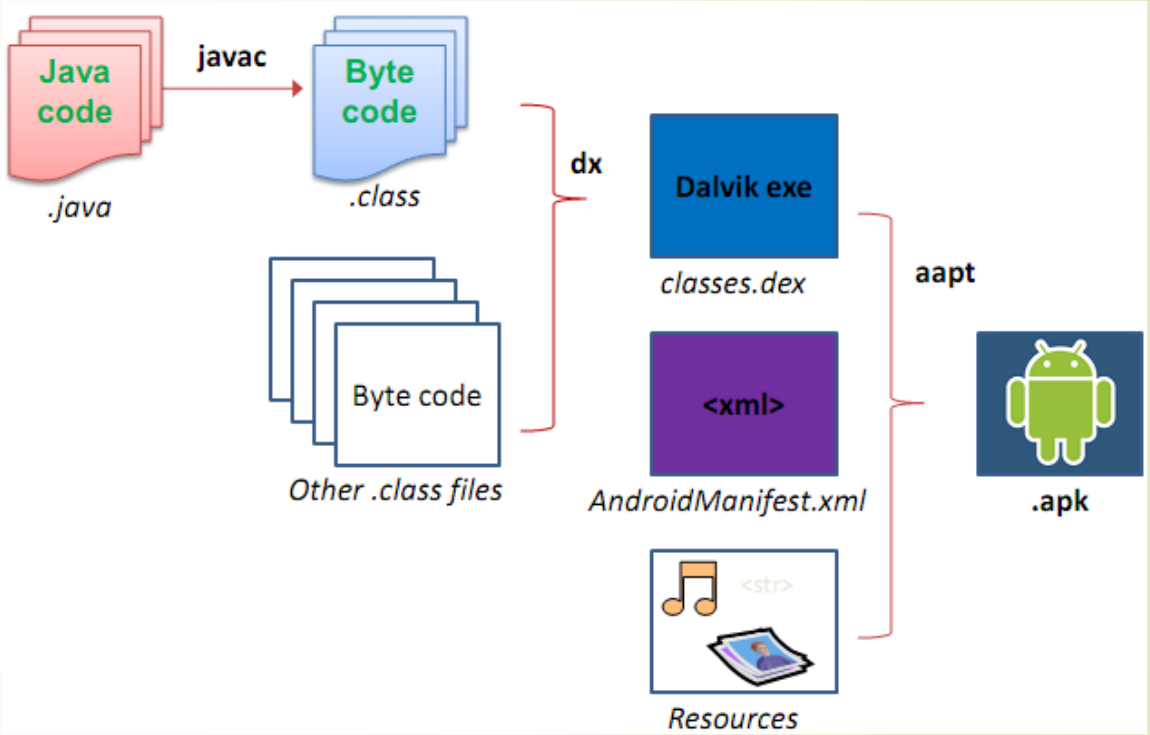
Separating the project resources

- Taking all the project resources **out** of the source code and manage them for each usage configuration separately
  - Achieve better **maintenance** of your app
- What resources we might have?
  - Images, strings, styles, UI layouts…
- What configurations we might have?
  - Different languages, screens sizes and resolutions, screen orientations, display directions, and more
- Let's demonstrate…
- See more details in this link…



The Manifest (XML) file

- Keeps and publish (to the OS)
  - general app settings
  - Required permissions
  - Global components (activities, services, …)
- Let's see details in this link…

