

Introduction to Android OS, Kotlin, Android Studio, Activity Lifecycle & Intents

IN721: Mobile Application Development

Kaiako: Grayson Orr

Today's Content

- Android OS
- Kotlin
- Android Studio
- Activity lifecycle
- Intents

Android OS

History

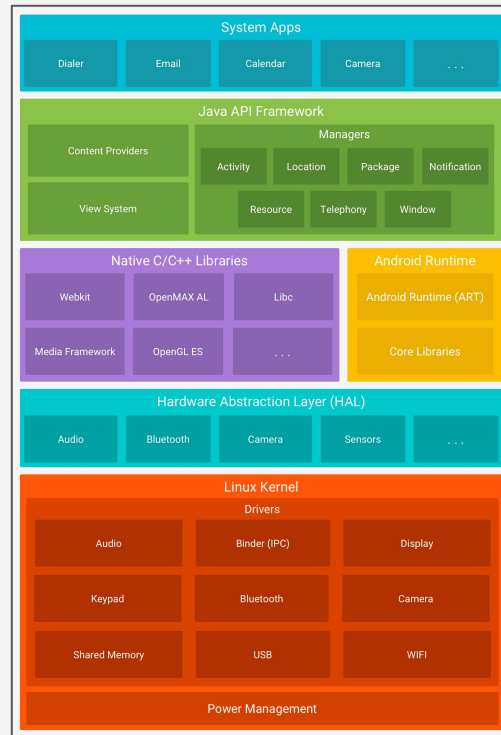
- Founded in Palo Alto, CA in October 2003
- Early intentions were to develop an advanced OS for digital cameras
- Google acquired Android Inc. in July 2005
- Developed by Open Handset Alliance (OHA)
- Defacto software for numerous smartphone manufacturing companies, i.e., Huawei, Meizu, Samsung, Xiaomi
- Android's mascot is a green robot
- Designed by Irina Block in November 2007
- No official name. The Android team at Google call it Bugdroid
- One of the most recognisable icons in the tech world

Linux Kernel

- Based on the Linux kernel long-term support (LTS) branches
- Android uses versions 4.14, 4.4 & 4.9 of the Linux kernel
- Kernel depends on the individual device
- A Linux distribution according to the Linux Foundation

Software Stack

- Apps
- Android framework
- Native libraries
- Android runtime
 - Android runtime (ART)
 - Core libraries
- Hardware abstraction layer (HAL)
 - Audio, bluetooth, camera
- Linux Kernel
 - Drivers
 - Power management



Kotlin

Additional Resources

- Documentation - <https://kotlinlang.org/docs/reference/>
- Style guide - <https://developer.android.com/kotlin/style-guide>
- Online sandbox - <https://play.kotlinlang.org/>

Comments

```
// This is a single-line comment
```

```
/* This is a  
   multi-line comment */
```

Variables

```
fun main() {  
    val x: Int = 1  
    val y = 2  
    val z: Int  
    z = 3  
    z = 4 // Val cannot be reassigned  
  
    var x = 1  
    x += 1  
    println("$x") // 2  
}
```

Functions

```
fun main() { // Entry point of a Kotlin app
    println("Hello World!")
}
```

Functions

```
fun add(x: Int, y: Int): Int { // Two Int params with Int return type
    return x + y
}
```

```
fun add(x: Int, y: Int) = x + y // Expression body with inferred return type
```

```
fun main() {
    val x = 1
    val y = 2
    println(add(x, y)) // 3
}
```

Functions

```
fun add(x: Int, y: Int): Unit { // Two Int params with Unit return type
    println("$x + $y = ${x + y}")
}
```

```
fun add(x: Int, y: Int) { // Omit Unit return type
    println("$x + $y = ${x + y}")
}
```

```
fun main() {
    val x = 1
    val y = 2
    add(x, y) // 1 + 2 = 3
}
```

String Templates

```
fun main() {  
    val x = 1  
    val y = 2  
    println("$x") // 1  
    println("$x + $y = ${x + y}") // 1 + 2 = 3  
}
```

Conditional Expressions

```
fun compare(x: Int, y: Int): Int {  
    if (x < y) {  
        return x  
    } else {  
        return y  
    }  
}
```

```
fun compare(x: Int, y: Int) = if (x < y) x else y
```

```
fun main() {  
    val x = 1  
    val y = 2  
    println(compare(x, y)) // 1  
}
```


For Loop

```
fun main() {  
    for (idx in 1..10) {  
        print(idx) // 12345678910  
    }  
  
    for (idx in 1..10 step 2) {  
        print(idx) // 13579  
    }  
  
    for (idx in 10 downTo 1) {  
        print(idx) // 10987654321  
    }  
}
```

While Loop

```
fun main() {  
    val vegetables = arrayOf("broccoli", "carrot", "potato")  
    var idx = 0  
    while (idx < vegetables.size) {  
        println("Vegetable at $idx is ${vegetables[idx]}") // Vegetable at 0 is broccoli  
                                                                // Vegetable at 1 is carrot  
                                                                // Vegetable at 2 is potato  
  
        idx++  
    }  
}
```

When Expression

```
fun greeting(isoCode: String): String =  
    when (isoCode) {  
        "en" -> "Hello"  
        "fr" -> "Bonjour"  
        "it" -> "Salve"  
        else -> "Greeting not found."  
    }  
  
fun main() {  
    var isoCode = "fr"  
    println(greeting(isoCode)) // Bonjour  
    isoCode = "es"  
    println(greeting(isoCode)) // Greeting not found.  
}
```

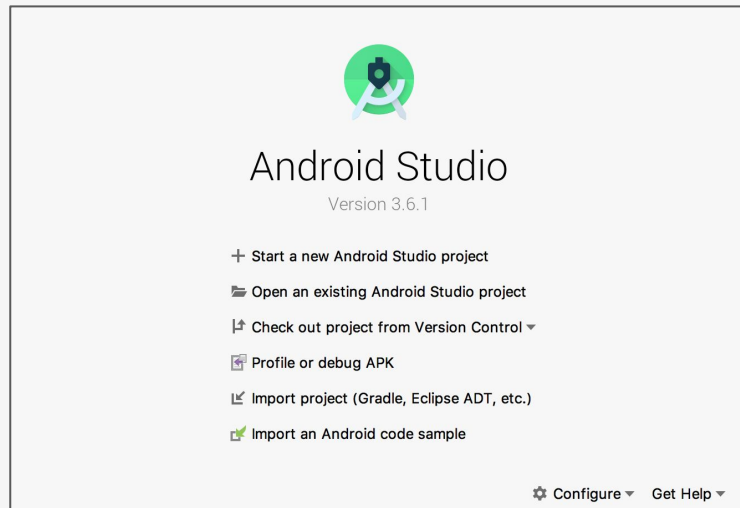
Android Studio

Android Studio

- Built on JetBrains' IntelliJ IDEA
 - Specifically designed for Android development
 - Available for download on Windows, macOS & Linux
- Supports Java, C++ & Kotlin
 - Kotlin replaced Java as Google's preferred language for Android development

