Name: Rutuja Patil
Class: BE-IT- B
Batch: B3
Roll No: 59

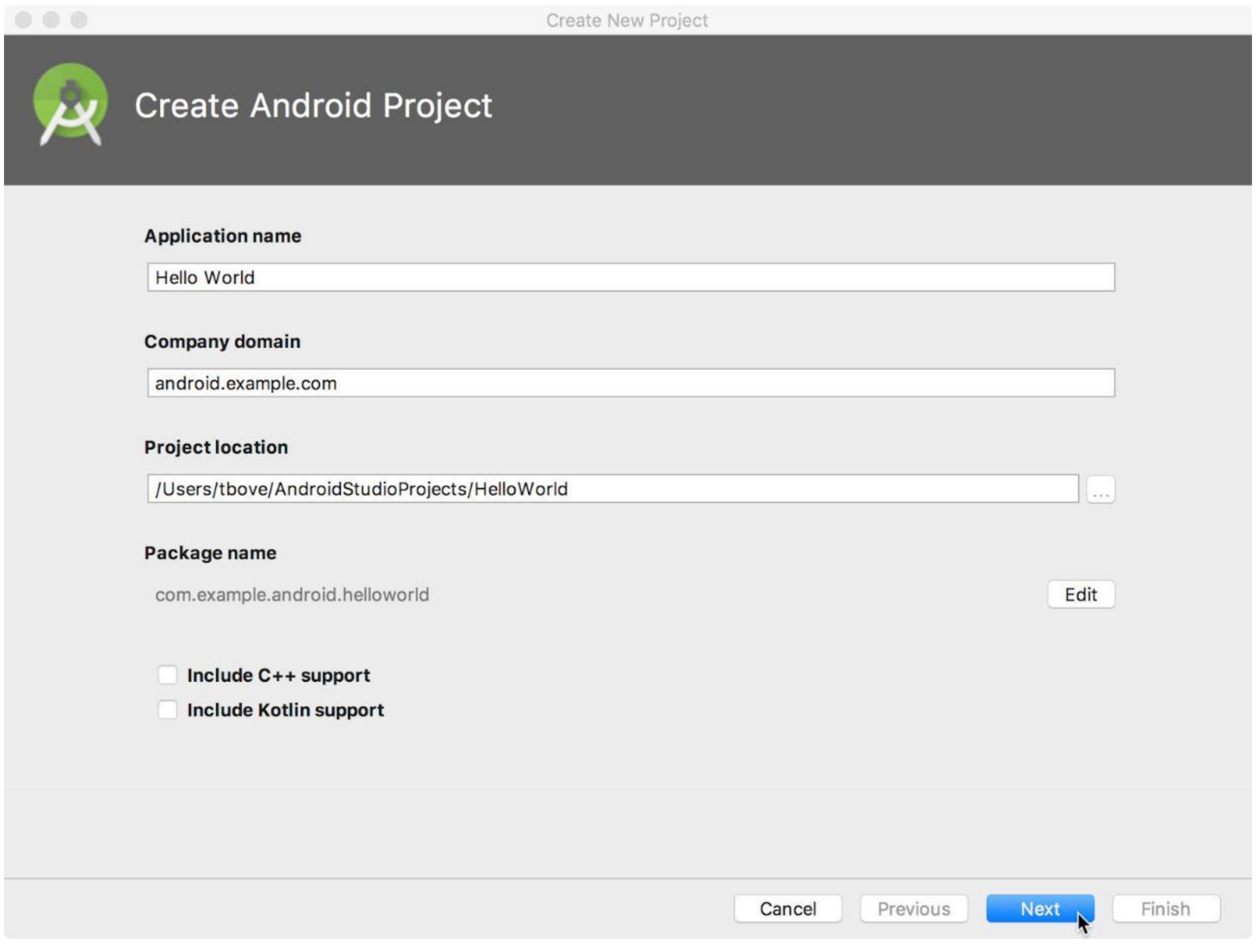
Experiment No. 2

Aim: Create the Hello World app.

Theory:
In this task, you will create an app that displays "Hello World" to verify that Android studio is correctly installed, and to learn the basics of developing with Android Studio.

