



Developing Mobile Apps

Mohd. Imran Khan



Mohd. Imran Khan

Teacher & App Developer
Alwar, Rajasthan

Jamnalal Bajaj Award 2019

National Teacher Award, 2017

National ICT Award, 2016

Bhamashah Award, 2016

Types of Mobile Apps

Three Types of Apps

```
graph TD; A[Three Types of Apps] --> B[Native Apps]; A --> C[Web Apps]; A --> D[Hybrid App];
```

Native Apps

Created for one specific Platform or Operating System

Web Apps

Responsive Versions of Websites

Hybrid App

Combinations of both Native and Web apps but wrapped within a native app
Ability to have its own icon

Skills Required

Native Apps

Objective-C
Swift
iOS SDK
Java
ADT
.NET(C#)

Hybrid Apps

HTML, CSS,
JavaScript,
Cordova/PhoneGap,
Cross platform Mobile
Development
Frameworks

Web Apps

HTML
CSS
JavaScript
JS frameworks

Uses

Native Apps

Games or consumer-focused apps where performance, graphics and overall user experience are more important

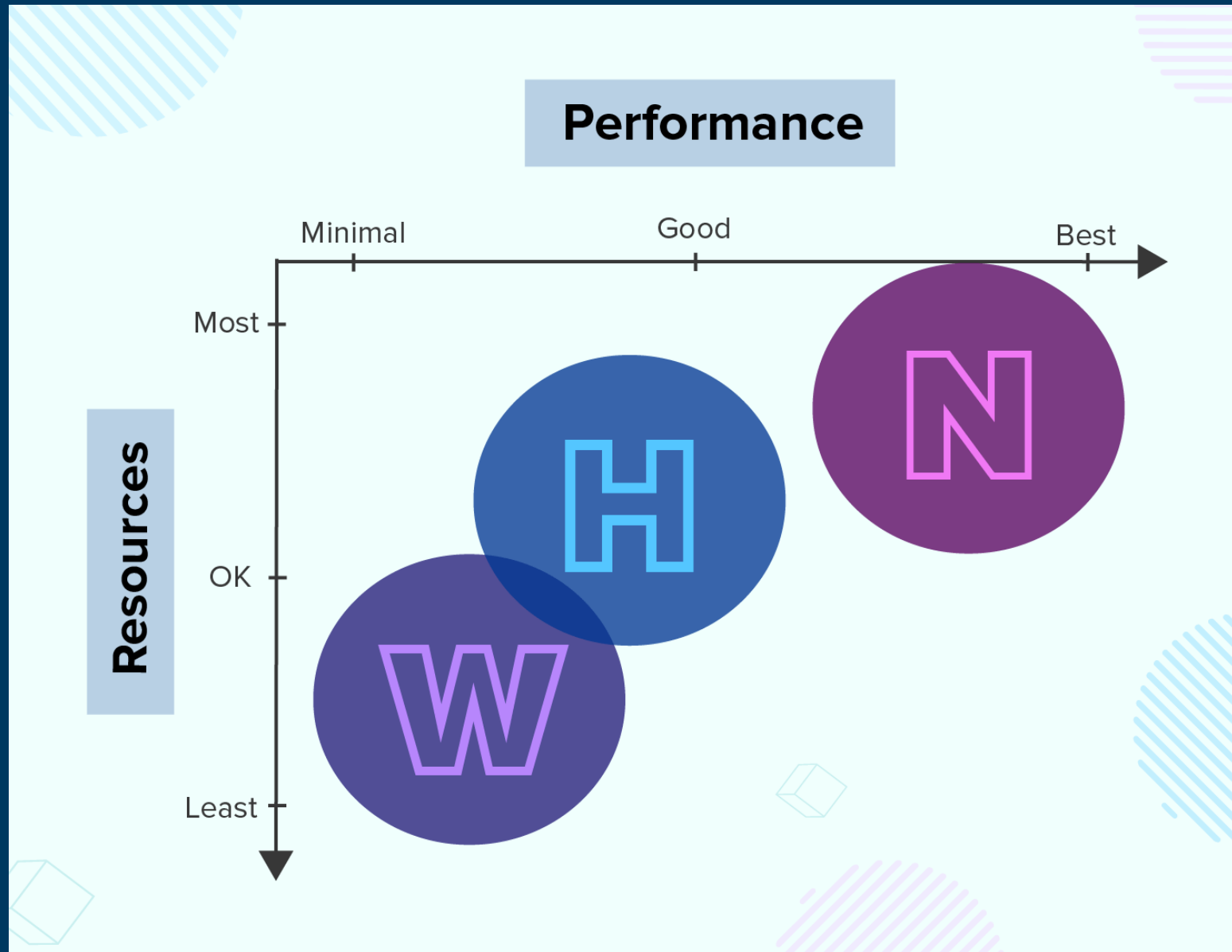
Hybrid Apps

Apps that do not have high performance requirements, but need full device access

Web Apps

No high-performance requirements
No need of push notifications or access to device functionality