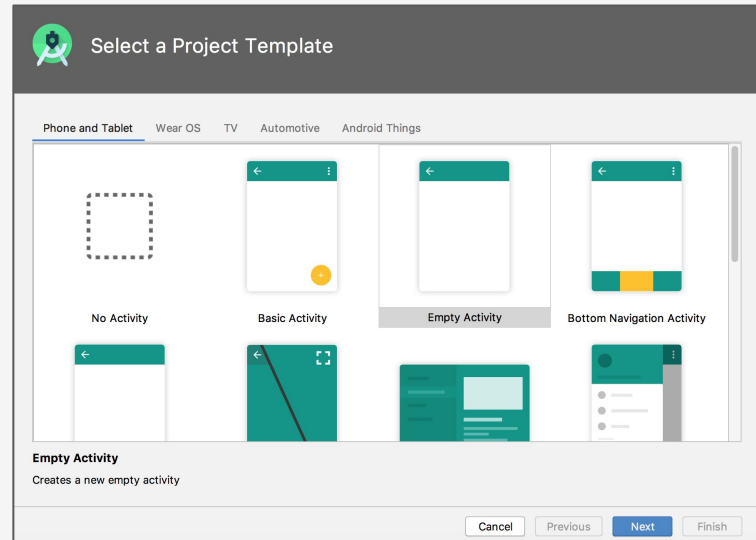
Startup

- Open Android Studio
- Start a new Android Studio project
- Download - <https://developer.android.com/studio>



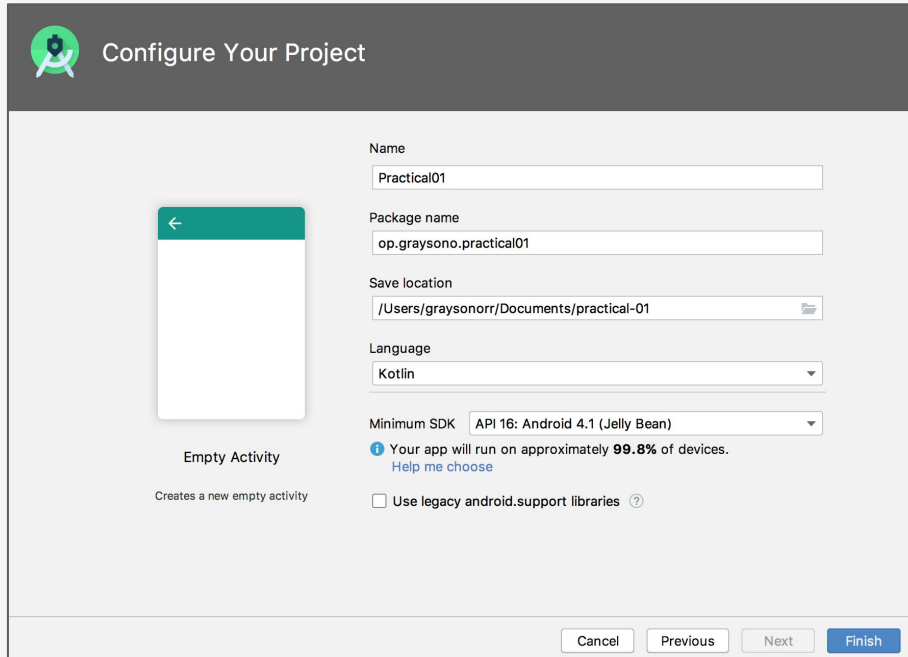
Project Template

- Select the empty activity template



Configure Project

- Name
- Package name
- Save location
- Language
- Minimum SDK



The screenshot shows the 'Configure Your Project' dialog box in Android Studio. On the left, there is a preview of an 'Empty Activity' with a green header bar and a back arrow. Below the preview, it says 'Empty Activity' and 'Creates a new empty activity'. On the right, there are several input fields and a checkbox. The 'Name' field is 'Practical01'. The 'Package name' field is 'op.graysono.practical01'. The 'Save location' field is '/Users/graysonorr/Documents/practical-01'. The 'Language' dropdown is set to 'Kotlin'. The 'Minimum SDK' dropdown is set to 'API 16: Android 4.1 (Jelly Bean)'. Below these fields, there is a blue information icon and text: 'Your app will run on approximately 99.8% of devices.' with a link 'Help me choose'. At the bottom right, there is a checkbox 'Use legacy android.support libraries' with a help icon. At the very bottom, there are four buttons: 'Cancel', 'Previous', 'Next', and 'Finish'.

Configure Your Project

Name
Practical01

Package name
op.graysono.practical01

Save location
/Users/graysonorr/Documents/practical-01

Language
Kotlin

Minimum SDK API 16: Android 4.1 (Jelly Bean)

Empty Activity
Creates a new empty activity

Information: Your app will run on approximately **99.8%** of devices.
[Help me choose](#)

☐ Use legacy android.support libraries ?

Buttons: Cancel Previous Next Finish

API Version Distribution

- Android platform version
- API level
- Cumulative distribution

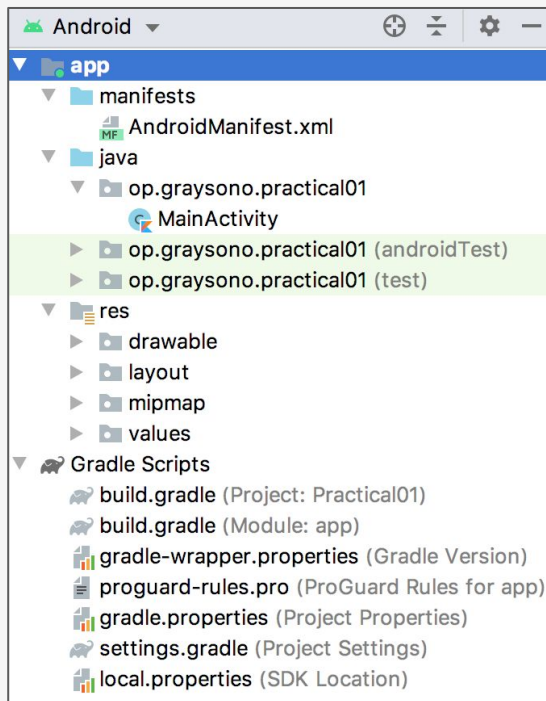
ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION	Jelly Bean	
4.0 Ice Cream Sandwich	15		App Components Isolated services Memory management Content providers Live wallpapers App stack navigation	Multimedia Enhanced RenderScript functionality
4.1 Jelly Bean	16	99.8%		
4.2 Jelly Bean	17	99.2%		
4.3 Jelly Bean	18	98.4%		
4.4 KitKat	19	98.1%	Multimedia Media codecs Record audio on cue Timed text tracks Gapless playback	Animations Activity launch animations Time animator
5.0 Lollipop	21	94.1%		
5.1 Lollipop	22	92.3%	Camera Auto focus movement Camera sounds	User Interface Improved notifications Controls for system UI Remote views More font families
6.0 Marshmallow	23	84.9%		
7.0 Nougat	24	73.7%	Connectivity Android Beam over Bluetooth Network service discovery Wi-Fi P2P service discovery Detect metered networks	Input Framework Multiple input devices Vibrate for input controllers
7.1 Nougat	25	66.2%		
8.0 Oreo	26	60.8%	Accessibility Accessibility service APIs Customizable app navigation More accessible widgets	
8.1 Oreo	27	53.5%		
9.0 Pie	28	39.5%	Copy and Paste Copy and paste with intents	
10, Android 10	29	8.2%		

<https://developer.android.com/about/versions/android-4.1.html>

Cancel OK

Project Structure

- app
 - manifests
 - java
 - res
- Gradle Scripts



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="op.graysono.practical01">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

AndroidManifest.xml

- The system must know that the component exists
- Reads the app's manifest file
- Identifies any user permissions
- Declares the minimum API level
- Declares hardware & software features
- Declares API libraries

Resource Directory

- Drawable
 - A general concept for a graphic that can draw to the screen
 - Resource: [Drawable Resource](#)
- Layout
 - Defines the architecture for the UI in an Activity or a component of a UI
 - View & ViewGroup
 - Resource: [Layout Resource](#)
- Mipmap
 - Drawable files for different launcher densities
 - Image Asset Studio
- Values
 - Colors
 - Strings
 - Styles