Procedure:
Create the app project

1. Open Android Studio if it is not already opened.
2. In the main Welcome to Android Studio window, click Start a new Android Studio project.
3. In the Create Android Project window, enter Hello World for the Application name.


Create New Project

Create Android Project

Application name: Hello World

Company domain: android.example.com

Project location: /Users/tbove/AndroidStudioProjects/HelloWorld

Package name: com.example.android.helloworld

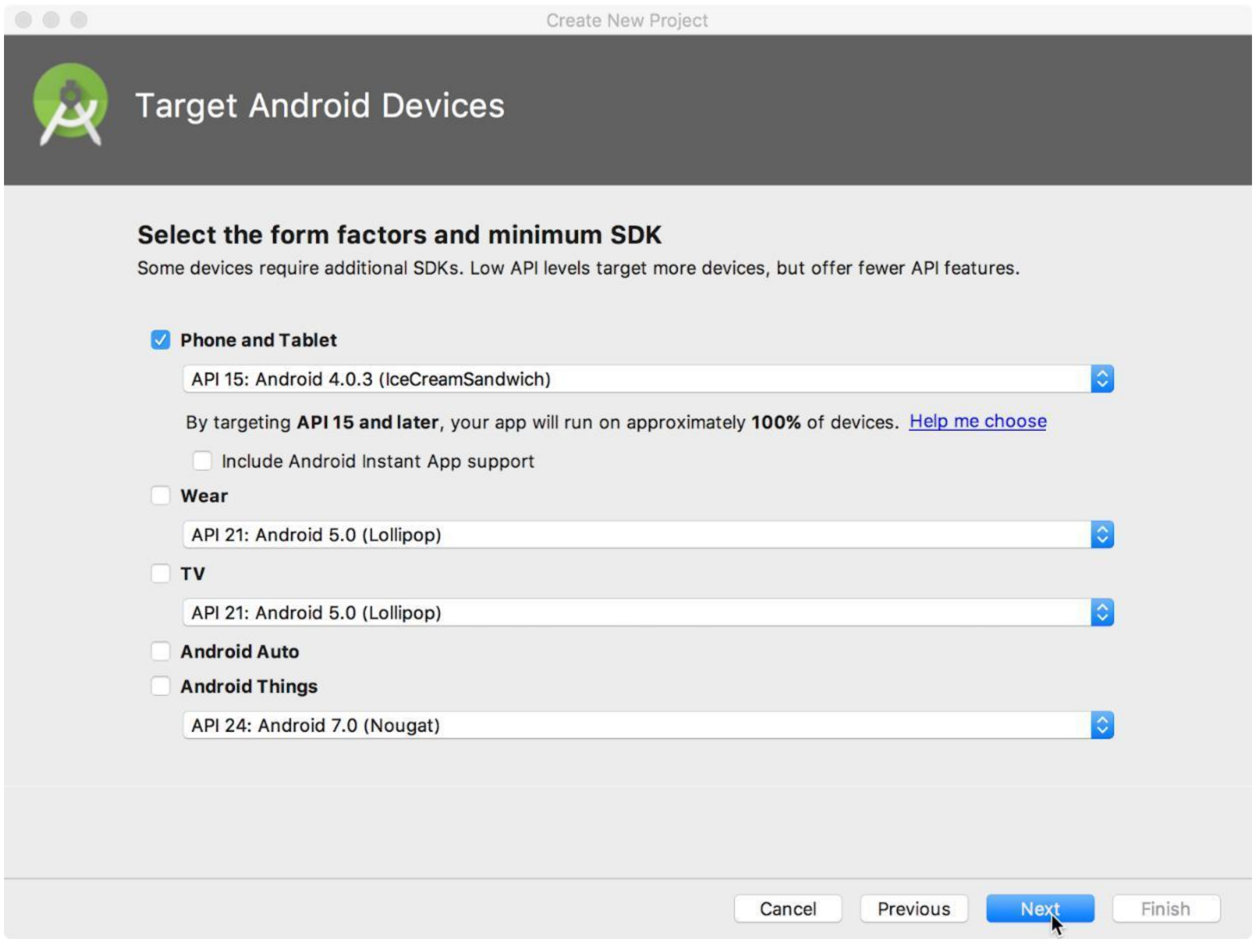
Include C++ support

Include Kotlin support

1.Verify that the default Project location is where you want to store your Hello World app and other Android Studio projects, or change it to your preferred directory.
2.Accept the default android.example.com for Company Domain, or create a unique company domain.

