# How to CREATE JAVA web applications with Android Studio

A Comprehensive Guide to become an Android Studio Java developer

By Vangelis Kakouras - www.studiowdev.click

# Preface

A Java developer walks into a bar and orders a drink. He then sees an Android developer sitting next to him and decides to strike up a conversation.

"Hey, I'm a Java developer. What do you do?" he asks.

"I'm an Android developer. I use Java too." the Android developer replies.

"Really? That's cool. What kind of apps do you make?" the Java developer asks.

"Well, I'm working on a social media app right now. It's called NullBook. It lets you share your thoughts and feelings with other users who have nothing in common with you." the Android developer says.

"NullBook? That sounds interesting. How does it work?" the Java developer asks.

"It's simple. You just create a profile with your name, age, gender, location, hobbies, interests, etc. Then you can post anything you want on your timeline. You can also follow other users and see their posts on your feed." the Android developer explains.

"Wow, that sounds awesome. How many users do you have?" the Java developer asks.

"None." the Android developer says.

"None? How come?" the Java developer asks.

"Well, every time someone tries to sign up or log in, they get a NullPointerException." the Android developer says.

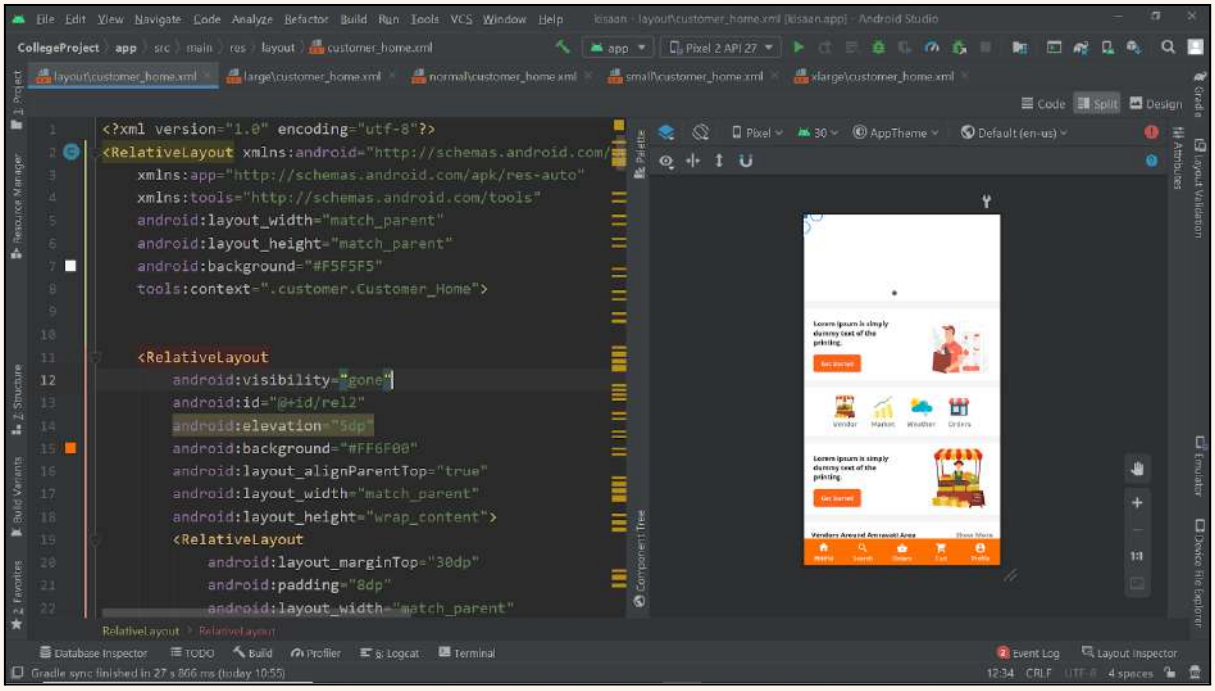
*Vangelis Kakouras*

ATHENS - March 2023

# Table of Contents

# Introduction

## What is Android Studio and why it is a powerful tool for developing Java web applications




Android Studio is the official integrated development environment (IDE) for Android application development. It is based on IntelliJ IDEA, a Java-integrated development environment for software, and incorporates its code editing and developer tools. Android Studio offers many features that enhance your productivity when building Android apps, such as:

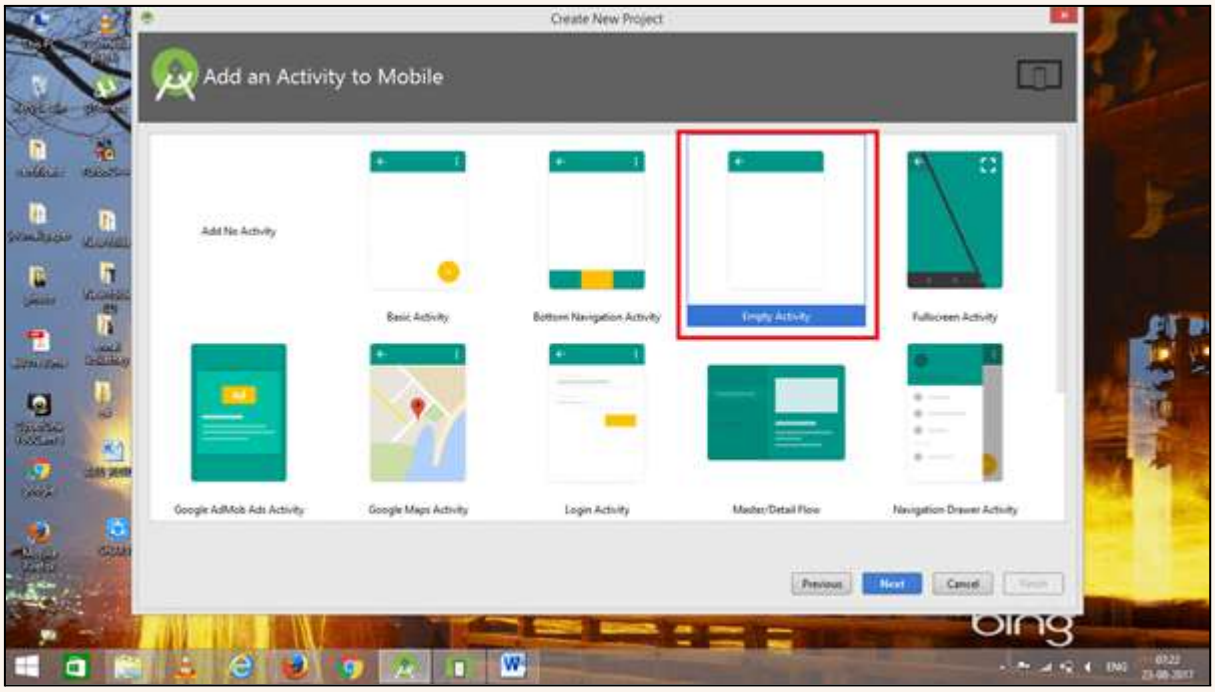
- ➔ A flexible Gradle-based build system that allows you to customise your app's dependencies, configurations, and outputs.
- ➔ An Android Emulator that lets you test your app on different devices and configurations without needing physical devices.
- ➔ Code templates and GitHub integration that helps you create standard app components and collaborate with other developers.
- ➔ A rich code editor that supports syntax highlighting, code completion, refactoring, debugging, testing, linting, and more.

- ➔ A variety of tools and plugins that help you design user interfaces, analyse performance issues, optimise your app's size and speed, add Firebase services, and more.

With Android Studio, you can develop Java web applications that run on Android devices and access web services using standard libraries such as HttpURLConnection or third-party libraries such as Retrofit or Volley. You can also use web technologies such as HTML5, CSS, JavaScript, jQuery, Bootstrap, AngularJS, etc. to create hybrid apps using frameworks such as Cordova or Ionic.

Android Studio is a powerful tool for developing Java web applications because it provides you with everything you need to create high-quality apps that run smoothly on a variety of devices. It also helps you learn the best practices of Android development and keep up with the latest trends and technologies in the industry.

## Examples of web apps that can be built with Android Studio




Android Studio allows you to build web apps that run on Android devices and access web services using standard or third-party libraries. You can also use web technologies such as HTML5, CSS, JavaScript, jQuery, Bootstrap, AngularJS, etc. to create hybrid apps that use a WebView component to display web content inside your app. *Some examples of web apps that can be built with Android Studio are:*

- A news app that fetches and displays articles from various sources using an API such as NewsAPI or RSS feeds.
- A weather app that shows the current and forecast weather conditions for different locations using an API such as OpenWeatherMap or Dark Sky.
- A social media app that allows users to post, like, comment, and share content from various platforms such as Facebook, Twitter, Instagram, etc. using their SDKs or APIs.
- A shopping app that lets users browse, search, compare, and buy products from different online stores using an API such as Amazon Product Advertising API or Google Shopping API.
- A fitness app that tracks and displays users' physical activity, calories burned, heart rate, etc. using sensors on their devices or wearable accessories such as Fitbit or Google Fit.

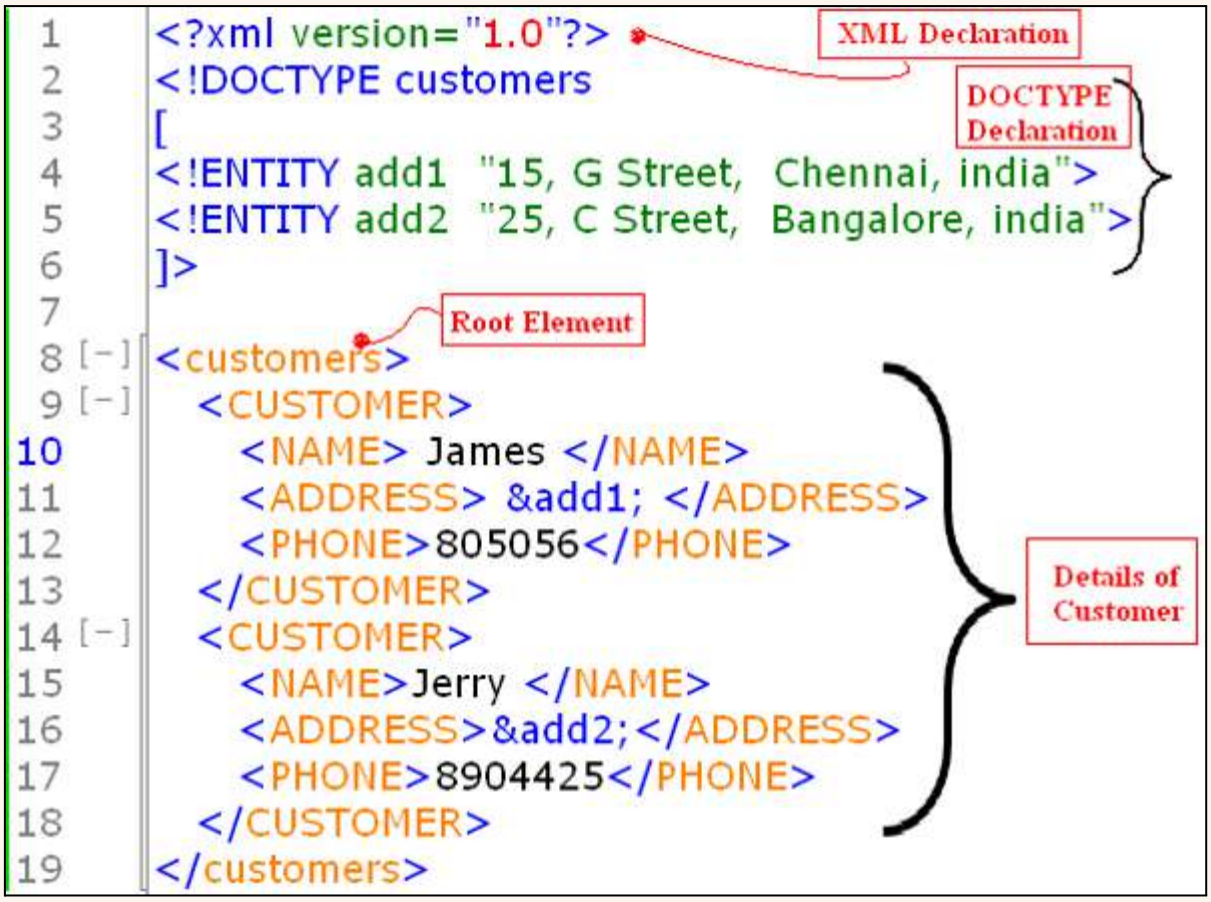
These are just some examples of web apps that can be built with Android Studio. You can also create your own custom web apps for any purpose or domain using your creativity and skills.

## The prerequisites for following this e-book

Before you start following this e-book, you need to make sure that you have installed Android Studio on your computer and that you have some basic knowledge of Java and XML. Android Studio is the official integrated development environment (IDE) for Android app development. You can download it from the official website.

- If you downloaded a .exe file (recommended), double-click to launch it.
- If you downloaded a .zip file, unpack the ZIP, copy the android-studio folder into your Program Files folder, and then open the android-studio > bin folder and launch studio64.exe (for 64-bit machines) or studio.exe (for 32-bit machines).
- Follow the setup wizard to install Android Studio and any necessary SDK tools.

Java is the primary programming language for Android app development. You need to have some basic knowledge of Java syntax, data types, variables, operators, control structures, loops, arrays, classes, objects, inheritance, polymorphism, interfaces, exceptions etc. You can learn more about Java from online **tutorials** or books.




XML is a markup language that is used to define user interfaces and resources for Android apps. You need to have some basic knowledge of XML syntax, elements, attributes, namespaces, schemas etc. You also need to know how to use XML tags such as <LinearLayout>, <TextView>, <Button>, <ImageView> etc. to create layouts and widgets for your app's screens. You can learn more about XML from online tutorials like **this one** or books.

# Creating a new project in Android Studio

## How to start a new Android Studio project

To start a new Android Studio project and create a web app, you need to follow these steps:

- ➔ Launch Android Studio and click on "Start a new Android Studio project" on the welcome screen.
- ➔ On the "Select a Project Template" screen, choose "Empty Activity" as your template. This will create a simple app with one activity and one layout file.

- ➔ On the "Configure your project" screen, enter your app name, package name, save location, language (Java or Kotlin), minimum SDK version etc. You can also change these settings later if needed.
- ➔ Click on "Finish" and wait for Android Studio to create your project.
- ➔ Once your project is created, you will see two files open in the editor: MainActivity.java (or MainActivity.kt) and activity_main.xml. These are the files where you will write your code and design your user interface respectively.

To create a web app, you need to add a WebView component to your layout file. A WebView is a view that displays web pages inside your app. You can use standard or third-party libraries to access web services from your WebView.

To add a WebView, open activity_main.xml and drag and drop a WebView from the Palette window to the Design window. You can also edit the XML code directly by switching to the Text tab at the bottom of the editor.