Colors

- colors.xml
- A color value defined in XML
- RGB value & alpha channel
- Color resource - @android:color/<name>
- Resource: [Color Resource](#)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#6200EE</color>
    <color name="colorPrimaryDark">#3700B3</color>
    <color name="colorAccent">#03DAC5</color>
</resources>
```

Strings

- strings.xml
- Provides text strings for an app
- Optional text styling & formatting
- String - a single string
- String array - array of strings
- Quantity strings - different strings for pluralisation
- Resource: [String Resource](#)

```
<resources>  
    <string name="app_name">Practical01</string>  
</resources>
```

Styles

- styles.xml
- Defines the format & look for a UI
- Can be applied to a View, activity or app
- Resource: [Style Resource](#)

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```


Gradle Scripts

- `build.gradle (Module)`
 - Defines the module-specific build configurations
- `build.gradle (Project)`
 - Defines your build configurations that apply to all modules

MainActivity.kt

- Default activity
- AppCompatActivity
- onCreate()
- setContentView()

```
package op.graysono.practical01

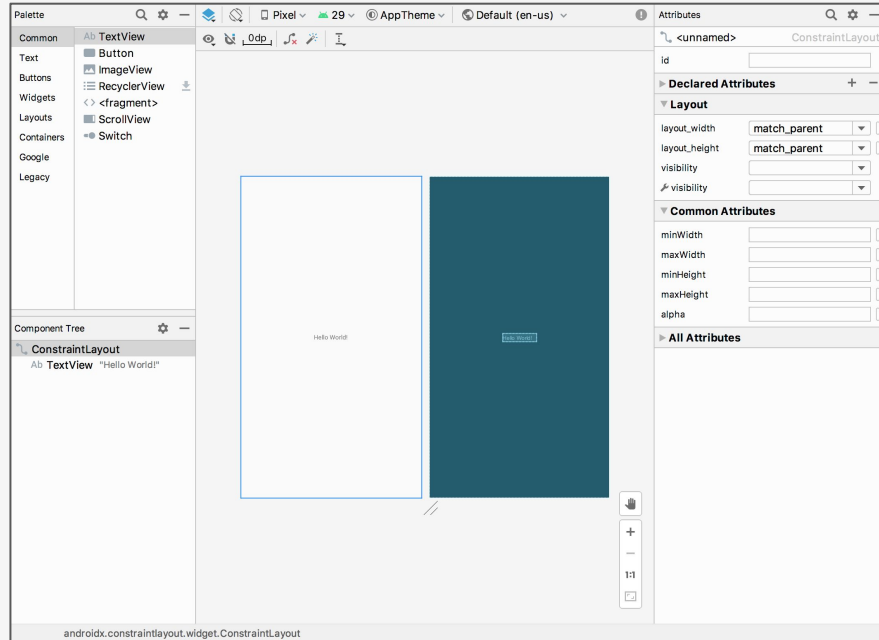
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

