

CIT 4404 Mobile App Development

Topic2: Android Application Development Using Android Studio

Dr. Fullgence Mwakondo

Institute of Computing and Informatics

Technical University of Mombasa

mwakondo@tum.ac.ke


Android Application Development Using Android Studio

- a. Exploring the IDE
- b. Using Development tools
- c. Coding Application
- d. Debugging Application
- e. Publishing application

Exploring the IDE

- Open the IDE
- Start a new project
- Select options
- Project view

Give the project name: IDEExplorer; use whatever domain name you like



Create Android Project

Application name

Company domain

Project location

 ...

Package name

 Edit

☐ Include C++ support

☐ Include Kotlin support

Cancel Previous **Next** Finish



Target Android Devices

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 24: Android 7.0 (Nougat)



By targeting **API 24 and later**, your app will run on approximately **37.1%** 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)



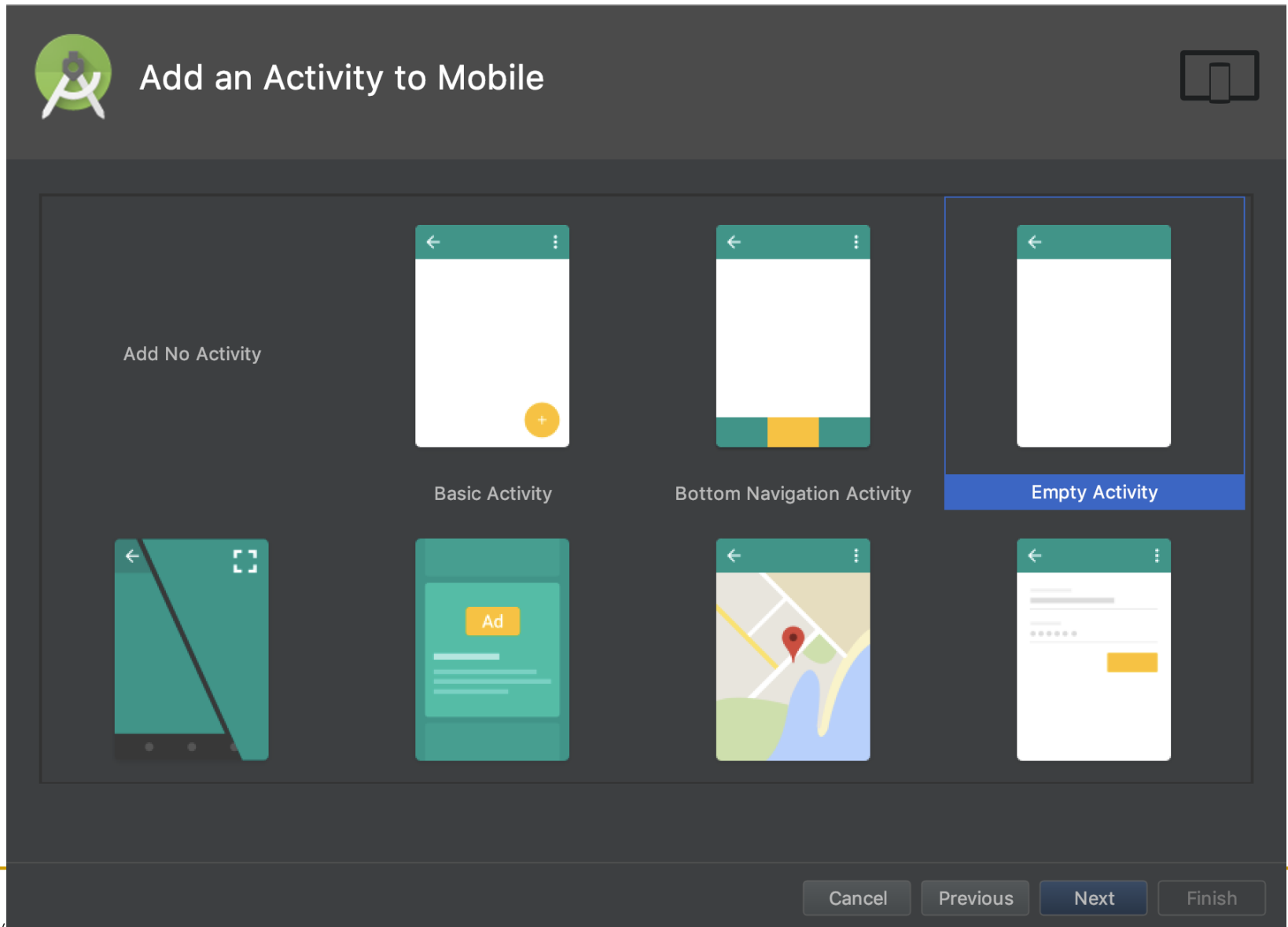
Cancel

Previous

Next

Finish

The default option is Empty Activity. This is the most useful for our examples because it creates a basic activity for you, with no code in it

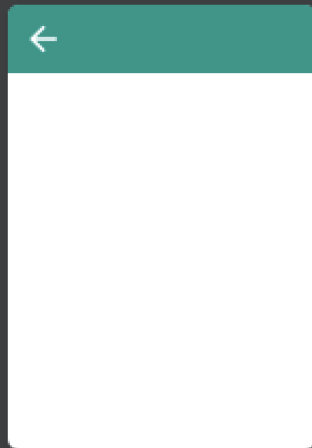




Configure Activity



It is accepted practice in Android development to name your main activity—that is, the Activity that is loaded on startup by your application—as MainActivity



Activity Name:

MainActivity



Generate Layout File

Layout Name:

activity_main



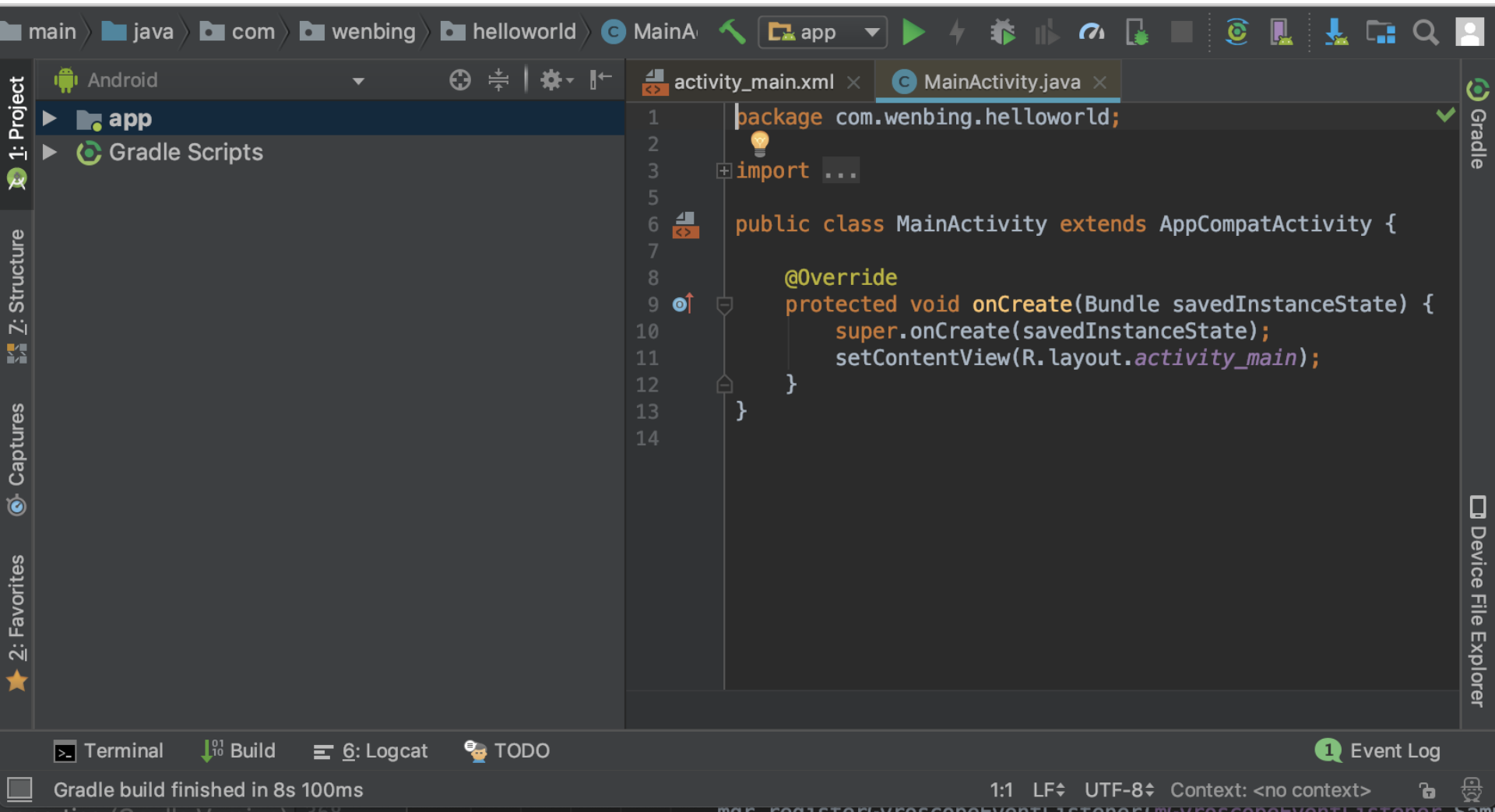
Backwards Compatibility (AppCompat)