How to Choose Just One

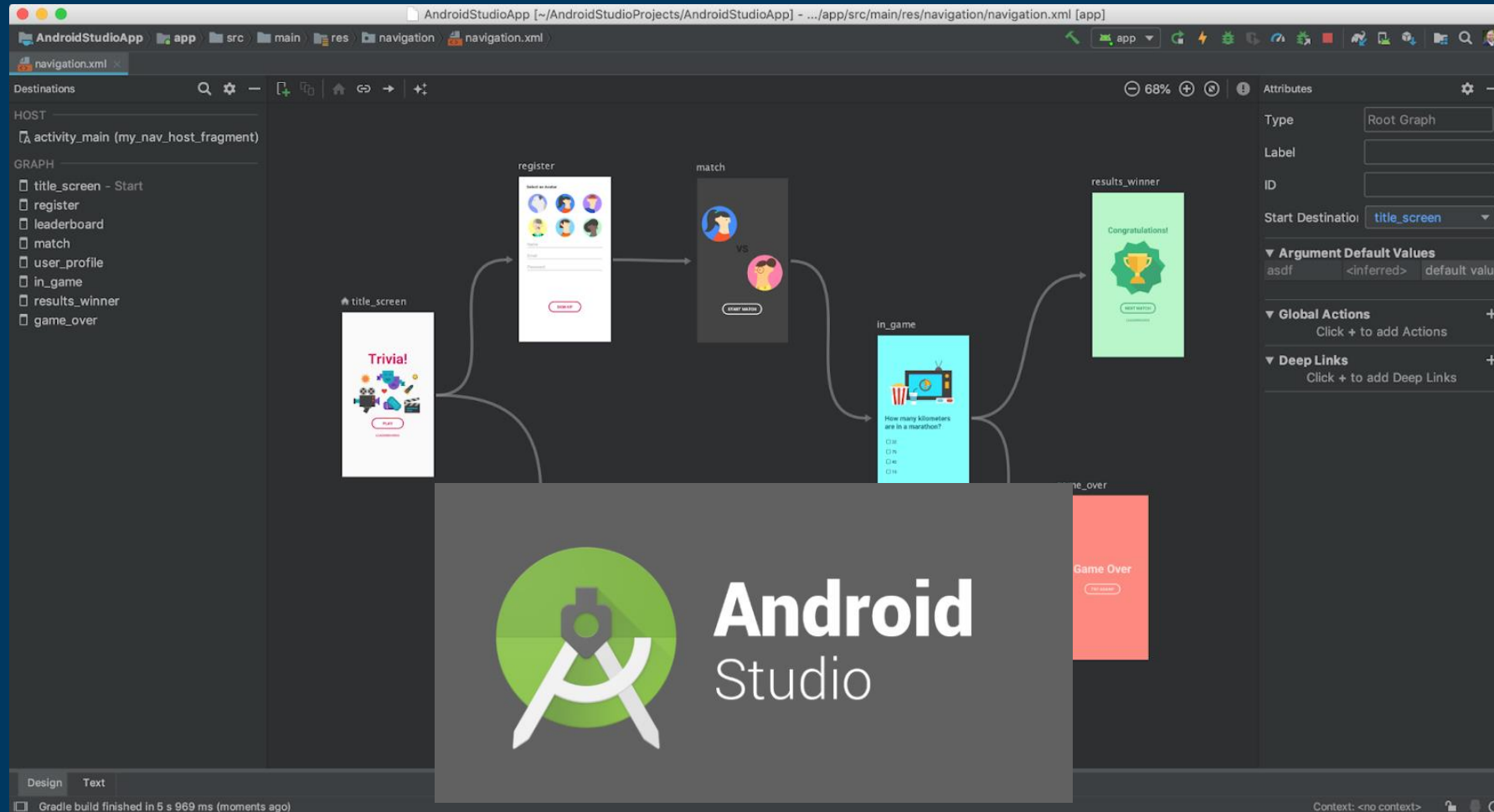


Android Development Tools

Java Development Kit



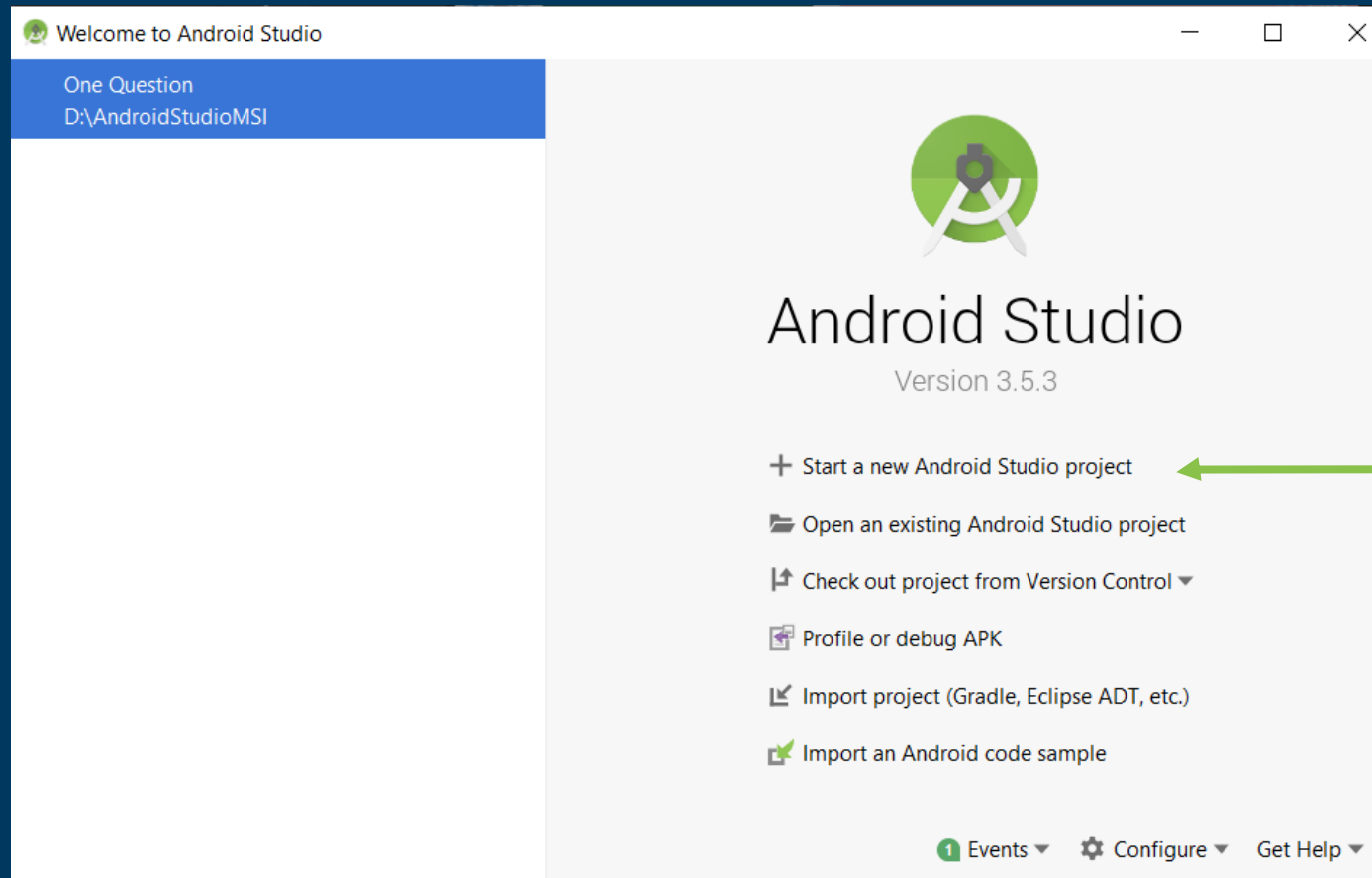
Android Studio and SDK Tools



Creating New Project

1

Select Start a new Android Studio Project on Welcome window



2

Select Activity Type

Create New Project

Choose your project

Phone and Tablet | Wear OS | TV | Android Auto | Android Things

Add No Activity

Basic Activity

Empty Activity

Bottom Navigation Activity

Fragment + ViewModel

Fullscreen Activity

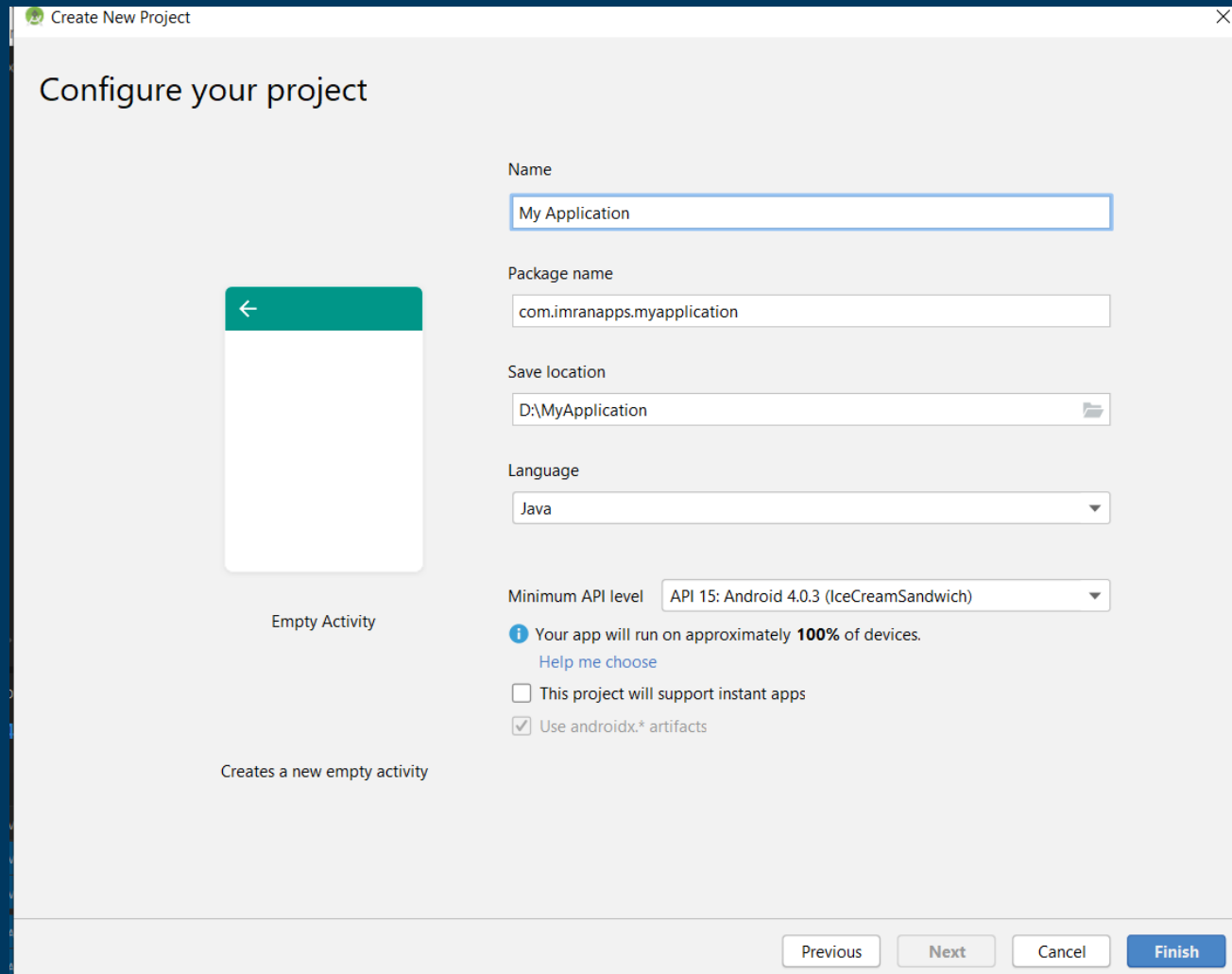
Master/Detail Flow

Navigation Drawer Activity

Empty Activity
Creates a new empty activity

Previous Next Cancel Finish

Type Application Name and Location



Create New Project

Configure your project

Name
My Application

Package name
com.imranapps.myapplication

Save location
D:\MyApplication

Language
Java

Minimum API level
API 15: Android 4.0.3 (IceCreamSandwich)

i Your app will run on approximately **100%** of devices.
[Help me choose](#)

This project will support instant apps

Use androidx.* artifacts

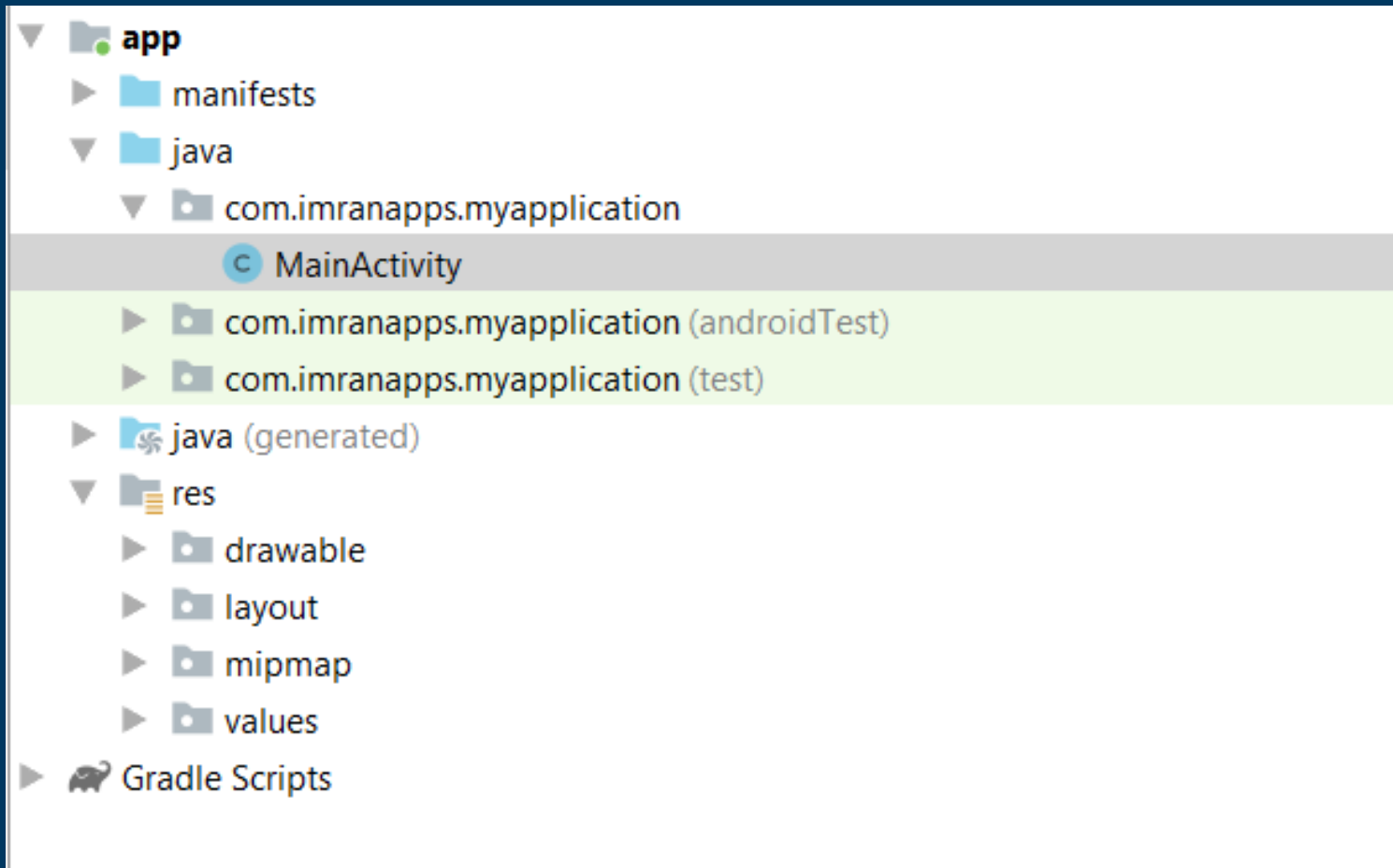
Empty Activity

Creates a new empty activity

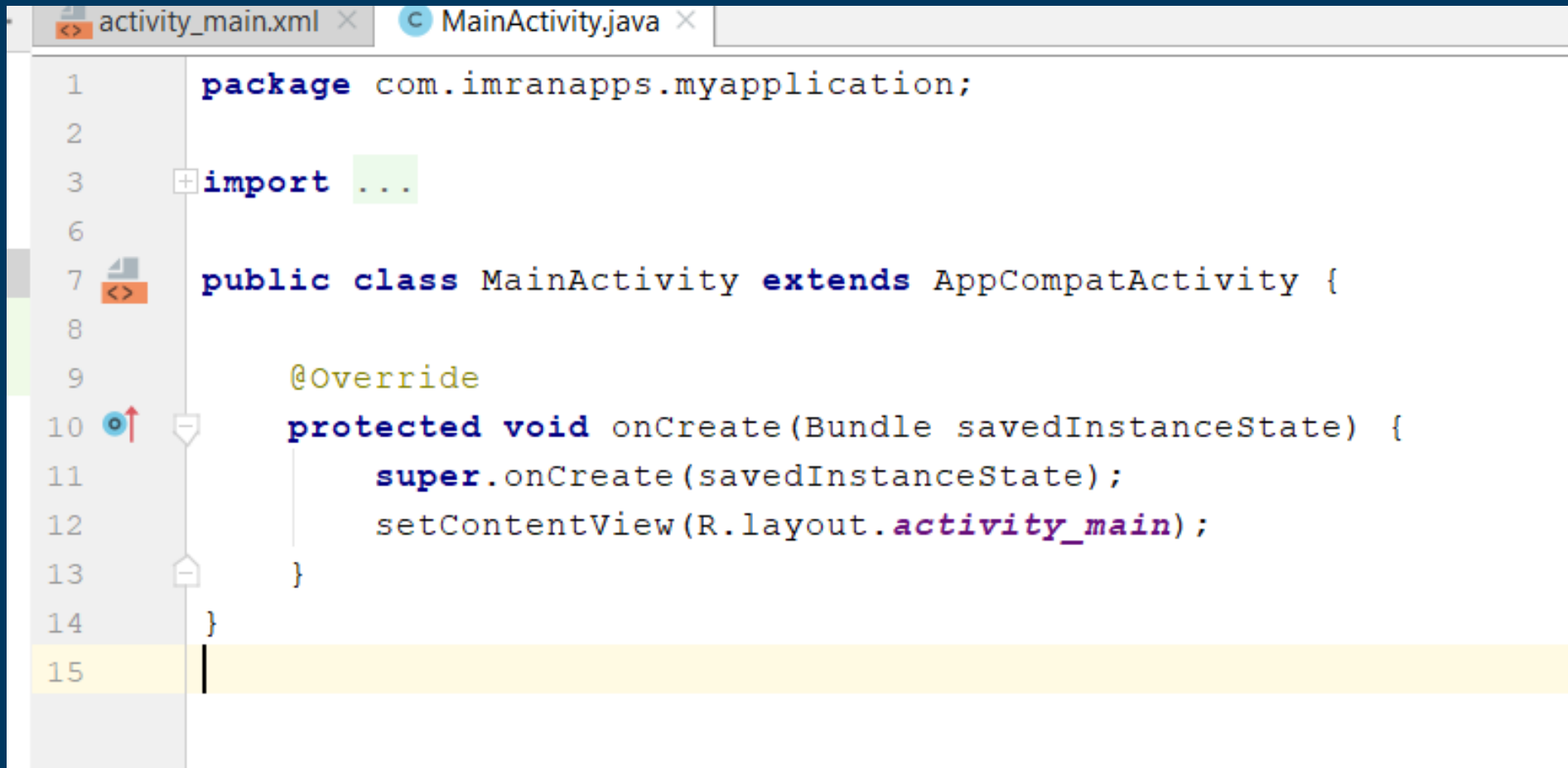
Previous Next Cancel Finish

Understand the Project Structure

Android project consist of manifest, java, res, and Gradle directories.