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

activity_main.xml

- Design view



activity_main.xml

- Code view

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

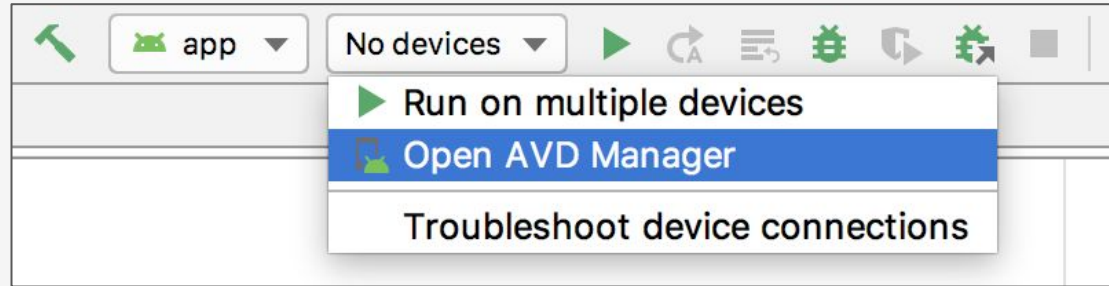
</androidx.constraintlayout.widget.ConstraintLayout>
```

Emulator

- Simulates an Android device on your computer
- You can test your app on a variety of devices & API levels
- Provides almost all of the capabilities of a real Android device

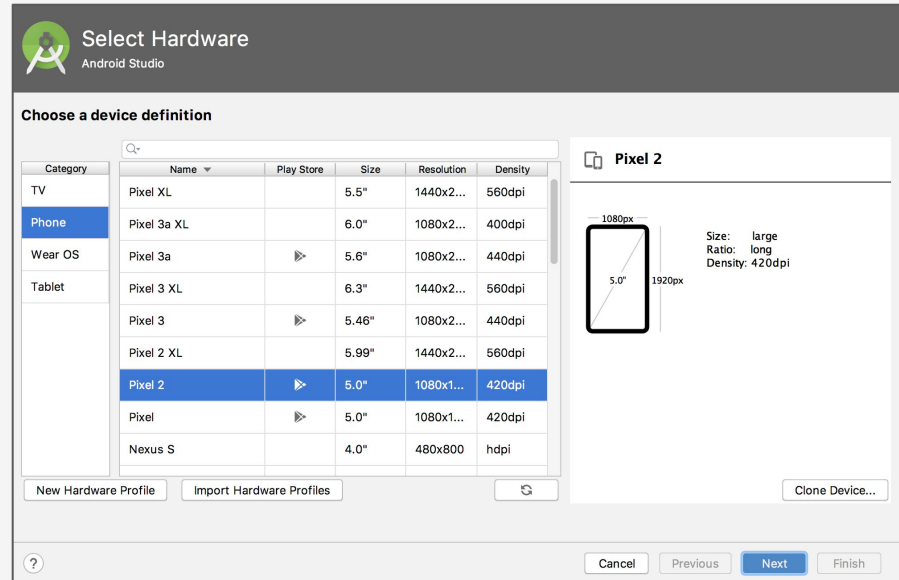
Emulator

- AVD manager



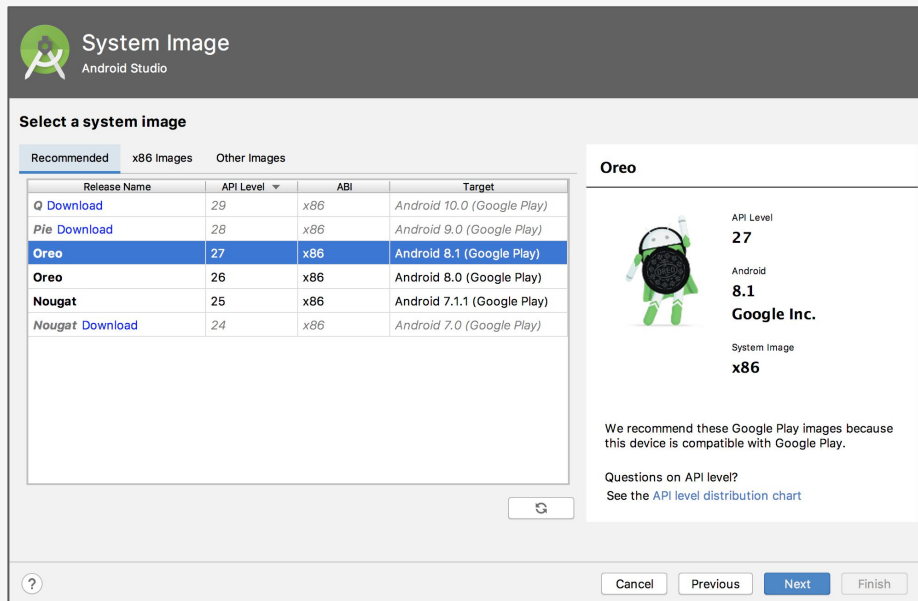
Emulator

- Choosing a device definition
 - Name
 - Play Store
 - Size
 - Resolution
 - Density



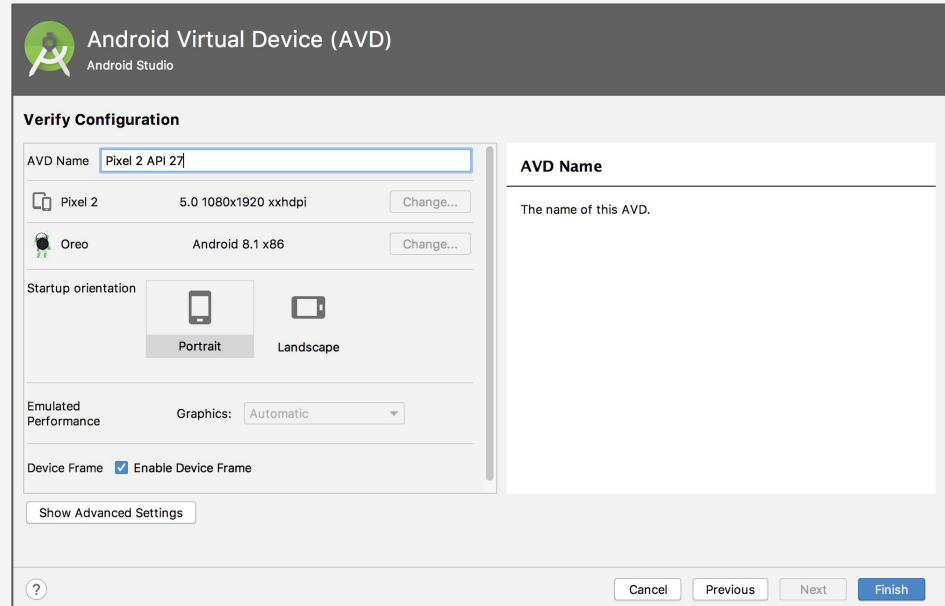
Emulator

- Selecting a system image
 - Release name
 - API level
 - ABI
 - Target



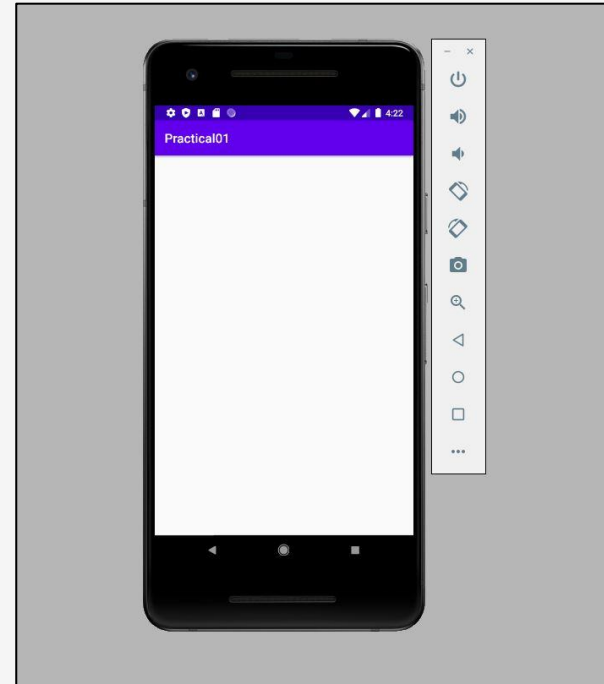
Emulator

- Verifying configuration
 - AVD name
 - Startup orientation



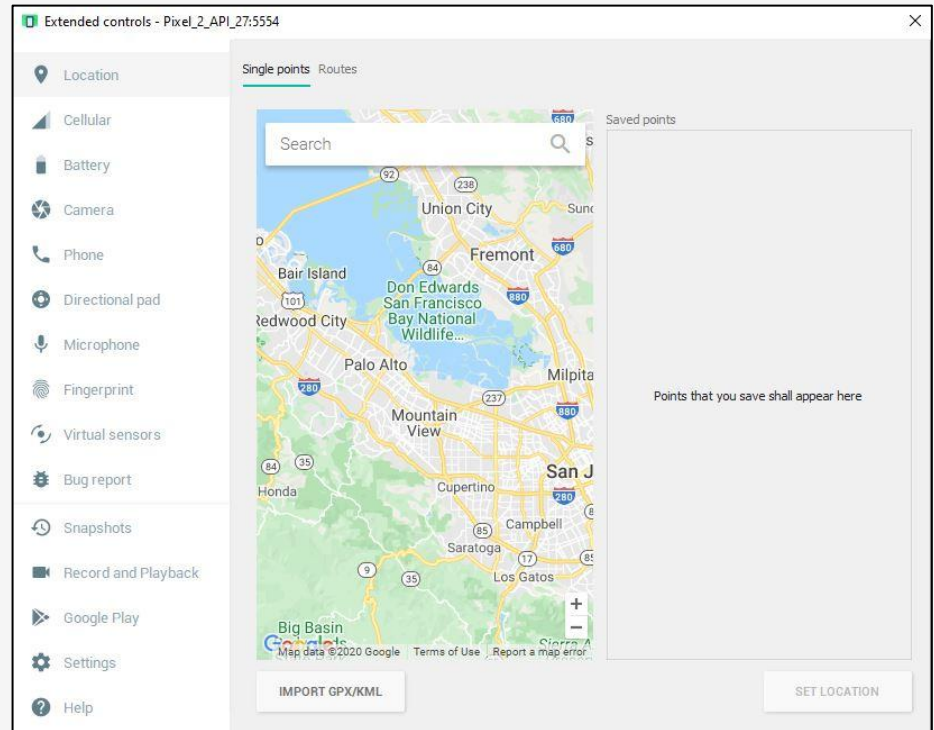
Emulator

- Click the green play button
- The emulator will boot
- Controls
 - Power
 - Volume
 - Rotate
 - Screenshot
 - Zoom
 - Back
 - Home
 - Overview
 - Extended controls (horizontal ellipsis)



Emulator

- Extended controls
 - Location
 - Cellular
 - Battery
 - Camera
 - Phone
 - Directional pad
 - Microphone
 - Fingerprint
 - Virtual sensors
 - Bug report
 - Snapshots
 - Record and Playback
 - Google Play
 - Settings
 - Help

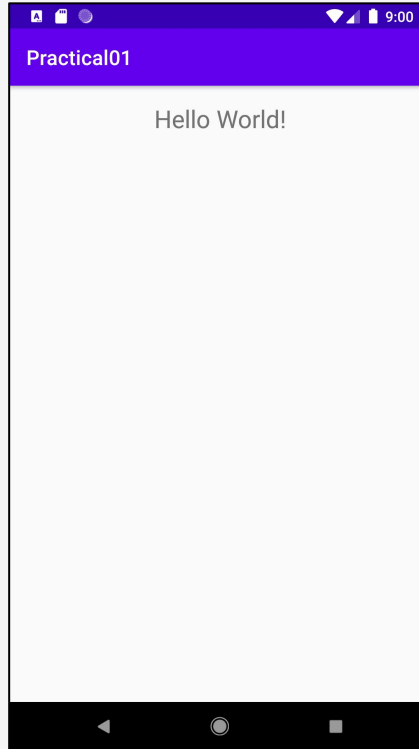


activity_main.xml

- Go to activity_main.xml
- Update the TextView widget
- Run app (Shift+F10)