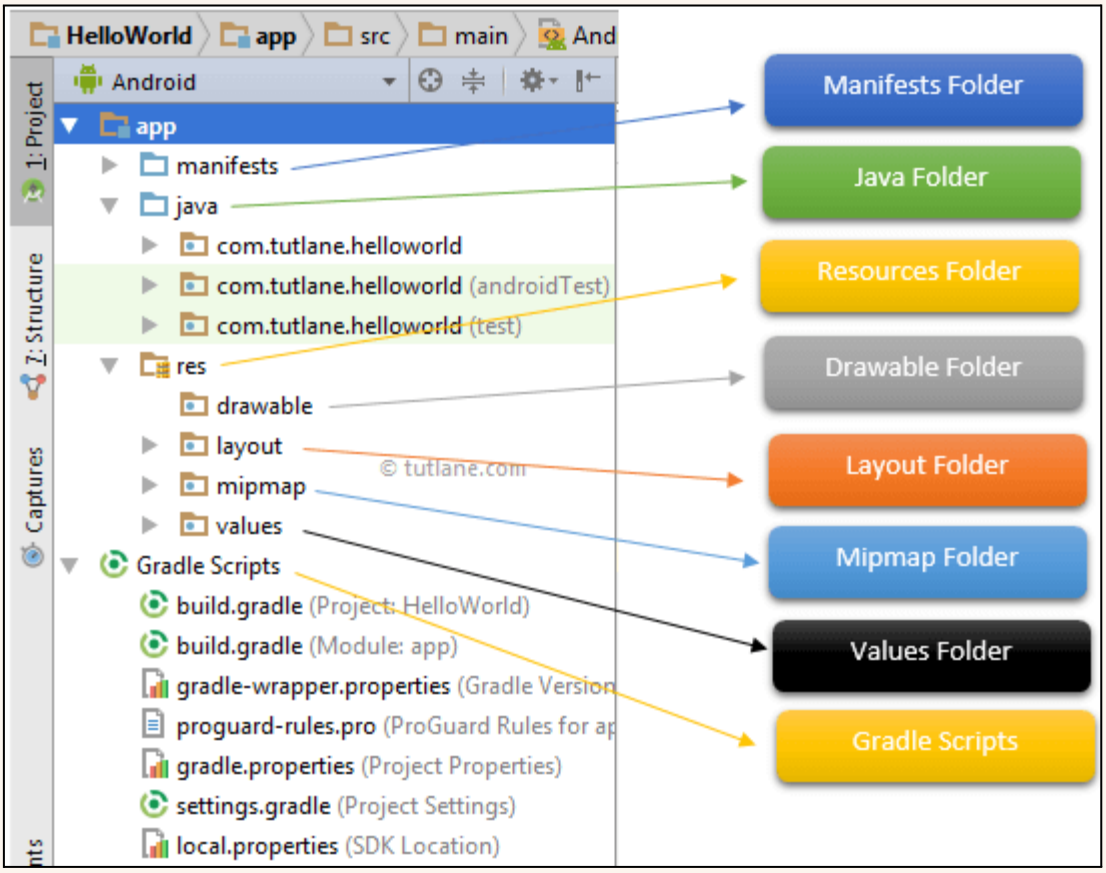
To load a web page in your WebView, you need to add some code in your MainActivity.java (or MainActivity.kt) file. First, you need to get a reference to your WebView using the findViewById() method. Then, you need to enable JavaScript for your WebView using WebSettings class. Finally, you need to load a URL using the loadUrl() method.

Here is an example of how your code might look like:

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;

public class MainActivity extends AppCompatActivity {

    private WebView webView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Get reference to WebView
        webView = findViewById(R.id.webView);

        // Enable JavaScript
        WebSettings webSettings = webView.getSettings();
        webSettings.setJavaScriptEnabled(true);
```

```
        // Load URL
        webView.loadUrl("https://www.example.com");
    }
}


// Kotlin code
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.webkit.WebSettings
import android.webkit.WebView

class MainActivity : AppCompatActivity() {

    private lateinit var webView: WebView

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

        // Get reference to WebView
        webView = findViewById(R.id.webView)

        // Enable JavaScript
        val webSettings: WebSettings = webView.settings
        webSettings.javaScriptEnabled = true

        // Load URL
        webView.loadUrl("https://www.example.com")
    }
}
```

# The structure of an Android project




An Android project is a collection of files and folders that define an app and its features. An Android project has the following main components:

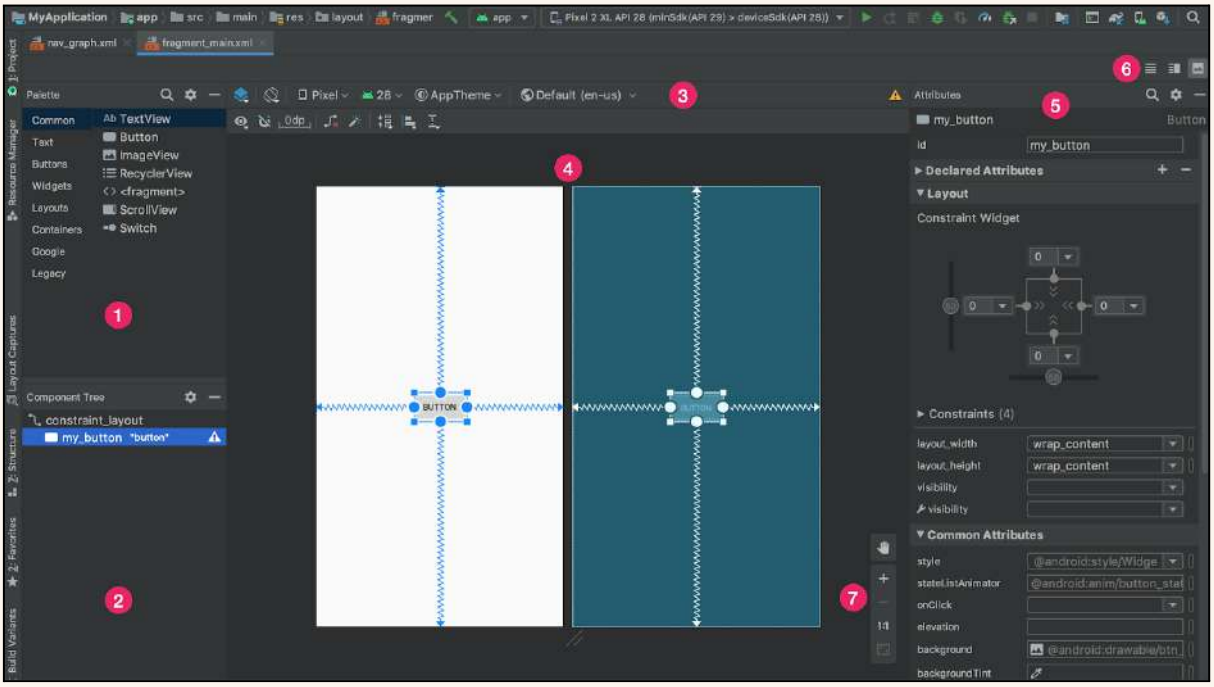
**Manifests folder**: This folder contains the AndroidManifest.xml file, which declares essential information about your app, such as its package name, permissions, activities, services, broadcast receivers, content providers etc. The manifest file also specifies how your app interacts with other apps and the system.
**Java folder**: This folder contains the source code files for your app's logic. These files are organised into packages that match your app's package name. The Java folder also contains test folders for unit tests and instrumented tests.
**res (Resources) folder**: This folder contains various types of resource files that define your app's appearance and behaviour. These include:
**Drawable folder**: This folder contains images and other drawable resources that can be used as backgrounds, icons, buttons etc.
**Layout folder:** This folder contains XML files that define the user interface layouts for your app's screens. Each layout file corresponds to an activity or a fragment.

**Mipmap folder**: This folder contains launcher icons for different screen densities. These icons are used by the system to display your app on the home screen or in the app drawer.
**Values folder:** This folder contains XML files that define simple values such as strings, colours, dimensions, styles etc. These values can be referenced by other resource files or code files.
**Gradle Scripts**: These are files that configure how your project is built, tested, and deployed using Gradle. Gradle is a powerful build automation tool that manages dependencies, generates APKs, runs tests etc.

These are the main components of an Android project. You can explore them in more detail using Android Studio's Project window. You can also add more components such as libraries, assets, native code etc. depending on your app's requirements.

# Designing a user interface with XML

## How to use the layout editor and drag-and-drop widgets




The layout editor is a tool that helps you design and preview your app's user interface (UI) by dragging and dropping UI components such as buttons, text fields, images, etc. onto a virtual device screen. You can also edit the properties of each component and see how they look on different device configurations.

To use the layout editor, you need to create a layout XML file that defines the structure and appearance of your UI. You can do this by selecting File > New > XML > Layout XML File from the main menu of Android Studio. You can also open an existing layout file from the Project window.

Once you have a layout file open, you can switch between two modes: Design and Code. The Design mode shows you a graphical representation of your UI, where you can drag and drop components from the Palette window onto the design surface. The Code mode shows you the XML code that corresponds to your UI design. You can switch between these modes by clicking on the tabs at the bottom of the editor window.

To add a component to your UI, you need to select it from the Palette window and drag it onto the design surface. You can also use keyboard shortcuts or right-click menus to add components. You can then adjust its position and size by dragging its handles or entering values in the Attributes window.

To enable drag-and-drop functionality in your app, you need to implement some methods that handle drag events such as starting, entering, exiting, dropping, etc. You also need to provide some data that represents what is being dragged and some metadata that describes it. You can start a drag-and-drop operation by calling startDragAndDrop() on any View in your current layout.

## How to use attributes, styles, themes, and resources to customize the appearance and behaviour of the widgets

**Attributes** are properties that specify the appearance and behaviour of a single View (such as a button, a text field, an image view, etc.). For example, you can use attributes to set the font colour, font size, background colour, padding, margin, etc. of a View. You can define attributes in XML by using the android: prefix followed by the attribute name and value. *For example:*

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me"
    android:textColor="#FFFFFF"
    android:background="#FF0000"/>
```

**Styles** are collections of attributes that specify the appearance for multiple Views. For example, you can create a style that defines a common look for all buttons in your app. You can define styles in XML by using the <style> tag inside a resource file (usually res/values/styles.xml). *For example:*

```
<style name="MyButtonStyle">
    <item name="android:textColor">#FFFFFF</item>
    <item name="android:background">#FF0000</item>
</style>
```

You can then apply a style to a View by using the style attribute with the name of the style. *For example:*

```
<Button
    style="@style/MyButtonStyle"
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me"/>
```

**Themes** are collections of styles that specify the appearance for an entire app or activity. For example, you can create a theme that defines a common colour scheme and typography for your app. You can define themes in XML by using the <style> tag with a parent attribute inside a resource file (usually res/values/themes.xml). *For example:*

```
<style name="MyTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorPrimary">#FF0000</item>
    <item name="colorPrimaryDark">#AA0000</item>
    <item name="colorAccent">#FFFF00</item>
</style>
```

You can then apply a theme to an app or activity by using the android:theme attribute with the name of the theme. *For example:*

```
<application
```

```
    ...
    android:theme="@style/MyTheme">

<activity
   ...
   android:theme="@style/MyTheme">
```

**Resources** are files that contain data or definitions that are not part of your app's code but are used by your app at runtime. For example, you can use resources to store strings, colours, dimensions, drawables (images), layouts (UI designs), etc. You can define resources in XML by using different tags inside different resource files (usually under res/values folder). *For example:*

```
<resources>
  <!-- Strings -->
  <string name="app_name">My App</string>
  <string name="button_text">Click me</string>

  <!-- Colors -->
  <color name="white">#FFFFFF</color>
  <color name="red">#FF0000</color>

  <!-- Dimensions -->
  <dimen name="button_width">100dp</dimen>
  <dimen name="button_height">50dp</dimen>

  <!-- Drawables -->
  <!-- See res/drawable folder -->

  <!-- Layouts -->
  <!-- See res/layout folder -->
</resources>
```

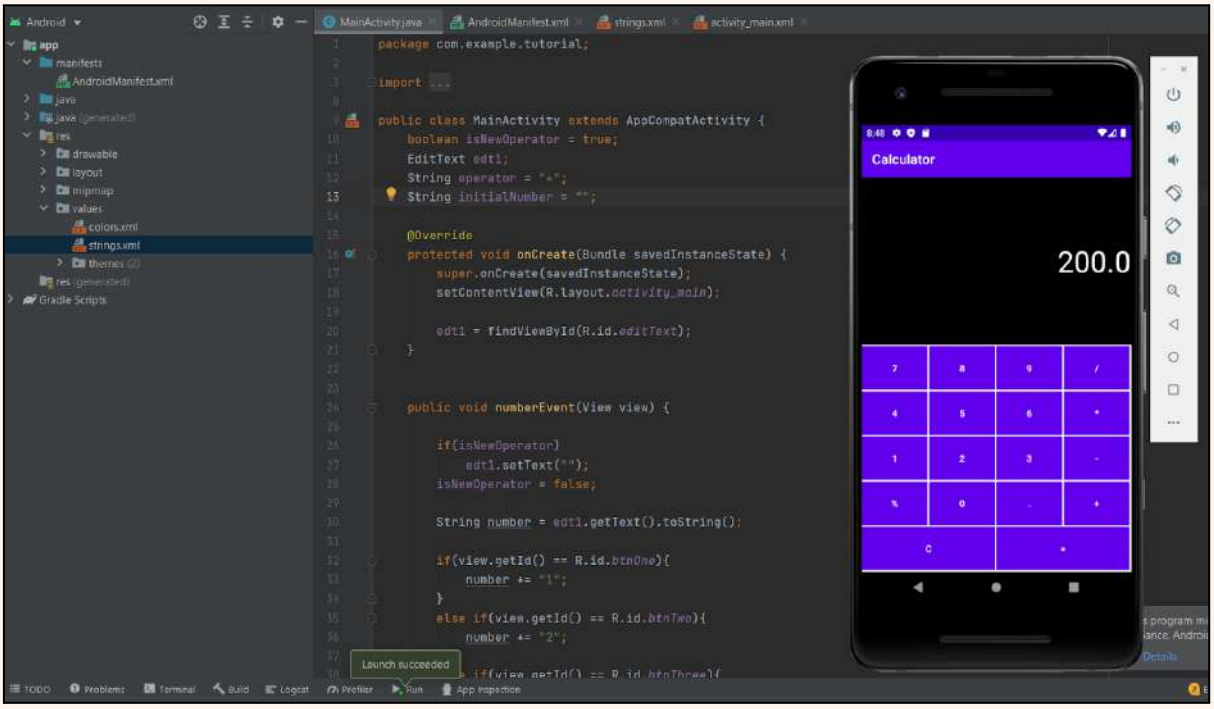
You can then reference resources in your code or XML by using @ followed by the resource type and name. *For example:*

```
String appName = getResources().getString(R.string.app_name);
int buttonWidth =
getResources().getDimensionPixelSize(R.dimen.button_width);
Drawable buttonBackground =
getResources().getDrawable(R.drawable.button_background);
```

```
<TextView
    ...
    android:text="@string/app_name"/>

<Button
    ...
    android:layout_width="@dimen/button_width"
    android:layout_height="@dimen/button_height"
    android:background="@drawable/button_background"/>
```

# Coding with Java

## How to use the code editor and write Java code




The code editor is a tool that helps you write, edit, debug, and run Java code for your Android app. You can access the code editor by opening any Java file (usually under src/main/java folder) from the Project window.

The code editor provides many features that make coding easier and faster, such as syntax highlighting, code completion, refactoring, formatting, linting, testing, etc. You can also use keyboard shortcuts or menus to perform various actions on your code.

To write Java code for your Android app, you need to follow some basic steps:

- ➔ Create an Android project in Android Studio by selecting File > New > New Project from the main menu.
- ➔ Choose a template for your app's main activity (such as Empty Activity) and provide some information about your app (such as name, package name, language, etc.).
- ➔ Write your app's logic and functionality in Java classes that extend from AppCompatActivity or other base classes. You can also create custom views or components by extending from View or other subclasses.

- ➔ Use the findViewById() method to access the UI components (such as buttons, text fields, images, etc.) that you defined in your layout XML files (usually under res/layout folder). You can also use view binding or data binding to simplify this process.
- ➔ Add event listeners or callbacks to handle user interactions (such as clicks, touches, swipes, etc.) with your UI components. You can also use intents or fragments to navigate between different screens or activities.
- ➔ Use Android APIs (such as sensors, camera, location, storage, networking, etc.) to access device features and services. You can also use third-party libraries or frameworks to add more functionality to your app.
- ➔ Test and debug your app using emulators or real devices. You can also use logcat, breakpoints, or unit tests to find and fix errors in your code.

## How to use variables, data types, operators, control structures, methods, classes, objects, inheritance, interfaces, exceptions, collections, generics, and annotations in Java

**Variables** are containers that store values of different types. You can declare a variable by specifying its name and data type. *For example:*

```
int x; // declare an integer variable named x
x = 10; // assign 10 to x
String name; // declare a string variable named name
name = "Alice"; // assign "Alice" to name
```

**Data types** are categories that define the size and type of variable values. There are two kinds of data types in Java: primitive and reference.