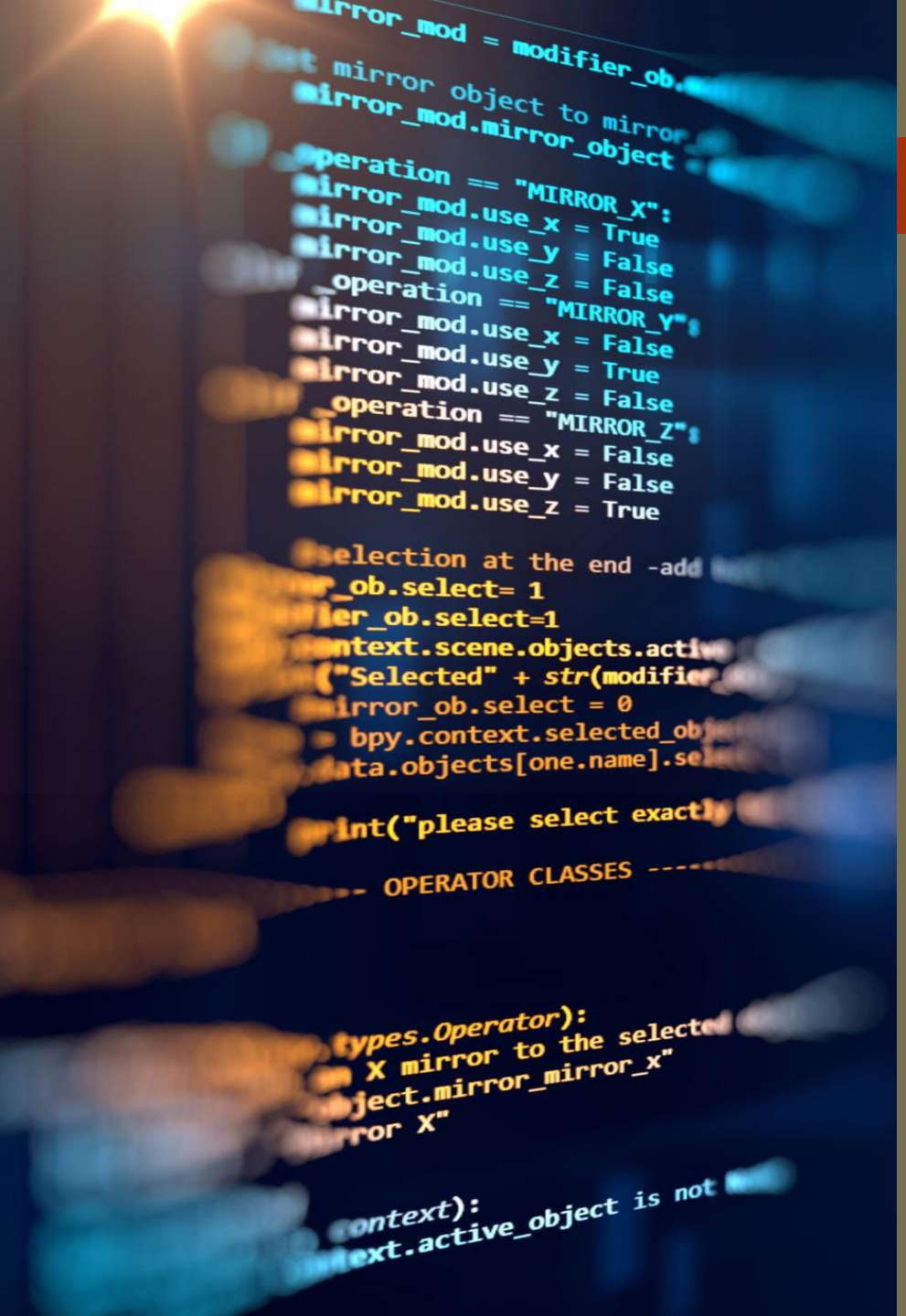


Creating & configuring the Emulator

- Allows you to test and debug your app without physically attaching a real device.
- Use the AVD manager to create and define various emulators
- Note that the emulator is a big resource killer !!!
- Let's demonstrate…
- See details in this link…

Attaching a real Android device

- This is the recommended way to test and debug your app
- Steps:
  1. you will need a real Android device
  2. Enable the "Developer options " by continuously tapping the "Settings-About Phone" entry
  3. Go to the "Settings->Developer option" and enable "USB debugging"
  4. You also might need to install USB driver from your phone manufacturer site
  5. Attach your phone to your PC and wait few seconds to let it recognize the device
  6. In this point, Android Studio should display your phone device
  7. Run your app in Android Studio and choose your device as the target
- See more info in this link…



Running & Debugging our code

- Built-in debugger (adb client-server module)
- This will be demonstrated next week…

Logging our ode

- Write and View Logs with Logcat screen
- This will be covered next week…

This week's issue: UI Design aspects

- Various types of Layouts
  - ConstraintLayout, LinearLayout, GridLayout, FrameLayout, TableLayout
- The strings resource file
- Widgets attributes
  - Measure units (dp, sp), see details in this link…
  - ID, height, width, padding, margin, background, constraints_…
- Different layout for landscape and portrait orientation
- The design screen
- Let's demonstrate…
- See more details on layout design in this link…

This week assignment – EX1

- 4 different layout (layout1 – layout4) in the same project
- How to create multiple layouts?
- How do I choose which layout the activity will load?
  - Comment out the relevant line