If you are not planning to publish your app, you can accept the default. Be aware that changing the package name of your app later is extra work.

1. Leave unchecked the options to Include C++ support and Include Kotlin support, and click Next.
2. On the Target Android Devices screen, Phone and Tablet should be selected. Ensure that API 15: Android 4.0.3 IceCreamSandwich is set as the Minimum SDK; if it is not, use the popup menu to set it.



Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☑ **Phone and Tablet**

API 15: Android 4.0.3 (IceCreamSandwich)

By targeting **API 15 and later**, your app will run on approximately **100%** of devices. Help me choose

☐ Include Android Instant App support

☐ **Wear**

API 21: Android 5.0 (Lollipop)

☐ **TV**

API 21: Android 5.0 (Lollipop)

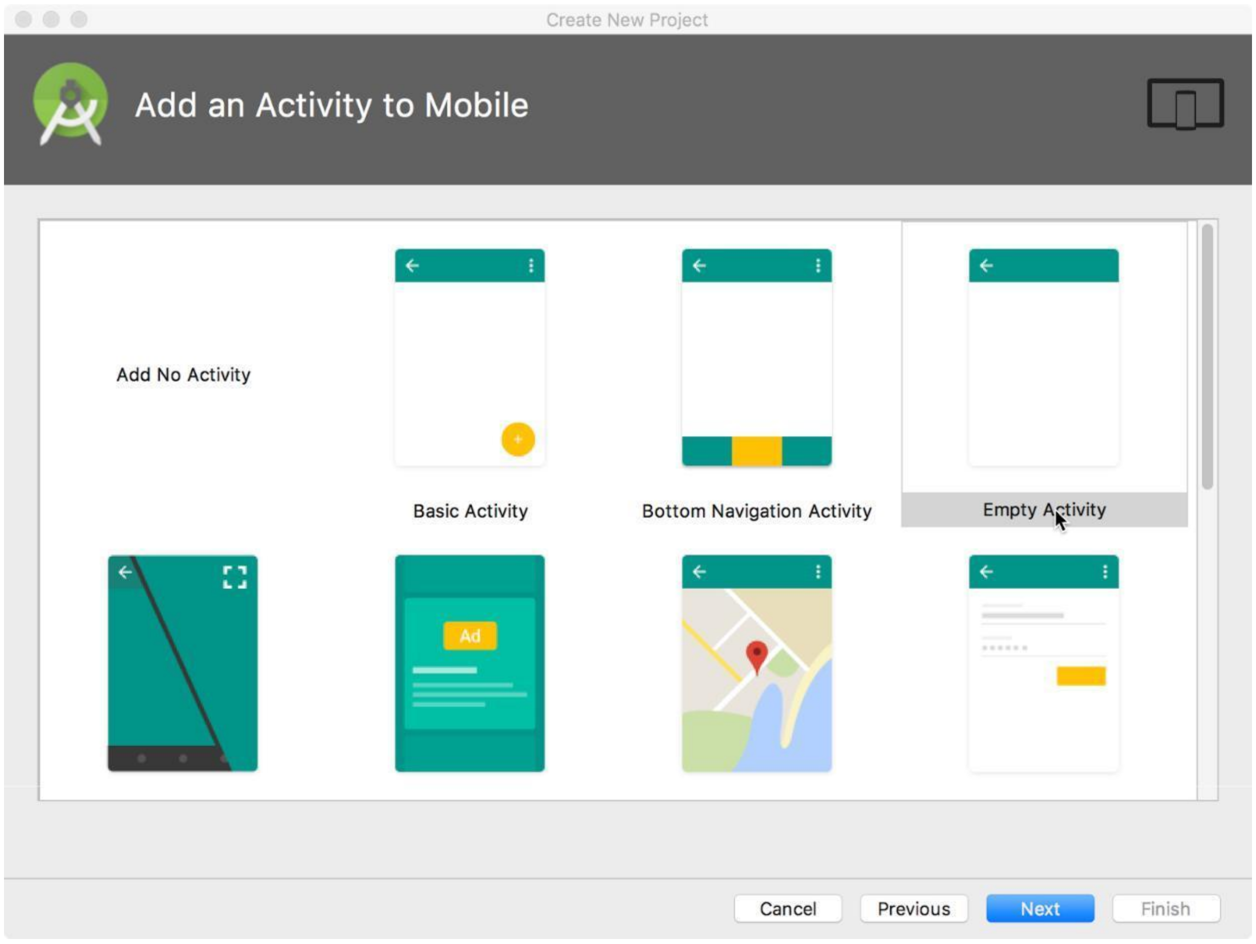
☐ **Android Auto**

☐ **Android Things**

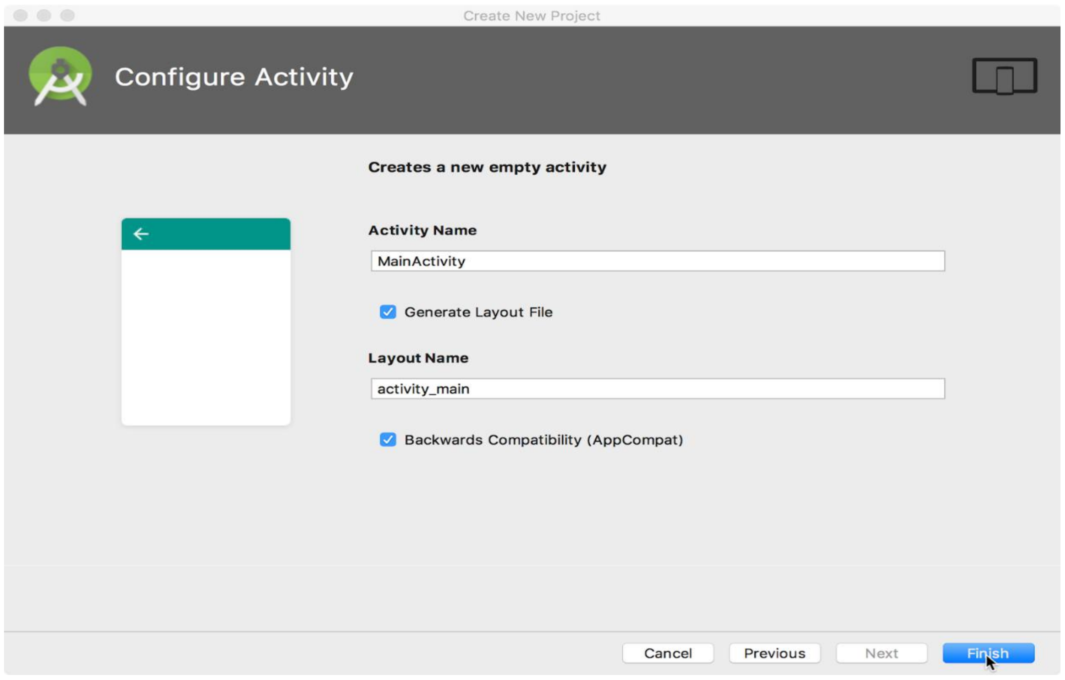
API 24: Android 7.0 (Nougat)

These are the settings used by the examples in the lessons for this course. As of this writing, these settings make your Hello World app compatible with 97% of Android devices active on the Google Play Store.

1. Leave unchecked the Include Instant App support and all other options. Then click Next. If your project requires additional components for your chosen target SDK, Android Studio will install them automatically.
2. The Add an Activity window appears. An Activityis a single, focused thing that the user can
do. It is a crucial component of any Android app. An Activitytypically has a layout associated with it that defines how UI elements appear on a screen. Android Studio provides Activity templates to help you get started. For the Hello World project, choose Empty Activity as shown below, and click Next.



1. The Configure Activity screen appears (which differs depending on which template you chose in the previous step). By default, the empty Activityprovided by the template is named
MainActivity. You can change this if you want, but this lesson uses MainActivity.