```
<TextView
    android:id="@+id/output_text"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginRight="16dp"
    android:gravity="center"
    android:text="Hello World!"
    android:textSize="24sp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Emulator



activity_main.xml

- Add a Button
- Why is `android:text="Click Me!"` highlighted orange?

```
<Button
    android:id="@+id/click_me_button"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Click Me!"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/output_text" />
```

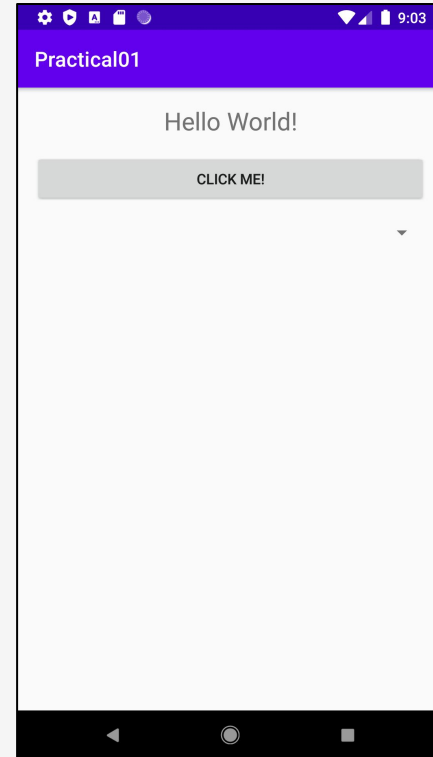
activity_main.xml

- Add a Spinner
- Run app

```
<Spinner
    android:id="@+id/months_spinner"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/click_me_button" />
```

Emulator

- **Click Me!** Button does not do anything
- Spinner has no items



MainActivity.kt

- Go to MainActivity.kt
- Extends AppCompatActivity

```
class MainActivity : AppCompatActivity() {  
  
    private lateinit var outputText: TextView  
    private lateinit var clickMeButton: Button  
    private lateinit var monthsSpinner: Spinner  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        outputText = findViewById(R.id.output_text)  
        clickMeButton = findViewById(R.id.click_me_button)  
        monthsSpinner = findViewById(R.id.months_spinner)  
    }  
}
```

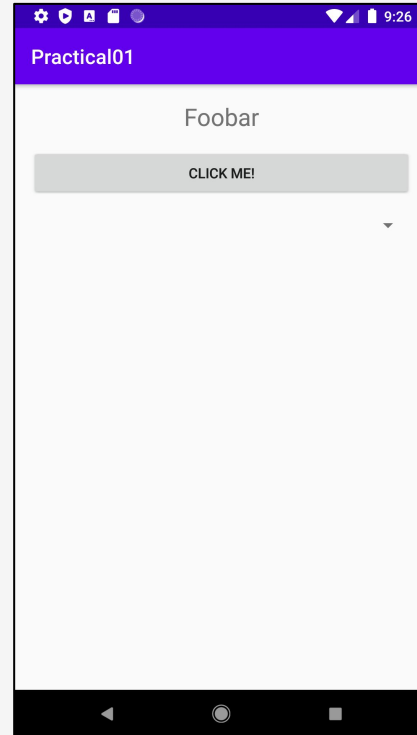
MainActivity.kt

- Add a click listener
- Run app

```
class MainActivity : AppCompatActivity() {  
  
    private lateinit var outputText: TextView  
    private lateinit var clickMeButton: Button  
    private lateinit var monthsSpinner: Spinner  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        outputText = findViewById(R.id.output_text)  
        clickMeButton = findViewById(R.id.click_me_button)  
        monthsSpinner = findViewById(R.id.months_spinner)  
  
        clickMeButton.setOnClickListener { outputText.text = getString(R.string.foobar) // Foobar }  
    }  
}
```

Emulator

- Click on the **Click Me!** Button
- TextView text will be set to Foobar



MainActivity.kt

- This is not very exciting...I want to display items in the Spinner
- ArrayAdapter
 - Returns a view for each object in a collection of data objects, i.e. months string array
 - Used with list-based widgets such as ListView (legacy) & Spinner
 - Resource: [ArrayAdapter](#)

strings.xml

- Go to strings.xml
- Add a string array

```
<string-array name="months">  
  <item>January</item>  
  <item>February</item>  
  <item>March</item>  
  <item>April</item>  
  <item>May</item>  
  <item>June</item>  
  <item>July</item>  
  <item>August</item>  
  <item>September</item>  
  <item>October</item>  
  <item>November</item>  
  <item>December</item>  
</string-array>
```

MainActivity.kt

- Create a private function called populateSpinner()
- Args - Spinner & Array<String>
- ArrayAdapter(Context context, int resource, List<T> objects)

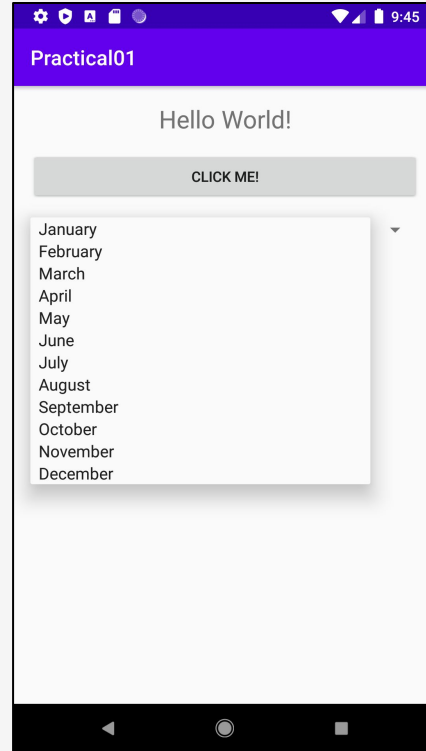
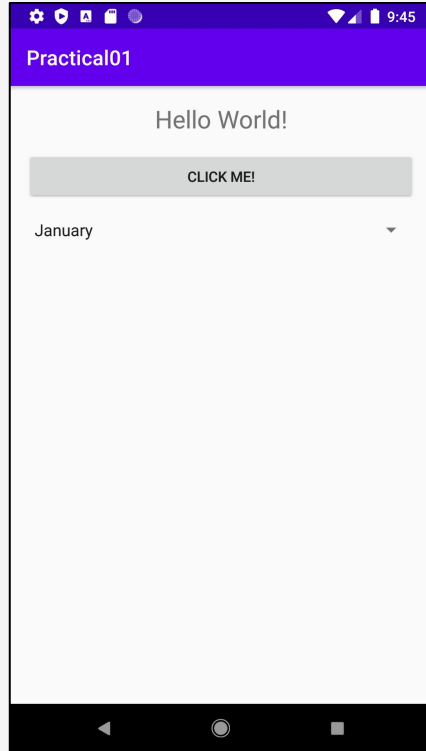
```
private fun populateSpinner(spinner: Spinner, array: Array<String>) {  
    val layoutID: Int = android.R.layout.simple_spinner_item  
    spinner.adapter = ArrayAdapter(this@MainActivity, layoutID, array)  
}
```

MainActivity.kt

- Call populateSpinner() in the onCreate()
- Run app

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    outputText = findViewById(R.id.output_text)  
    clickMeButton = findViewById(R.id.click_me_button)  
    monthsSpinner = findViewById(R.id.months_spinner)  
  
    clickMeButton.setOnClickListener { outputText.text = getString(R.string.foobar) }  
  
    populateSpinner(monthsSpinner, resources.getStringArray(R.array.months))  
}
```

Emulator



MainActivity.kt

- Alternative