Primitive data types are predefined by the language and have no additional methods. There are eight primitive data types in Java:

byte, short, int, long, float, double, char, and boolean. *For example:*

```
byte b = 127; // a byte can store values from -128 to 127
short s = 32767; // a short can store values from -32768 to 32767
int i = 2147483647; // an int can store values from -2147483648 to
2147483647
long l = 92233720368547758O7L; // a long can store values from
```

```
-9223372036854775808L to 9223372036854775807L
float f = 3.14f; // a float can store decimal numbers with up to
6-7 digits of precision
double d = 3.141592653589793d; // a double can store decimal
numbers with up to 15-16 digits of precision
char c = 'A'; // a char can store a single character using Unicode
encoding
boolean bool = true; // a boolean can store either true or false
values
```

**Reference data types** are defined by classes and have additional methods. A reference data type stores the address or reference of an object (an instance of a class) rather than the actual value. For example:

```
String str = "Hello"; // a string is an object that represents a
sequence of characters
Integer num = new Integer(10); // an integer is an object that
wraps an int value as an object
Scanner sc = new Scanner(System.in); // a scanner is an object
that allows reading input from various sources (such as keyboard)
```

**Operators** are symbols that perform operations on one or more operands (variables or values). There are different kinds of operators in Java: arithmetic, assignment, relational, logical, bitwise, unary, ternary, and others. *For example:*

```
// arithmetic operators perform basic mathematical operations such
as addition (+), subtraction (-), multiplication (*), division
(/), modulus (%), and exponentiation (**)
int sum = x + y; // add x and y and assign the result to sum
int diff = x - y; // subtract y from x and assign the result to
diff
int prod = x * y; // multiply x and y and assign the result to
prod
int quot = x / y; // divide x by y and assign the result to quot
int rem = x % y; // get the remainder of dividing x by y and
assign it to rem
double pow = Math.pow(x,y); // raise x to the power of y and
assign it to pow

// assignment operators assign values to variables using symbols
such as (=), (+=), (-=), (*=), (/=), (%=), etc.
```

```
x += 5; // equivalent to x = x + 5;
y -= 2; // equivalent to y = y - 2;
z *= 3; // equivalent to z = z * 3;
w /= 4; / equivalent t w w / ;
v %= ; / equivalent t v v % ;

// relational operators compare two operands and return a boolean
value based on their relationship such as equal (==),
not equal (!=),
greater than (>),
less than (<),
greater than or equal (>=),
less than or equal (<=)
bool b1 b == ; / b true if b equals ;
bool b2 b != ; / b true if b does not equal ;
bool b3 b > ; / b true if b greater than ;
bool b4 < ; /
```

**Control structures** are statements that control the flow of execution in a program. There are different kinds of control structures in Java: conditional (if/else), iterative for/while/do-while), and branching (break/continue/return). *For example:*

```
// conditional statements execute a block of code based on a
condition
if (x > 10) { // if x is greater than 10
  System.out.println("x is large"); // print "x is large"
} else { // otherwise
  System.out.println("x is small"); // print "x is small"
}

// iterative statements execute a block of code repeatedly until a
condition is met
for (int i = 0; i < 10; i++) { // for i from 0 to 9
  System.out.println(i); // print i
}

while (x < 100) { // while x is less than 100
  x = x * 2; // double x
}

do {
  y = y + 1; // increment y
} while (y < 50); // do this while y is less than 50
```

```
// branching statements alter the normal flow of execution by
jumping to another point in the program
break; // exit the current loop or switch statement
continue; // skip the rest of the current iteration and start a
new one
return z; // return z as the value of the current method and exit
it
```

**Methods** are blocks of code that perform a specific task and can be reused. You can define a method by specifying its name, parameters, return type, and body. You can also call a method by using its name and passing arguments. *For example:*

```
// define a method that calculates the area of a circle given its
radius as a parameter and returns it as a double value

public static double areaOfCircle(double radius) {
   double area = Math.PI * radius * radius; // calculate area using
Math.PI constant and radius parameter
   return area; / return area as result
}

// call the method by using its name and passing an argument

double circleArea = areaOfCircle(5); / call areaOfCircle method with
argument ; assign returned value t circleArea variable
System.out.println(circleArea); / print circleArea variable
```

**Classes** are user-defined blueprints or prototypes from which objects are created. They represent the set of properties or attributes (fields) and behaviours or actions (methods) that are common to all objects of one type. You can define a class by using the class keyword followed by its name and body. You can also create an object by using the new keyword followed by the class name and constructor parameters. *For example:*

```
// define a class that represents a person with fields for name,
age, and gender, and methods for getting and setting these fields

public class Person {
   private String name; / declare private field for name
   private int age; / declare private field for age
```

```
   private char gender; / declare private field for gender

   public Person(String name, int age, char gender) { / define
constructor with parameters for name age gender
      this.name = name; / assign parameter value t field using
this keyword
      this.age = age;
      this.gender = gender;
   }

   public String getName() { / define getter method for name field
      return this.name;
   }

   public void setName(String newName) { / define setter method
for name field with parameter for new value
      this.name = newName;
   }

   public int getAge() { / define getter method for age field
      return this.age;
   }
```

**Objects** are instances of classes that have state (fields) and behaviour (methods). You can create an object by using the new keyword followed by the class name and constructor parameters. You can also access and modify the fields and methods of an object by using the dot operator (.). *For example:*

```
// create an object of Person class with name "Bob", age 25, and
gender 'M'
Person bob = new Person("Bob", 25, 'M');

// access and print the name field of bob object
System.out.println(bob.getName());

// modify the name field of bob object to "Robert"
bob.setName("Robert");

// access and print the name field of bob object again
System.out.println(bob.getName());
```

**Inheritance** is a mechanism that allows one class to acquire or inherit the properties and behaviours of another class. The class that inherits is called a subclass or a child class. The class that is inherited from is called a superclass or a parent class. You can use inheritance to reuse existing code, avoid duplication, and establish relationships between classes. You can use the extends keyword to indicate that a subclass inherits from a superclass. *For example:*

```
// define a subclass named Student that inherits from Person
superclass

public class Student extends Person {
   private int id; // declare private field for id
   private double gpa; // declare private field for gpa

   public Student(String name, int age, char gender, int id,
double gpa) { // define constructor with parameters for name age
gender id gpa
      super(name, age, gender); // call superclass constructor
with parameters for name age gender using super keyword
      this.id = id; // assign parameter value to field using this
keyword
      this.gpa = gpa;
   }

   public int getId() { // define getter method for id field
      return this.id;
   }

   public void setId(int newId) { // define setter method for id
field with parameter for new value
      this.id = newId;
   }

   public double getGpa() { // define getter method for gpa field
      return this.gpa;
   }

   public void setGpa(double newGpa) { // define setter method for
gpa field with parameter for new value
      this.gpa = newGpa;
   }
}
```

```
// create an object of Student subclass with name "Alice", age 20,
gender 'F', id 1234, and gpa 3.5

Student alice = new Student("Alice", 20, 'F', 1234, 3.5);

// access and print the fields inherited from Person superclass

System.out.println(alice.getName());
System.out.println(alice.getAge());
System.out.println(alice.getGender());

// access and print the fields defined in Student subclass

System.out.println(alice.getId());
System.out.println(alice.getGpa());
```

**Interfaces** are abstract types that specify a set of abstract methods that a class must implement if it wants to conform to that interface. Interfaces are used to define contracts or behaviours that classes must follow without specifying how they do so. You can use interfaces to achieve abstraction, polymorphism, and multiple inheritances in Java. You can use the interface keyword to define an interface followed by its name and body. You can also use the implements keyword to indicate that a class implements one or more interfaces. *For example:*

```
// define an interface named Animal that specifies two abstract
methods: makeSound() and move()

public interface Animal {
   public void makeSound(); / declare abstract method makeSound
without body
```

**How to use interfaces in Java**

- ➔ An interface is a blueprint of behaviour that defines what methods a class must implement.
- ➔ To use an interface, you write a class that implements the interface and provides a method body for each of the methods declared in the interface.
- ➔ You can use interfaces to achieve security, abstraction, and multiple inheritances in Java.

- ➔ You can also use constants, variables, static methods, default methods, and empty interfaces in an interface.

**How to create and implement interfaces in Java**

- ➔ To declare an interface, use the interface keyword followed by the interface name. You can optionally specify other interfaces that this interface extends using the extends keyword.
- ➔ To create an interface, you can declare constant fields, abstract methods, static methods, and default methods inside curly braces.
- ➔ To implement an interface, use the implements keyword followed by the interface name. You must provide a method body for each of the abstract methods declared in the interface.

*Here is a code block that illustrates these steps:*

```
// Declare an interface named Animal
public interface Animal {
  // Declare a constant field
  public static final int LEGS = 4;
  // Declare an abstract method
  public void makeSound();
  // Declare a static method
  public static void sleep() {
    System.out.println("Zzz...");
  }
  // Declare a default method
  public default void eat() {
    System.out.println("Yum!");
  }
}

// Implement an interface named Animal
public class Dog implements Animal {
  // Provide a method body for makeSound()
  public void makeSound() {
    System.out.println("Woof!");
  }
}

// Create an object of the Dog class and call its methods
public class Main {
```

```
  public static void main(String[] args) {
    Dog d = new Dog();
    d.makeSound(); // Woof!
    d.eat(); // Yum!
    Animal.sleep(); // Zzz...
    System.out.println(Animal.LEGS); // 4
  }
}
```

**How to use exceptions in Java**

- ➔ An exception is an unwanted or unexpected event that occurs during the execution of a program and disrupts its normal flow.
- ➔ To use exceptions in Java, you need to use try...catch blocks that allow you to handle different types of exceptions.
- ➔ You can also create your own custom exceptions by extending Exception or RuntimeException classes.

*Here is a code block that illustrates these steps:*

```
// Import java.util package
import java.util.*;

// Create an ArrayList of String
List<String> names = new ArrayList<>();
// Add some elements to it
names.add("Alice");
names.add("Bob");
names.add("Charlie");
names.add("David");
// Print the list
System.out.println(names); // [Alice, Bob, Charlie, David]

// Create a HashSet of Integer
Set<Integer> numbers = new HashSet<>();
// Add some elements to it
numbers.add(10);
numbers.add(20);
numbers.add(30);
numbers.add(40);
// Print the set
```

```
System.out.println(numbers); // [40, 10 , 30 , 20]

// Create a HashMap of String and Integer
Map<String,Integer> scores = new HashMap<>();
// Put some key-value pairs to it
scores.put("Alice",90);
scores.put("Bob",80);
scores.put("Charlie",70);
scores.put("David",60);
// Print the map
System.out.println(scores); // {Alice=90 , Bob=80 , Charlie=70 ,
David=60}
```

**How to use generics in Java**

- ➔ Generics are a feature of Java that allows you to write code that can work with different types of data without repeating yourself.
- ➔ Generics use type parameters (such as <T>) to represent generic types that can be replaced by actual types when using generics.
- ➔ Generics can be used with classes, interfaces, methods, and constructors.
- ➔ Generics provide type safety, code reusability, and performance benefits.

*Here is a code block that illustrates how to use generics with a class:*

```
// Define a generic class
class Box<T> {
  // Declare a generic instance variable
  private T item;
  // Define a constructor with a generic parameter
  public Box(T item) {
    this.item = item;
  }
  // Define a getter method for the generic variable
  public T getItem() {
    return item;
  }
}

// Create an object of Box class with Integer type
Box<Integer> intBox = new Box<>(10);
// Get the item from intBox
```

```
int num = intBox.getItem();
// Print the item
System.out.println(num); // 10

// Create an object of Box class with String type
Box<String> strBox = new Box<>("Hello");
// Get the item from strBox
String str = strBox.getItem();
// Print the item
System.out.println(str); // Hello
```

**How to use annotations in Java**

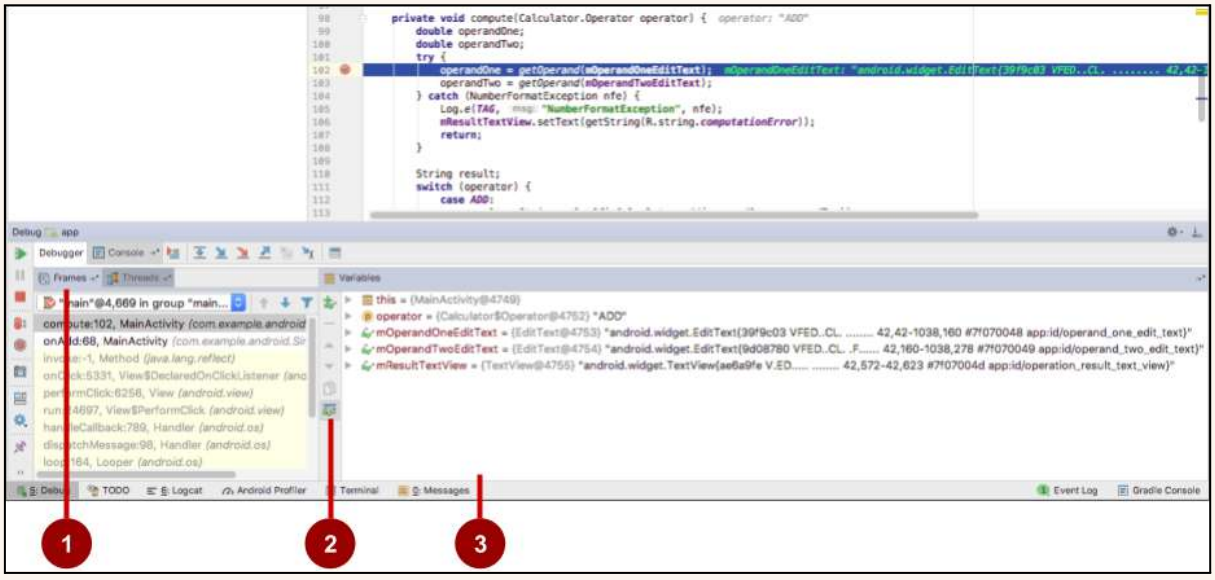
Annotations are a form of metadata that provide data about a program that is not part of the program itself. They have no direct effect on the operation of the code they annotate. They can be used for various purposes, such as information for the compiler, documentation generation, code analysis, testing frameworks, etc.

To use annotations in Java, you need to place them above a declaration (such as class, method, field, etc.) or before a type (such as return type of a method). You also need to import or define the annotation type. *For example:*

```
import java.lang.annotation.Documented;

@Documented // this is an annotation
public @interface MyAnnotation { // this is an annotation type
  String value(); // this is an annotation element
}

@MyAnnotation("Hello") // this is an annotation with value
public class MyClass {
  @MyAnnotation("World") // this is another annotation with value
  public void myMethod() {
    // some code here
  }
}
```

# Testing and debugging with Android Studio

## How to use the emulator and logcat to test and debug a web app

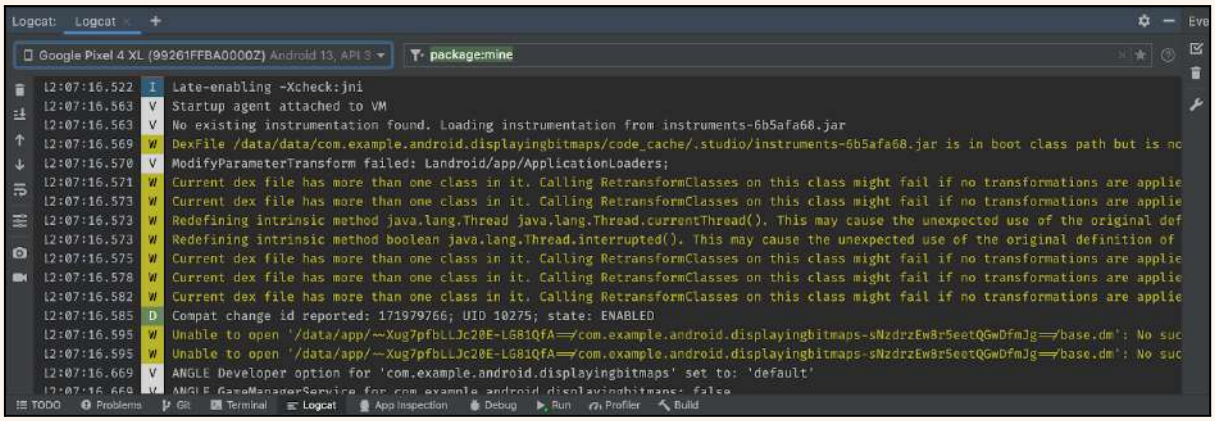



To test and debug a web app on different devices and configurations in Android Studio, you can use the emulator and logcat tools. The emulator is a virtual device that simulates a real Android device on your computer. You can create different emulators with different settings, such as screen size, API level, hardware features, etc. Logcat is a tool that displays logs from your device or emulator in real-time. You can use logcat to view messages that you added to your app with the Log class, messages from services that run on Android, or system messages.