Creates a new empty activity

Activity Name: MainActivity

☑ Generate Layout File

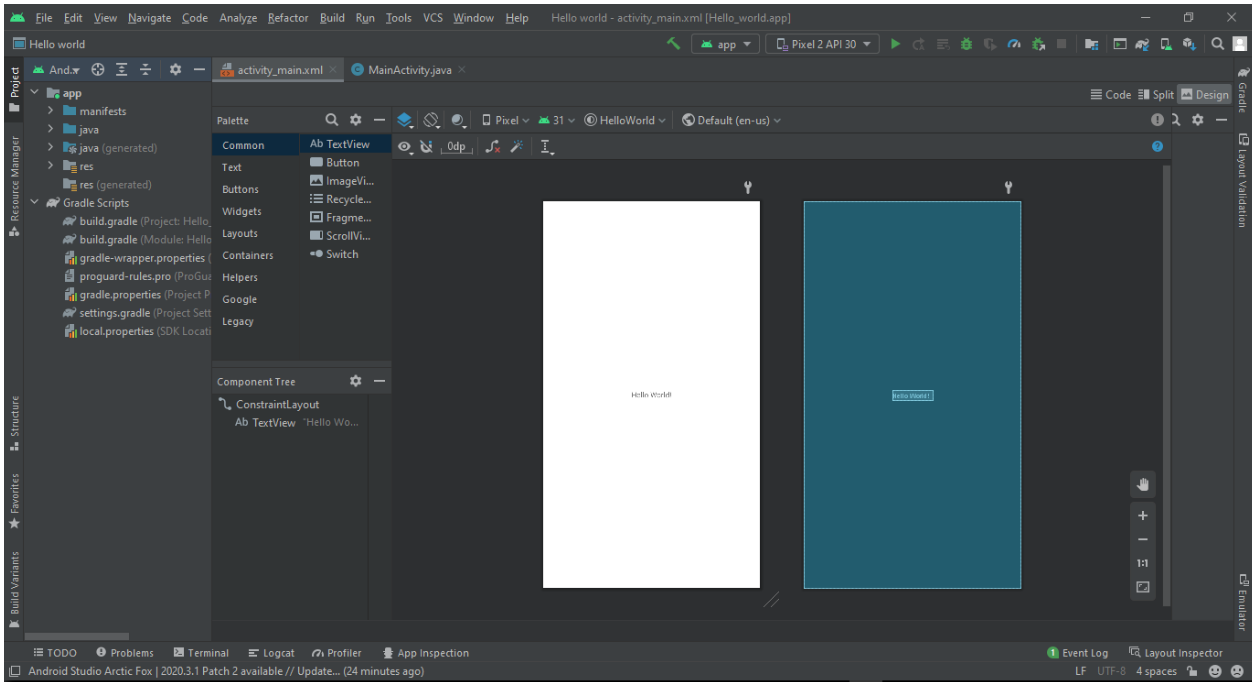
Layout Name: activity_main

☑ Backwards Compatibility (AppCompat)