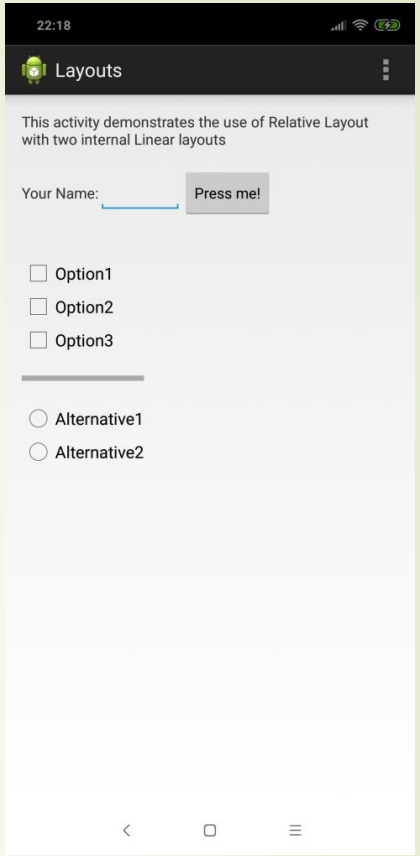
1. *Right-click the layout folder*
2. *New->Layout Resource file*
3. *Give it a name, ex: 'layout1'*
4. *Set the Root element type, ex: 'ConstraintLayout'*
5. *Press OK*

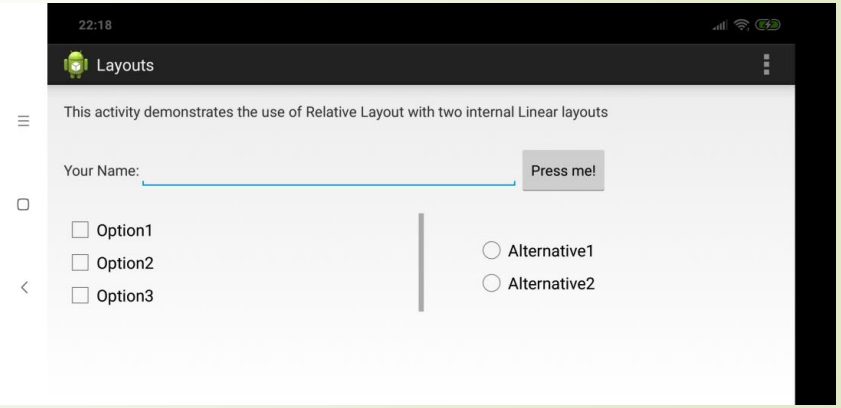
```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       // setContentView(R.layout.activity_main);
        setContentView(R.layout.layout1);
        setContentView(R.layout.layout2);
        setContentView(R.layout.layout3);
        setContentView(R.layout.layout4);
    }
}
```

Layout 1

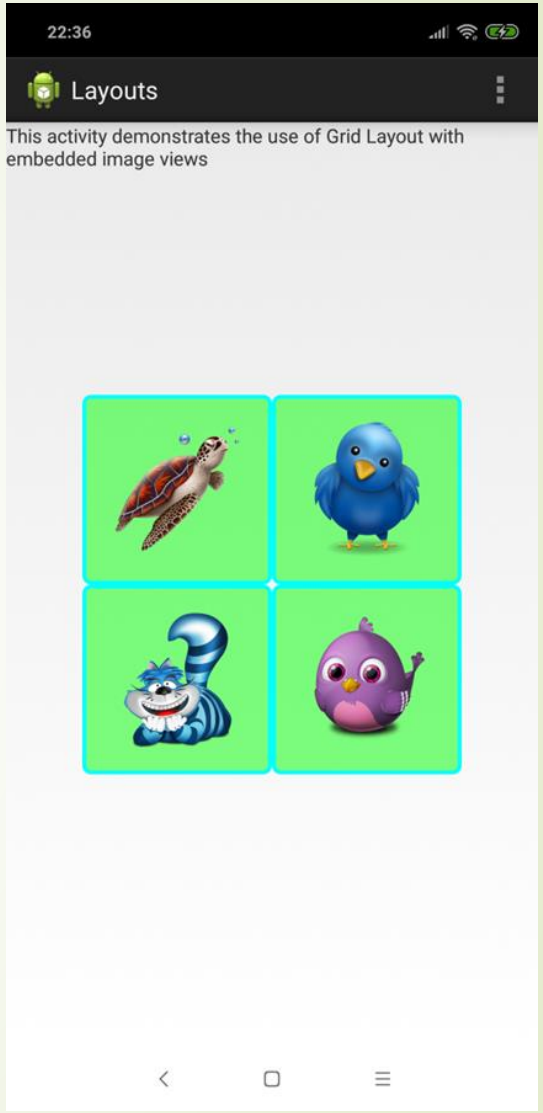
- Root Layout: ConstraintLayout
- The two radio buttons must be inside a ‘RadioGroup’ widget to make them related
- Create first the ‘portrait’ version of the layout and then clone it to the ‘landscape’ version and then refine it accordingly …