To use logcat with an emulator, you need to run logcat as an adb command or directly in a shell prompt on your emulator. You can also use the Logcat window in Android Studio to view and filter logs. To use logcat with a web app, you need to enable WebView debugging in your app's manifest file.

To create an emulator in Android Studio, you need to follow these steps:

- ➔ Open Android Studio and click on Tools > AVD Manager
- ➔ Click on Create Virtual Device button
- ➔ Choose a device definition and click Next
- ➔ Choose a system image and click Next
- ➔ Review the configuration settings and click Finish




To enable WebView debugging in your web app, you need to add this line to your app's manifest file:

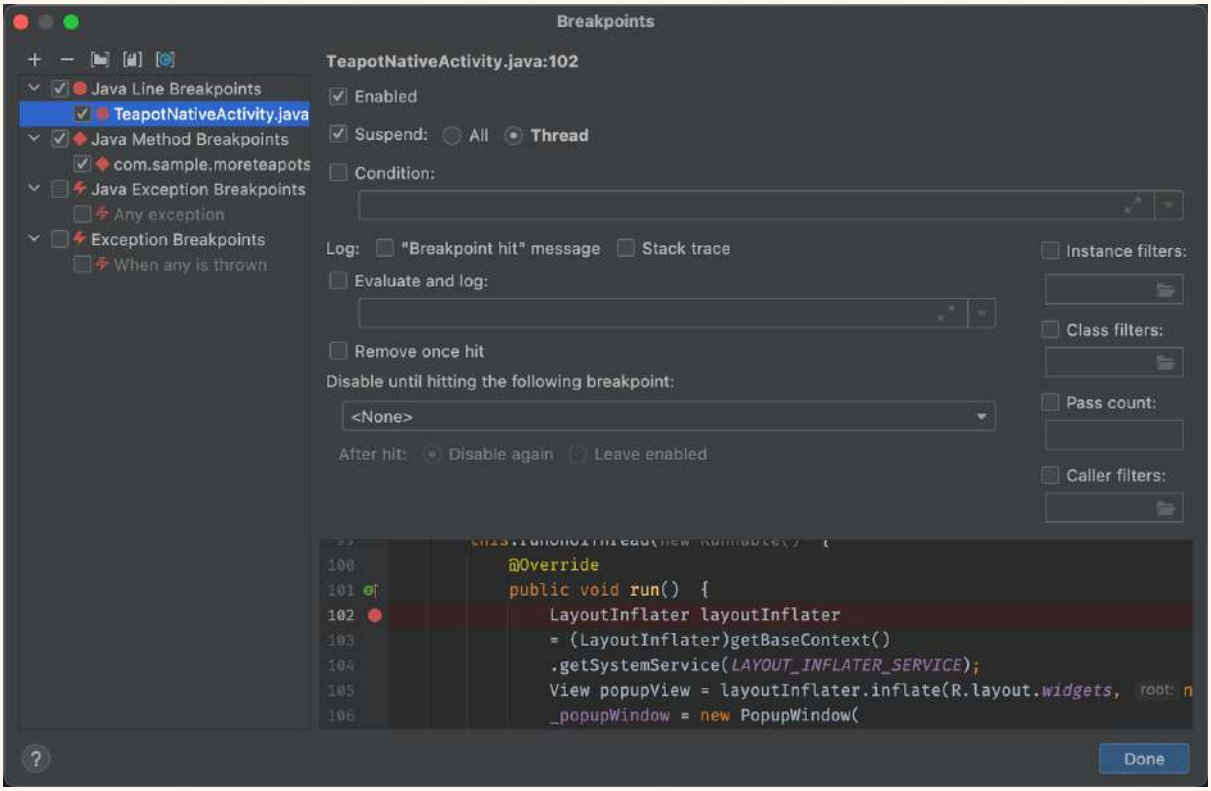
```
<application android:debuggable="true">
```

And this line to your activity's onCreate method:

```
WebView.setWebContentsDebuggingEnabled(true);
```

This will allow you to use Chrome DevTools to inspect and debug your web app on an emulator or a device.

## How to use breakpoints



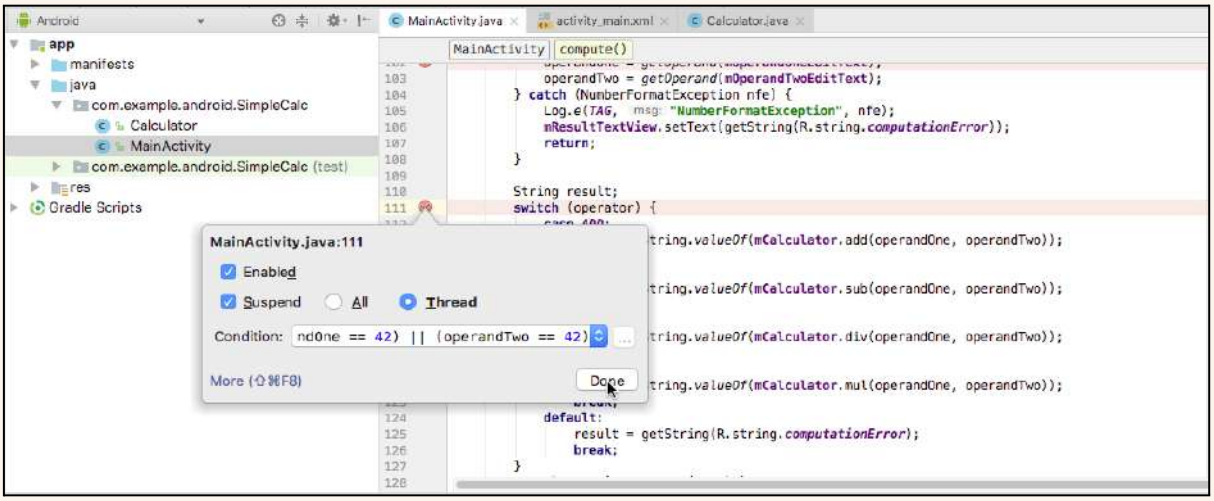

Breakpoints are a debugging tool that allows you to pause the execution of your app at a specific line of code. You can use breakpoints to inspect the state of your app, evaluate expressions, modify variables, and resume execution with different conditions.

```
98      private void compute(Calculator.Operator operator) {
99          double operandOne;
100         double operandTwo;
101         try {
102             operandOne = getOperand(mOperandOneEditText);
103             operandTwo = getOperand(mOperandTwoEditText);
104         } catch (NumberFormatException nfe) {
105             Log.e(TAG,  msg: "NumberFormatException", nfe);
106             mResultTextView.setText(getString(R.string.computationError));
107             return;
108         }
109
110         String result;
111         switch (operator) {
```

To use breakpoints in Android Studio, you need to follow these steps:

➔ Open your project and locate the code where you want to set a breakpoint

- ➔ Click on the narrow column to the left of the code line number. A red dot will appear indicating that a breakpoint is set
- ➔ Run your app in debug mode by clicking on Debug icon
- ➔ When your app reaches the breakpoint, it will pause and show you the Debug window with different panes and options
- ➔ You can use the buttons on top of Debug window to resume, step over, step into, step out, or stop debugging
- ➔ You can also right-click on a breakpoint to edit its properties, such as condition, log message, suspend policy etc.




*Here is an example of using breakpoints in Android Studio:*

Let's say you have a simple app that displays a message when you click a button. The code for MainActivity.kt looks like this:

```
package com.example.debugging

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.TextView

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

```
        val button = findViewById<Button>(R.id.button)
        val textView = findViewById<TextView>(R.id.textView)

        button.setOnClickListener {
            textView.text = "Hello World!"
        }
    }
}
```

Now let's introduce a bug by changing the text to "Hello Bug!" instead of "Hello World!". To debug this bug, we can set a breakpoint at line 17 where we assign the text to textView. To do that, we click on the column to the left of line 17 and see a red dot appear.

Next, we run our app in debug mode by clicking on Debug icon. When our app reaches line 17, it will pause and show us the Debug window with different panes and options.

We can see that textView.text is "Hello Bug!" which is not what we want. We can also see that button.text is "Click Me" which is correct. We can use the Variables pane to inspect and modify these values.

We can also use the Evaluate Expression option to execute any expression in the context of our app. For example, we can type textView.text = "Hello World!" and click Evaluate to fix our bug temporarily.

To resume our app execution, we can click on Resume icon. We will see that our app now displays "Hello World!" when we click the button.

This is how we can use breakpoints to debug our app in Android Studio.

## How to watch an expression

A watch expression is an expression that you want to evaluate while debugging your app. You can use watch expressions to monitor the values of variables or expressions that are not visible in the Variables pane.

To watch an expression in Android Studio, you need to follow these steps:

- ➔ Run your app in debug mode and pause at a breakpoint
- ➔ In the Debug window, select the Watches tab
- ➔ Click on + icon to add a new watch expression
- ➔ Enter the expression you want to watch in the text field and press Enter

➔ You will see the value of your expression displayed below it. You can also edit or remove your watch expression by right-clicking on it

To watch expressions in Android Studio debugging, you need to use breakpoints and watches. Breakpoints are markers that tell the debugger where to pause your app's execution so you can inspect its state. Watches are expressions that you can evaluate while debugging and see their values change as you step through your code.

To add a breakpoint, click on the left gutter of the code editor next to the line where you want to pause your app. A red dot will appear indicating a breakpoint. To add a watch, right-click on a variable in the Variables window and select "Add to watches". You can also type an expression in the Watches window and press Enter.

*Here is an example of how to watch expressions in Android Studio debugging:*

```
fun main() {
    val numbers = listOf(1, 2, 3)
    var sum = 0
    for (n in numbers) {
        sum += n // Add a breakpoint here
    }
    println(sum)
}
```

In this code snippet, we have added a breakpoint at line 5 where we update the sum variable. When we run the app in debug mode, it will pause at this line and show us the values of n and sum in the Variables window. We can also add watches for expressions like `sum / n` or `numbers.size` and see how they change as we step over each iteration of the loop.

## How to step over/into/out commands

To step over/into/out commands in Android Studio debugging, you need to use the buttons on the Debug toolbar or the keyboard shortcuts. These commands allow you to control how your app's execution flows through your code and jump into or out of methods.

➔ Step Over executes the next line of code in the current class and method, executing all of the method calls on that line and remaining in the same file. To use this command, click the Step Over icon, select Run > Step Over, or type F8.

- ➔ Step Into jumps into the execution of a method call on the current line (versus just executing that method and remaining on the same line). To use this command, click the Step Into icon, select Run > Step Into, or type F7.
- ➔ Step Out resumes execution until it reaches a return statement in a method call that was previously stepped into. To use this command, click the Step Out icon, select Run > Step Out, or type Shift+F8.

*Here is an example of how to use these commands in Android Studio debugging:*

```
fun main() {
    val numbers = listOf(1, 2, 3)
    var sum = 0
    for (n in numbers) {
        sum += n // Add a breakpoint here
    }
    println(sum)
}

fun add(a: Int, b: Int): Int {
    return a + b // Add another breakpoint here
}
```

In this code snippet, we have added two breakpoints: one at line 5 where we update the sum variable using a loop, and another at line 11 where we define a simple add function. When we run the app in debug mode, it will pause at line 5 and show us the values of n and sum in the Variables window.

If we want to see how our add function works internally, we can replace line 5 with `sum = add(sum,n)` and then use **Step Into** to jump into that function. We will see that it pauses at line 11 and shows us the values of a and b.

If we want to skip over our add function and just see its result without jumping into it, we can use **Step Over** instead. We will see that it executes line 5 without pausing at line 11 and updates the sum accordingly.

If we want to exit our add function after stepping into it and resume execution until it reaches a return statement (line 12), we can use **Step Out**. We will see that it returns to line 5 with the sum updated.

## How to evaluate expressions

To evaluate expressions in Android Studio debugging, you need to use the Evaluate Expression window. This window allows you to validate different expressions and the current state of variables at any breakpoint.

To open the Evaluate Expression window, right-click on a variable or expression in your code and select Evaluate Expression. Alternatively, you can press Alt+F8 on Windows or Option+F8 on Mac. You can also access this window from the Debug toolbar by clicking on [evaluate expression icon](https://developer.android.com/images/tools/debugging/evaluate-expression.png).

In the Evaluate Expression window, you can type any expression that is valid in your current context and press Enter to see its value. You can also modify the value of variables by assigning new values to them. For example, if you have a variable called `name` that holds a string value `"Alice"`, you can type `name = "Bob"` and press Enter to change its value.

*Here is an example of how to evaluate expressions in Android Studio debugging:*

```
fun main() {
    val numbers = listOf(1, 2, 3)
    var sum = 0
    for (n in numbers) {
        sum += n // Add a breakpoint here
    }
    println(sum)
}
```

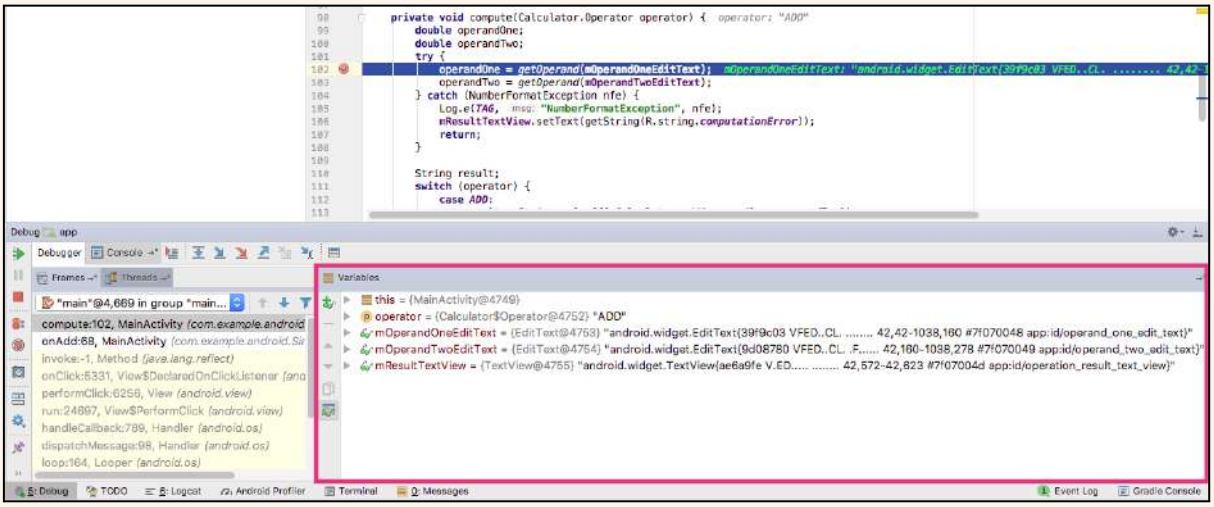
In this code snippet, we have added a breakpoint at line 5 where we update the sum variable using a loop. When we run the app in debug mode, it will pause at this line and show us the values of n and sum in the Variables window.

If we want to see what would be the value of the sum if we added another number to it without changing its actual value, we can open the Evaluate Expression window and type `sum + 4` and press Enter. We will see that it returns `10` as expected.

If we want to change the value of the sum permanently while debugging, we can type `sum = 100` and press Enter. We will see that it updates the sum both in the Evaluate Expression window and in the Variables window.

## How to inspect variables/values/memory/threads/call stack

To inspect variables/values/memory/threads/call stack in Android Studio debugging, you need to use the Debug window. This window provides various tabs and tools that help you monitor and manipulate your app's execution while debugging.




To open the Debug window, click on [debug icon](https://developer.android.com/images/tools/debugging/debug.png) on the toolbar or press Shift+F9. Alternatively, you can select Run > Debug from the menu bar.

In the Debug window, you can use the following tabs and tools to inspect different aspects of your app:

- **Variables**: This tab shows you the values of variables in your current scope. You can expand or collapse variables to see their fields or elements. You can also modify their values by double-clicking on them and typing a new value.
- **Memory**: This tab shows you how much memory your app is using and how it changes over time. You can also perform garbage collection, capture heap dumps, and analyze memory leaks using this tab.
- **Threads**: This tab shows you all the threads that are running in your app. You can switch between threads by clicking on them. You can also suspend or resume threads using the buttons on this tab.

- **Call Stack**: This tool shows you the sequence of method calls that led to your current breakpoint. You can navigate through the call stack by clicking on each method name. You can also jump to a specific line of code by double-clicking on it.

*Here is an example of how to inspect variables/values/memory/threads/call stack in Android Studio debugging:*