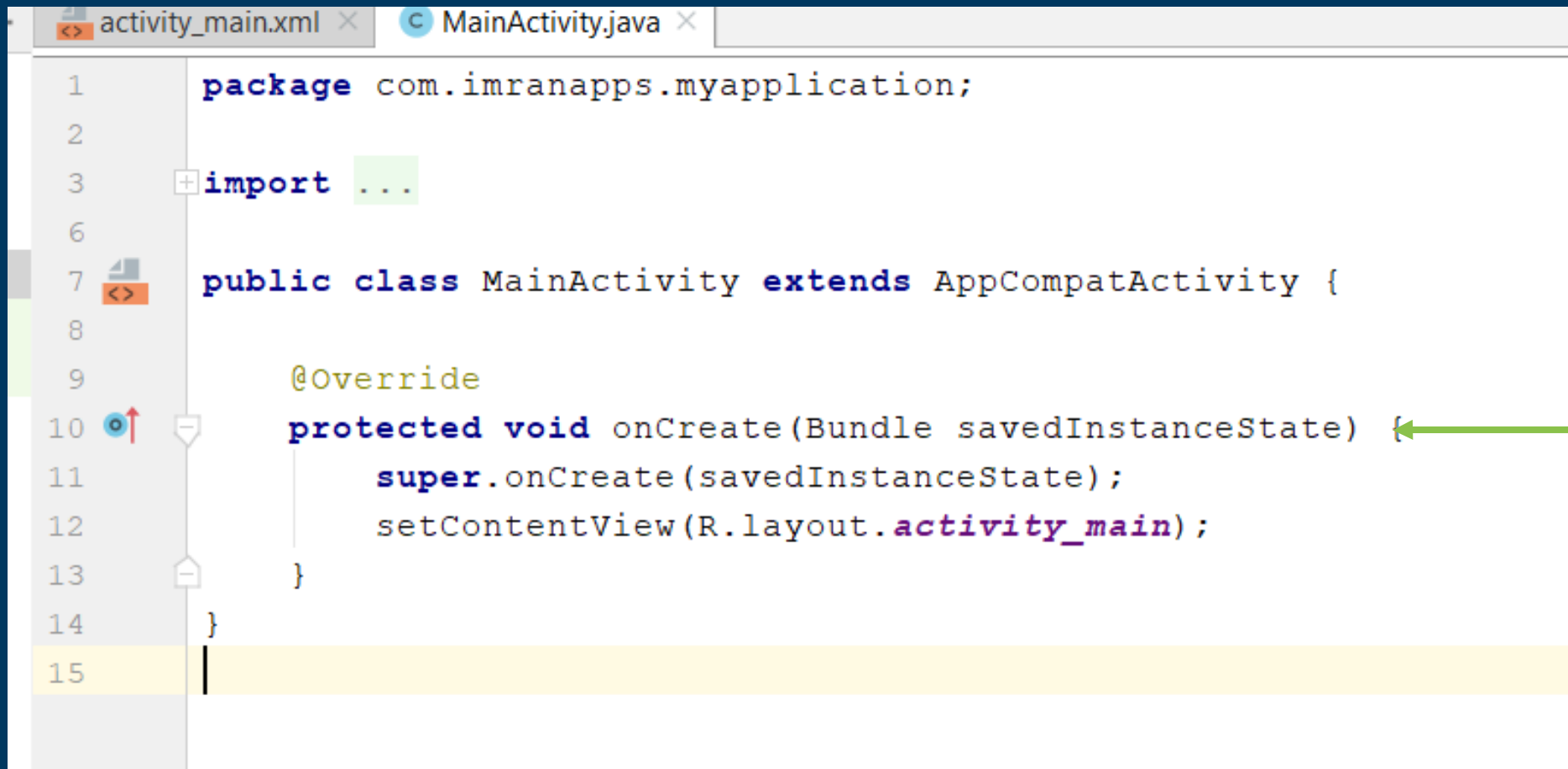
Activity works as a page in application.
Located in java directory.

A screenshot of an IDE window showing the MainActivity.java file. The window title bar includes 'activity_main.xml' and 'MainActivity.java'. The code is as follows:

```
1 package com.imranapps.myapplication;
2
3 import ...
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
15
```

3

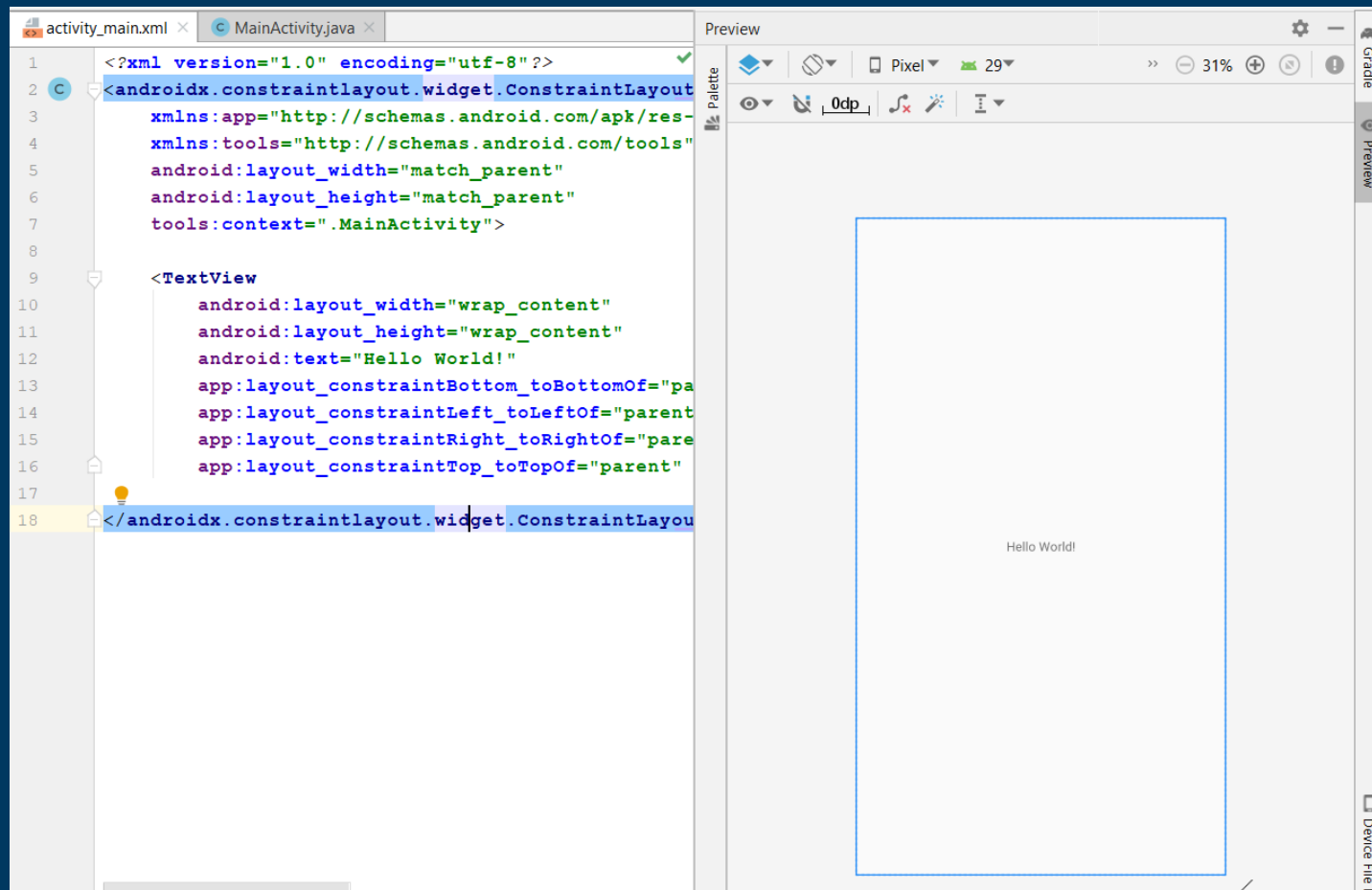
The first method that will be executed when app run is **onCreate()**.



```
1 package com.imranapps.myapplication;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
15
```

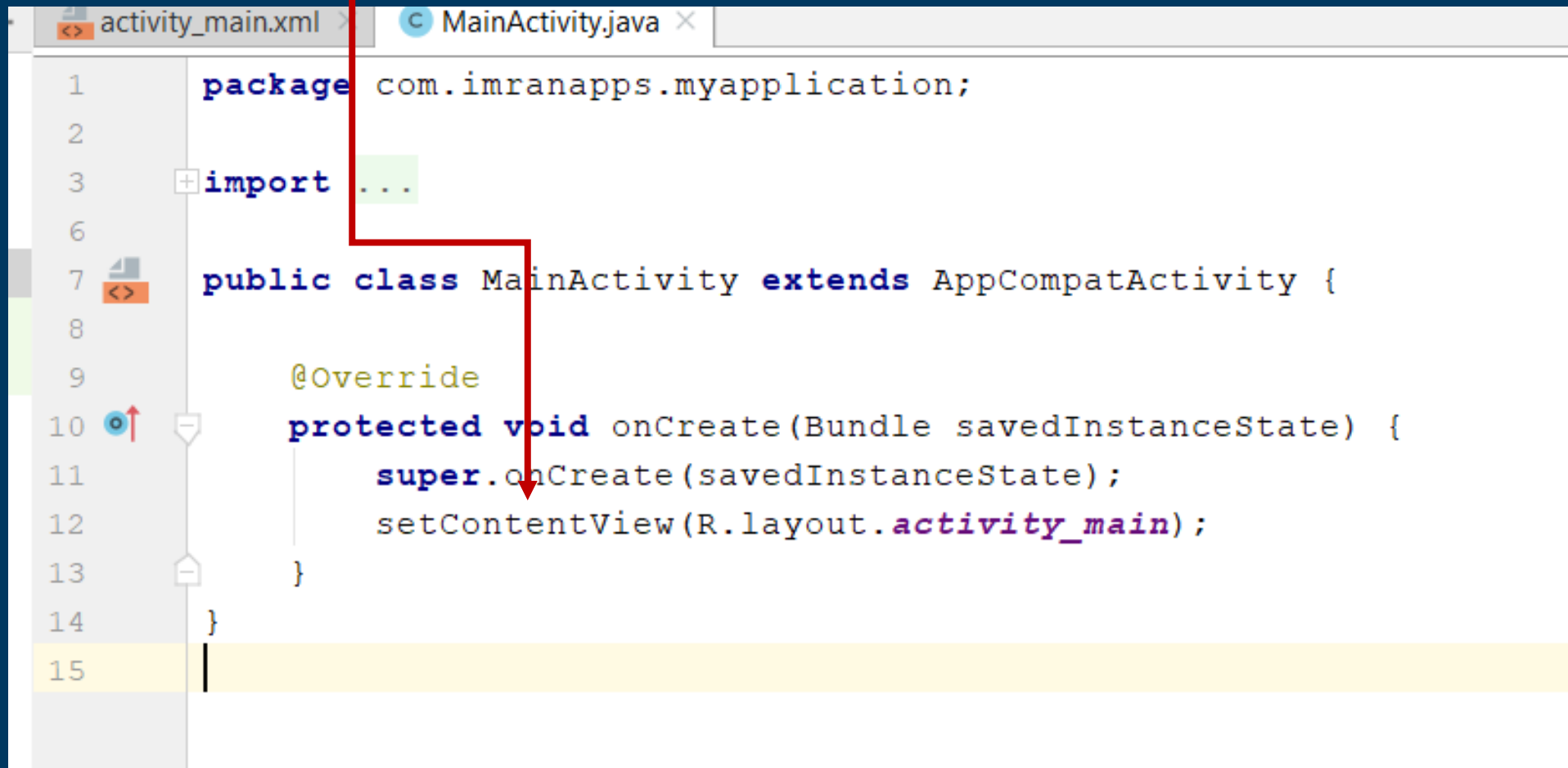
4

Every activity has layout file as its user interface located in **res/layout** directory



5

To connect layout and activity, **setContentView()** must be defined.

A screenshot of an IDE window showing the MainActivity.java file. The code is as follows:

```
1 package com.imranapps.myapplication;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
15
```

A red arrow originates from the text 'setContentView()' in the main text above and points to the corresponding line in the code. The line number 15 is highlighted in yellow.

Every activity created must be registered to **AndroidManifest.xml**.