```
private fun populateSpinnerAlt(spinner: Spinner, array: Int) {  
    val layoutID: Int = android.R.layout.simple_spinner_item  
    ArrayAdapter.createFromResource(  
        this@MainActivity,  
        array,  
        layoutID  
    ).also { adapter ->  
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)  
        spinner.adapter = adapter  
    }  
}
```

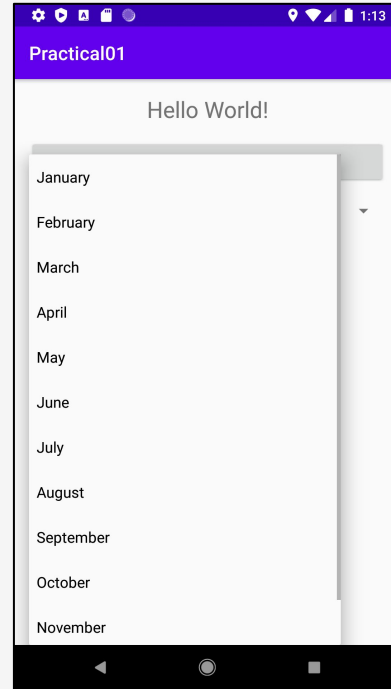
MainActivity.kt

- Comment out populateSpinner()
- Call populateSpinnerAlt() in the onCreate()
- Run app

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    outputText = findViewById(R.id.output_text)  
    clickMeButton = findViewById(R.id.click_me_button)  
    monthsSpinner = findViewById(R.id.months_spinner)  
  
    clickMeButton.setOnClickListener { outputText.text = getString(R.string.foobar) }  
  
    populateSpinnerAlt(monthsSpinner, R.array.months)  
}
```

Emulator

- Different drop down layout file



Practical Part 1

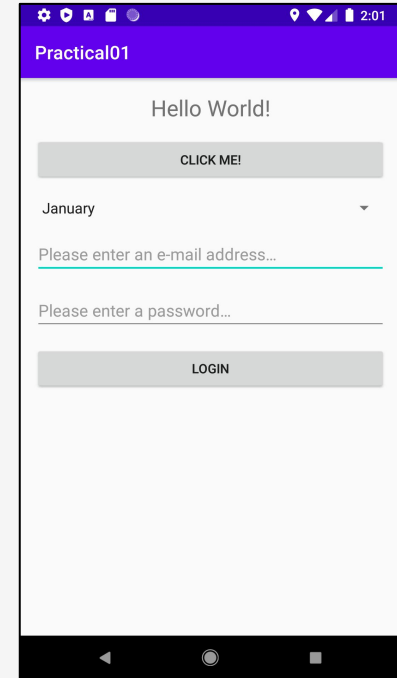
- Please use the current app
- Independent tasks:
 - Implement the code as specified in the previous lecture slides
 - Refactor the **Click Me!** Button click listener so the TextView text is set to the value of the selected Spinner item
 - In activity_main.xml, under the Spinner, add two EditText (one e-mail & one password) widgets
 - In activity_main.xml, under the EditText widgets, add one Button with the text **Login**
 - Ensure each EditText has an autofillHints attribute
 - Resource: [Autofill Optimize](#)
 - Create a **Login** Button click listener which gets the value from both EditText widgets & sets it to the TextView text
 - Ensure correct handling of inputs, i.e. empty inputs
 - Resources: [isBlank](#) & [error](#)

Practical Part 1 - Additional Resources

- Additional resources:
 - Special characters - [Escaping Quotes](#)
 - Formatted strings - [Formatting Strings](#)
 - Dimension - [Dimension Resource](#)

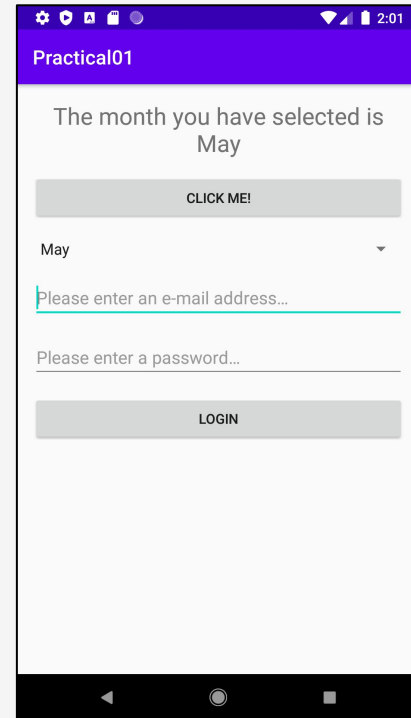
Practical Part 1 - Expected Output

- I will leave the UI upto you...just make sure you implement the required functionality



Practical Part 1 - Expected Output

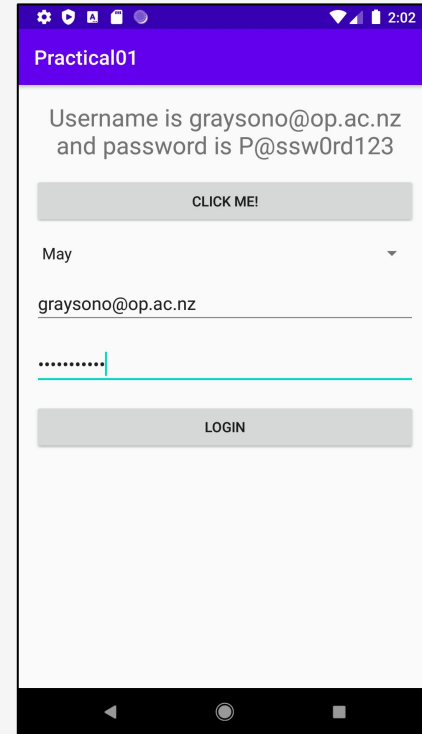
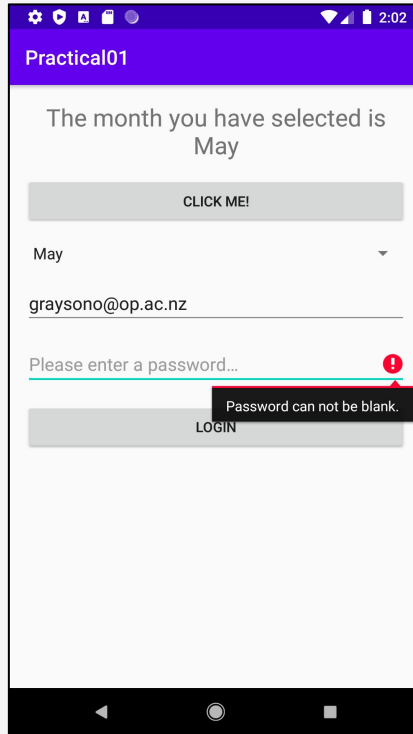
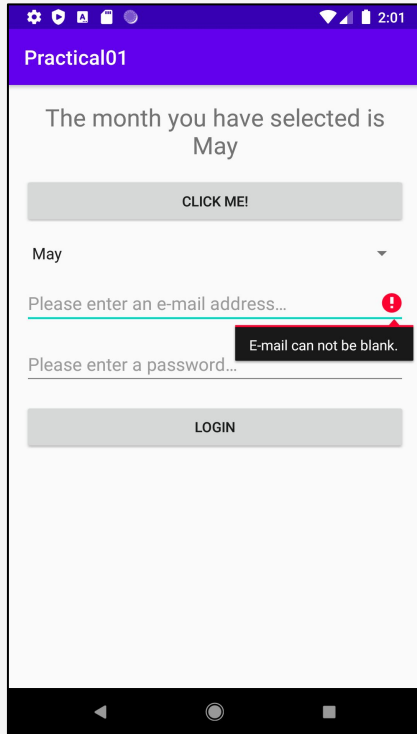
- Default Spinner item is January
- Change the Spinner item to May
- Click the **Click Me!** Button
- TextView text has changed