The startup layout, that is the layout for the screen elements that will be displayed when your application is started by the user, is the activity_main layout. All other layouts should be named according to the activity that they support (activity_input, activity_delete)

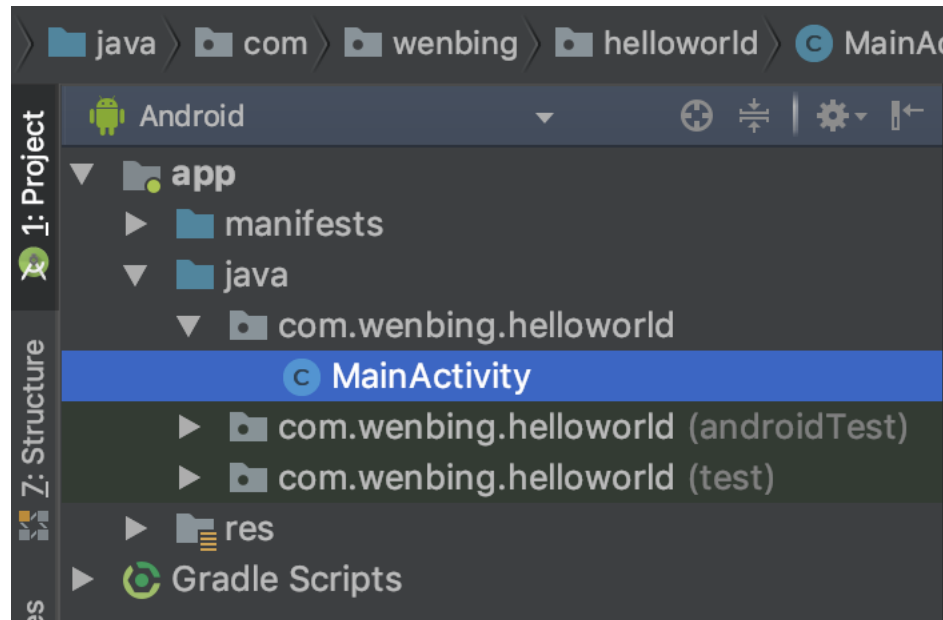
The name of the activity class to create

Click the Finish button to finish creating the project and jump into exploring the IDE

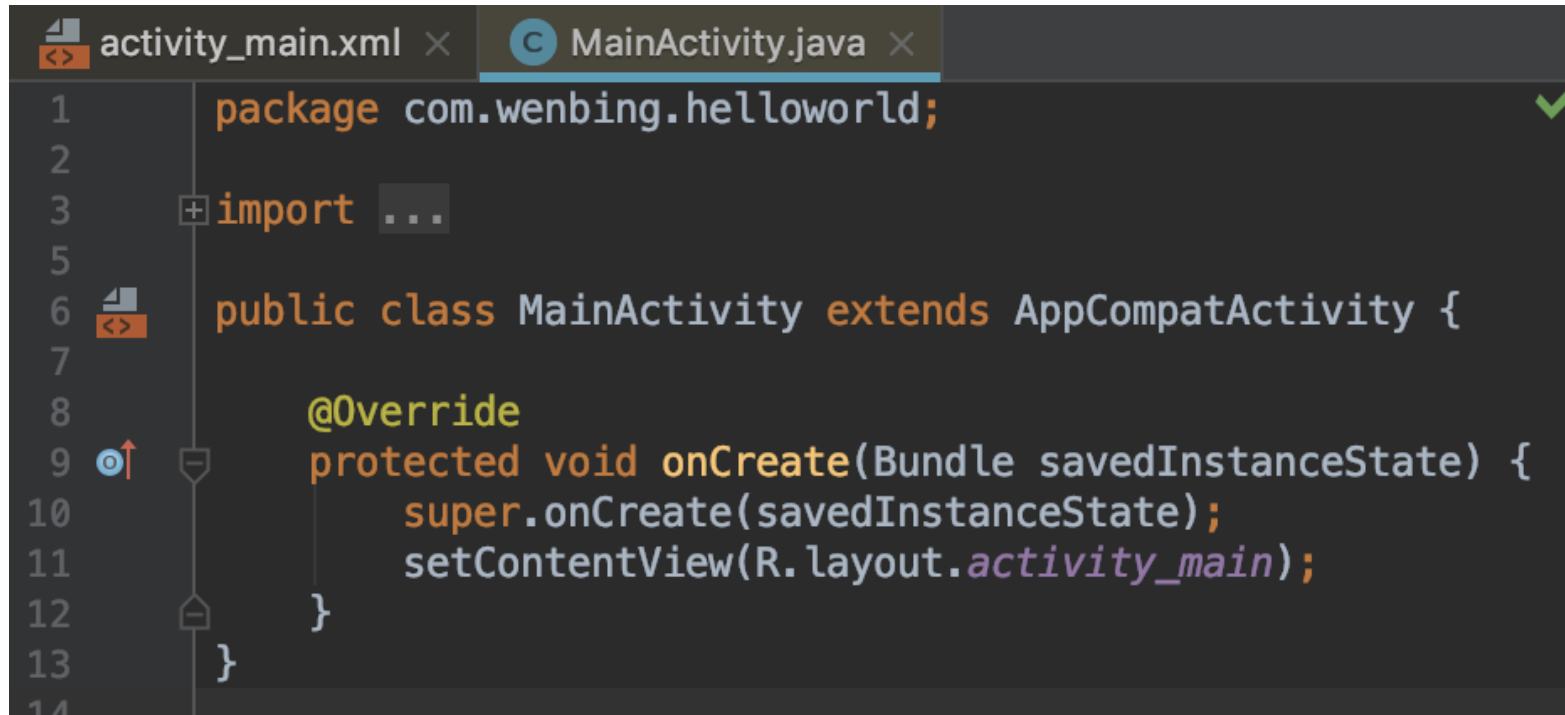
The Android Studio IDE



The left side of the IDE shows the Project window. The Project window enables you to quickly navigate the files within your project.



On the right side of the IDE are the Editor tabs. The Editor tabs are where you write and work with your code files.

A screenshot of an IDE's editor window. At the top, there are two tabs: 'activity_main.xml' and 'MainActivity.java'. The 'MainActivity.java' tab is active and highlighted. The code in the editor is as follows:

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

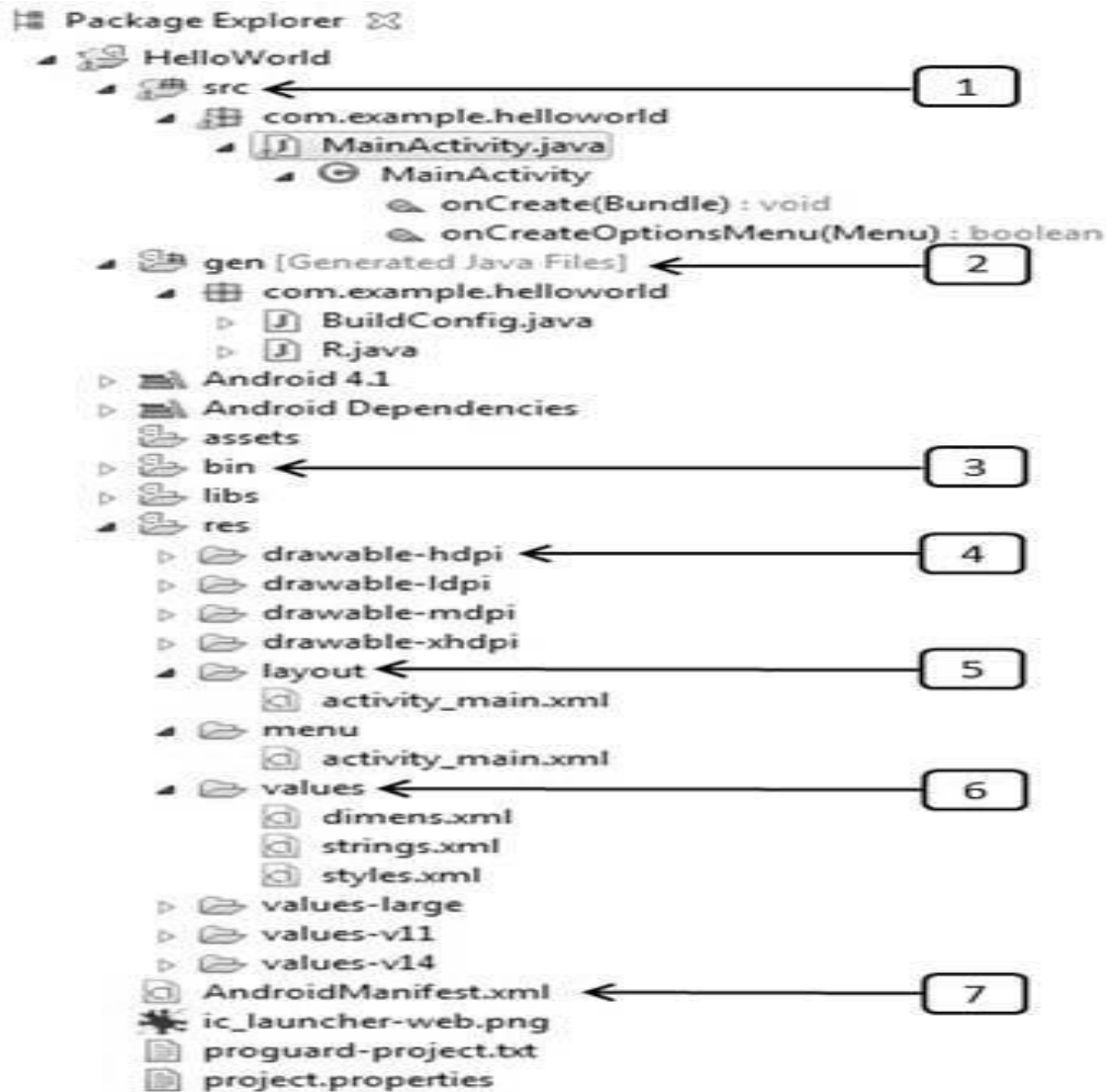
The code is color-coded: 'package' is orange, 'import' is orange, 'public class' is orange, 'extends' is orange, '@Override' is yellow, 'protected void' is orange, 'onCreate' is orange, 'Bundle' is grey, 'savedInstanceState' is grey, 'super' is orange, 'setContentView' is orange, and 'R.layout.activity_main' is purple. A green checkmark is visible in the top right corner of the code area.

- To work on a new file, simply locate the file in the Project window and double-click it to open a new Editor tab that contains that file's code.
- If you need to create a new file from scratch, right-click the directory into which you want to place your file, and select New ➞ <File Type> from the context menu.
- At the bottom of the IDE, you should see a button labeled LogCat. Logcat displays most of the helpful messages that are output by your application while you are trying to debug it.

```
Logcat
Emulator Nexus_5X_API_2  com.wenbing.helloworld (4)  Ve...  Q  [X] Regex  Show only selected applicati
01-14 01:14:27.973 4294-4294/? I/art: Not late-enabling -Xcheck:jni (already on)
01-14 01:14:27.973 4294-4294/? W/art: Unexpected CPU variant for X86 using defaults: x86_64
01-14 01:14:28.297 4294-4294/com.wenbing.helloworld W/ActivityThread: Application com.wenbing.helloworld is waiting for the debugger on port
01-14 01:14:28.301 4294-4294/com.wenbing.helloworld I/System.out: Sending WAIT chunk
01-14 01:14:30.993 4294-4302/com.wenbing.helloworld I/art: Debugger is active
01-14 01:14:31.499 4294-4294/com.wenbing.helloworld I/System.out: Debugger has connected
>> waiting for debugger to settle...
```

Terminal Build Logcat Android Profiler Run Debug TODO Event Log

Project structure of Android App



S.N. Folder, File & Description

1 src

This contains the .java source files for your project. By default, it includes an MainActivity.java source file having an activity class that runs when your app is launched using the app icon.

2 gen

This contains the .R file, a compiler-generated file that references all the resources found in your project. You should not modify this file.

3 bin

This folder contains the Android package files .apk built by the ADT during the build process and everything else needed to run an Android application.

4 res/drawable-hdpi

This is a directory for drawable objects that are designed for high-density screens.

5 res/layout

This is a directory for files that define your app's user interface.

6 res/values

This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.

7 AndroidManifest.xml

This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

Coding Application

- Important parts of Android App
- Creating your first App

Important parts of Android App

Important parts of an Android application include the following:

- **Activities**

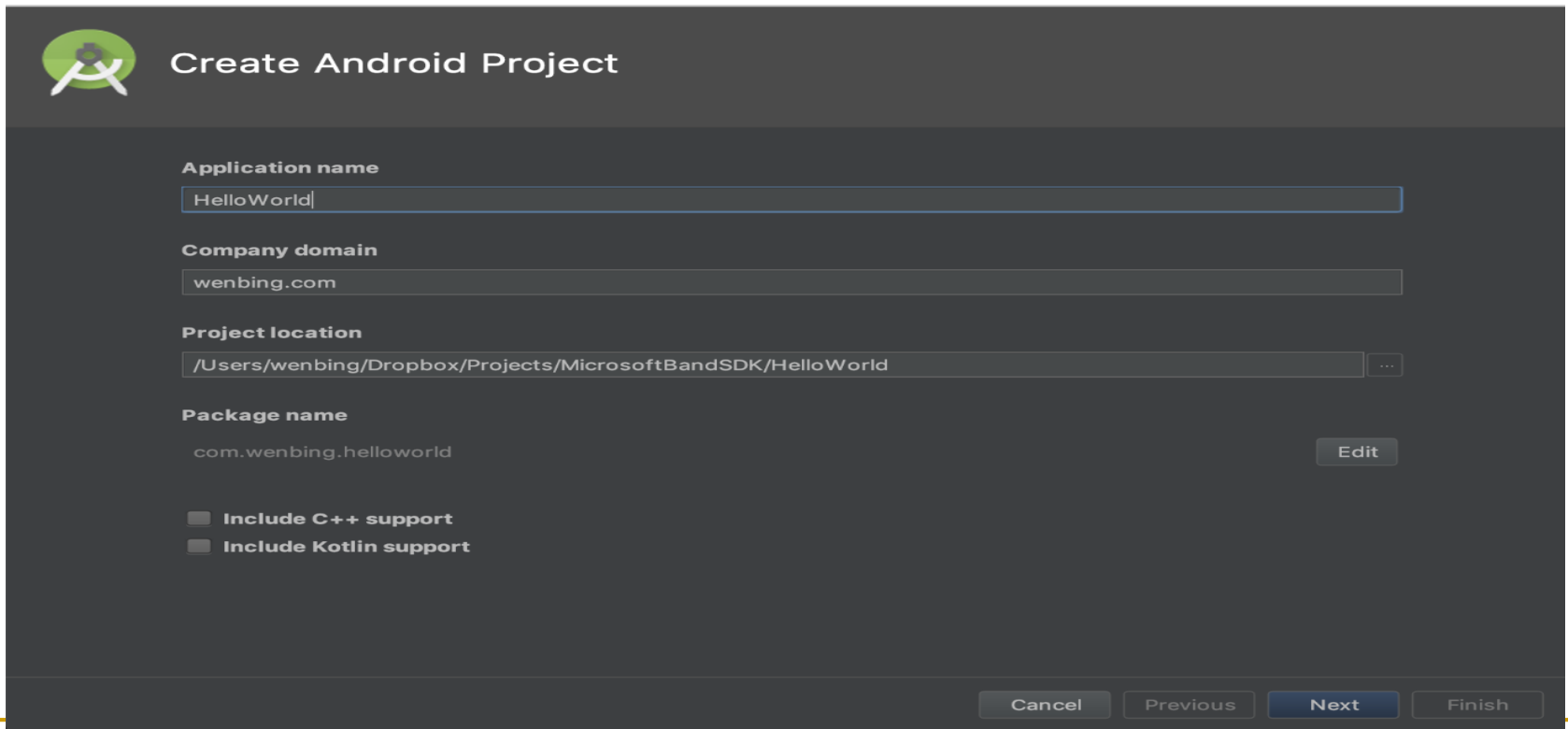
The Activities are the main Java classes, that contain the Android code with which we are going to develop, **what do we want the application to do.**

- **Layouts**

The Layouts are the main xml files, that contain the Android xml code with which we are going to develop, **how will the application views look like.**

Creating your first Android App

- File -> New->New Project
- Name the app: HelloWorld
- Then select default option for all remaining steps



The screenshot shows the 'Create Android Project' dialog in Android Studio. The dialog has a dark gray background and a title bar with the Android Studio logo and the text 'Create Android Project'. The main content area contains several input fields and checkboxes. The 'Application name' field is filled with 'HelloWorld'. The 'Company domain' field is filled with 'wenbing.com'. The 'Project location' field is filled with '/Users/wenbing/Dropbox/Projects/MicrosoftBandSDK/HelloWorld'. The 'Package name' field is filled with 'com.wenbing.helloworld'. There are two checkboxes: 'Include C++ support' and 'Include Kotlin support', both of which are unchecked. At the bottom right, there is an 'Edit' button next to the package name field. At the bottom of the dialog, there are four buttons: 'Cancel', 'Previous', 'Next', and 'Finish'. The 'Next' button is highlighted in blue.