```
fun main() {
    val numbers = listOf(1, 2, 3)
    var sum = 0
    for (n in numbers) {
        sum += n // Add a breakpoint here
    }
    println(sum)
}

fun add(a: Int, b: Int): Int {
    return a + b // Add another breakpoint here
}
```

In this code snippet, we have added two breakpoints: one at line 5 where we update the sum variable using a loop, and another at line 11 where we define a simple add function. When we run the app in debug mode, it will pause at line 5 and show us the following information in the Debug window:

- **Variables**: We can see that n has a value of 1 and sum has a value of 0 at this point. We can also see that numbers are a list with three elements.
- **Memory**: We can see that our app is using about 10 MB of memory at this point. We can also see a graph that shows how memory usage changes over time.
- **Threads**: We can see that our app has one thread called main which is currently suspended at line 5.
- **Call Stack**: We can see that our app has called main() which has called for (n in numbers) which has reached line 5.

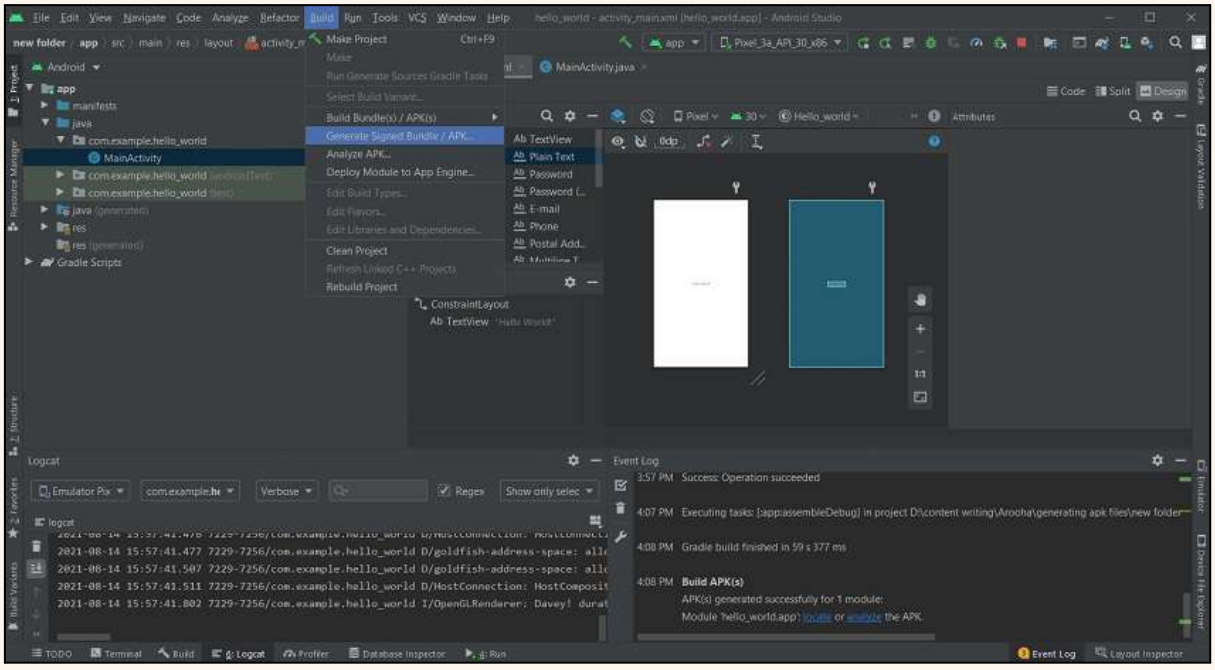
If we want to see how our add function works internally, we can replace line 5 with `sum = add(sum,n)` and then use **Step Into** to jump into that function. We will see that it pauses at line 11 and shows us different information in the Debug window:

- **Variables**: We can see that a has a value of 0 and b has a value of 1 at this point. We can also see that sum and n are out of scope now.
- **Memory**: We can see that our app is still using about 10 MB of memory at this point.
- **Threads**: We can see that our app still has one thread called main which is now suspended at line 11.

- **Call Stack:** We can see that our app has called main() which has called for (n in numbers) which has called add(a,b) which has reached line 11.

# Deploying and publishing a web app

## How to generate a signed APK file for a web app

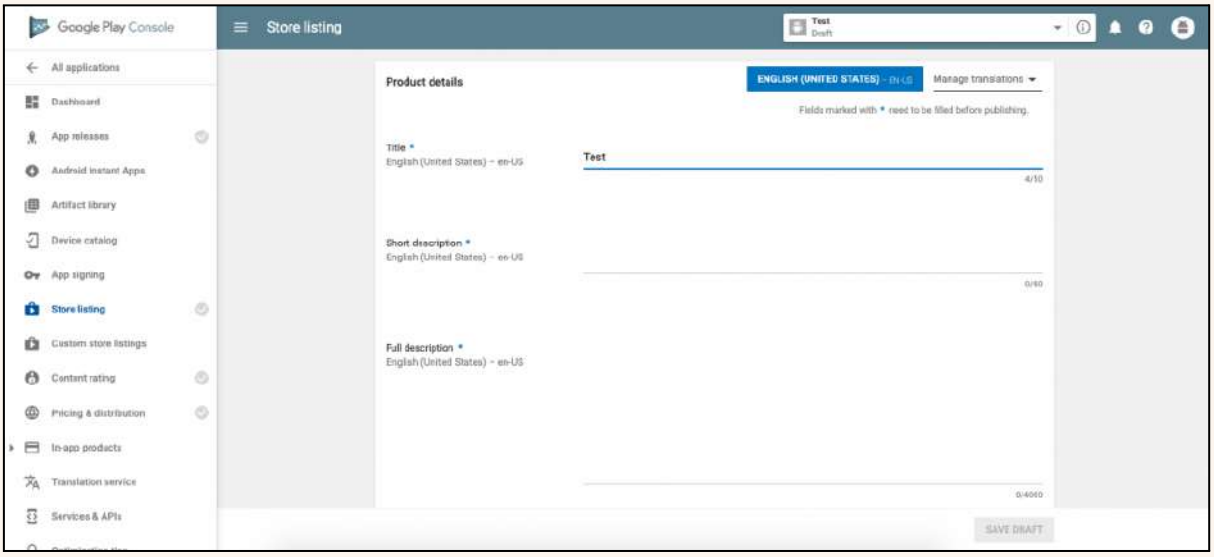



To generate a signed APK file for a web app using Android Studio, you need to follow these steps:

- ➔ Make sure your web app is ready for release and has no errors or warnings.
- ➔ Go to Android Studio's menu and select Build > Generate Signed Bundle/APK.
- ➔ In the Generate Signed Bundle/APK dialogue box, select APK and click Next.
- ➔ In the next window, you need to provide your app signing key information. If you already have an existing key store file, you can browse to it and enter its password and alias. If you don't have one, you can create a new one by clicking on Create new... button. You need to enter details such as key store path, key store password, key alias, key password, certificate validity period and certificate information. Click OK when done.
- ➔ In the next window, you need to select the build type as release and choose a destination folder for your signed APK file. You can also enable V1 (Jar Signature) and V2 (Full APK Signature) options if you want[5]. Click Finish when done.

- ➔ Wait until Android Studio generates your signed APK file successfully. You will see a message like "APK(s) generated successfully" at the bottom of Android Studio.
- ➔ You can find your signed APK file in the destination folder that you specified earlier. You can also locate it by clicking on Locate in the message box or by clicking on Show in Explorer/Reveal in Finder/Show in Files button on the right side of Android Studio.

## How to upload the APK file to Google Play Store or other platforms for distribution




To upload the APK file to Google Play Store for distribution, you need to follow these steps:

- ➔ Create a Google Play Developer account if you don't have one already. You need to pay a one-time registration fee of $25 to access the Play Console.
- ➔ Link your developer account with a Google Wallet Merchant account if you want to sell your app or offer in-app purchases.
- ➔ Create an application entry in the Play Console by clicking on Create Application button and entering your app's name and default language.
- ➔ Fill out the app store listing details such as title, description, screenshots, icon, category, etc. You can also add translations for other languages if you want.
- ➔ Upload your app bundles or APK files to Google Play by clicking on App Bundle Explorer > Upload App Bundle / APK button. You can also use Android Studio's Build > Generate Signed Bundle/APK option to upload your app directly from there. Note that since August 2021, Google Play only accepts Android App Bundles (.aab files)

instead of APK files (.apk files) for new apps. If you had an existing app that used APK files, you could still update it with APK files until November 2021.

- ➔ Provide a content rating for your app by filling out a questionnaire about its content and features. This helps Google Play determine the appropriate age groups for your app.
- ➔ Set up your app's pricing and distribution options such as countries/regions, free/paid status, device compatibility, etc. You can also opt-in for various programs and features such as Google Play Instant, Wear OS by Google, etc.
- ➔ Publish your app by clicking on Review > Start Rollout To Production button. Your app will be reviewed by Google Play before it becomes available to users. You can also use testing tracks such as internal testing, closed testing or open testing to test your app with a limited number of users before publishing it publicly.

To upload the APK file to other platforms for distribution (such as Amazon Appstore or Samsung Galaxy Store), you need to follow their specific guidelines and requirements which may differ from Google Play Store. You can find more information on their respective websites.

# Conclusion & References

## Recap of what has been learned in this e-book

1. What is Android Studio and why it is a powerful tool for developing Java web applications
2. Examples of web apps that can be built with Android Studio
3. The prerequisites for following this e-book
4. How to start a new Android Studio project
5. The structure of an Android project
6. How to use the layout editor and drag-and-drop widgets
7. How to use attributes, styles, themes, and resources to customise the appearance and behaviour of the widgets
8. How to use the code editor and write Java code
9. How to use variables, data types, operators, control structures, methods, classes, objects, inheritance, interfaces, exceptions, collections, generics, and annotations in Java
10. How to use the emulator and logcat to test and debug a web app
11. How to use breakpoints, watch expressions, step over/into/out commands, evaluate expressions, inspect variables/values/memory/threads/call stack etc.
12. How to generate a signed APK file for a web app
13. How to upload the APK file to Google Play Store or other platforms for distribution

## Tips and resources for further learning

Here are some tips, links and resources for further learning on the subject 'How to create Java web applications with Android Studio':

- ➔ You can start by following this **codelab** that guides you through creating a simple app in Android Studio using Java. It covers topics such as creating a new project, designing a user interface, adding interactivity and testing your app on an emulator or a device.
- ➔ You can also take this **course** that teaches you the basics of creating apps with Jetpack Compose, Android's modern toolkit for developing user interfaces. You'll learn the basics of the Kotlin programming language and the fundamentals of app development.

➔ You can also check out this **wikiHow** article that shows you how to create an app with Android Studio in 8 steps. It covers topics such as selecting an activity, configuring your project settings, adding code and running your app.

## Kotlin code and its benefits

Kotlin code is a concise and expressive programming **language** that is 100% compatible with Java and widely used for Android development. It allows developers to write less boilerplate code and more type-safe code. Kotlin is fully supported by Android Studio, the official IDE for Android development. This means that developers can get code completion, type checking, debugging, testing, and other features to help them write Kotlin code efficiently. Kotlin also interoperates seamlessly with Java code and libraries, so developers can use existing Java resources and APIs from Kotlin.

Some of the benefits of using Kotlin over Java are:

➔ Kotlin has *null safety* features that prevent null pointer exceptions, which are a common source of errors in Java.
➔ Kotlin is *more concise* than Java, which means it requires less code to achieve the same functionality and reduces boilerplate code. This makes the code more readable, maintainable, and less prone to bugs.
➔ Kotlin is *fully interoperable* with Java, which means it can use existing Java libraries, frameworks, and tools without any hassle. It also allows developers to mix Kotlin and Java code in the same project without any issues.
➔ Kotlin has some *modern language features* that Java lacks, such as coroutines, data classes, extension functions, lambdas, smart casts, and more. These features make Kotlin more expressive and powerful than Java.

*Here is a simple example of Kotlin code that prints "Hello World" to the console:*

```
fun main() {
    println("Hello World")
}
```

*Here is another example of Kotlin code that defines a data class and creates an instance of it:*

```
data class Person(val name: String, val age: Int)

fun main() {
    val person = Person("Alice", 25)
```

```
    println(person.name) // prints Alice
    println(person.age) // prints 25
}
```

*Here is another example of Kotlin code that uses a coroutine to perform an asynchronous task:*

```
import kotlinx.coroutines.*

fun main() {
    GlobalScope.launch { // launch a new coroutine in background
and continue
        delay(1000L) // non-blocking delay for 1 second (default
time unit is ms)
        println("World!") // print after delay
    }
    println("Hello") // main thread continues while coroutine is
delayed
    Thread.sleep(2000L) // block main thread for 2 seconds to keep
JVM alive
}
```

*Here is an example using Kotlin code that creates a class with 3 attributes and 2 methods plus a subclass with 1 attribute and 1 method and finally creates 1 object for each class:*

```
// Define a class named Animal with 3 attributes and 2 methods
class Animal(val name: String, val color: String, var age: Int) {
    // A method to make a sound
    fun makeSound() {
        println("$name makes a sound.")
    }
    // A method to increase the age by one year
    fun growOlder() {
        age++
        println("$name is now $age years old.")
    }
}

// Define a sub class named Dog that inherits from Animal and has
1 additional attribute and 1 additional method
class Dog(name: String, color: String, age: Int, val breed:
String) : Animal(name, color, age) {
    // A method to fetch a ball
```

```
    fun fetchBall() {
        println("$name fetches a ball.")
    }
}

// Create an object of Animal class
val lion = Animal("Leo", "golden", 5)
// Call the methods of Animal class on lion object
lion.makeSound()
lion.growOlder()

// Create an object of Dog class
val dog = Dog("Max", "brown", 3, "Labrador")
// Call the methods of Animal and Dog classes on dog object
dog.makeSound()
dog.growOlder()
dog.fetchBall()
```

## An Example project from scratch

Here is a straightforward (Intermediate level) new project with all the needed code and explanations.

*Create an Android app that converts any kind of a numeral system value to another equivalent numeral system value (Numeral Systems Converter).*

**Step 1**: Create a new Android Studio project

Create a new Android Studio project by selecting "Start a new Android Studio project" on the welcome screen, or by going to File > New > New Project. Choose Empty Activity as a template (default one).

**Step 2:** Set up the user interface

In the "res" folder of your project, create a new layout file named "activity_intro.xml". In this file, create two TextView's for the title and subtitle of your intro page. Then you should create a LinearLayout to incorporate the 4 checkboxes representing 4 different numeral systems.Right after, create the 4 actual CheckBox elements. Below the LinearLayout you should add a Button element that will lead to the respective screens based on the selection of the user from the checkboxes. Finally, at the bottom of the intro page create if you wish another TextView with any message you wish to show.

The final output of the “activity_intro.xml” file should look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background_image">

    <TextView
        android:id="@+id/titleTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"
        android:text="Μετατροπέας αριθμητικών συστημάτων"
        android:textColor="@color/purple_700"
        android:textStyle="bold"
        android:gravity="center"
        android:textSize="36sp" />

    <TextView
        android:id="@+id/subtitleTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/titleTextView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="26dp"
        android:text="Επιλέξτε 2 συστήματα"
        android:textStyle="bold"
        android:textColor="@color/purple_700"
        android:textSize="26sp" />

    <LinearLayout
        android:id="@+id/checkBoxLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/subtitleTextView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:layout_marginStart="220dp"
        android:orientation="vertical">

        <CheckBox
```

```
        android:id="@+id/binaryCheckBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:text="2-αδικό"
        android:textColor="@color/purple_700"
        android:textStyle="bold"
        android:button="@drawable/custom_checkbox"
        android:layout_weight="1" />

    <CheckBox
        android:id="@+id/octalCheckBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:text="8-αδικό"
        android:textColor="@color/purple_700"
        android:textStyle="bold"
        android:button="@drawable/custom_checkbox"
        android:layout_marginTop="14dp"
        android:layout_weight="1" />

    <CheckBox
        android:id="@+id/decimalCheckBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:text="10-αδικό"
        android:textColor="@color/purple_700"
        android:textStyle="bold"
        android:button="@drawable/custom_checkbox"
        android:layout_marginTop="14dp"
        android:layout_weight="1" />

    <CheckBox
        android:id="@+id/hexadecimalCheckBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="24sp"
        android:text="16-αδικό"
        android:textColor="@color/purple_700"
        android:textStyle="bold"
        android:button="@drawable/custom_checkbox"
        android:layout_marginTop="14dp"
        android:layout_weight="1" />
```

```
    </LinearLayout>

    <Button
        android:id="@+id/startButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/checkBoxLayout"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:clickable="true"
        android:text="Start"
        android:background="@color/citro"
        android:textColor="@color/purple_700"
        android:textSize="24sp"
        android:focusable="true" />

    <TextView
        android:id="@+id/bottomtitleTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/titleTextView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="416dp"
        android:text="Η πίστη σώζει!!!"
        android:textStyle="bold"
        android:textColor="@color/purple_700"
        android:textSize="26sp" />

</RelativeLayout>
```

Here is a screen shot of the intro page in the Design view of Android Studio:


In the XML code of the intro page layout, you notice that I'm using a background image that is a file named background_image, and we refer to that file with the entry: android:background="@drawable/background_image" at the top RelativeLayout.