Practical Part 1 - Expected Output

- Click the **Login** Button
- If you have made the correct check, it should display an error on the e-mail EditText
- Enter a value into the email EditText
- Click the **Login** Button again
- Again, if you have made the correct check it should display an error on the password EditText
- Enter a value into the password EditText
- Click the **Login** Button again
- TextView text has changed

Practical Part 1 - Expected Output



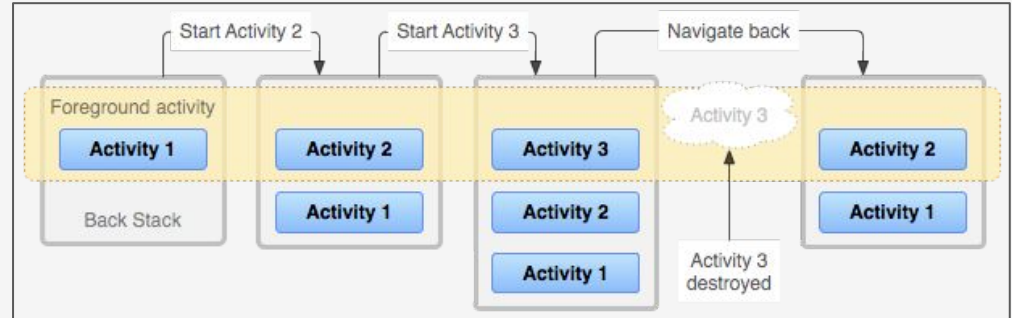
Formative Assessment

- Please write your answers to the following questions in your app:
 - What is the difference between a View & ViewGroup?
 - What does an ArrayAdapter return?

Activity Lifecycle

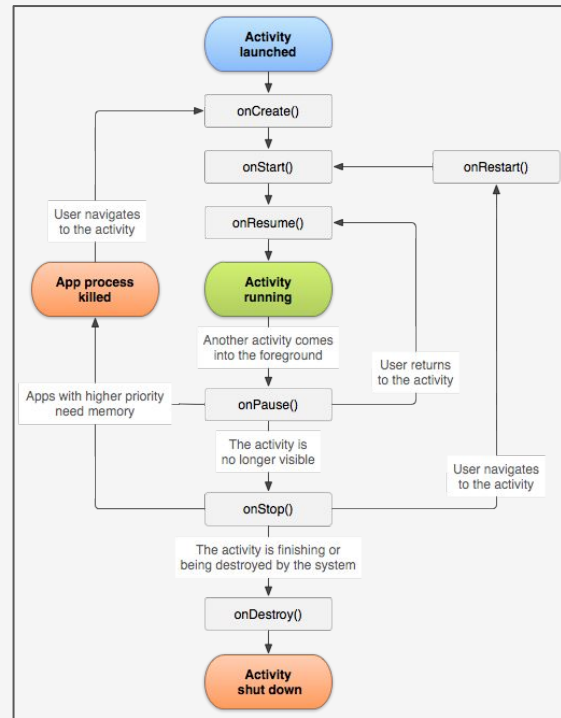
Activity Lifecycle

- What is happening here?



Activity Lifecycle

- Callback methods
 - onCreate()
 - onStart()
 - onRestart()
 - onResume()
 - onPause()
 - onStop()
 - onDestroy()
- Resource: [Activity Lifecycle](#)

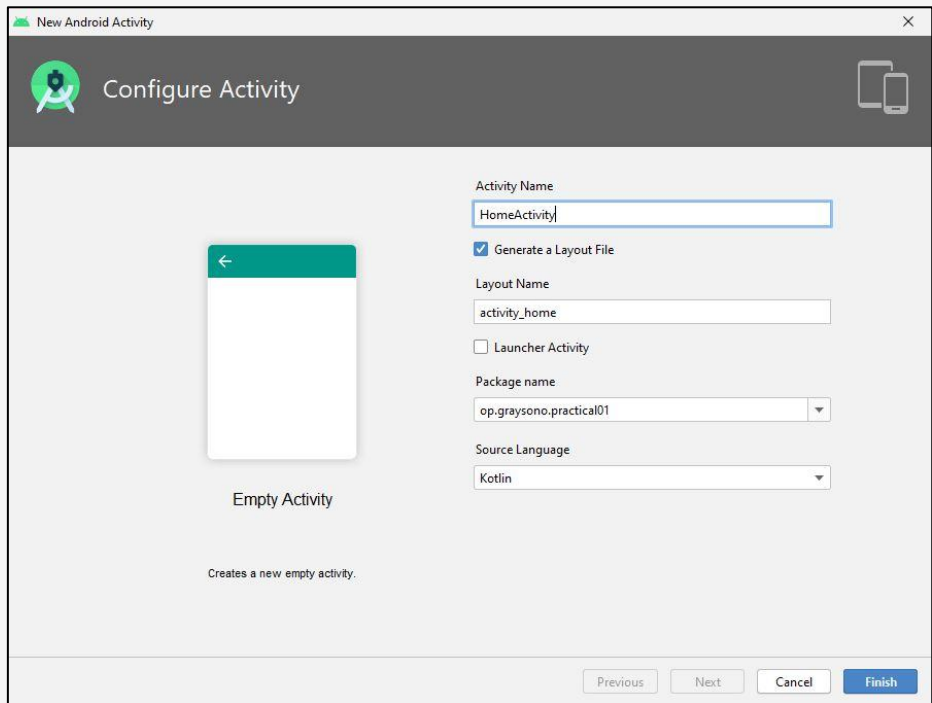


New Activity

- How do we add a new activity to our app?
 - Right click on the package name, for example, `op.graysono.practical01`
 - New > Activity > Empty Activity

New Activity

- Configure the activity
 - Activity name
 - Layout name
 - Package name
 - Source language



The screenshot shows the 'New Android Activity' dialog box in Android Studio. The title bar says 'New Android Activity'. The main header is 'Configure Activity' with a gear icon and a close button. The dialog is divided into two main sections. On the left, there is a preview of an 'Empty Activity' with a green header bar and a white body, with a back arrow icon. Below the preview, it says 'Empty Activity' and 'Creates a new empty activity.' On the right, there are configuration options: 'Activity Name' (text field with 'HomeActivity'), 'Generate a Layout File' (checked checkbox), 'Layout Name' (text field with 'activity_home'), 'Launcher Activity' (unchecked checkbox), 'Package name' (dropdown menu with 'op.graysono.practical01'), and 'Source Language' (dropdown menu with 'Kotlin'). At the bottom, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

New Android Activity

Configure Activity

Activity Name
HomeActivity

☒ Generate a Layout File

Layout Name
activity_home

☐ Launcher Activity

Package name
op.graysono.practical01

Source Language
Kotlin

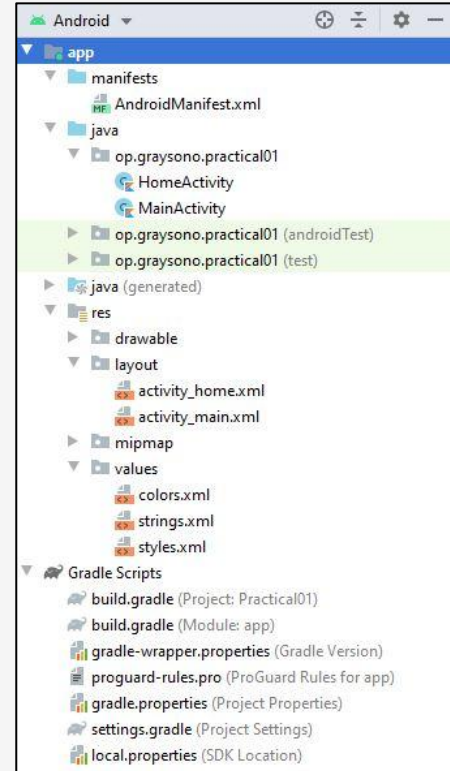
Empty Activity

Creates a new empty activity.