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

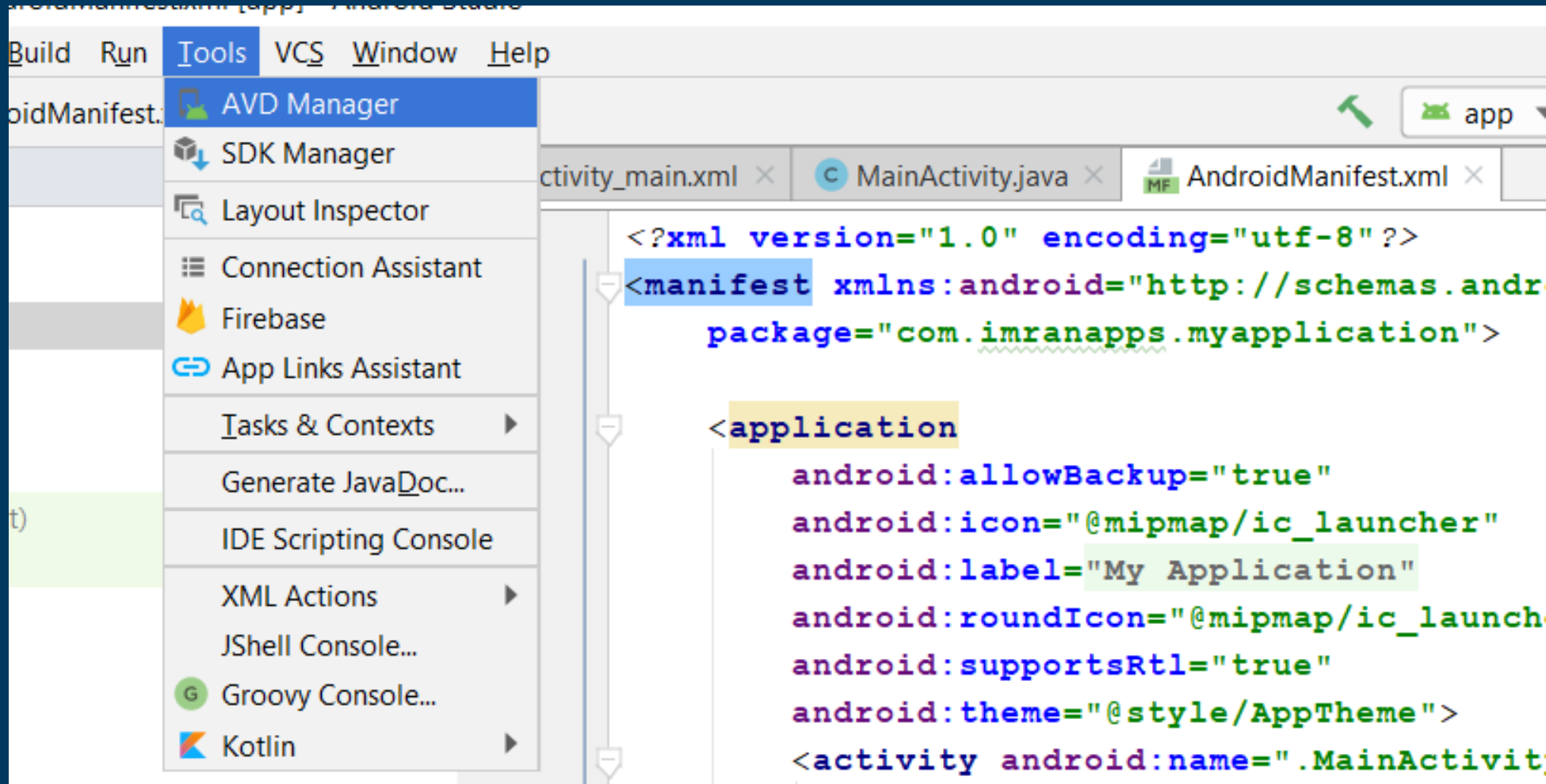
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">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