Actually, we have created in advance another xml file named 'background_image.xml' that is located in the 'drawable' folder with the following code inside:

```
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/intro_gradient"/>
```

And of course the file 'intro_gradient' in the above code, refers to our actual image file named 'intro_gradient.jpg' that we should have already uploaded to the folder 'drawable' as well.

Concerning the various colors you notice in the intro layout, those have been placed in another created xml file named 'colors.xml' inside the 'values' subfolder of 'drawable' folder and its content is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="orange">#FFF5A65B</color>
    <color name="sage">#FFC1CC99</color>
    <color name="citro">#FFffbe0b</color>
    <color name="pale_brown">#FFbb9457</color>
    <color name="bright_pink">#FFf72585</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

**Step 3**: Create the intro_activity.java file

Now it's time to create the java code for the activity_intro.xml layout UI. The name of the file we create is 'intro_activity.java' and it is located in the java folder, in its subfolder related to com.example.'the name of our project'. The content of that java file looks like the following:

```
package com.example.numeralsystemsconverter;

import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.CheckBox;
```

```
public class intro_activity extends AppCompatActivity {

    private CheckBox binaryCheckBox;
    private CheckBox octalCheckBox;
    private CheckBox decimalCheckBox;
    private CheckBox hexadecimalCheckBox;

    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_intro);

        // initialize the UI elements
        binaryCheckBox = findViewById(R.id.binaryCheckBox);
        octalCheckBox = findViewById(R.id.octalCheckBox);
        decimalCheckBox = findViewById(R.id.decimalCheckBox);
        hexadecimalCheckBox =
findViewById(R.id.hexadecimalCheckBox);

        Button startButton = findViewById(R.id.startButton);

        // set an OnClickListener for the start button
        startButton.setOnClickListener(v -> {
            // get the number of selected checkboxes
            int numChecked = 0;
            if (binaryCheckBox.isChecked()) {
                numChecked++;
            }
            if (octalCheckBox.isChecked()) {
                numChecked++;
            }
            if (decimalCheckBox.isChecked()) {
                numChecked++;
            }
            if (hexadecimalCheckBox.isChecked()) {
                numChecked++;
            }

            // create an intent to start the appropriate
conversion activity based on the selected checkboxes
            Intent intent;
            if (numChecked == 2) {
                if (binaryCheckBox.isChecked() &&
```

```
octalCheckBox.isChecked()) {
                    intent = new Intent(intro_activity.this,
BinaryAndOctal.class);
                } else if (binaryCheckBox.isChecked() &&
decimalCheckBox.isChecked()) {
                    intent = new Intent(intro_activity.this,
BinaryAndDecimal.class);
                } else if (binaryCheckBox.isChecked() &&
hexadecimalCheckBox.isChecked()) {
                    intent = new Intent(intro_activity.this,
BinaryAndHexadecimal.class);
                } else if (octalCheckBox.isChecked() &&
decimalCheckBox.isChecked()) {
                    intent = new Intent(intro_activity.this,
OctalAndDecimal.class);
                } else if (octalCheckBox.isChecked() &&
hexadecimalCheckBox.isChecked()) {
                    intent = new Intent(intro_activity.this,
OctalAndHexadecimal.class);
                } else {
                    intent = new Intent(intro_activity.this,
DecimalAndHexadecimal.class);
                }
            } else {
                // If more or less than 2 checkboxes are selected,
do nothing
                return;
            }

            // start the selected activity
            startActivity(intent);
        });
    }
}
```

The java file basically reads the input of checkboxes that are selected by the user when picks the numeral systems he/she prefers to make numeral system conversions and creates appropriate new intent (redirection to appropriate new layout/class) that operates the actual conversions between the chosen 2 numeral systems.

**Step 4**: Create the 6 pairs of layouts (xml files) and java files (classes) that operate the actual conversions.

File: activity_binary_and_decimal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/purple_200"
    android:padding="16dp">

    <TextView
        android:id="@+id/title_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="38sp"
        android:gravity="center"
        android:textColor="@color/black"
        android:text="Δυαδικό σε Δεκαδικό"
        android:textStyle="bold"
        android:layout_marginTop="50dp"/>


    <EditText
        android:id="@+id/binary_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:textSize="26sp"
        android:hint="Εισάγετε 2-αδικό αριθμό"
        android:layout_marginTop="56dp"/>


    <EditText
        android:id="@+id/decimal_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:textSize="26sp"
        android:hint="Εισάγετε 10-αδικό αριθμό"
        android:layout_marginTop="36dp"/>

    <Button
        android:id="@+id/convert_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
```

```
        android:text="Μετατροπή"
        android:layout_marginTop="46dp"/>

    <TextView
        android:id="@+id/output_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        android:textColor="@color/black"
        android:gravity="center"
        android:layout_marginTop="55dp"/>

</LinearLayout>
```

File: activity_binary_and_hexadecimal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/orange"
    android:padding="16dp">

    <TextView
        android:id="@+id/title_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="38sp"
        android:gravity="center"
        android:textColor="@color/black"
        android:text="Δυαδικό σε Δεκαεξαδικό"
        android:textStyle="bold"
        android:layout_marginTop="50dp"/>

    <EditText
        android:id="@+id/binary_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:textSize="26sp"
        android:hint="Εισάγετε 2-αδικό αριθμό"
        android:layout_marginTop="56dp"/>
```

```
    <EditText
        android:id="@+id/hexadecimal_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:textSize="26sp"
        android:hint="Εισάγετε 16-αδικό αριθμό"
        android:layout_marginTop="36dp"/>

    <Button
        android:id="@+id/convert_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Μετατροπή"
        android:layout_marginTop="46dp"/>

    <TextView
        android:id="@+id/output_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        android:textColor="@color/black"
        android:gravity="center"
        android:layout_marginTop="55dp"/>
</LinearLayout>
```

File: activity_binary_and_octal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/sage"
    android:padding="16dp">

    <TextView
        android:id="@+id/title_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="38sp"
```

```
        android:gravity="center"
        android:textColor="@color/black"
        android:text="Δυαδικό σε Οκταδικό"
        android:textStyle="bold"
        android:layout_marginTop="50dp"/>


    <EditText
        android:id="@+id/binary_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:textSize="26sp"
        android:hint="Εισάγετε 2-αδικό αριθμό"
        android:layout_marginTop="56dp"/>


    <EditText
        android:id="@+id/octal_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:textSize="26sp"
        android:hint="Εισάγετε 8-αδικό αριθμό"
        android:layout_marginTop="36dp"/>

    <Button
        android:id="@+id/convert_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Μετατροπή"
        android:layout_marginTop="46dp"/>

    <TextView
        android:id="@+id/output_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        android:textColor="@color/black"
        android:gravity="center"
        android:layout_marginTop="55dp"/>

</LinearLayout>
```

File: activity_decimal_and_hexadecimal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/citro"
    android:padding="16dp">

    <TextView
        android:id="@+id/title_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="38sp"
        android:gravity="center"
        android:textColor="@color/black"
        android:text="Δεκαδικό σε Δεκαεξαδικό"
        android:textStyle="bold"
        android:layout_marginTop="50dp"/>


    <EditText
        android:id="@+id/decimal_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:textSize="26sp"
        android:hint="Εισάγετε 10-αδικό αριθμό"
        android:layout_marginTop="56dp"/>


    <EditText
        android:id="@+id/hexadecimal_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:textSize="26sp"
        android:hint="Εισάγετε 16-αδικό αριθμό"
        android:layout_marginTop="36dp"/>

    <Button
        android:id="@+id/convert_button"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Μετατροπή"
        android:layout_marginTop="46dp"/>

    <TextView
        android:id="@+id/output_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        android:textColor="@color/black"
        android:gravity="center"
        android:layout_marginTop="55dp"/>

</LinearLayout>
```

File: activity_octal_and_decimal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/pale_brown"
    android:padding="16dp">

    <TextView
        android:id="@+id/title_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="38sp"
        android:gravity="center"
        android:textColor="@color/black"
        android:text="Οκταδικό σε Δεκαδικό"
        android:textStyle="bold"
        android:layout_marginTop="50dp"/>


    <EditText
        android:id="@+id/octal_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
```

```
        android:textSize="26sp"
        android:hint="Εισάγετε 8-αδικό αριθμό"
        android:layout_marginTop="56dp"/>


    <EditText
        android:id="@+id/decimal_edit_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberSigned|numberDecimal"
        android:textSize="26sp"
        android:hint="Εισάγετε 10-αδικό αριθμό"
        android:layout_marginTop="36dp"/>

    <Button
        android:id="@+id/convert_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Μετατροπή"
        android:layout_marginTop="46dp"/>

    <TextView
        android:id="@+id/output_text_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        android:textColor="@color/black"
        android:gravity="center"
        android:layout_marginTop="55dp"/>

</LinearLayout>
```

File: activity_octal_and_hexadecimal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/bright_pink"
    android:padding="16dp">
```

```
<TextView
    android:id="@+id/title_text_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="38sp"
    android:gravity="center"
    android:textColor="@color/black"
    android:text="Οκταδικό σε Δεκαεξαδικό"
    android:textStyle="bold"
    android:layout_marginTop="50dp"/>


<EditText
    android:id="@+id/octal_edit_text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="numberSigned|numberDecimal"
    android:textSize="26sp"
    android:hint="Εισάγετε 8-αδικό αριθμό"
    android:layout_marginTop="56dp"/>


<EditText
    android:id="@+id/hexadecimal_edit_text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="text"
    android:textSize="26sp"
    android:hint="Εισάγετε 16-αδικό αριθμό"
    android:layout_marginTop="36dp"/>

<Button
    android:id="@+id/convert_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="Μετατροπή"
    android:layout_marginTop="46dp"/>

<TextView
    android:id="@+id/output_text_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="26sp"
    android:textColor="@color/black"
```

```
        android:gravity="center"
        android:layout_marginTop="55dp"/>

</LinearLayout>
```

File: BinaryAndDecimal.java

```
package com.example.numeralsystemsconverter;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class BinaryAndDecimal extends AppCompatActivity {

    EditText binaryEditText, decimalEditText;
    TextView outputTextView, titleTextView;
    Button convertButton;

    @SuppressLint({"MissingInflatedId", "SetTextI18n"})
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_binary_and_decimal);

        // Get references to the UI elements
        titleTextView = findViewById(R.id.title_text_view);
        binaryEditText = findViewById(R.id.binary_edit_text);
        decimalEditText = findViewById(R.id.decimal_edit_text);
        outputTextView = findViewById(R.id.output_text_view);
        convertButton = findViewById(R.id.convert_button);

        // Set click listener for the convert button
        convertButton.setOnClickListener(v -> {

            // Convert binary to decimal
            String binaryString =
binaryEditText.getText().toString();
            if (!binaryString.isEmpty()) {
                if (validateBinaryInput(binaryString)) {
```

```
                    double decimal = 0.0;
                    boolean isNegative = false;
                    if (binaryString.startsWith("-")) {
                        isNegative = true;
                        binaryString = binaryString.substring(1);
                    }
                    if (binaryString.contains(".")) {
                        // Handle decimal binary numbers
                        String[] parts =
binaryString.split("\\.");
                        for (int i = 0; i < parts[0].length();
i++) {
                            if (parts[0].charAt(i) == '1') {
                                decimal += Math.pow(2,
parts[0].length() - i - 1);
                            }
                        }
                        for (int i = 0; i < parts[1].length();
i++) {
                            if (parts[1].charAt(i) == '1') {
                                decimal += Math.pow(2, -i - 1);
                            }
                        }
                    } else {
                        decimal = Integer.parseInt(binaryString,
2);
                    }
                    if (isNegative) {
                        decimal = -decimal;
                    }
                    if (decimal % 1 == 0) {

outputTextView.setText(Integer.toString((int) decimal));
                    } else {

outputTextView.setText(Double.toString(decimal));
                    }

                } else {
                    outputTextView.setText("μή έγκυρη τιμή");
                }
            }

            // Convert decimal to binary
            String decimalString =
```

```
decimalEditText.getText().toString();
            if (!decimalString.isEmpty()) {
                double decimal = 0.0;
                try {
                    decimal = Double.parseDouble(decimalString);
                } catch (NumberFormatException e) {
                    outputTextView.setText("μή έγκυρη τιμή");
                    return;
                }
                String binary = "";
                if (decimal < 0) {
                    binary = "-";
                    decimal = -decimal;
                }
                int integerPart = (int) decimal;
                double decimalPart = decimal - integerPart;

                // Add condition to check if there is a fractional
part before appending the decimal point
                if (decimalPart != 0) {
                    binary += Integer.toBinaryString(integerPart)
+ ".";
                } else {
                    binary += Integer.toBinaryString(integerPart);
                }

                int maxDecimalPlaces = 10; // Set a maximum number
of decimal places
                while (decimalPart > 0 && maxDecimalPlaces > 0) {
                    decimalPart *= 2;
                    if (decimalPart >= 1) {
                        binary += "1";
                        decimalPart -= 1;
                    } else {
                        binary += "0";
                    }
                    maxDecimalPlaces--;
                }

                outputTextView.setText(binary);
            }
        });
    }
    private boolean validateBinaryInput(String binaryString) {
        if (binaryString.isEmpty()) {
```

```
            return false;
        }
        boolean isNegative = false;
        if (binaryString.charAt(0) == '-') {
            isNegative = true;
            binaryString = binaryString.substring(1);
        }
        boolean hasDecimalPoint = false;
        for (int i = 0; i < binaryString.length(); i++) {
            char c = binaryString.charAt(i);
            if (c == '.') {
                if (hasDecimalPoint) {
                    return false; // only one decimal point
allowed
                }
                hasDecimalPoint = true;
            } else if (c != '0' && c != '1') {
                return false; // invalid character
            }
        }
        if (isNegative) {
            binaryString = '-' + binaryString;
        }
        return true;
    }
}
```