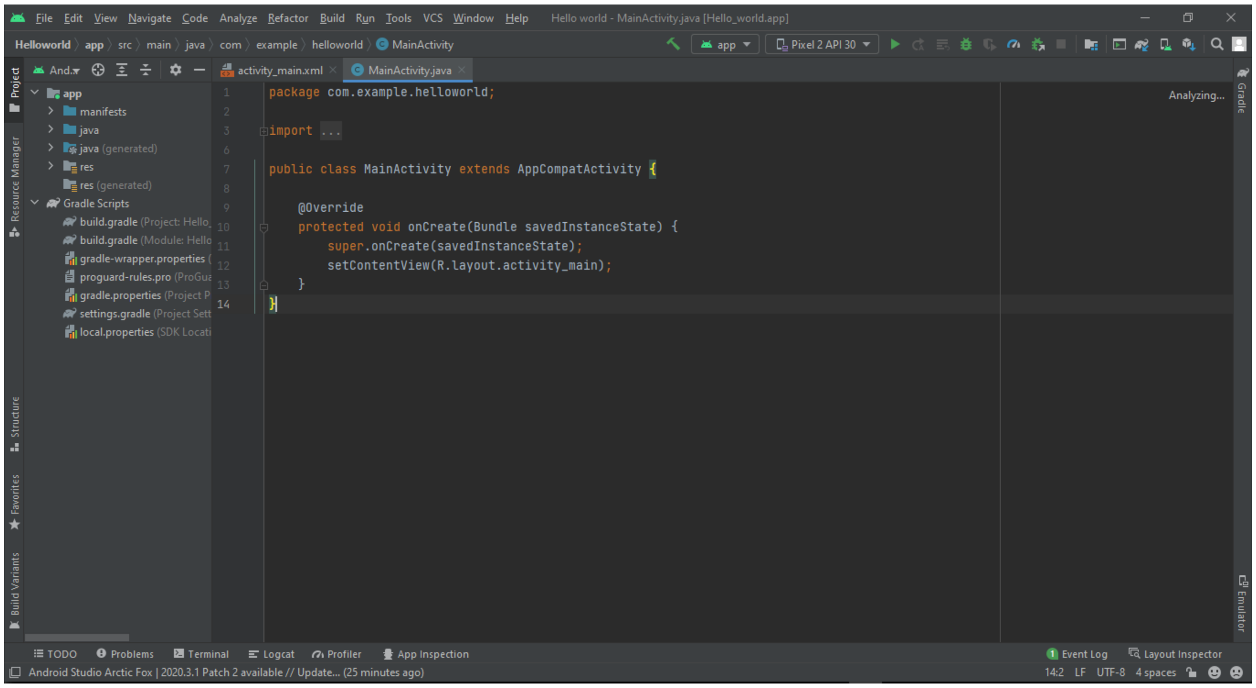
1. Make sure that the Generate Layout file option is checked. The layout name by default is activity_main. You can change this if you want, but this lesson uses activity_main.
2. Make sure that the Backwards Compatibility (App Compat) option is checked. This ensures that your app will be backwards-compatible with previous versions of Android.
3. Click Finish.

Android Studio creates a folder for your projects, and builds the project with Gradle(this may take a few moments).
Tip: See the Configure your builddeveloper page for detailed information.
You may also see a "Tip of the day" message with keyboard shortcuts and other useful tips. Click Close to close the message.
The Android Studio editor appears. Follow these steps:

1. Click the activity_main.xml tab to see the layout editor.
2. Click the layout editor Design tab, if not already selected, to show a graphical rendition of the layout as shown below.



1.Click the MainActivity.java tab to see the code editor as shown below.