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

</manifest>
```

Creating Android Emulator

Open **AVD Manager** via Tools > AVD Manager



Select **Create Virtual Device** Button

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 29		1080 × 1920: 420dpi	29	Android 10.0 (Google APIs)	x86	3.5 GB	
	Pixel 2 API 27		1080 × 1920: 420dpi	27	Android 8.1 (Google APIs)	x86	3.6 GB	
	Pixel 3a API 29		1080 × 2220: 440dpi	29	Android 10.0 (Google APIs)	x86	4.6 GB	

+ Create Virtual Device...

Select **Device Type** and Screen Resolution.

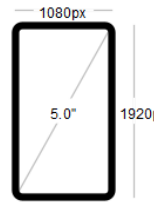
Virtual Device Configuration

Select Hardware
Android Studio

Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
TV	Pixel XL		5.5"	1440x2560	560dpi
Phone	Pixel 3a XL		6.0"	1080x2160	400dpi
Wear OS	Pixel 3a	▶	5.6"	1080x2220	440dpi
Tablet	Pixel 3 XL		6.3"	1440x2960	560dpi
	Pixel 3	▶	5.46"	1080x2160	440dpi
	Pixel 2 XL		5.99"	1440x2880	560dpi
	Pixel 2	▶	5.0"	1080x1920	420dpi
	Pixel	▶	5.0"	1080x1920	420dpi
	Nexus S		4.0"	480x800	hdpi

Pixel 2




Size: large
Ratio: long
Density: 420dpi

New Hardware Profile Import Hardware Profiles Clone Device...

Previous Next Cancel Finish Help

Select Android Version

Virtual Device Configuration


 System Image
Android Studio

Select a system image

Recommended x86 Images Other Images


Release Name	API Level	ABI	Target
R Download	R	x86	Android API R (Google Play)
Q Download	29	x86	Android 10.0 (Google Play)
Pie Download	28	x86	Android 9.0 (Google Play)
Oreo Download	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)


R

 API Level
R
Android
Google Inc.
System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

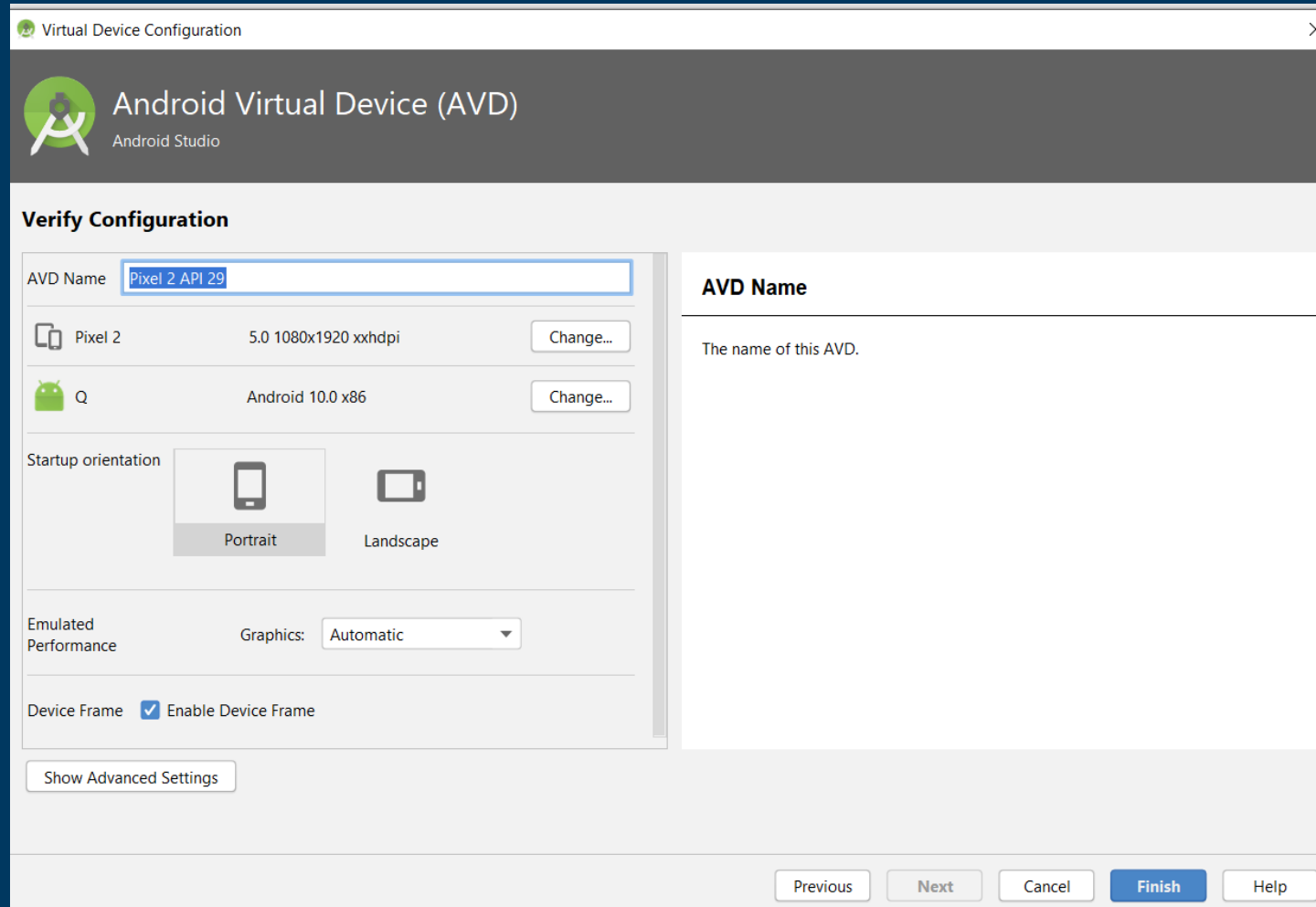
Questions on API level?
See the [API level distribution chart](#)



 A system image must be selected to continue.

Previous Next **Cancel** Finish Help

Type Emulator Name



Select the emulator name and click **Launch icon or Play Button**

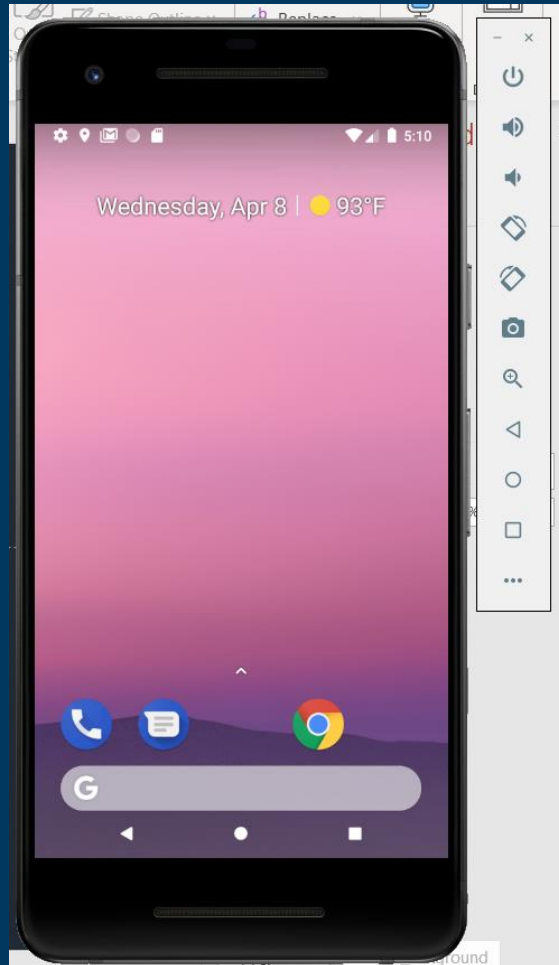
Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5X API 29		1080 × 1920: 420dpi	29	Android 10.0 (Google APIs)	x86	3.5 GB	
	Pixel 2 API 27		1080 × 1920: 420dpi	27	Android 8.1 (Google APIs)	x86	3.6 GB	
	Pixel 2 API 29		1080 × 1920: 420dpi	29	Android 10.0 (Google APIs)	x86	513 MB	
	Pixel 3a API 29		1080 × 2220: 440dpi	29	Android 10.0 (Google APIs)	x86	4.6 GB	

+ Create Virtual Device...

6

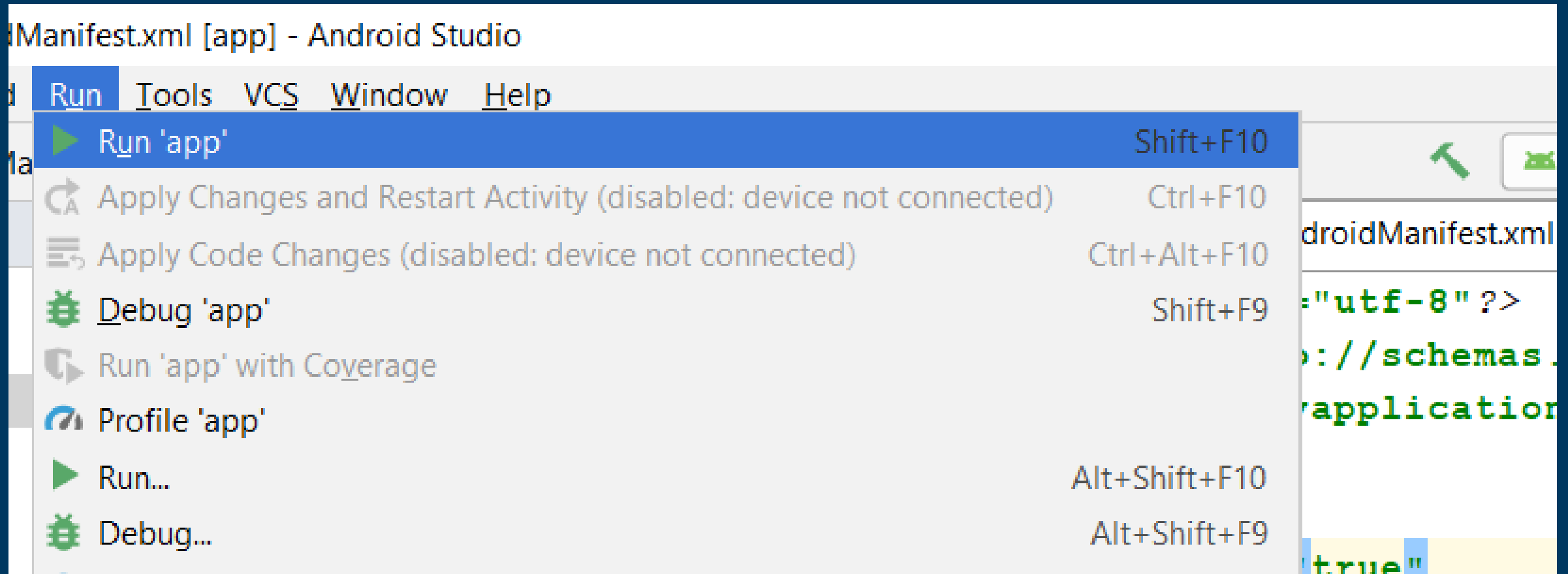
Finally **Emulator** will Start



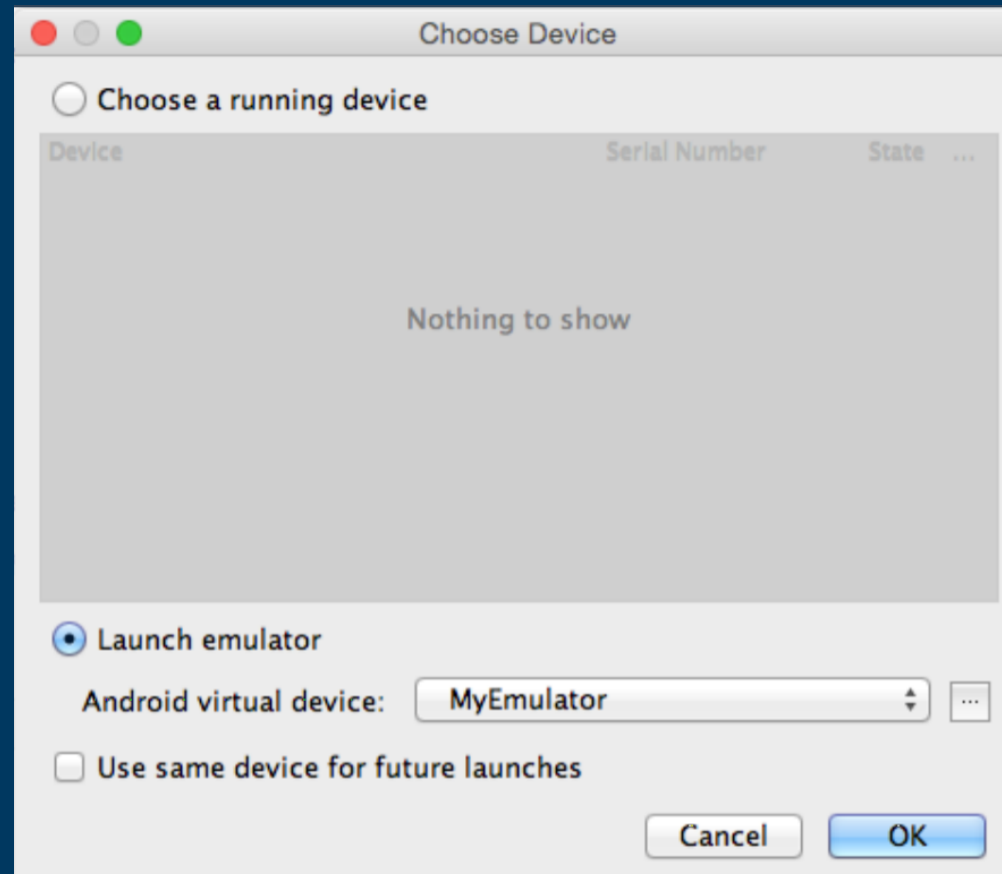
Running Android Project on Emulator

1

Select Run > **Run 'app'**.



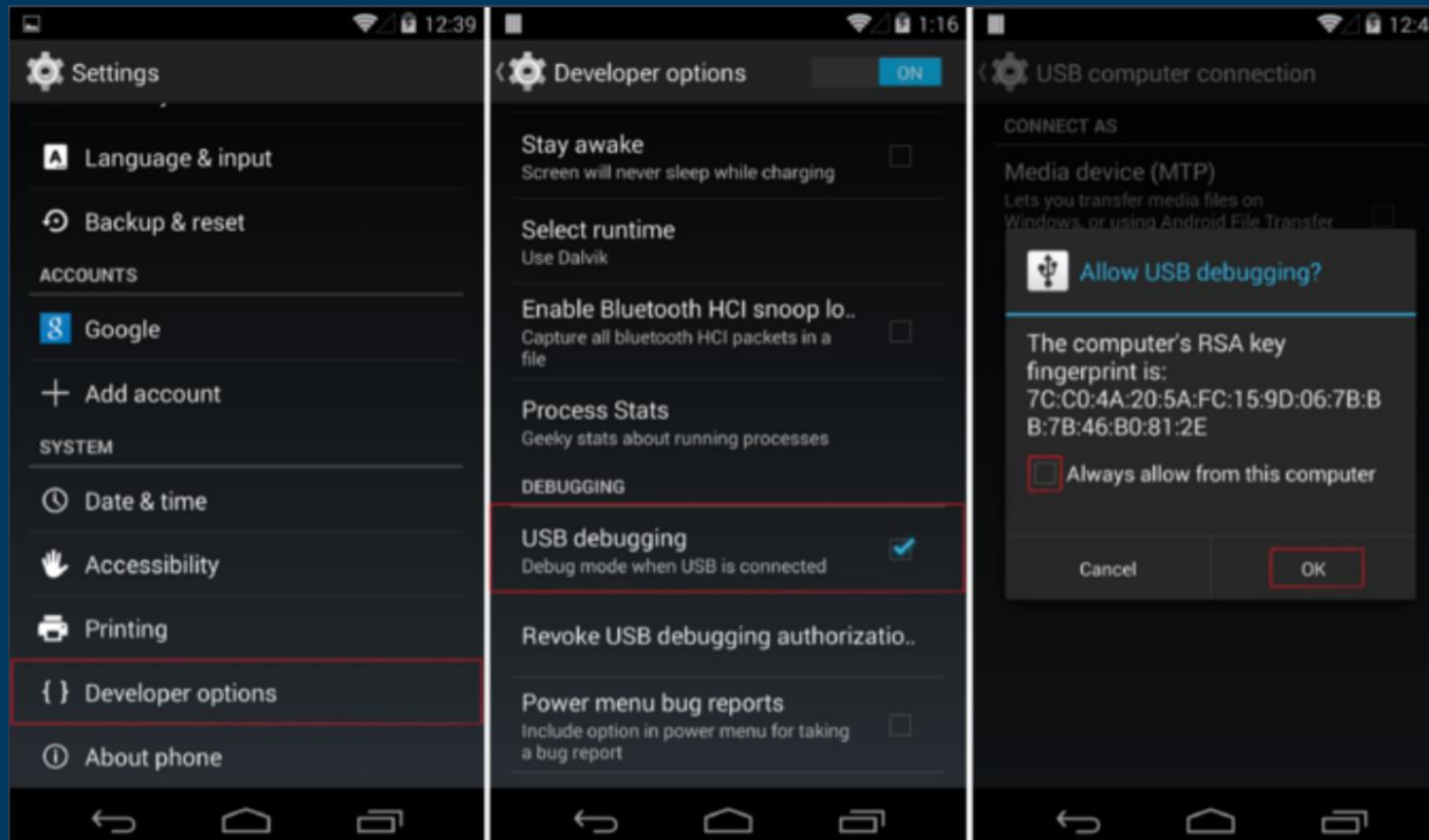
Select **Launch emulator** and emulator name or select **Choose a running device** if you have running emulator



Running Android Project on Android Device

1

On Android device, select
Settings > Developer Options.
Enable USB Debugging.



Connect Android device to the computer via **USB cable**



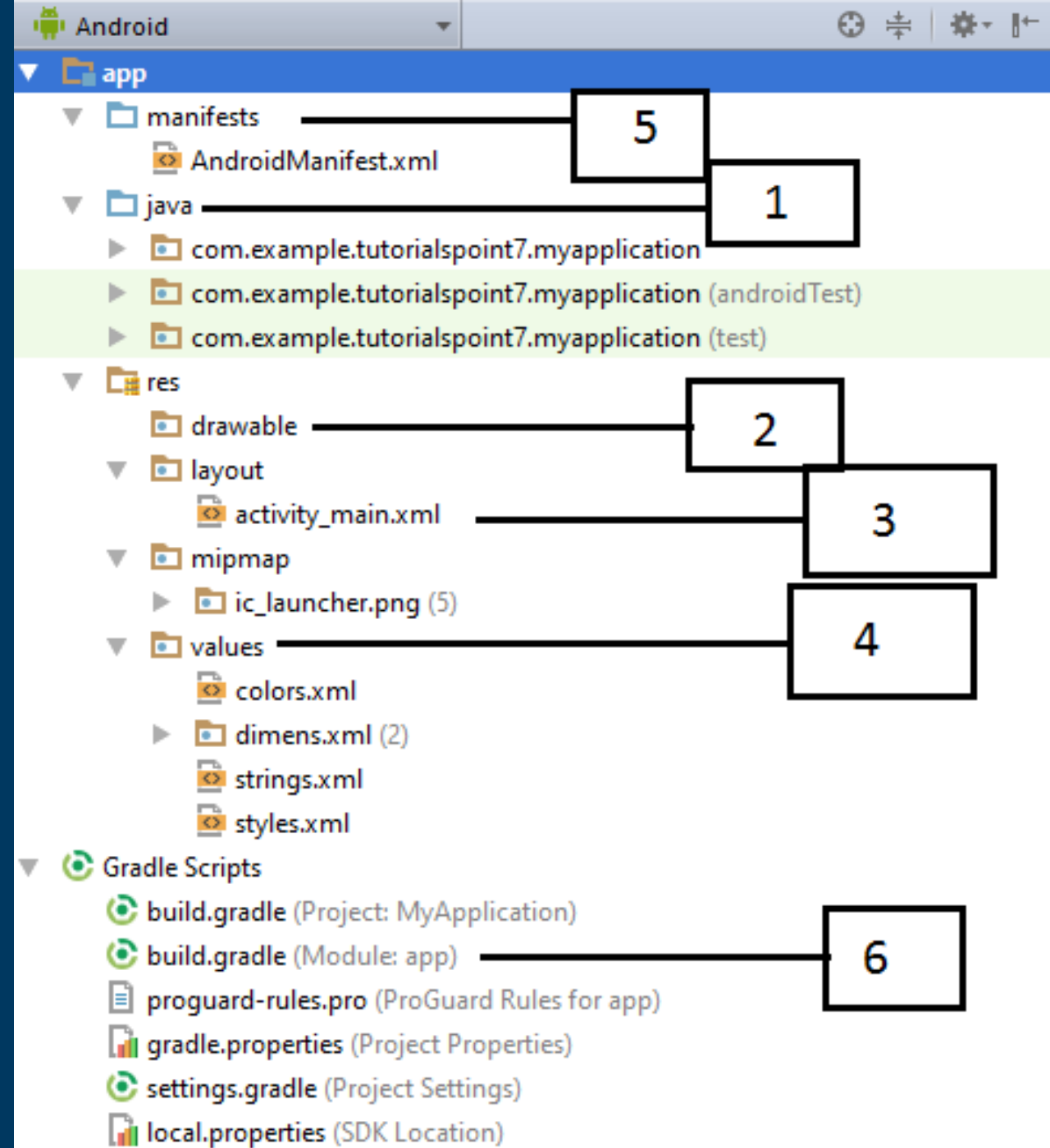
**Build your project on
your computer**



**Test it in real-time on
your device**

Anatomy of Android Application

- 1 Java
- 2 res/drawable-hdpi
- 3 res/layout
- 4 res/values
- 5 AndroidManifest.xml
- 6 build.gradle



Directory & Resource Type

- 1 anim/
- 2 color/
- 3 drawable/
- 4 layout/
- 5 menu/
- 6 raw/
- 7 values/
- 8 xml/

Common Layouts

Linear Layout



Relative Layout



Web View

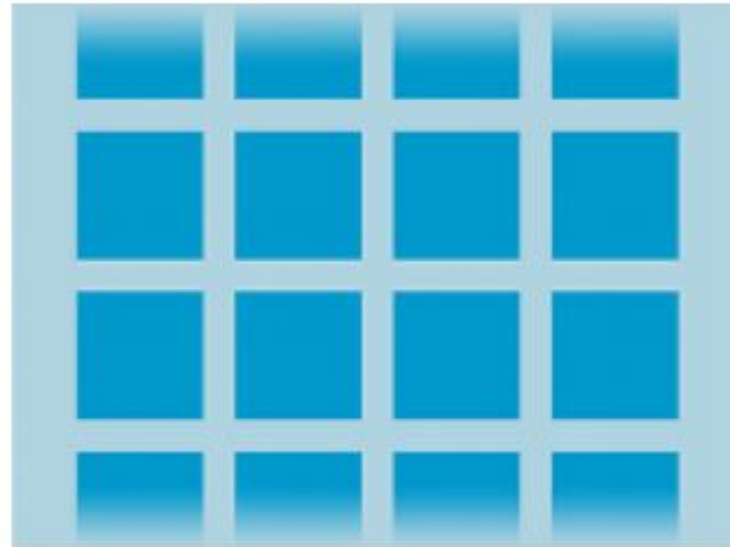


Layouts with an **Adapter**

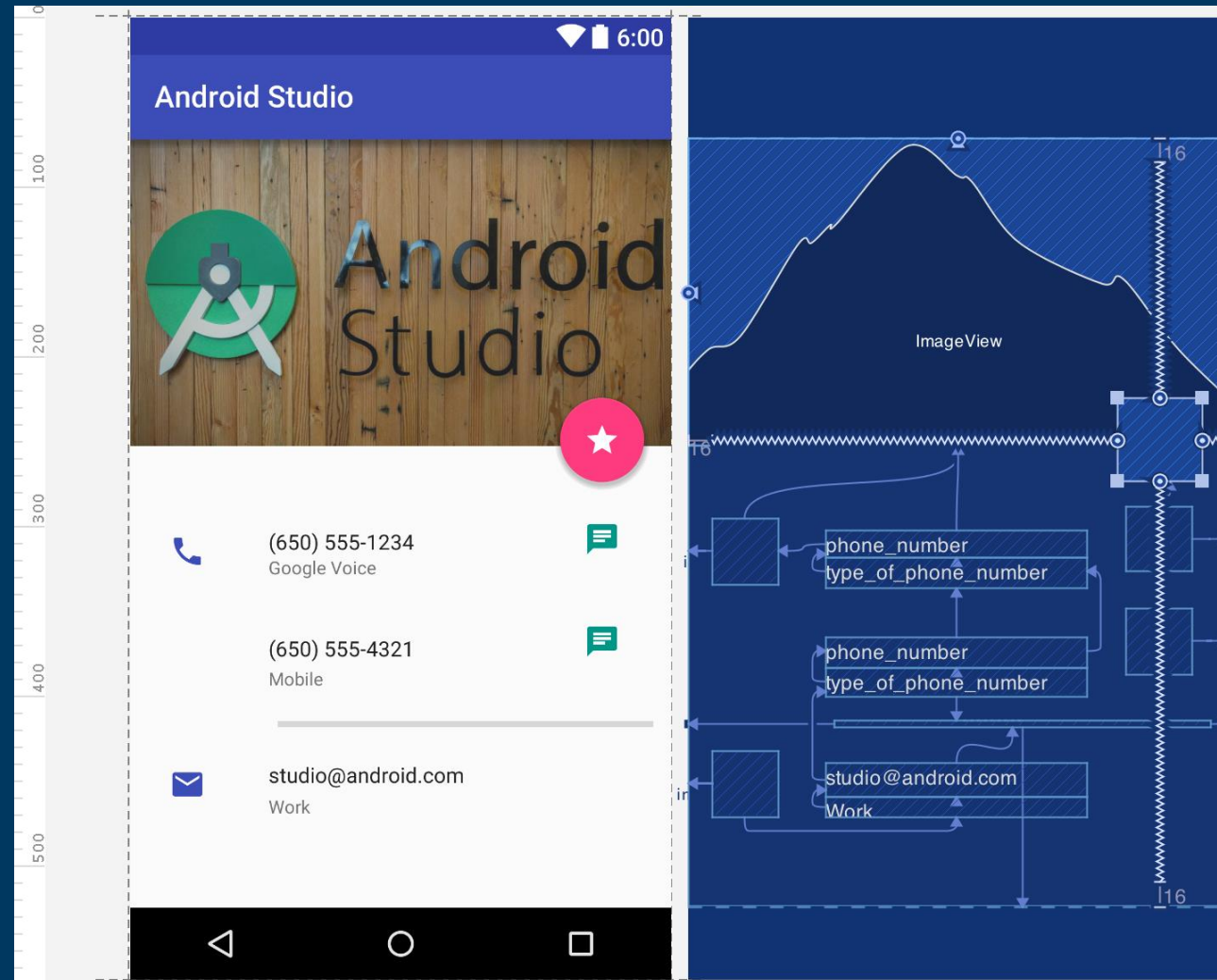
List View



Grid View



Constraint Layout



Android **Permissions**

Protect the privacy of an Android user

Types of Permissions