File: BinaryAndHexadecimal.java

```
package com.example.numeralsystemsconverter;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class BinaryAndHexadecimal extends AppCompatActivity {

    EditText binaryEditText, hexadecimalEditText;
    TextView outputTextView, titleTextView;
    Button convertButton;
```

```
    @SuppressLint({"MissingInflatedId", "SetTextI18n"})
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_binary_and_hexadecimal);

        // Get references to the UI elements
        titleTextView = findViewById(R.id.title_text_view);
        binaryEditText = findViewById(R.id.binary_edit_text);
        hexadecimalEditText =
findViewById(R.id.hexadecimal_edit_text);
        outputTextView = findViewById(R.id.output_text_view);
        convertButton = findViewById(R.id.convert_button);

        // Set click listener for the convert button
        convertButton.setOnClickListener(v -> {

            // Convert binary to hexadecimal

            String binaryString =
binaryEditText.getText().toString();

            if (!isValidBinary(binaryString)) {
                outputTextView.setText("μή έγκυρη τιμή");
            } else {
                // Continue with conversion
                // Check if the binary number is negative
                if (binaryString.length() > 0) {
                    boolean isNegative = false;
                    if (binaryString.charAt(0) == '-') {
                        isNegative = true;
                        binaryString = binaryString.substring(1);
                    }
                    // Split the binary number into integer and
fractional parts
                    String[] parts = binaryString.split("\\.");
                    String integerPart = parts[0];
                    String fractionalPart = parts.length > 1 ?
parts[1] : "";

                    // Convert the integer part to hexadecimal
                    StringBuilder hexBuilder = new
StringBuilder();
                    int missingZeros = 4 - integerPart.length() %
```

```
4;
                    if (missingZeros != 4) {
                        integerPart = "0".repeat(missingZeros) +
integerPart;
                    }
                    for (int i = 0; i < integerPart.length(); i +=
4) {
                        String fourBits = integerPart.substring(i,
i + 4);
                        int decimalValue =
Integer.parseInt(fourBits, 2);

hexBuilder.append(Integer.toHexString(decimalValue).toUpperCase())
;
                    }
                    // Convert the fractional part to hexadecimal
                    if (!fractionalPart.isEmpty()) {
                        hexBuilder.append(".");
                        missingZeros = 4 - fractionalPart.length()
% 4;
                        if (missingZeros != 4) {
                            fractionalPart +=
"0".repeat(missingZeros);
                        }
                        for (int i = 0; i <
fractionalPart.length(); i += 4) {
                            String fourBits =
fractionalPart.substring(i, i + 4);
                            int decimalValue =
Integer.parseInt(fourBits, 2);

hexBuilder.append(Integer.toHexString(decimalValue).toUpperCase())
;
                        }
                    }
                    // Add a negative sign if the original binary
number was negative
                    if (isNegative) {
                        hexBuilder.insert(0, "-");
                    }
                    // The final hexadecimal representation of the
binary number
                    String hexString = hexBuilder.toString();
                    outputTextView.setText(hexString);
                }
```

```
            }

            // Convert Hexadecimal to Binary

            String hexadecimalString =
hexadecimalEditText.getText().toString();

            if (hexadecimalString.matches("-?[0-9a-fA-F]+")) {
                // hexadecimalString is a valid hexadecimal number
                // Check if the hexadecimal number is negative
                if (hexadecimalString.length() > 0) {
                    boolean isHexNegative = false;
                    if (hexadecimalString.charAt(0) == '-') {
                        isHexNegative = true;
                        hexadecimalString =
hexadecimalString.substring(1);
                    }

                    // Split the hexadecimal number into integer
and fractional parts
                    String[] parts2 =
hexadecimalString.split("\\.");
                    String integerPart2 = parts2[0];
                    String fractionalPart2 = parts2.length > 1 ?
parts2[1] : "";

                    // Convert the integer part to binary
                    StringBuilder binaryBuilder2 = new
StringBuilder();
                    for (int i = 0; i < integerPart2.length();
i++) {
                        char hexDigit = integerPart2.charAt(i);
                        int decimalValue2 =
Integer.parseInt(String.valueOf(hexDigit), 16);
                        String fourBits =
Integer.toBinaryString(decimalValue2);
                        binaryBuilder2.append("0".repeat(4 -
fourBits.length())).append(fourBits);
                    }

                    // Remove leading zeros
                    while (binaryBuilder2.length() > 1 &&
binaryBuilder2.charAt(0) == '0') {
                        binaryBuilder2.deleteCharAt(0);
                    }
```

```
                    // Convert the fractional part to binary
                    if (!fractionalPart2.isEmpty()) {
                        binaryBuilder2.append(".");
                        for (int i = 0; i <
fractionalPart2.length(); i++) {
                            char hexDigit =
fractionalPart2.charAt(i);
                            int decimalValue =
Integer.parseInt(String.valueOf(hexDigit), 16);
                            String fourBits =
Integer.toBinaryString(decimalValue);
                            binaryBuilder2.append("0".repeat(4 -
fourBits.length())).append(fourBits);
                        }
                    }

                    // Add a negative sign if the original
hexadecimal number was negative
                    if (isHexNegative) {
                        binaryBuilder2.insert(0, "-");
                    }

                    // The final binary representation of the
hexadecimal number
                    String binaryString2 =
binaryBuilder2.toString();
                    outputTextView.setText(binaryString2);
                } else {
                    outputTextView.setText("μή έγκυρη τιμή");
                }
                }
            });
        }

        private boolean isValidBinary(String binaryString) {
            // Check if the binary string is empty
            if (binaryString.isEmpty()) {
                return false;
            }
            // Check if the binary string contains only 0s, 1s and a
single decimal point
            int pointCount = 0;
            for (int i = 0; i < binaryString.length(); i++) {
                char c = binaryString.charAt(i);
```

```
            if (c != '0' && c != '1' && c != '.' && c != '-') {
                return false;
            }
            if (c == '.') {
                pointCount++;
                if (pointCount > 1) {
                    return false;
                }
            }
            // Check that minus sign is only at the beginning of
the string
            if (c == '-' && i != 0) {
                return false;
            }
        }
        return true;
    }
}
```

File: BinaryAndOctal.java

```
package com.example.numeralsystemsconverter;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class BinaryAndOctal extends AppCompatActivity {

    private EditText binaryEditText;
    private EditText octalEditText;
    private TextView outputTextView;

    @SuppressLint("SetTextI18n")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_binary_and_octal);

        binaryEditText = findViewById(R.id.binary_edit_text);
```

```
        octalEditText = findViewById(R.id.octal_edit_text);
        Button convertButton = findViewById(R.id.convert_button);
        outputTextView = findViewById(R.id.output_text_view);

        convertButton.setOnClickListener(view -> {

            // Convert Binary to Octal
            String binaryString =
binaryEditText.getText().toString().trim();

            if (isValidBinary(binaryString)) {
                // The input binaryString is valid
                if (binaryString.length() > 0) {
                    boolean isNegative = false;
                    if (binaryString.charAt(0) == '-') {
                        isNegative = true;
                        binaryString = binaryString.substring(1);
                    }

                    int pointIndex = binaryString.indexOf('.');
                    if (pointIndex == -1) {
                        pointIndex = binaryString.length();
                    }

                    // Convert the integer part to decimal
                    int intPartDecimal = 0;
                    for (int i = 0; i < pointIndex; i++) {
                        intPartDecimal +=
Character.getNumericValue(binaryString.charAt(i)) * Math.pow(2,
pointIndex - i - 1);
                    }

                    // Convert the fractional part to decimal
                    double fracPartDecimal = 0;
                    for (int i = pointIndex + 1; i <
binaryString.length(); i++) {
                        fracPartDecimal +=
Character.getNumericValue(binaryString.charAt(i)) * Math.pow(2,
pointIndex - i);
                    }

                    // Convert the integer part to octal
                    StringBuilder intPartOctalBuilder = new
StringBuilder();
                    while (intPartDecimal > 0) {
```

```
                    int remainder = intPartDecimal % 8;
                    intPartOctalBuilder.append(remainder);
                    intPartDecimal /= 8;
                }
                intPartOctalBuilder.reverse();

                // Convert the fractional part to octal
                StringBuilder fracPartOctalBuilder = new
StringBuilder(".");
                while (fracPartDecimal > 0) {
                    fracPartDecimal *= 8;
                    int digit = (int) fracPartDecimal;
                    fracPartOctalBuilder.append(digit);
                    fracPartDecimal -= digit;
                }

                // Combine the integer and fractional parts
and add negative sign if needed
                StringBuilder resultBuilder = new
StringBuilder();
                if (isNegative) {
                    resultBuilder.append('-');
                }
                resultBuilder.append(intPartOctalBuilder);
                if (fracPartOctalBuilder.length() > 1) { //
check if there are any digits after the decimal point

resultBuilder.append(fracPartOctalBuilder);
                }


outputTextView.setText(resultBuilder.toString());
                }
            } else {
                // The input binaryString is not valid
                outputTextView.setText("μή έγκυρη τιμή");
            }



            // Convert Octal to Binary

            String octalString =
octalEditText.getText().toString().trim();
```

```
            if (isValidOctal(octalString)) {
                // The input octalString is valid
                if (octalString.length() > 0) {
                    boolean isOctalNegative = false;
                    if (octalString.charAt(0) == '-') {
                        isOctalNegative = true;
                        octalString = octalString.substring(1);
                    }

                    int pointOctalIndex =
octalString.indexOf('.');
                    if (pointOctalIndex == -1) {
                        pointOctalIndex = octalString.length();
                    }

                    // Convert the integer part to decimal
                    int intOctalPartDecimal = 0;
                    for (int i = 0; i < pointOctalIndex; i++) {
                        intOctalPartDecimal +=
Character.getNumericValue(octalString.charAt(i)) * Math.pow(8,
pointOctalIndex - i - 1);
                    }

                    // Convert the fractional part to decimal
                    double fracOctalPartDecimal = 0;
                    for (int i = pointOctalIndex + 1; i <
octalString.length(); i++) {
                        fracOctalPartDecimal +=
Character.getNumericValue(octalString.charAt(i)) * Math.pow(8,
pointOctalIndex - i);
                    }

                    // Convert the integer part to binary
                    StringBuilder intPartBinaryBuilder = new
StringBuilder();
                    while (intOctalPartDecimal > 0) {
                        int remainder = intOctalPartDecimal % 2;
                        intPartBinaryBuilder.append(remainder);
                        intOctalPartDecimal /= 2;
                    }
                    intPartBinaryBuilder.reverse();

                    // Convert the fractional part to binary
                    StringBuilder fracPartBinaryBuilder = new
StringBuilder(".");
```

```
                    while (fracOctalPartDecimal > 0) {
                        fracOctalPartDecimal *= 2;
                        int digit = (int) fracOctalPartDecimal;
                        fracPartBinaryBuilder.append(digit);
                        fracOctalPartDecimal -= digit;
                    }

                    // Combine the integer and fractional parts
and add negative sign if needed
                    StringBuilder resultBinaryBuilder = new
StringBuilder();
                    if (isOctalNegative) {
                        resultBinaryBuilder.append('-');
                    }

resultBinaryBuilder.append(intPartBinaryBuilder);
                    if (fracPartBinaryBuilder.length() > 1) { //
check if there are any digits after the decimal point

resultBinaryBuilder.append(fracPartBinaryBuilder);
                    }


outputTextView.setText(resultBinaryBuilder.toString());
            } else {
                // The input octalString is not valid
                outputTextView.setText("μή έγκυρη τιμή");
            }


            }
        });
    }

    private boolean isValidBinary(String binaryString) {
        // Check if the binary string is empty
        if (binaryString.isEmpty()) {
            return false;
        }
        // Check if the binary string contains only 0s, 1s and a
single decimal point
        int pointCount = 0;
        for (int i = 0; i < binaryString.length(); i++) {
            char c = binaryString.charAt(i);
            if (c != '0' && c != '1' && c != '.' && c != '-') {
```

```
                return false;
            }
            if (c == '.') {
                pointCount++;
                if (pointCount > 1) {
                    return false;
                }
            }
            // Check that minus sign is only at the beginning of
the string
            if (c == '-' && i != 0) {
                return false;
            }
        }
        return true;
    }

    private boolean isValidOctal(String octalString) {
        // Check if the octal string is empty
        if (octalString.isEmpty()) {
            return false;
        }
        // Check if the octal string contains only valid digits of
the octal numeral system plus accepting fractional part and plus
accepting negative sign at front
        int pointCount = 0;
        for (int i = 0; i < octalString.length(); i++) {
            char c = octalString.charAt(i);
            if ((c < '0' || c > '7') && c != '.' && c != '-') {
                return false;
            }
            if (c == '.') {
                pointCount++;
                if (pointCount > 1) {
                    return false;
                }
            }
            // Check that minus sign is only at the beginning of
the string
            if (c == '-' && i != 0) {
                return false;
            }
        }
        return true;
    }
```

```
}
```

File: DecimalAndHexadecimal.java

```
package com.example.numeralsystemsconverter;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.text.TextUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class DecimalAndHexadecimal extends AppCompatActivity {

    private EditText decimalEditText;
    private EditText hexadecimalEditText;
    private TextView outputTextView;

    @SuppressLint("SetTextI18n")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_decimal_and_hexadecimal);

        decimalEditText = findViewById(R.id.decimal_edit_text);
        hexadecimalEditText =
findViewById(R.id.hexadecimal_edit_text);
        outputTextView = findViewById(R.id.output_text_view);

        Button convertButton = findViewById(R.id.convert_button);
        convertButton.setOnClickListener(v -> {
            String decimalString =
decimalEditText.getText().toString().trim();
            String hexadecimalString =
hexadecimalEditText.getText().toString().trim();

            // Check if both fields are empty
            if (TextUtils.isEmpty(decimalString) &&
TextUtils.isEmpty(hexadecimalString)) {
                outputTextView.setText("Εισάγετε μια τιμή για
μετατροπή.");
```

```
                return;
            }

            // Check if both fields are filled in
            if (!TextUtils.isEmpty(decimalString) &&
!TextUtils.isEmpty(hexadecimalString)) {
                outputTextView.setText("Εισάγετε μόνο μια τιμή σε
ένα πεδίο.");
                return;
            }

            // Check if the decimal input is valid
            if (!TextUtils.isEmpty(decimalString) &&
!isValidDecimal(decimalString)) {
                outputTextView.setText("Εισάγετε έναν έγκυρο
δεκαδικό αριθμό.");
                return;
            }

            // Check if the hexadecimal input is valid
            if (!TextUtils.isEmpty(hexadecimalString) &&
!isValidHexadecimal(hexadecimalString)) {
                outputTextView.setText("Εισάγετε έναν έγκυρο
δεκαεξαδικό αριθμό.");
                return;
            }

            // Perform the conversion
            if (!TextUtils.isEmpty(decimalString)) {
                double decimal =
Double.parseDouble(decimalString);
                int integerPart = (int) decimal;
                double fractionalPart = Math.abs(decimal -
integerPart);
                String hexadecimal;
                if (decimal < 0) {
                    hexadecimal = "-" +
Integer.toHexString(Math.abs(integerPart));
                } else {
                    hexadecimal =
Integer.toHexString(integerPart);
                }
                if (fractionalPart > 0) {
                    hexadecimal += ".";
                    while (fractionalPart > 0) {
```

```
                    fractionalPart *= 16;
                    int digit = (int) fractionalPart;
                    hexadecimal += Integer.toHexString(digit);
                    fractionalPart -= digit;
                }
            }
            outputTextView.setText(hexadecimal.toUpperCase());
        } else {
            int decimal = Integer.parseInt(hexadecimalString,
16);
            outputTextView.setText(Integer.toString(decimal));
        }
    });
}

private boolean isValidDecimal(String decimalString) {
    // Regular expression for a decimal string with fractional
part and optional negative sign
    String regex = "-?[0-9]+(\\.[0-9]+)?";
    return decimalString.matches(regex);
}

private boolean isValidHexadecimal(String hexadecimalString) {
    // Regular expression for a hexadecimal string with
optional negative sign
    String regex = "-?[0-9A-Fa-f]+";
    return !hexadecimalString.contains(".") &&
hexadecimalString.matches(regex);
}
}
```