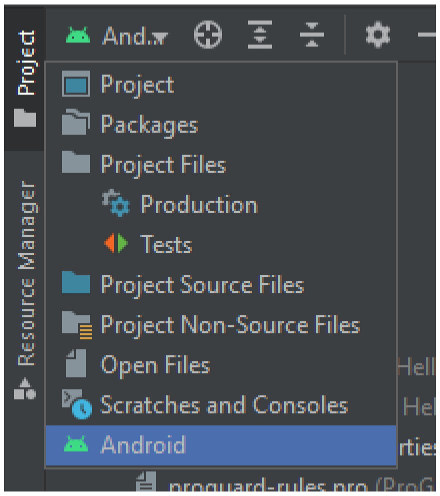

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Hello world - MainActivity.java [Hello_world.app]
Helloworld app src main java com example helloworld MainActivity
app
Pixel 2 API 30
activity_main.xml
MainActivity.java
Analyzing...
app
manifests
java
java (generated)
res
res (generated)
Gradle Scripts
build.gradle (Project: Hello_
build.gradle (Module: Hello
gradle-wrapper.properties
proguard-rules.pro (ProGua
gradle.properties (Project P
settings.gradle (Project Sett
local.properties (SDK Locati
TODO
Problems
Terminal
Logcat
Profiler
App Inspection
Event Log
Layout Inspector
Android Studio Arctic Fox | 2020.3.1 Patch 2 available // Update... (25 minutes ago)
14:2 LF UTF-8 4 spaces

```java
package com.example.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

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

Explore the Project > Android pane

In this practical, you will explore how the project is organized in Android Studio.

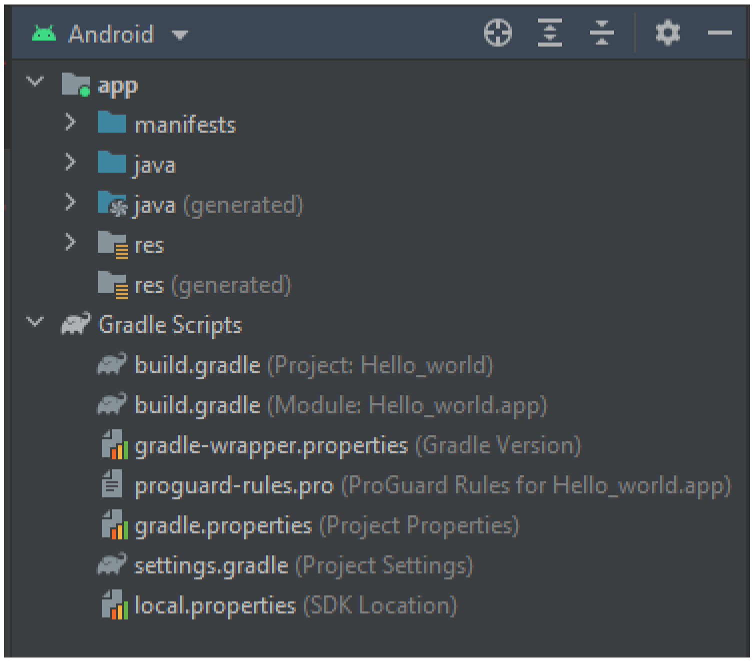
1. If not already selected, click the Project tab in the vertical tab column on the left side of the Android Studio window. The Project pane appears.
2. To view the project in the standard Android project hierarchy, choose Android from the popup menu at the top of the Project pane, as shown below.



Note: This chapter and other chapters refer to the Project pane, when set to Android, as the Project > Android pane.

Explore the Gradle Scripts folder

The Gradle build system in Android Studio makes it easy to include external binaries or other library modules to your build as dependencies.
When you first create an app project, the Project > Android pane appears with the Gradle


Android
app
manifests
java
java (generated)
res
res (generated)
Gradle Scripts
build.gradle (Project: Hello_world)
build.gradle (Module: Hello_world.app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for Hello_world.app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

Scripts folder expanded as shown below.

Follow these steps to explore the Gradle system:

1.If the Gradle Scripts folder is not expanded, click the triangle to expanit.
This folder contains all the files needed by the build system.

1.Look for the build.gradle(Project: HelloWorld) file.

This is where you'll find the configuration options that are common to all of the modules that make up your project. Every Android Studio project contains a single, top-level Gradle build file. Most of the time, you won't need to make any changes to this file, but it's still useful to understand its contents.

By default, the top-level build file uses the buildscriptblock to define the Gradle repositories and dependencies that are common to all modules in the project. When your dependency is something other than a local library or file tree, Gradle looks for the files in whichever online repositories are specified in the repositories block of this file. By default, new Android Studio projects declare
JCenter and Google (which includes the Google Maven repository) as the repository locations:

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
  repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:7.0.0"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

1. Look for the build.gradle(Module:app) file.

In addition to the project-level build.gradlefile, each module has a build.gradlefile of its own, which allows you to configure build settings for each specific module (the HelloWorld app has only one module). Configuring these build settings allows you to provide custom packaging options, such as
additional build types and product flavors. You can also override settings in the AndroidManifest.xml file or the top-level build.gradlefile.
This file is most often the file to edit when changing app-level configurations, such as declaring dependencies in the dependenciessection. You can declare a library dependency using one of several different dependency configurations. Each dependency configuration provides Gradle different instructions about how to use the library. For example, the statement implementation fileTree(dir: 'libs', include: ['*.jar'])adds a dependency of all ".jar" files inside the libsdirectory.
The following is the build.gradle(Module:app) file for the HelloWorld app:

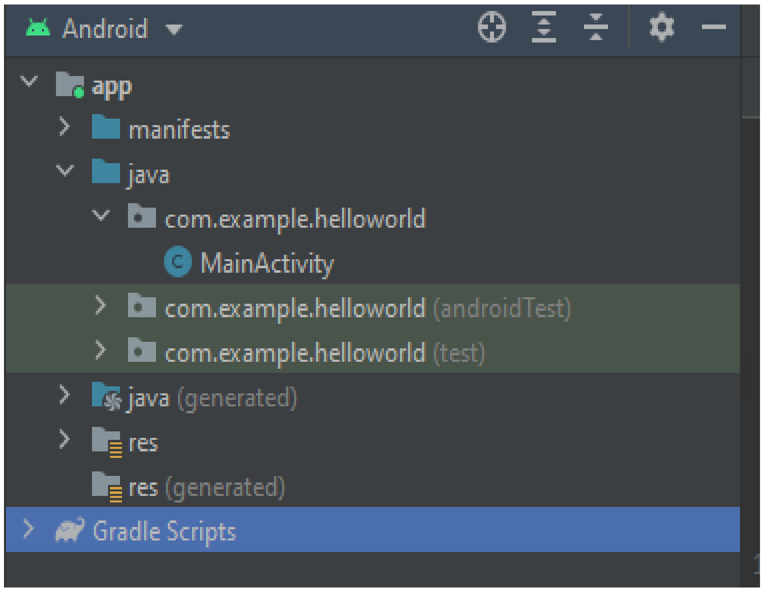
```
plugins {
    id 'com.android.application'
}