Create Android Project

Application name
HelloWorld

Company domain
wenbing.com

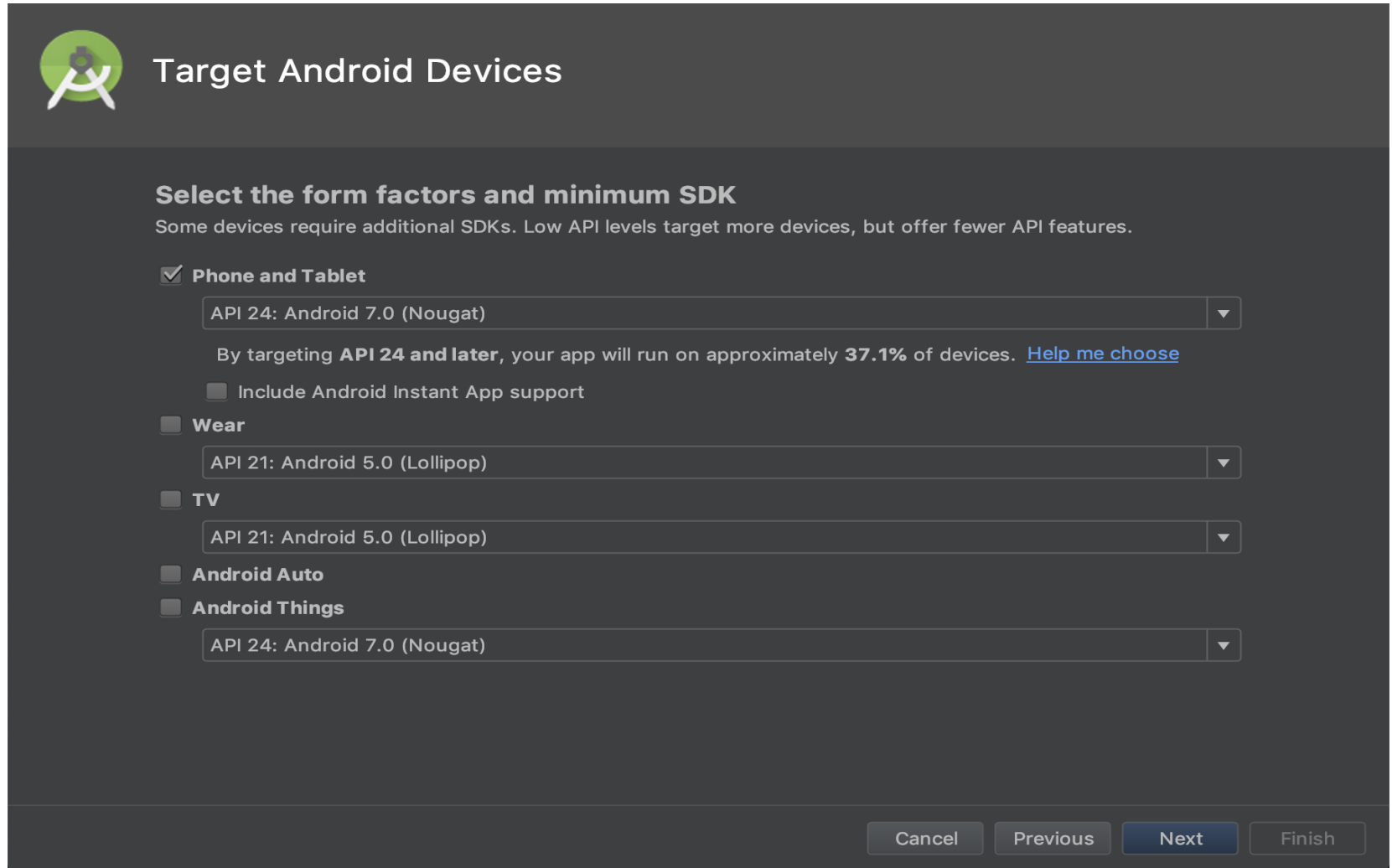
Project location
/Users/wenbing/Dropbox/Projects/MicrosoftBandSDK/HelloWorld

Package name
com.wenbing.helloworld Edit

☐ Include C++ support
☐ Include Kotlin support

Cancel Previous Next Finish

Creating your first Android App



The image shows the 'Target Android Devices' screen in Android Studio. At the top left is the Android Studio logo. The title 'Target Android Devices' is in the top bar. Below it, the section 'Select the form factors and minimum SDK' is displayed. A subtitle explains that some devices require additional SDKs and that low API levels target more devices but offer fewer API features. There are five form factor options: 'Phone and Tablet' (checked), 'Wear', 'TV', 'Android Auto', and 'Android Things'. Each has a dropdown menu for the minimum SDK. For 'Phone and Tablet', the dropdown shows 'API 24: Android 7.0 (Nougat)' and a note indicates that targeting API 24 and later will run on approximately 37.1% of devices, with a link to 'Help me choose'. There is also a checkbox for 'Include Android Instant App support'. At the bottom right are four buttons: 'Cancel', 'Previous', 'Next', and 'Finish'.

Target Android Devices

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 24: Android 7.0 (Nougat) ▼

By targeting **API 24 and later**, your app will run on approximately **37.1%** of devices. [Help me choose](#)