Previous Next Cancel Finish

New Activity

- View the file structure. The app should have a new Activity class file & layout file
- What has been added to AndroidManifest.xml ?



Intents

Intent

- Abstract description of an operation to be performed
- Late runtime binding between the code in different apps
- Used mostly in the launching of activities
- Resource: [Intent](#)

Intent Structure

- Two primary components of information in an intent:
 - Action - general action to be performed
 - Data - data to operate on
 - Resource: [Intent Structure](#)

Intent Resolution

- Two primary forms of intents:
 - Explicit - specified a component
 - Implicit - not specified a component
 - Resource: [Intent Resolution](#)

Starting Activity From Another

- startActivity() - the started activity does not need to return a result
- startActivityForResult()
- Resource: [Starting One Activity From Another](#)

```
val intent = Intent(this@MainActivity, HomeActivity::class.java)
startActivity(intent)
```

Sending Data From Another

- putExtra()
- getExtra()

```
/* MainActivity.kt.
```

```
Note: In activity_main.xml, add a new Button. Create a Button  
click listener. This code will go in there */
```

```
val intent = Intent(this@MainActivity, HomeActivity::class.java)  
intent.putExtra("month", monthsSpinner.selectedItem as String)  
startActivity(intent)
```

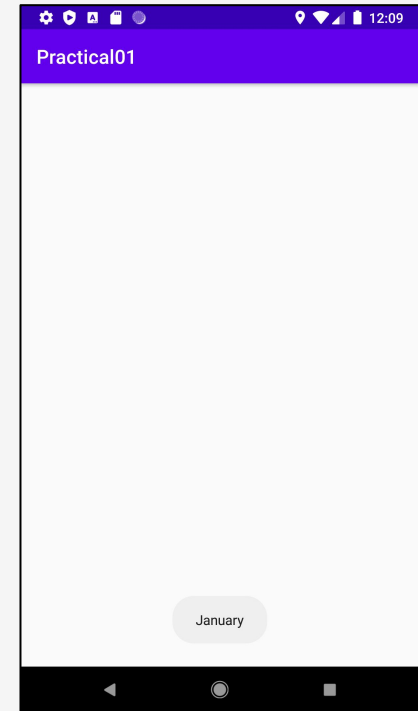
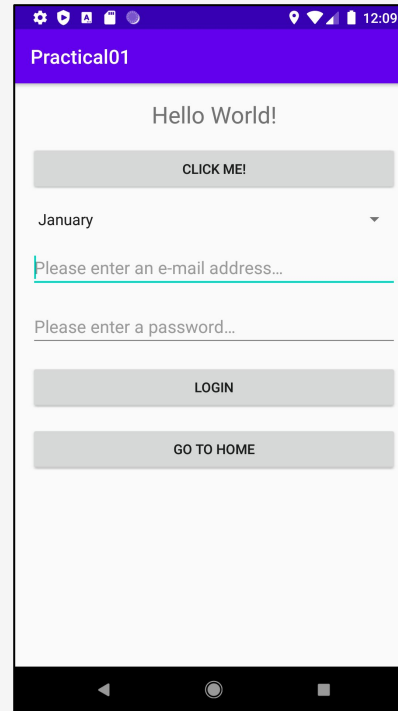
```
/* HomeActivity.kt
```

```
Note: This code will go in the onCreate() */
```

```
val bundle: Bundle? = intent.extras  
val month = bundle?.get("month")  
Toast.makeText(this@HomeActivity, month as String, Toast.LENGTH_LONG).show()
```

Emulator

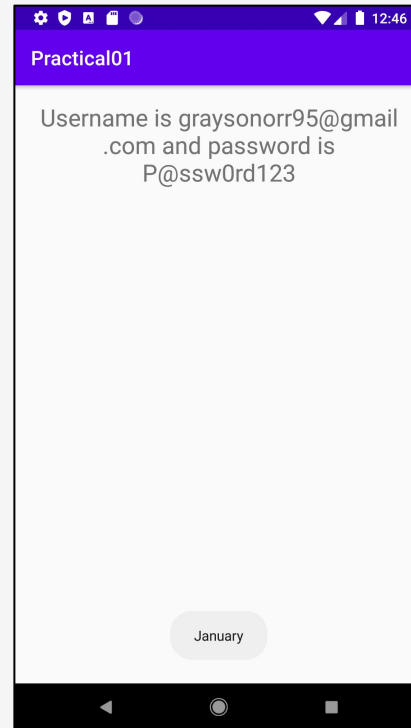
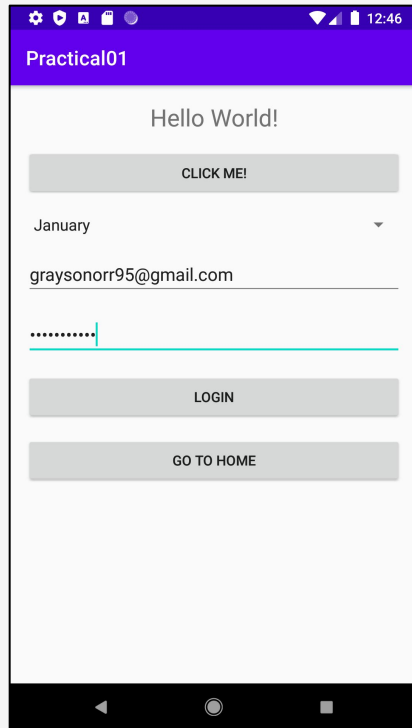
- Click the **Go to Home** Button
- Create a new intent
- Put the value into the intent
- Start activity
- Get the value from the intent
- Display the value in a Toast



Practical Part 2

- Please use the current app
- Independent tasks:
 - Implement the code as specified in the previous lecture slides
 - In activity_home.xml, add a TextView. If you wish, you can copy & paste the TextView XML code from activity_main.xml. Ensure you change the TextView id
 - In the **Go to Home** Button click listener, get the username & password values from the two EditText widgets and put them into the intent. Ensure you make the correct error checking as specified in part 1
 - Start activity
 - Get the value from the intent & display it in the TextView
 - **Research:** Implement an implicit intent in your app
 - Once you have completed the practical, create a branch named **01-submission**, push the app to the branch, make a pull request & set **Grayson-Orr** as the reviewer
 - If you do not set **Grayson-Orr** as a reviewer, I will not mark off your practical
 - **DO NOT MERGE YOUR OWN PULL REQUEST!**
 - **Due Date/Time:** 27/7/2020 @ 7.59 am

Practical Part 2 - Expected Output



Formative Assessment

- Please write your answers to the following questions in your app:
 - In the practical part 2, you implemented an intent. What type of intent is this?
 - Which activity lifecycle method is called when the activity is no longer visible to the user?