```
graph TD; A[Types of Permissions] --> B[Normal Permission]; A --> C[Dangerous Permission];
```

Normal Permission

Dangerous Permission

```
<!--Normal Permissions-->
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>

<!--Dangerous Permission-->
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Different Pixel densities

1x

1.5x

2x

3x

4x

BASELINE



MDPI

~160 DPI



HDPI

~240 DPI



XHDPI

~320 DPI



XXHDPI

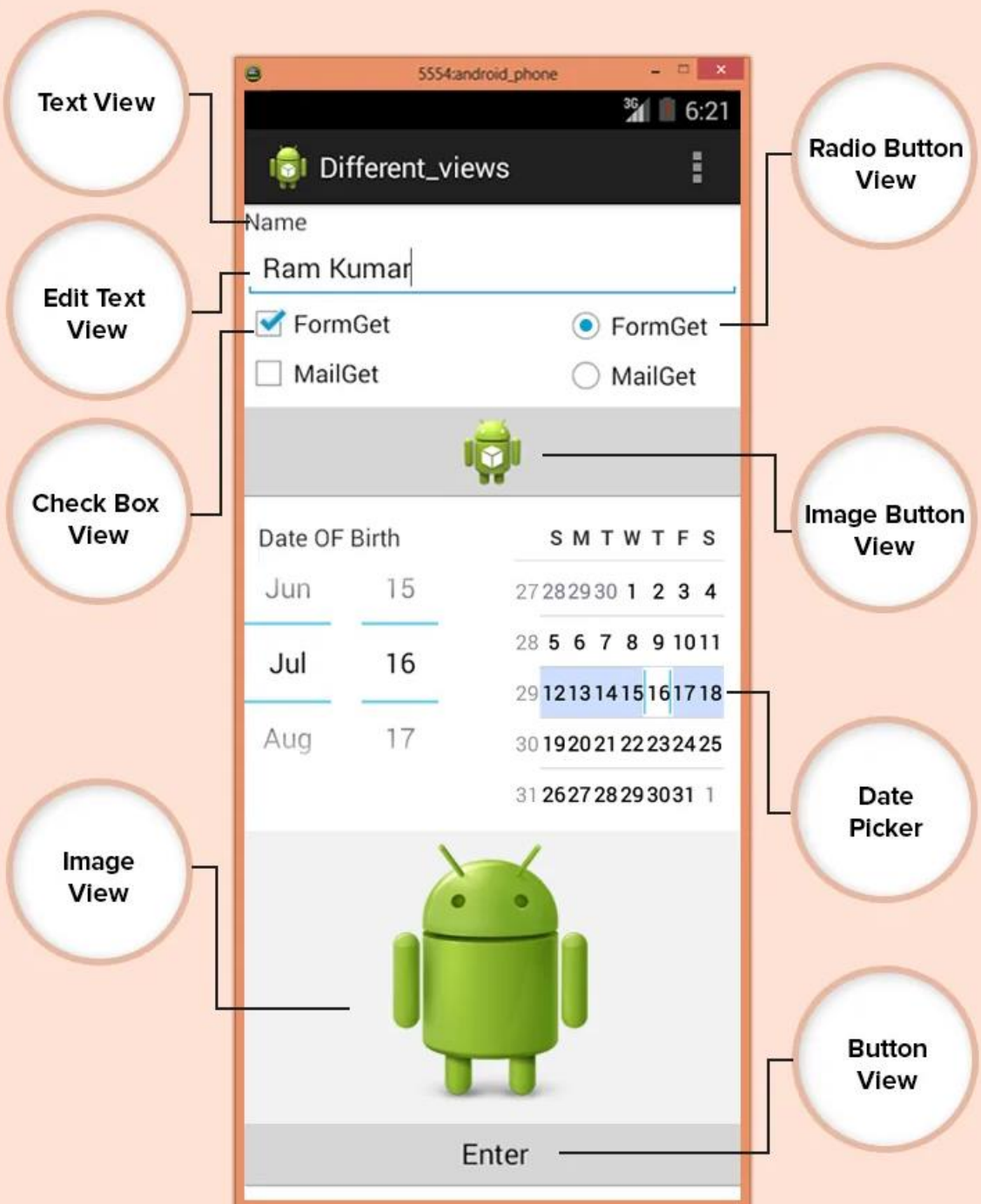
~480 DPI



XXXHDPI

~640 DPI

Most Used Android View Classes

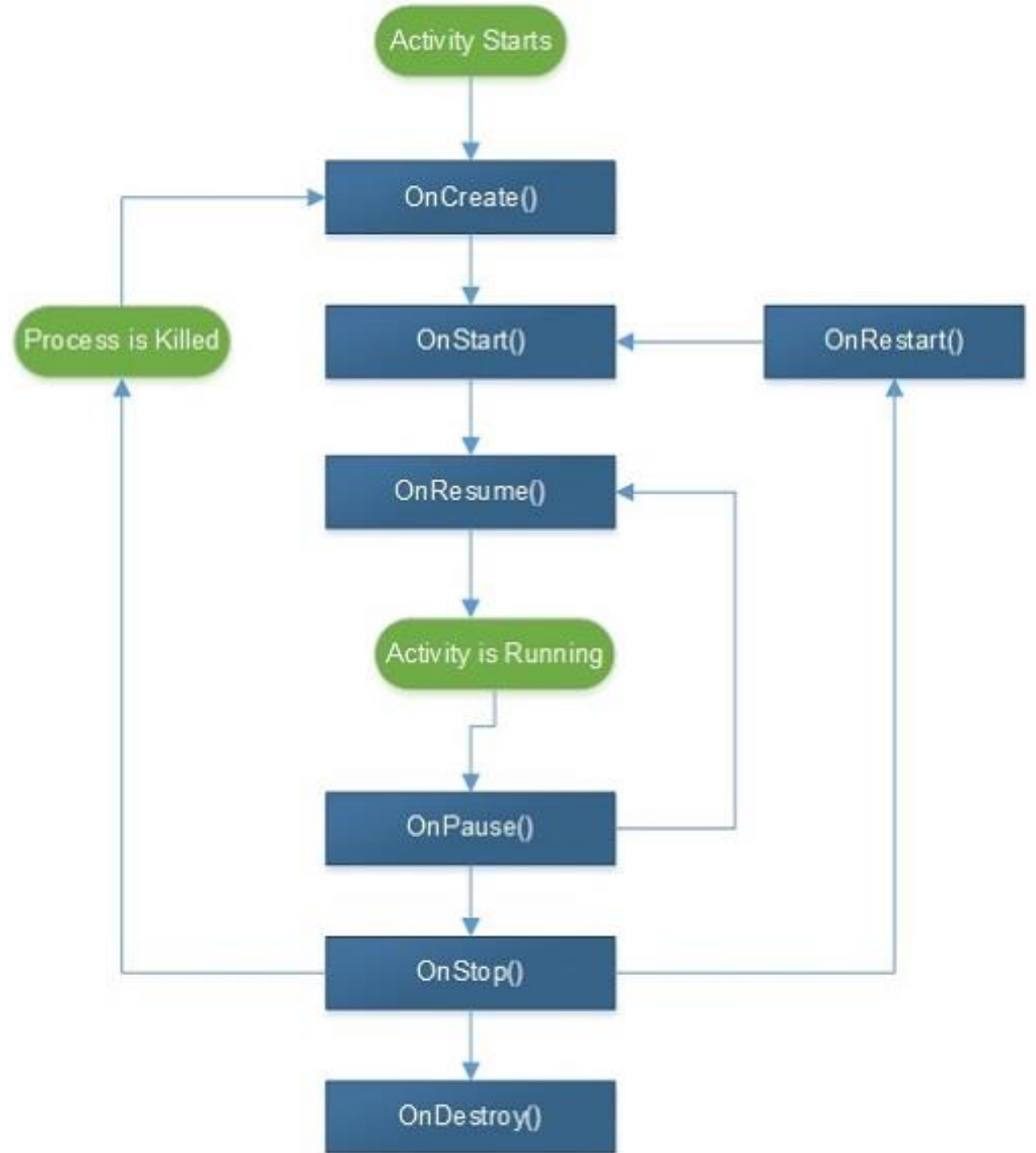


- 1 Text View
- 2 EditText
- 3 Button
- 4 ImageView
- 5 ImageButton
- 6 CheckBox
- 7 Radio button
- 8 RadioGroup
- 9 Spinner