☐ Include Android Instant App support

☐ **Wear**

API 21: Android 5.0 (Lollipop) ▼

☐ **TV**

API 21: Android 5.0 (Lollipop) ▼

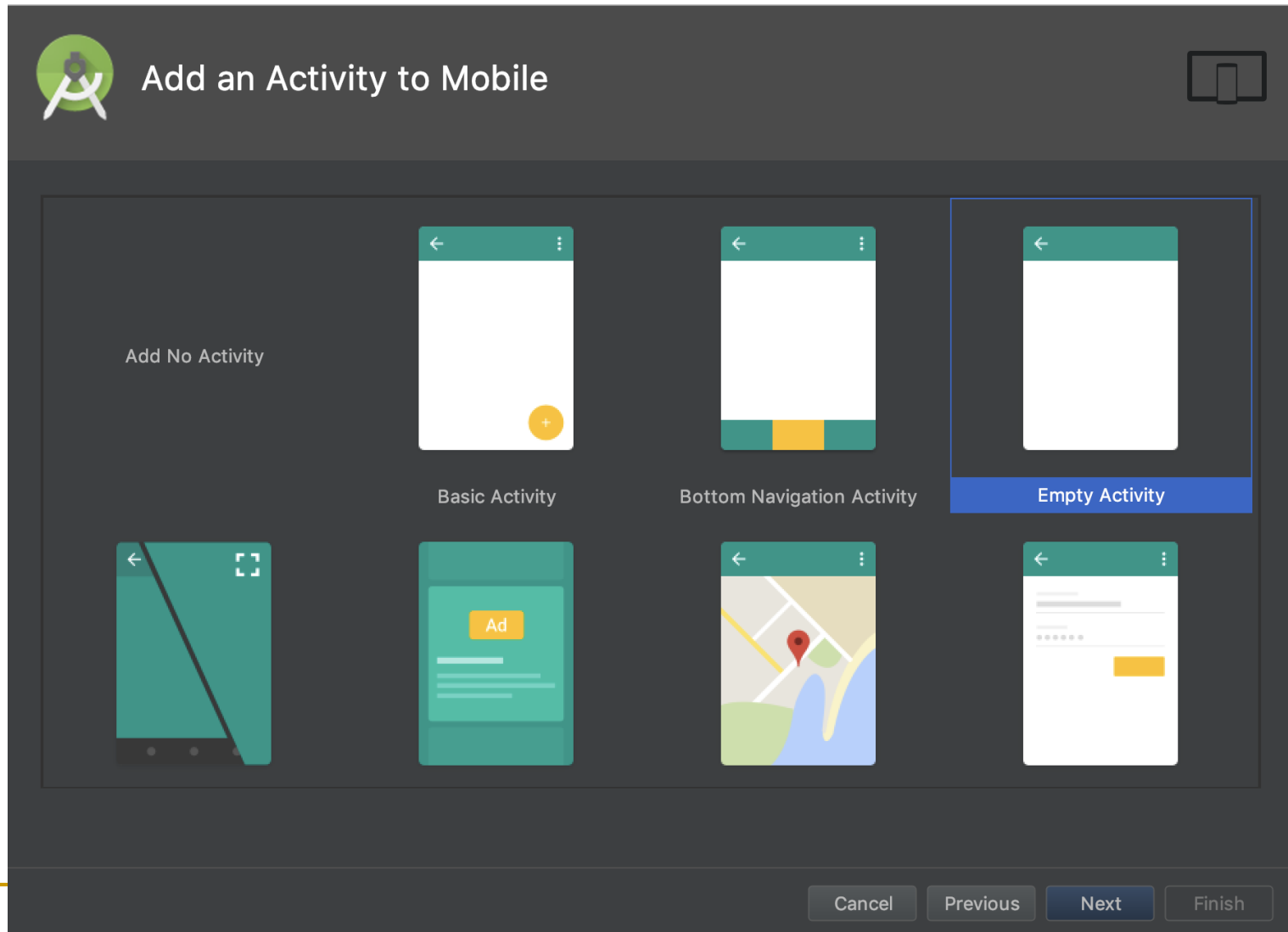
☐ **Android Auto**

☐ **Android Things**

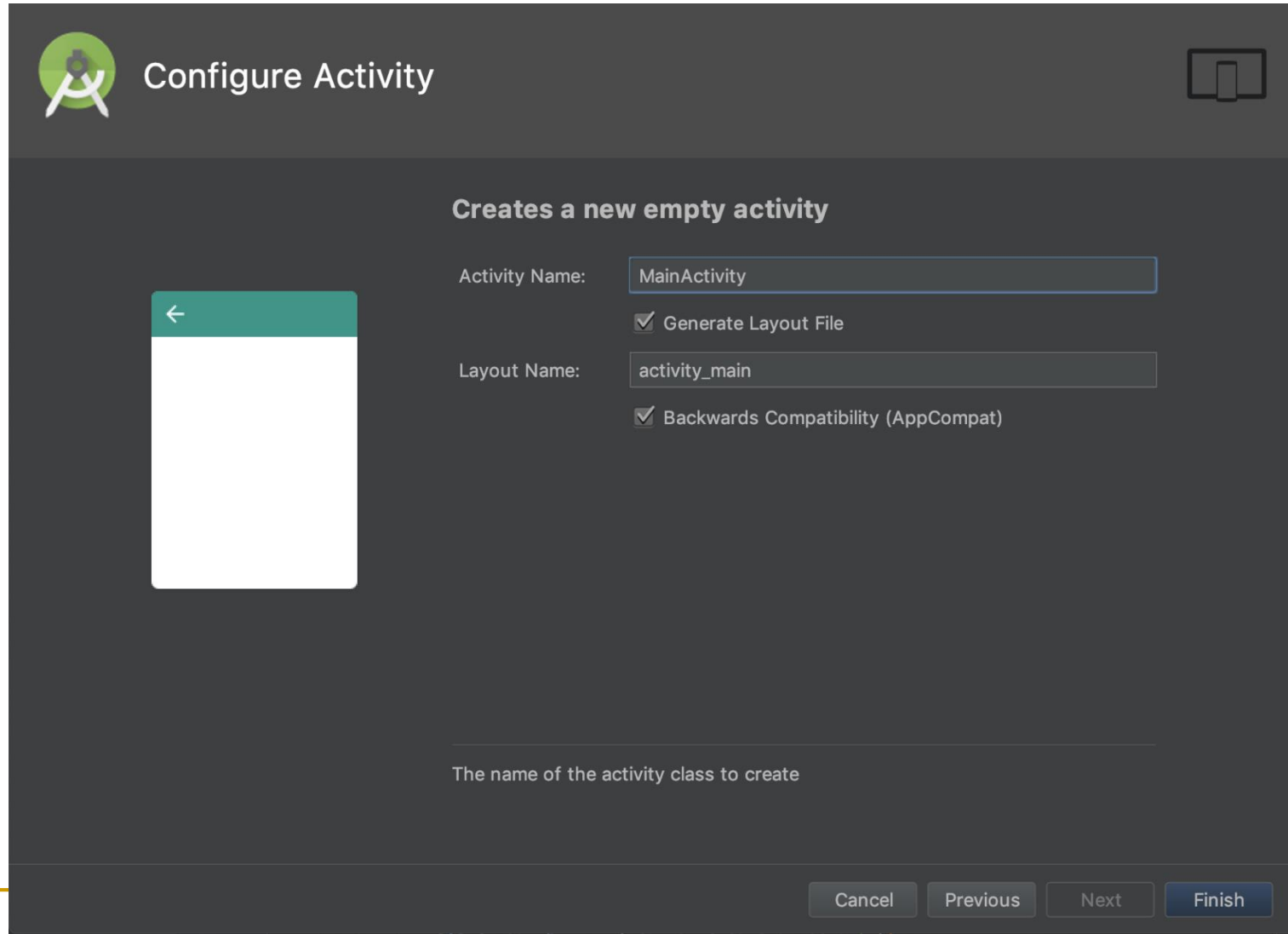
API 24: Android 7.0 (Nougat) ▼

Cancel Previous **Next** Finish

Creating your first Android App



Creating your first Android App



Configure Activity

Creates a new empty activity

Activity Name:

☒ Generate Layout File

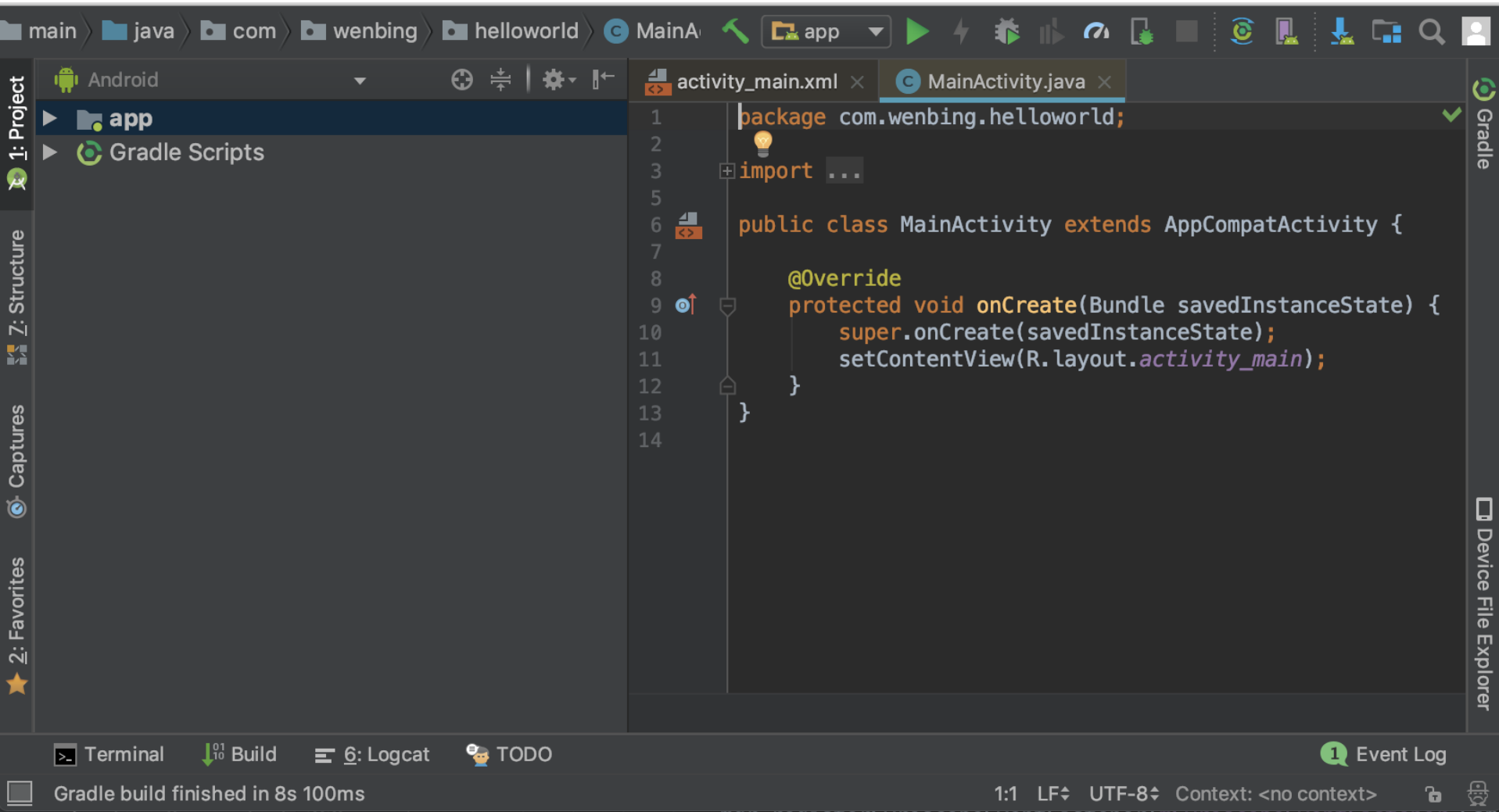
Layout Name:

☒ Backwards Compatibility (AppCompat)

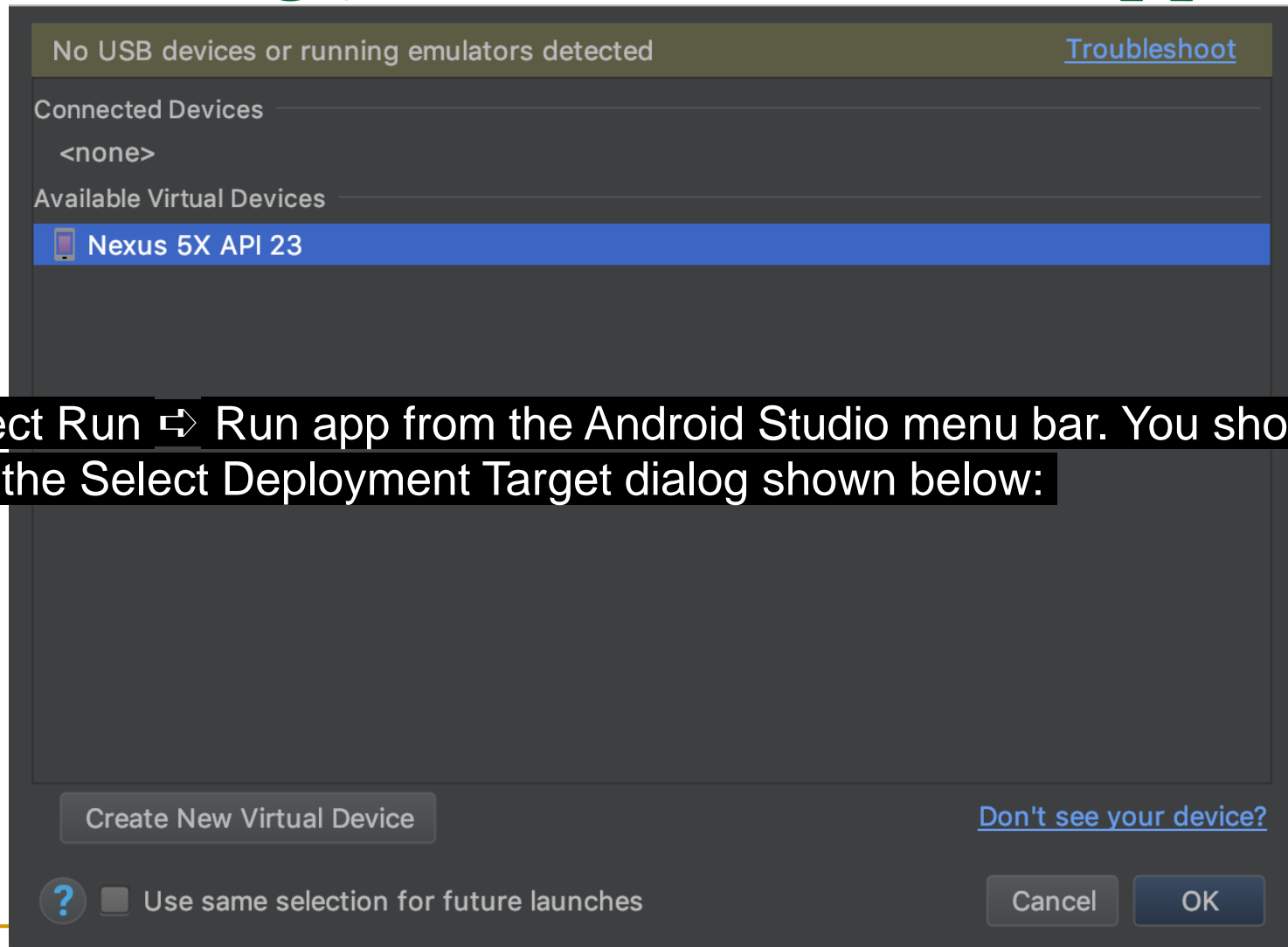
The name of the activity class to create

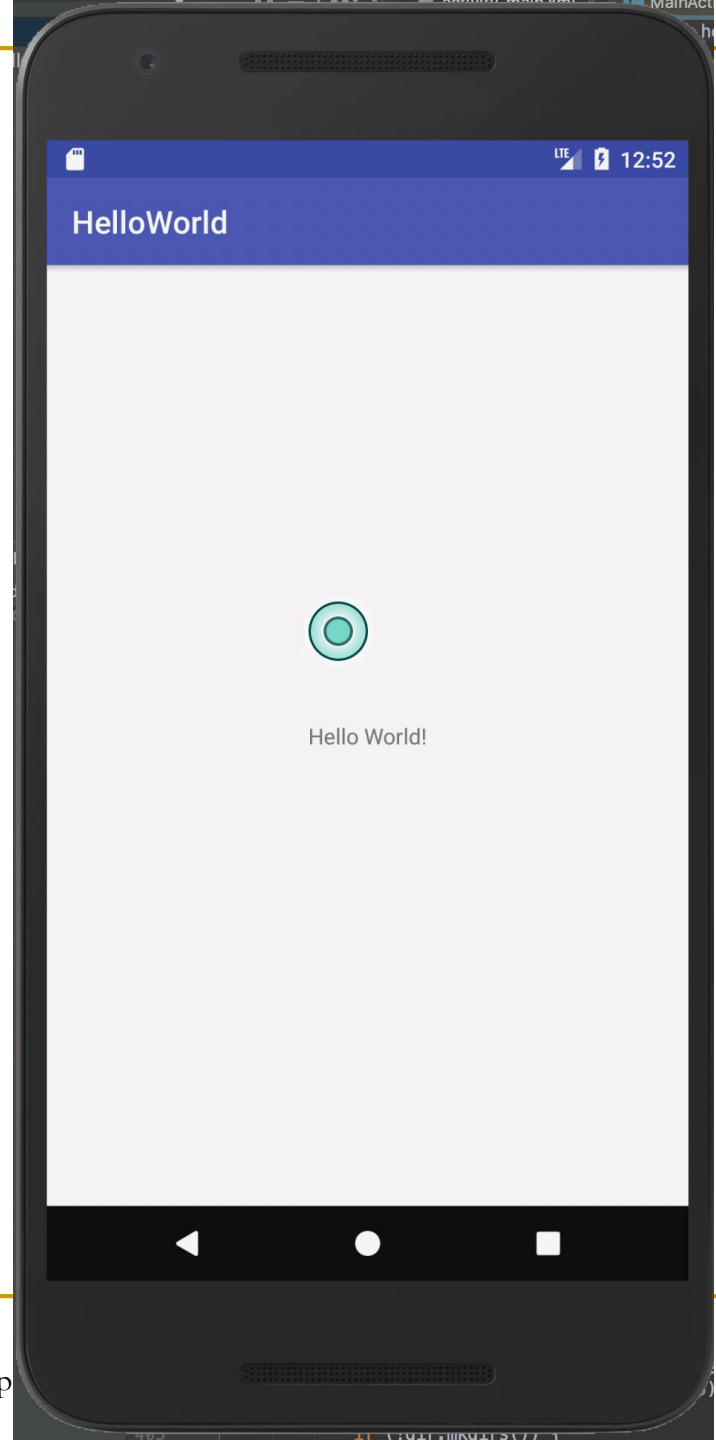
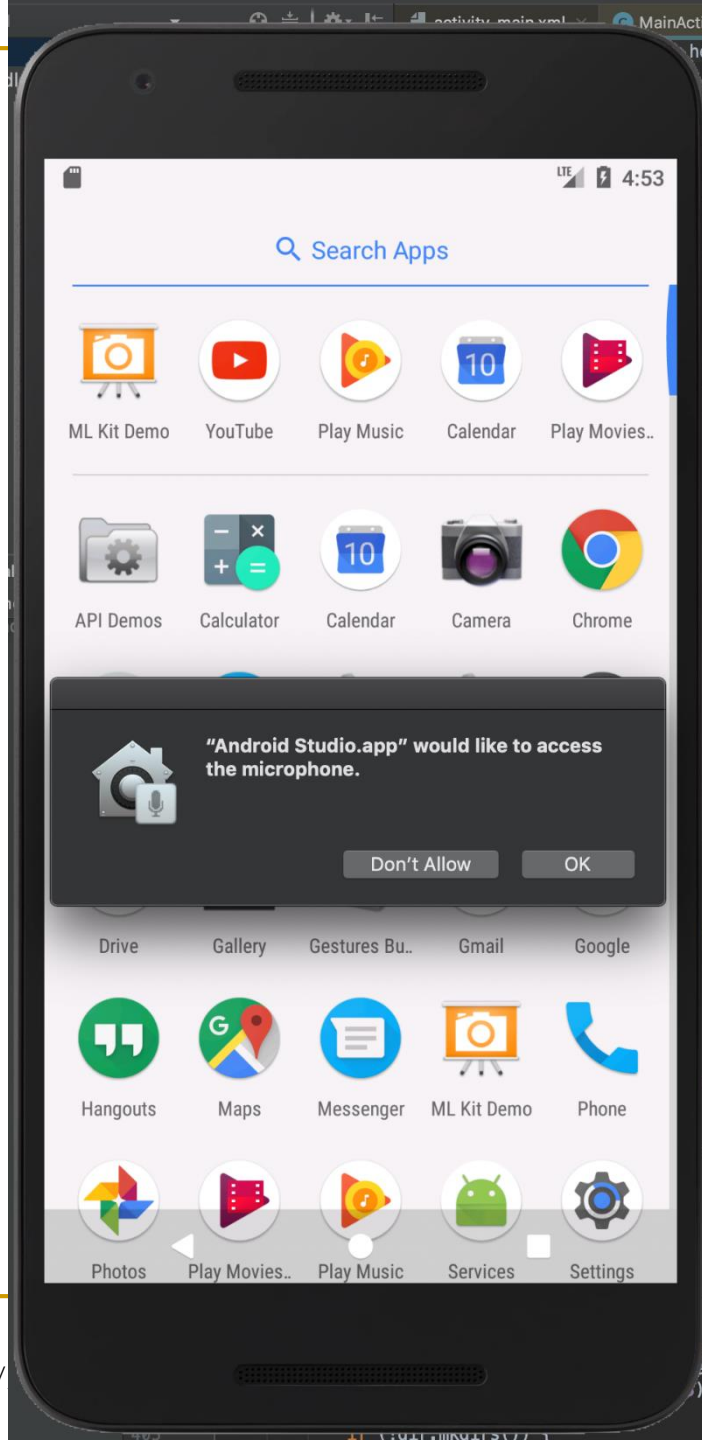
Cancel Previous Next Finish