android {
    compileSdk 31

    defaultConfig {
        applicationId "com.example.helloworld"
        minSdk 16
        targetSdk 31
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
```

```
                              proguardFiles    getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

1.Click the triangle to close Gradle Scripts.

2.4 Explore the app and res folders
All code and resources for the app are located within the appand resfolders.

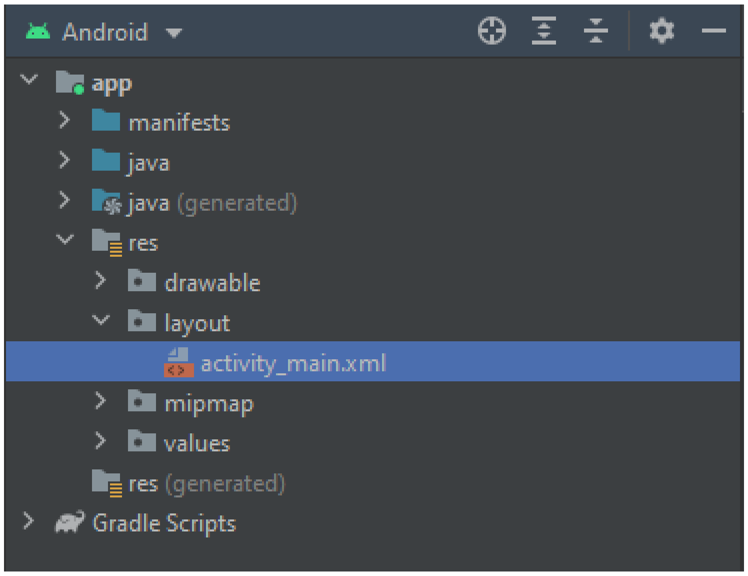
1. Expand the app folder, the java folder, and the com.example.android.hello world folder to see the MainActivity java file. Double-clicking the file opens it in the code editor.



The java folder includes Java class files in three subfolders, as shown in the figure above. The com.example.hello.helloworld (or the domain name you have specified) folder contains all the

files for an app package. The other two folders are used for testing and described in another lesson. For
the Hello World app, there is only one package and it contains MainActivity.java. The name of the
first Activity(screen) the user sees, which also initializes app-wide resources, is customarily called MainActivity (the file extension is omitted in the Project > Androidpane)

1. Expand the res folder and the layout folder, and double-click the activity_main.xml file to open it in the layout editor.



The res folder holds resources, such as layouts, strings, and images. An Activityis usually associated with a layout of UI views defined as an XML file. This file is usually named after its Activity.

Explore the manifests folder

The manifestsfolder contains files that provide essential information about your app to the Android system, which the system must have before it can run any of the app's code.

1. Expand the manifests folder.
2. Open the AndroidManifest.xml file.

The AndroidManifest.xmlfile describes all of the components of your Android app. All components
for an app, such as each Activity,must be declared in this XML file. In other course lessons you will modify this file to add features and feature permissions. For an introduction, see App Manifest Overview.

Conclusion:

You just need to explore the interface of the Android Studio and it contain already a "Hello world" program