File: OctalAndDecimal.java

```
package com.example.numeralsystemsconverter;

import android.os.Bundle;
import android.text.TextUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class OctalAndDecimal extends AppCompatActivity {
```

```
    private EditText octalEditText, decimalEditText;
    private TextView outputTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_octal_and_decimal);

        octalEditText = findViewById(R.id.octal_edit_text);
        decimalEditText = findViewById(R.id.decimal_edit_text);
        outputTextView = findViewById(R.id.output_text_view);
        Button convertButton = findViewById(R.id.convert_button);

        convertButton.setOnClickListener(v -> {
            String octalString =
octalEditText.getText().toString().trim();
            String decimalString =
decimalEditText.getText().toString().trim();
            if (TextUtils.isEmpty(octalString) &&
TextUtils.isEmpty(decimalString)) {
                outputTextView.setText("Εισάγετε έναν οκταδικό ή
δεκαδικό αριθμό.");
            } else if (!TextUtils.isEmpty(octalString) &&
!TextUtils.isEmpty(decimalString)) {
                outputTextView.setText("Εισάγετε μόνο, έναν
οκταδικό ή δεκαδικό αριθμό.");
            } else if (!TextUtils.isEmpty(octalString)) {
                if (!isValidOctal(octalString)) {
                    outputTextView.setText("Εισάγετε έναν έγκυρο
οκταδικό αριθμό.");
                } else {
                    double decimalValue =
octalToDecimal(octalString);

outputTextView.setText(String.valueOf(decimalValue));
                }
            } else if (!TextUtils.isEmpty(decimalString)) {
                if (!isValidDecimal(decimalString)) {
                    outputTextView.setText("Εισάγετε έναν έγκυρο
δεκαδικό αριθμό.");
                } else {
                    double decimalValue =
Double.parseDouble(decimalString);
                    String octalValue =
decimalToOctal(decimalValue);
```

```
                outputTextView.setText(octalValue);
            }
        }
    });
}

private boolean isValidOctal(String octalString) {
    if (octalString.charAt(0) == '-') {
        octalString = octalString.substring(1);
    }
    String[] parts = octalString.split("\\.");
    for (int i = 0; i < parts[0].length(); i++) {
        char c = parts[0].charAt(i);
        if (c < '0' || c > '7') {
            return false;
        }
    }
    if (parts.length > 1) {
        for (int i = 0; i < parts[1].length(); i++) {
            char c = parts[1].charAt(i);
            if (c < '0' || c > '7') {
                return false;
            }
        }
    }
    return true;
}

private double octalToDecimal(String octalString) {
    boolean isNegative = false;
    if (octalString.charAt(0) == '-') {
        isNegative = true;
        octalString = octalString.substring(1);
    }
    String[] parts = octalString.split("\\.");
    double decimalValue = 0.0;
    int n = parts[0].length();
    for (int i = 0; i < n; i++) {
        char c = parts[0].charAt(i);
        int digit = c - '0';
        decimalValue += digit * Math.pow(8, n - i - 1);
    }
    if (parts.length > 1) {
        n = parts[1].length();
        for (int i = 0; i < n; i++) {
```

```
                char c = parts[1].charAt(i);
                int digit = c - '0';
                decimalValue += digit * Math.pow(8, -(i + 1));
            }
        }
        return isNegative ? -decimalValue : decimalValue;
    }

    private boolean isValidDecimal(String decimalString) {
        try {
            Double.parseDouble(decimalString);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }

    private String decimalToOctal(double decimalValue) {
        StringBuilder octalValue = new StringBuilder();
        if (decimalValue == 0) {
            return "0";
        }
        boolean isNegative = false;
        if (decimalValue < 0) {
            isNegative = true;
            decimalValue = -decimalValue;
        }
        int integerPart = (int) decimalValue;
        double fractionalPart = decimalValue - integerPart;
        while (integerPart > 0) {
            int remainder = Math.floorMod(integerPart, 8);
            octalValue.append(remainder);
            integerPart /= 8;
        }
        if (isNegative) {
            octalValue.append("-");
        }
        octalValue.reverse();

        if (fractionalPart > 0) {
            octalValue.append(".");
            while (fractionalPart > 0 && octalValue.length() < 20)
{ // Limit the length of the result
                fractionalPart *= 8;
                int digit = (int) fractionalPart;
```

```
                octalValue.append(digit);
                fractionalPart -= digit;
            }
        }

        return octalValue.toString();
    }
}
```

File: OctalAndHexadecimal.java

```
package com.example.numeralsystemsconverter;

import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class OctalAndHexadecimal extends AppCompatActivity {

    private EditText octalEditText;
    private EditText hexadecimalEditText;
    private TextView outputTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_octal_and_hexadecimal);

        octalEditText = findViewById(R.id.octal_edit_text);
        hexadecimalEditText =
findViewById(R.id.hexadecimal_edit_text);
        outputTextView = findViewById(R.id.output_text_view);

        Button convertButton = findViewById(R.id.convert_button);
        convertButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String octalString =
```

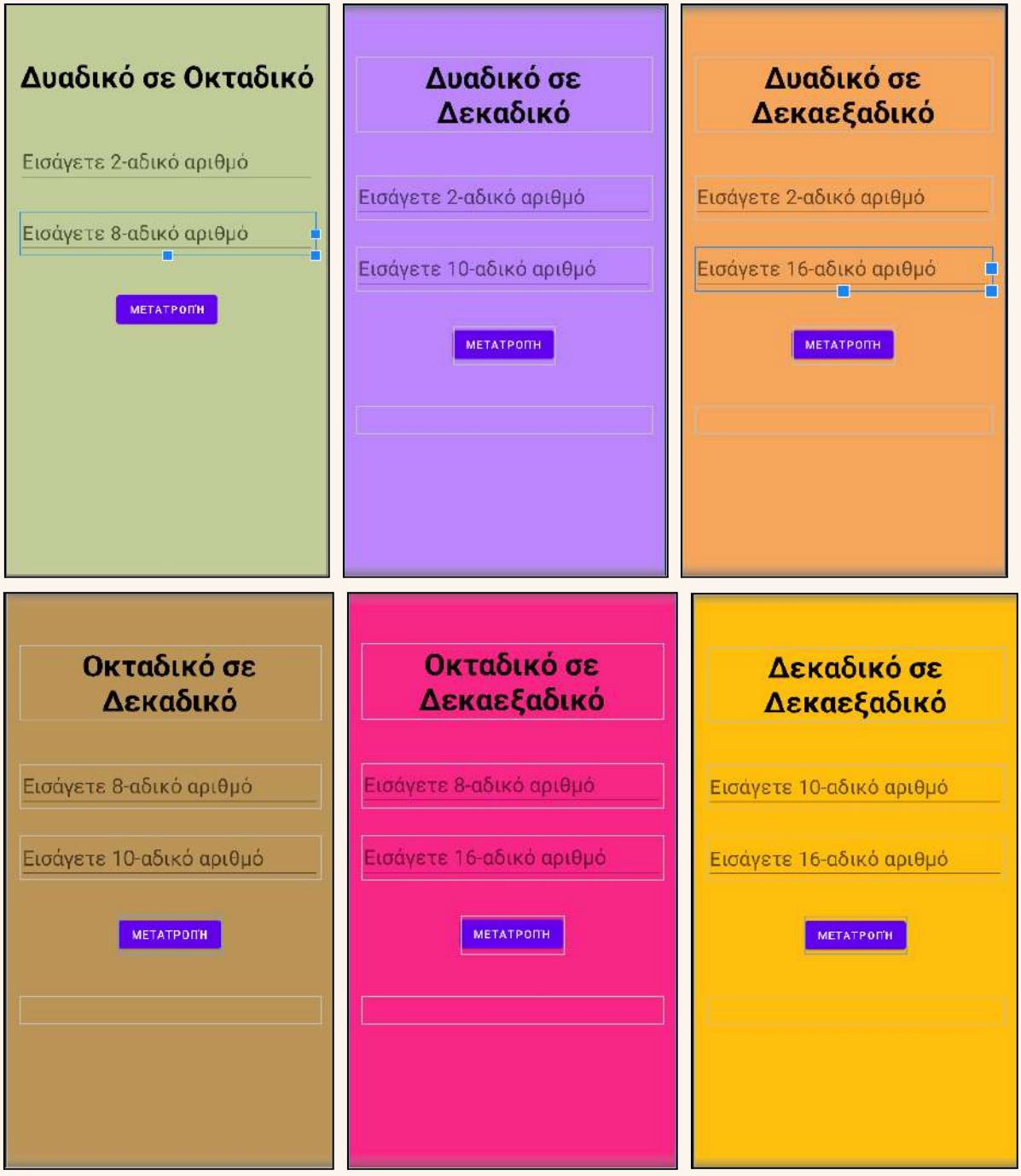
```
octalEditText.getText().toString().trim();
                String hexadecimalString =
hexadecimalEditText.getText().toString().trim();

                // Check if both fields are empty
                if (TextUtils.isEmpty(octalString) &&
TextUtils.isEmpty(hexadecimalString)) {
                    outputTextView.setText("Εισάγετε μια τιμή για
μετατροπή.");
                    return;
                }

                // Check if both fields are filled in
                if (!TextUtils.isEmpty(octalString) &&
!TextUtils.isEmpty(hexadecimalString)) {
                    outputTextView.setText("Εισάγετε μόνο μια τιμή
σε ένα πεδίο.");
                    return;
                }

                // Check if the octal input is valid
                if (!TextUtils.isEmpty(octalString) &&
!isValidOctal(octalString)) {
                    outputTextView.setText("Εισάγετε έναν έγκυρο
οκταδικό αριθμό.");
                    return;
                }

                // Check if the hexadecimal input is valid
                if (!TextUtils.isEmpty(hexadecimalString) &&
!isValidHexadecimal(hexadecimalString)) {
                    outputTextView.setText("Εισάγετε έναν έγκυρο
δεκαεξαδικό αριθμό.");
                    return;
                }

                // Perform the conversion
                if (!TextUtils.isEmpty(octalString)) {
                    boolean isNegative = false;
                    if (octalString.charAt(0) == '-') {
                        isNegative = true;
                        octalString = octalString.substring(1);
                    }
                    String[] parts = octalString.split("\\.");
                    int decimalValue = Integer.parseInt(parts[0],
```

```
8);
                    String hexadecimalValue =
Integer.toHexString(decimalValue).toUpperCase();

                    if (parts.length > 1) {
                        StringBuilder fractionalHexadecimalValue =
new StringBuilder();
                        double fractionalPart = 0.0;
                        for (int i = 0; i < parts[1].length();
i++) {
                            char c = parts[1].charAt(i);
                            int digit = c - '0';
                            fractionalPart += digit * Math.pow(8,
-(i + 1));
                        }
                        while (fractionalPart > 0 &&
fractionalHexadecimalValue.length() < 6) { // Limit the length of
the result
                            fractionalPart *= 16;
                            int digit = (int) fractionalPart;
                            String hexDigit =
Integer.toHexString(digit).toUpperCase();

fractionalHexadecimalValue.append(hexDigit);
                            fractionalPart -= digit;
                        }
                        hexadecimalValue += "." +
fractionalHexadecimalValue.toString();
                    }

                    if (isNegative) {
                        hexadecimalValue = "-" + hexadecimalValue;
                    }

                    outputTextView.setText(hexadecimalValue);
                } else {
                    boolean isNegative = false;
                    if (hexadecimalString.charAt(0) == '-') {
                        isNegative = true;
                        hexadecimalString =
hexadecimalString.substring(1);
                    }
                    int decimalValue =
Integer.parseInt(hexadecimalString, 16);
                    String octalValue =
```

```
Integer.toOctalString(decimalValue);
                    if (isNegative) {
                        octalValue = "-" + octalValue;
                    }
                    outputTextView.setText(octalValue);
                }
            }
        });
    }

    private boolean isValidOctal(String octalString) {
        // Regular expression for an octal string with optional
negative sign and fractional part
        String regex = "-?[0-7]+(\\.[0-7]+)?";
        return octalString.matches(regex);
    }

    private boolean isValidHexadecimal(String hexadecimalString) {
        // Regular expression for a hexadecimal string with
optional negative sign
        String regex = "-?[0-9A-Fa-f]+";
        return hexadecimalString.matches(regex);
    }
}
```

It is obvious that the most difficult part in terms of coding for all the above 6 java files is the needed algorithms that perform the conversion calculations, especially when the input values contain both integer parts as well as fractional parts.

Here are the screenshots of the 6 pairs of generated layouts:




**Step 5:** Create the AndroidManifest.xml

One last step before we finalise the app and proceed in testing is to create the AndroidManifest.xml file in 'manifests' folder

The content code in that file is as follows:

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.NumeralSystemsConverter"
        tools:targetApi="31">
        <activity
            android:name=".DecimalAndHexadecimal"
            android:label="Decimal and Hexadecimal Conversion"
            android:exported="true">
            <intent-filter>
                <action
android:name="com.example.numeralsystemsconverter.DECIMAL_AND_HEXA
DECIMAL" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".OctalAndHexadecimal"
            android:label="Octal and Hexadecimal Conversion"
            android:exported="true">
            <intent-filter>
                <action
android:name="com.example.numeralsystemsconverter.OCTAL_AND_HEXADE
CIMAL" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".OctalAndDecimal"
            android:label="Octal and Decimal Conversion"
            android:exported="true">
            <intent-filter>
                <action
android:name="com.example.numeralsystemsconverter.OCTAL_AND_DECIMA
```

```
L" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".BinaryAndHexadecimal"
            android:label="Binary and Hexadecimal Conversion"
            android:exported="true">
            <intent-filter>
                <action
android:name="com.example.numeralsystemsconverter.BINARY_AND_HEXAD
ECIMAL" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".BinaryAndDecimal"
            android:label="Binary and Decimal Conversion"
            android:exported="true">
            <intent-filter>
                <action
android:name="com.example.numeralsystemsconverter.BINARY_AND_DECIM
AL" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".BinaryAndOctal"
            android:label="Binary and Octal Conversion"
            android:exported="true">
            <intent-filter>
                <action
android:name="com.example.numeralsystemsconverter.BINARY_AND_OCTAL
" />
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity"
            android:label="Main Activity"
            android:exported="true">
```

```
            <intent-filter>
                <action
android:name="com.example.numeralsystemsconverter.MAIN_ACTIVITY"
/>
                <category
android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".intro_activity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>

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

</manifest>
```

This is where we declare what will be the introduction screen of our application as well as we declare all the rest pages/screens of our app

**Step 6**: Test the app

Run the app on an emulator or Android device, and test the conversion functionality by entering valid binary, octal, decimal or hexadecimal numbers in the appropriate EditText views and clicking the "Convert" button. Verify that the converted result is displayed correctly in the "Result" TextView.

Congratulations! You have now created an Android app that can convert any numeral system value to another numeral system using Java.


Binary and Octal Conversion
Δυαδικό σε Οκταδικό
-11.001
Εισάγετε 8-αδικό αριθμό
ΜΕΤΑΤΡΟΠΗ
-3.1
Binary and Decimal Conversion
Δυαδικό σε Δεκαδικό
-101.001
Εισάγετε 10-αδικό αριθμό
ΜΕΤΑΤΡΟΠΗ
-5.125
Binary and Hexadecimal Conversion
Δυαδικό σε Δεκαεξαδικό
-1001.1101
Εισάγετε 16-αδικό αριθμό
ΜΕΤΑΤΡΟΠΗ
-9.D
Octal and Decimal Conversion
Οκταδικό σε Δεκαδικό
-56.75
Εισάγετε 10-αδικό αριθμό
ΜΕΤΑΤΡΟΠΗ
-46.953125
Decimal and Hexadecimal Conversion
Δεκαδικό σε Δεκαεξαδικό
-456.234
Εισάγετε 16-αδικό αριθμό
ΜΕΤΑΤΡΟΠΗ
-1C8.3BE76C8B439
Octal and Hexadecimal Conversion
Οκταδικό σε Δεκαεξαδικό
-343.671
Εισάγετε 16-αδικό αριθμό
ΜΕΤΑΤΡΟΠΗ
-E3.DC8

# GOOD LUCK!