Creating your first Android App



Launching your first Android App

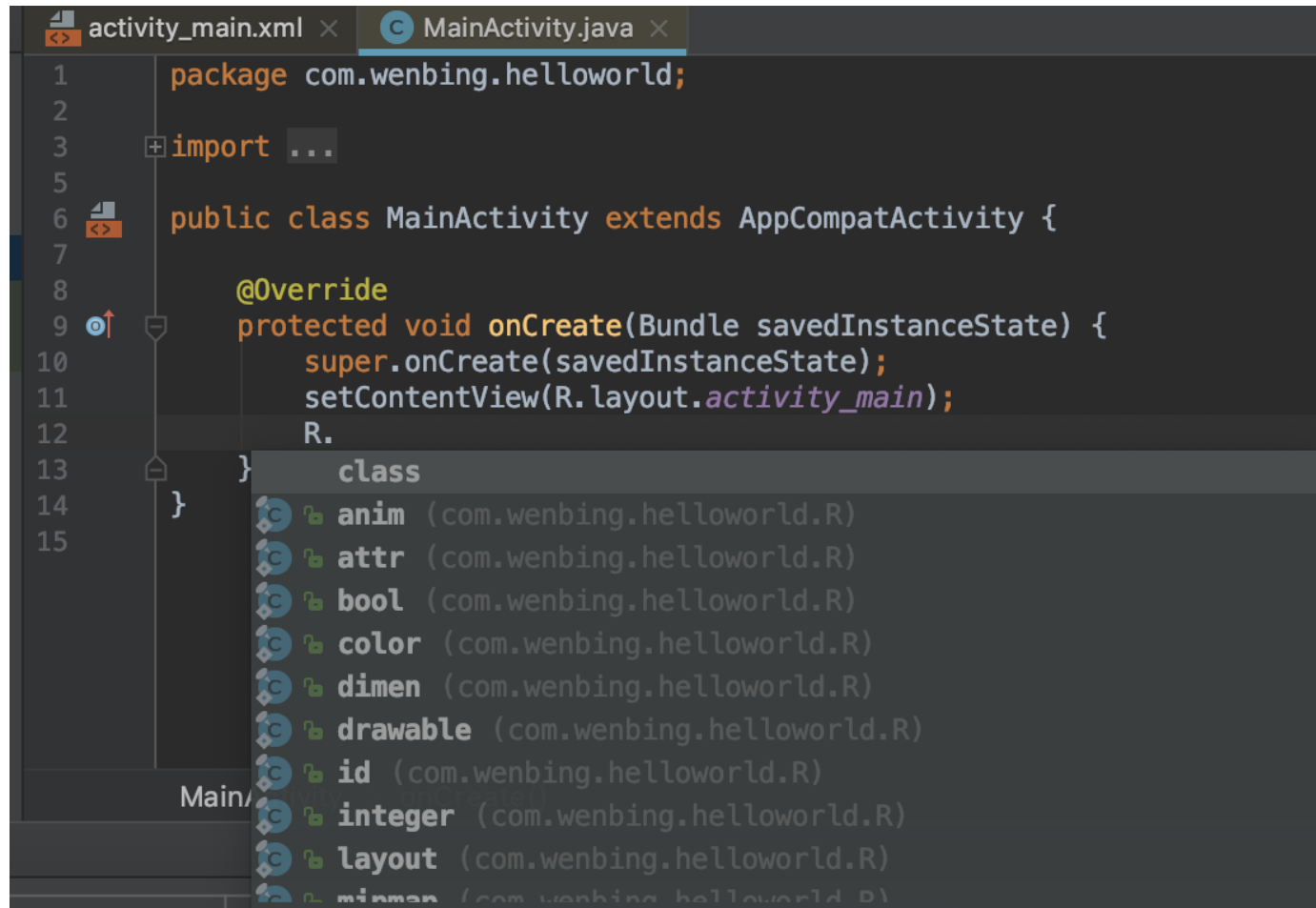




Using Code Completion

- Code completion: a tool that shows contextual options for completing the piece of code that you are trying to write
- Example:
 - ❑ In the editor tab for the MainActivity.java file, locate the line that reads
 - ❑ `setContentView(R.layout.activity_main);`
 - ❑ Place your cursor after this line and press the Enter key. On the new line, type the letter R, and then type a period, as shown here:
 - ❑ `R.`
 - ❑ Android Studio Code Completion should display a list of values that you could use to try to complete the code statement

Code completion example



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

class

- anim (com.wenbing.helloworld.R)
- attr (com.wenbing.helloworld.R)
- bool (com.wenbing.helloworld.R)
- color (com.wenbing.helloworld.R)
- dimen (com.wenbing.helloworld.R)
- drawable (com.wenbing.helloworld.R)
- id (com.wenbing.helloworld.R)
- integer (com.wenbing.helloworld.R)
- layout (com.wenbing.helloworld.R)
- minmax (com.wenbing.helloworld.R)

If the code completion window does not open, press Ctrl+Space to force it to open.