Layout Attributes

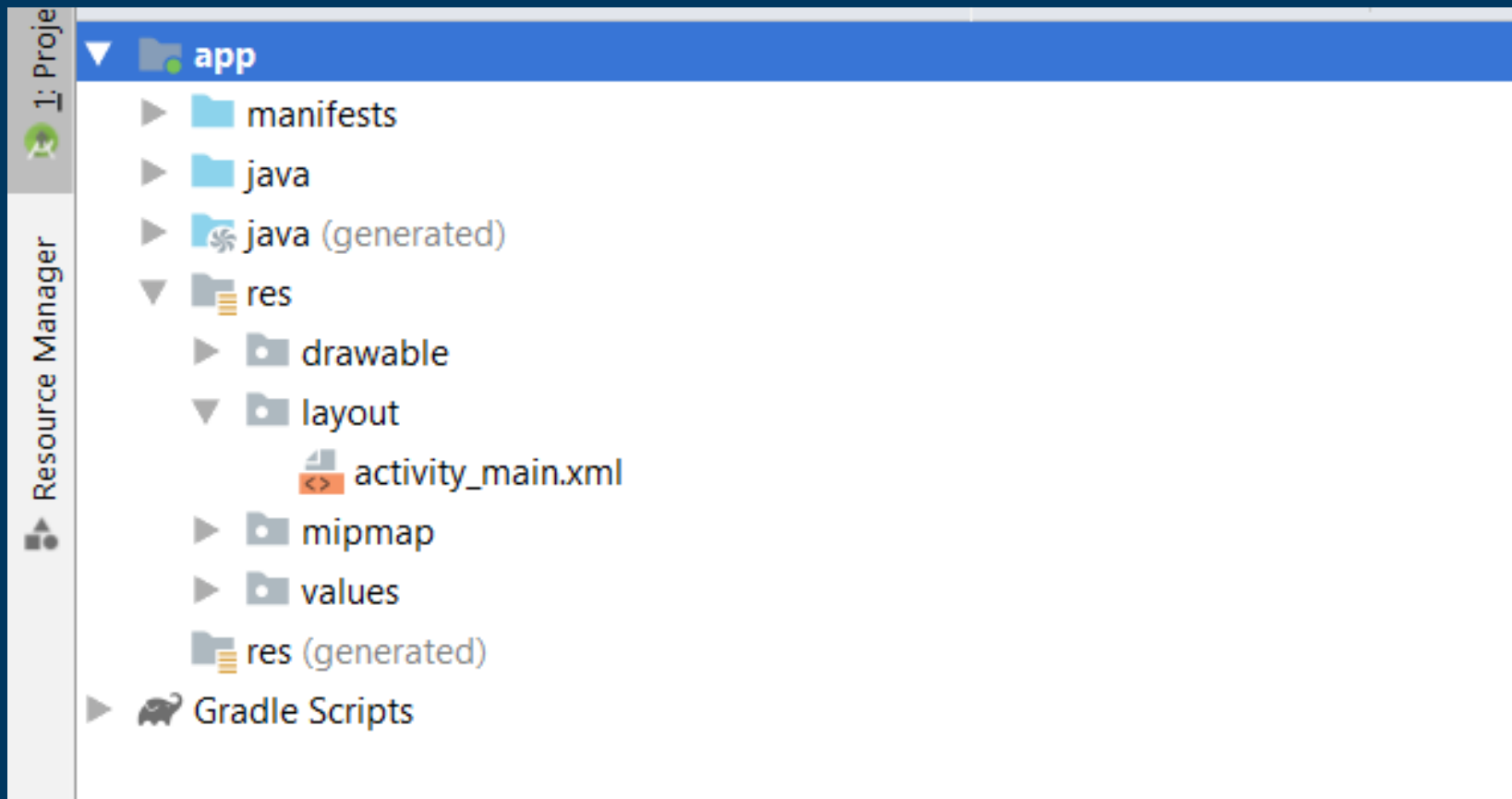
- 1 android:id
- 2 android:layout_width
- 3 android:layout_height
- 4 android:layout_margin
- 5 android:layout_gravity
- 6 android:layout_weight
- 7 android:layout_x
- 8 android:layout_y
- 9 android:padding

Activity Lifecycle



Creating User Interface

Open your xml **layout file** in layout directory



Add widget like Text , Button , List and Configure Widget Properties

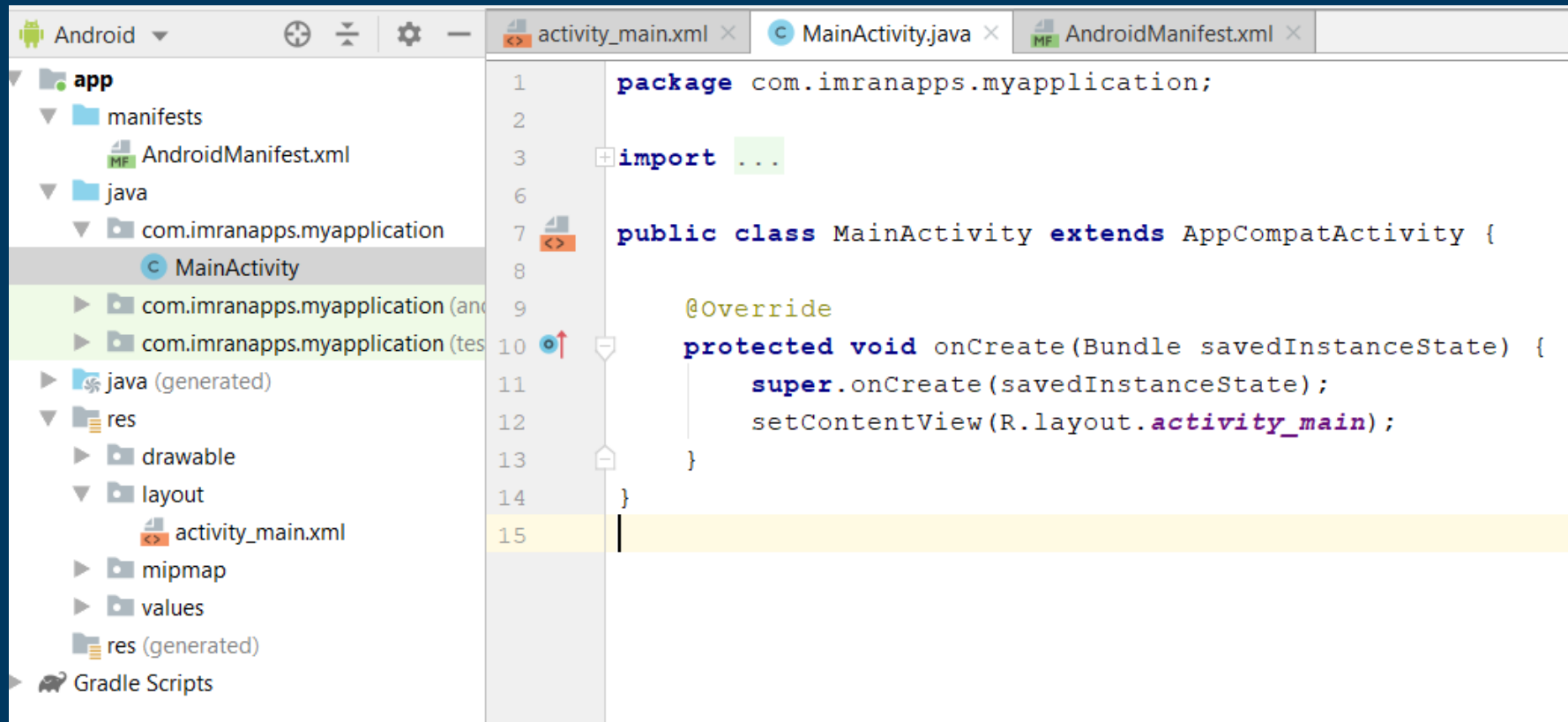
The screenshot displays the Android Studio interface with the XML editor on the left and the Preview window on the right. The XML code defines a `ConstraintLayout` containing a `TextView` widget. The `TextView` is configured with the following properties:

- `android:layout_width="wrap_content"`
- `android:layout_height="wrap_content"`
- `android:text="Hello World!"`
- `app:layout_constraintBottom_toBottomOf="parent"`
- `app:layout_constraintLeft_toLeftOf="parent"`
- `app:layout_constraintRight_toRightOf="parent"`
- `android:textSize="30dp"`
- `app:layout_constraintTop_toTopOf="parent" />`

The Preview window shows a visual representation of the layout, displaying a large rectangular box with the text "Hello World!" centered inside. The interface includes a Palette on the left, a toolbar with various drawing tools, and a status bar at the bottom showing "Pixel" and "29".

Manipulating Widget

Open activity file that use **activity_main.xml** layout in java directory.



```
1 package com.imranapps.myapplication;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
15
```

Create objects of the widgets in activity file that you want to manipulate.

```
public class MainActivity extends ActionBarActivity {  
    TextView txtName;  
    Button btnProcess;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

Connect the objects with widget id in xml layout inside onCreate() method

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

    txtName = (TextView) findViewById(R.id.textView);
    btnProcess = (Button) findViewById(R.id.button);
}
}
```

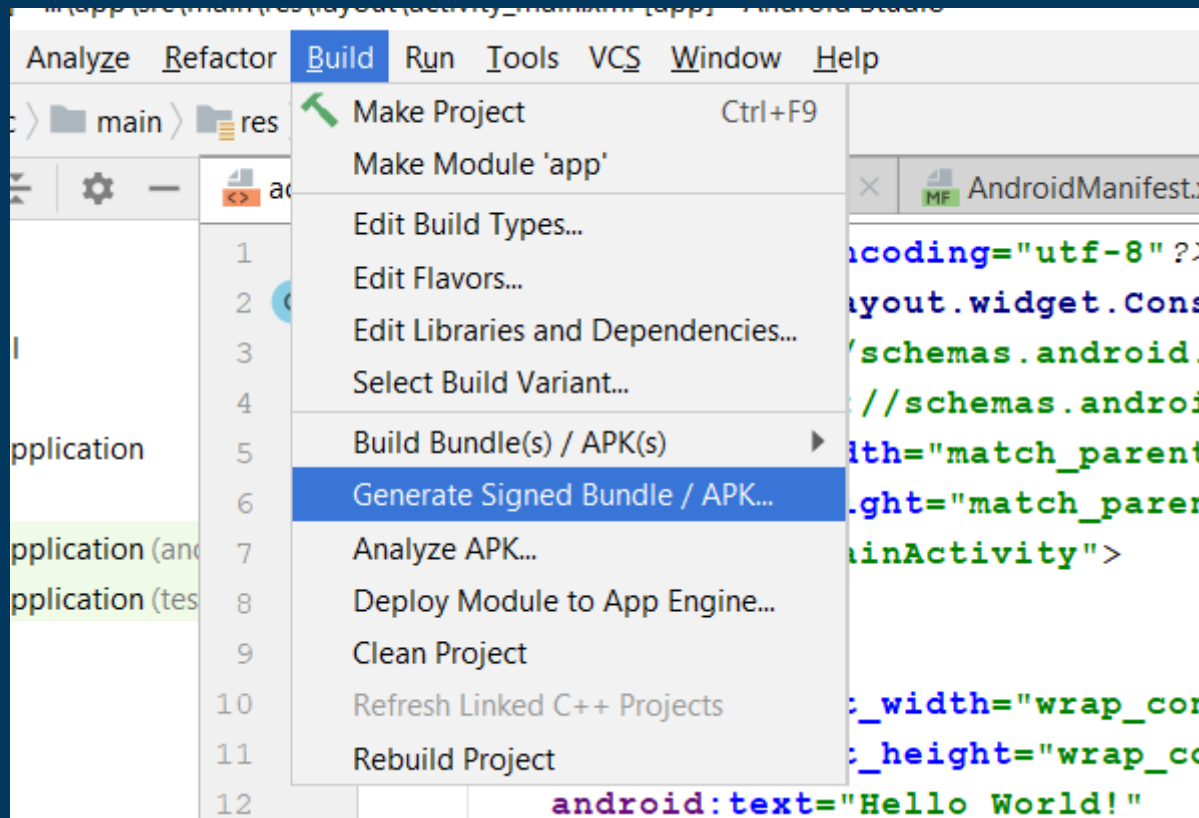
Add **event handling** to button object.

```
btnProcess.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
  
    }  
});  
}
```