Layout 2

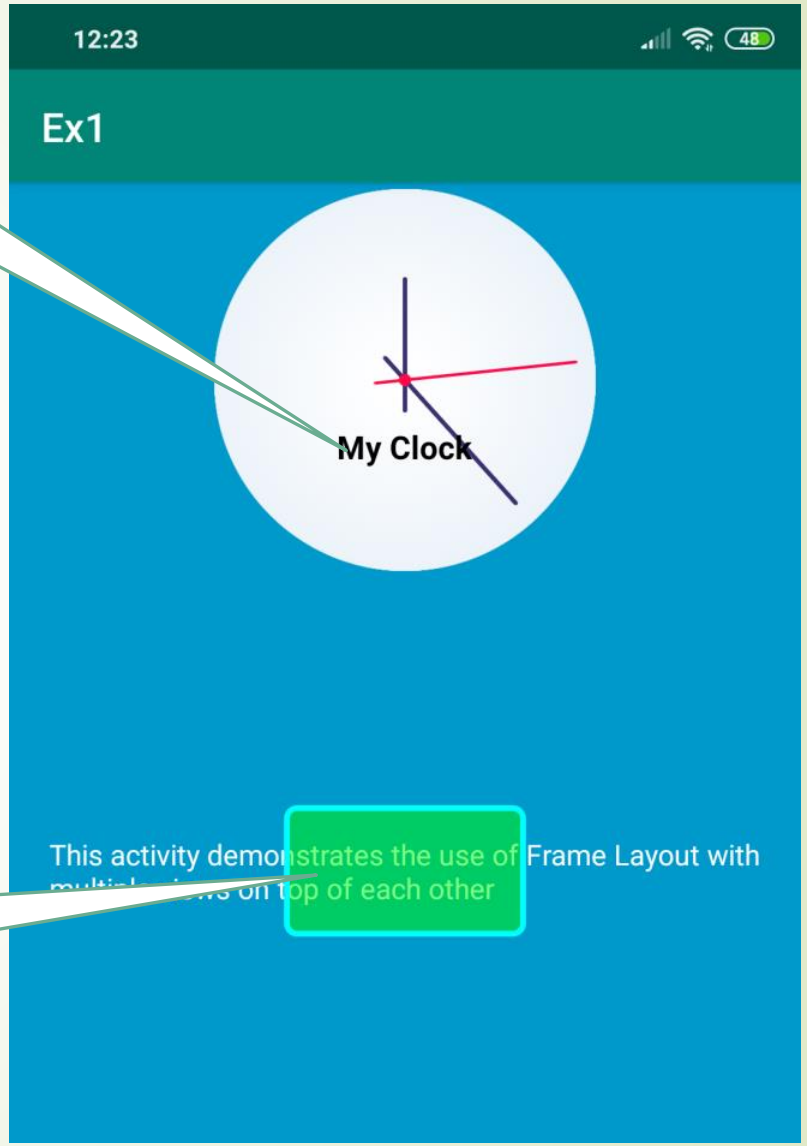
- Root layout: ConstraintLayout
- Use GrideLayout for the images' container
- Use ImageView widget for each image
- All the images will use the same border (XML) file in the 'drawable' folder (see example)



Layout 3

- Root Layout: FrameLayout
- The clock: use this link for it

TextView on top of the Clock widget



TextView with rounded corners on top of the text. Use Opacity background color to expose the text underneath

Layout 4

- Root layout: ConstraintLayout
- Use TableLayout for the table
- Each row in the table will use separate TableRow widget
- Each cell is actually a TextView widget
  - All the cells will use the same border (XML) file in the 'drawable' folder (see example)
  - Use layout_span attribute in the TextView to span over more than one column