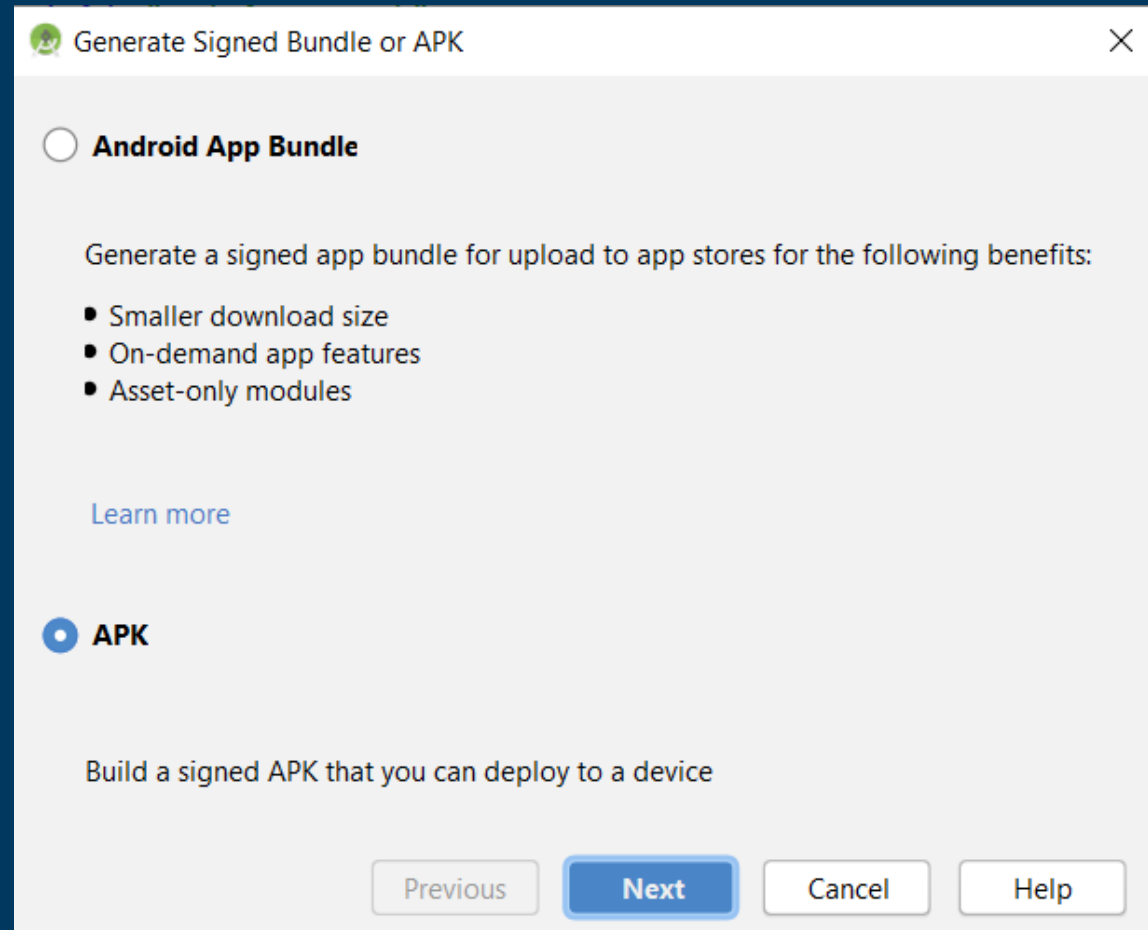
Building an **APK File**

1

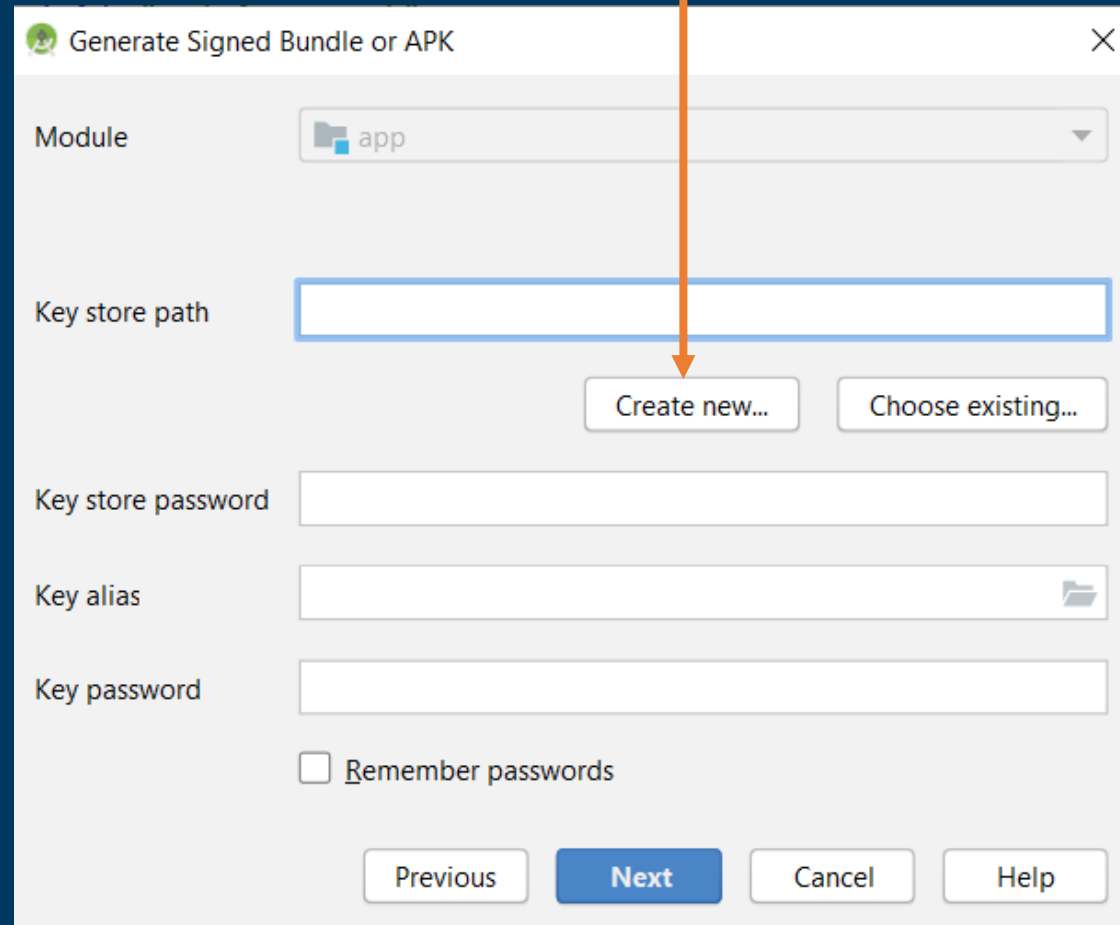
On Build, select
Generate Signed Bundle/APK.



Choose Android App Bundle or **APK** and Click Next



Now you will need to create **KeyStore path**. Click on **Create new**.



The screenshot shows the 'Generate Signed Bundle or APK' dialog box. The 'Module' dropdown is set to 'app'. The 'Key store path' field is empty and highlighted with a blue border. Below it are two buttons: 'Create new...' and 'Choose existing...'. The 'Key store password', 'Key alias', and 'Key password' fields are also empty. There is a checkbox for 'Remember passwords' which is unchecked. At the bottom, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Help'. The 'Next' button is highlighted in blue.

4

Now locate **key store path** in your system where you want to save **jks file** of your project. Fill the other details and click OK.

New Key Store

Key store path: D:\MyApplication\appstorekeyjks

Password: ... Confirm: ...

Key

Alias: key0

Password: ... Confirm: ...

Validity (years): 25

Certificate

First and Last Name:

Organizational Unit:

Organization:

City or Locality:

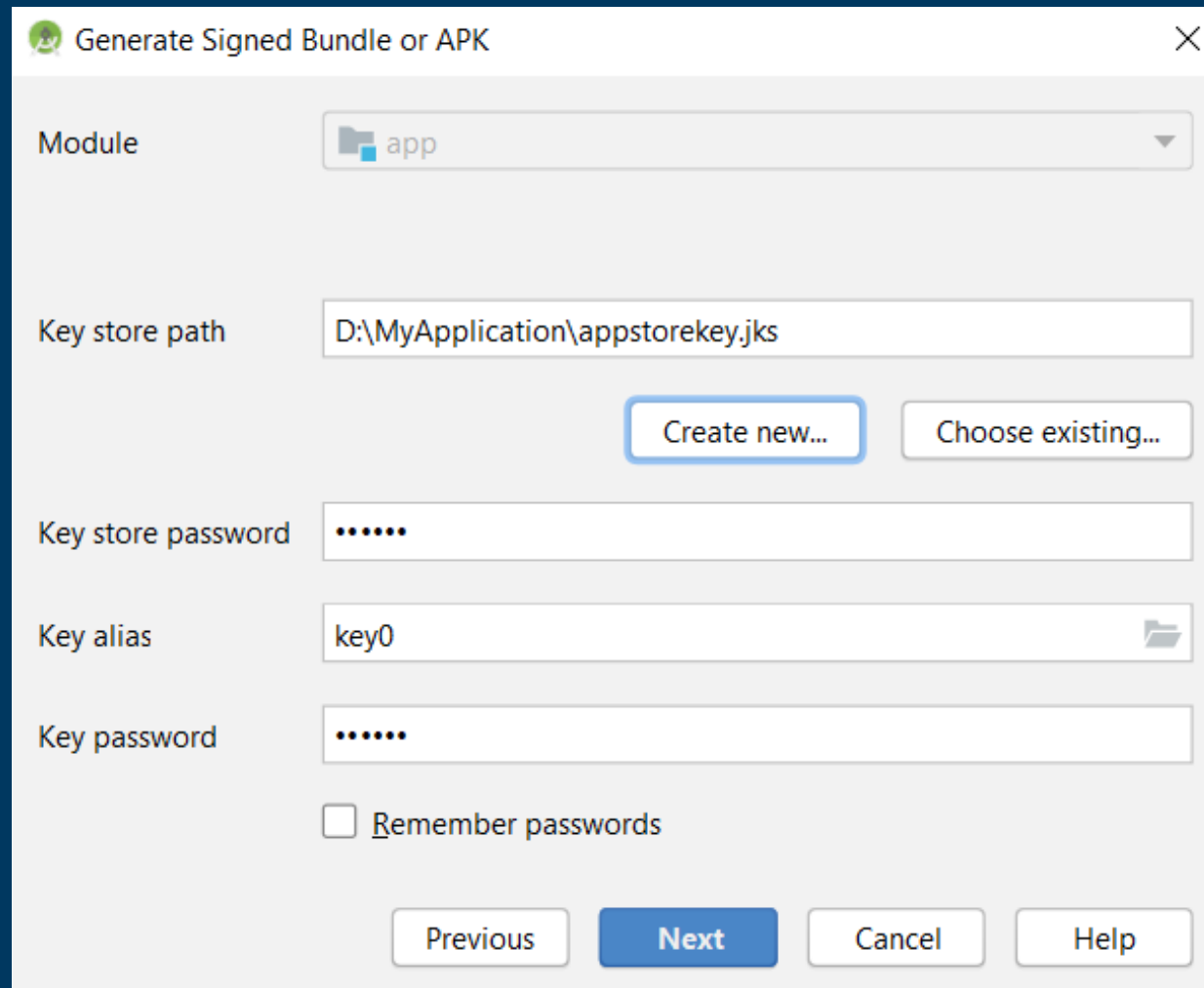
State or Province:

Country Code (XX):

OK Cancel

5

Click Next.



Generate Signed Bundle or APK

Module: app

Key store path: D:\MyApplication\appstorekey.jks

Buttons: Create new... (highlighted), Choose existing...

Key store password:

Key alias: key0

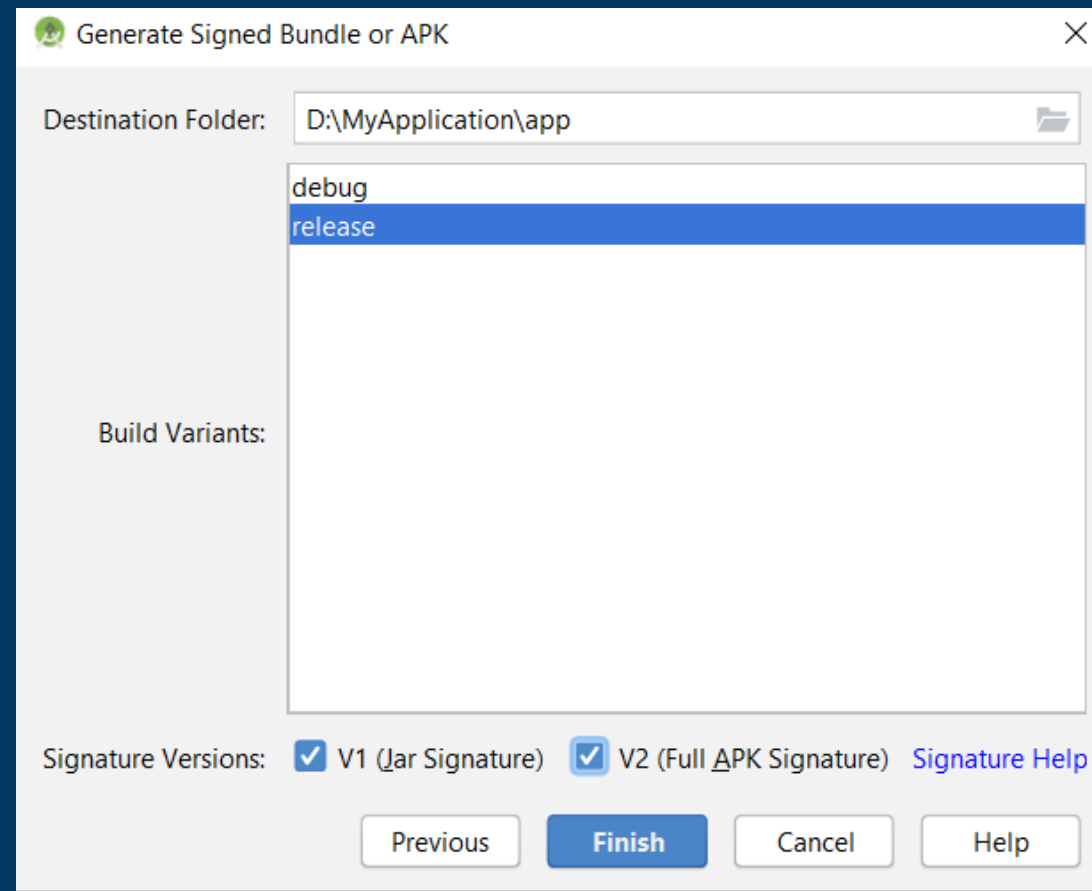
Key password:

Remember passwords

Buttons: Previous, Next (highlighted), Cancel, Help

5


Now edit the destination folder of signed **apk file**, choose build type and select signature versions. Finally click Finish.





 @imrankhanonnet

 @imrankhanonnet

 imranapps.com



Join Hands to **Shape Future** of Millions