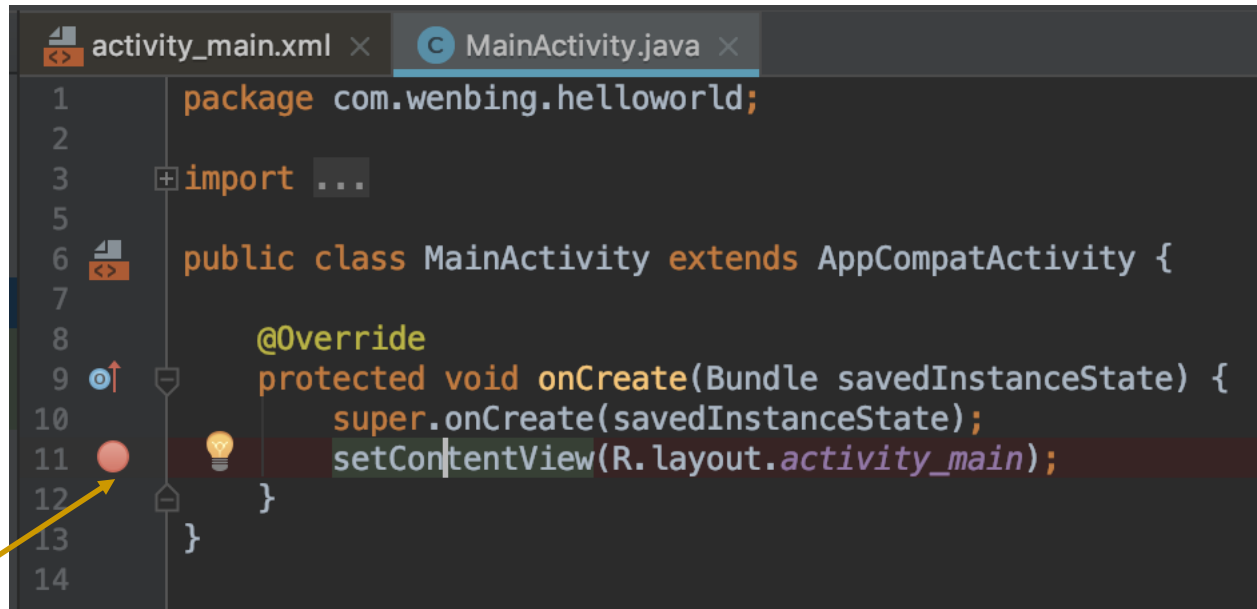
Debugging Your Application

- Setting Breakpoints
- Navigating paused code

Setting Breakpoints

- Common way to debug: set breakpoints to help you find what is going on with your code
- Breakpoints are a mechanism by which you can tell Android Studio to temporarily pause execution of your code, which allows you to examine the condition of your application
 - You can check on the values of variables in your application while you are debugging it
 - You can check whether certain lines of code are being executed as expected—or at all

Click the margin of the editor tab next to line of code you want to break at, to set a breakpoint. A red circle is placed in the margin, and the corresponding line is highlighted in red (clicked it again to remove the breakpoint)



Method
Breakpoint

A method breakpoint is represented by a red circle containing four dots placed at the method signature

Android Studio pauses execution when the method is hit, and it also automatically sets a corresponding breakpoint and pauses at the end of the method

Temporary Breakpoints

The screenshot shows the Android Studio IDE with a project named 'IDEExplorer'. The 'MainActivity.java' file is open, showing the following code:

```
package com.jfdimarzio.ideexplorer;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.bluetooth.BluetoothAdapter;  
  
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

A temporary breakpoint is set on line 12, which is the line containing `setContentView(R.layout.activity_main);`. A dialog box is open for configuring the breakpoint, showing:

- ☒ Enabled
- ☒ Suspend ☐ All ☒ Thread
- Condition: (empty field)
- More (Ctrl+Shift+F8)
- Done

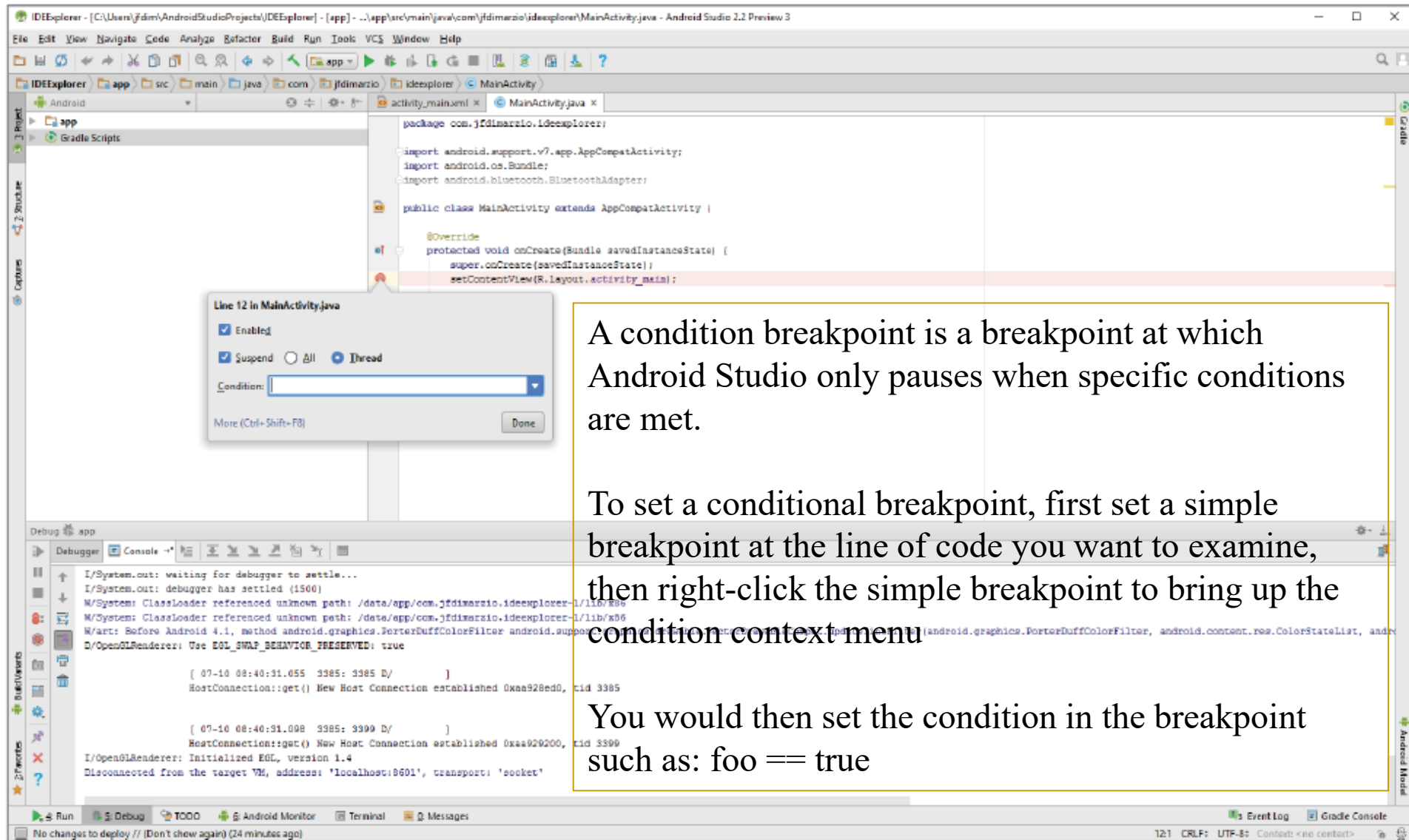
The bottom panel shows the 'Debugger' tab with the following log output:

```
I/System.out: waiting for debugger to settle...  
I/System.out: debugger has settled (1500)  
W/System: ClassLoader referenced unknown path: /data/app/com.jfdimarzio.ideexplorer-1/lib/x86  
W/System: ClassLoader referenced unknown path: /data/app/com.jfdimarzio.ideexplorer-1/lib/x86  
W/art: Before Android 4.1, method android.graphics.PorterDuffColorFilter android.support.graphics.drawable.VectorDrawableCompat.updateTintFilter(android.graphics.PorterDuffColorFilter, android.content.res.ColorStateList, and  
D/OpenGLESRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true  
  
[ 07-10 08:40:31.055 3385: 3385 D/ ]  
HostConnection::get() New Host Connection established 0xaa928ed0, tid 3385  
  
[ 07-10 08:40:31.098 3385: 3390 D/ ]  
HostConnection::get() New Host Connection established 0xaa929200, tid 3390  
I/OpenGLESRenderer: Initialized EGL, version 1.4  
Disconnected from the target VM, address: 'localhost:8601', transport: 'socket'
```

Annotations on the image:

- Useful in a loop**: A yellow box highlights the code area.
- To set a temporary breakpoint, place your cursor at the location in the code where you want it to break and select Run ⇌ Toggle Temporary Line Breakpoint.**: A yellow box highlights the text.
- Android Studio only stops at this breakpoint the first time your code enters it**: A yellow box highlights the text.

Conditional Breakpoints



The screenshot shows the Android Studio IDE with a project named 'IDEExplorer'. The 'MainActivity.java' file is open, and a breakpoint is set on line 12. A context menu is displayed over the breakpoint, showing options to 'Enable', 'Suspend', 'All', and 'Thread'. The 'Condition' field is empty, and the 'More (Ctrl+Shift+F8)' option is visible. The bottom panel shows the 'Debugger' tab with a console log.

Line 12 in MainActivity.java

- ☒ Enabled
- ☒ Suspend ☐ All ☒ Thread
- Condition:
- More (Ctrl+Shift+F8)
- Done

Debugger Console Log:

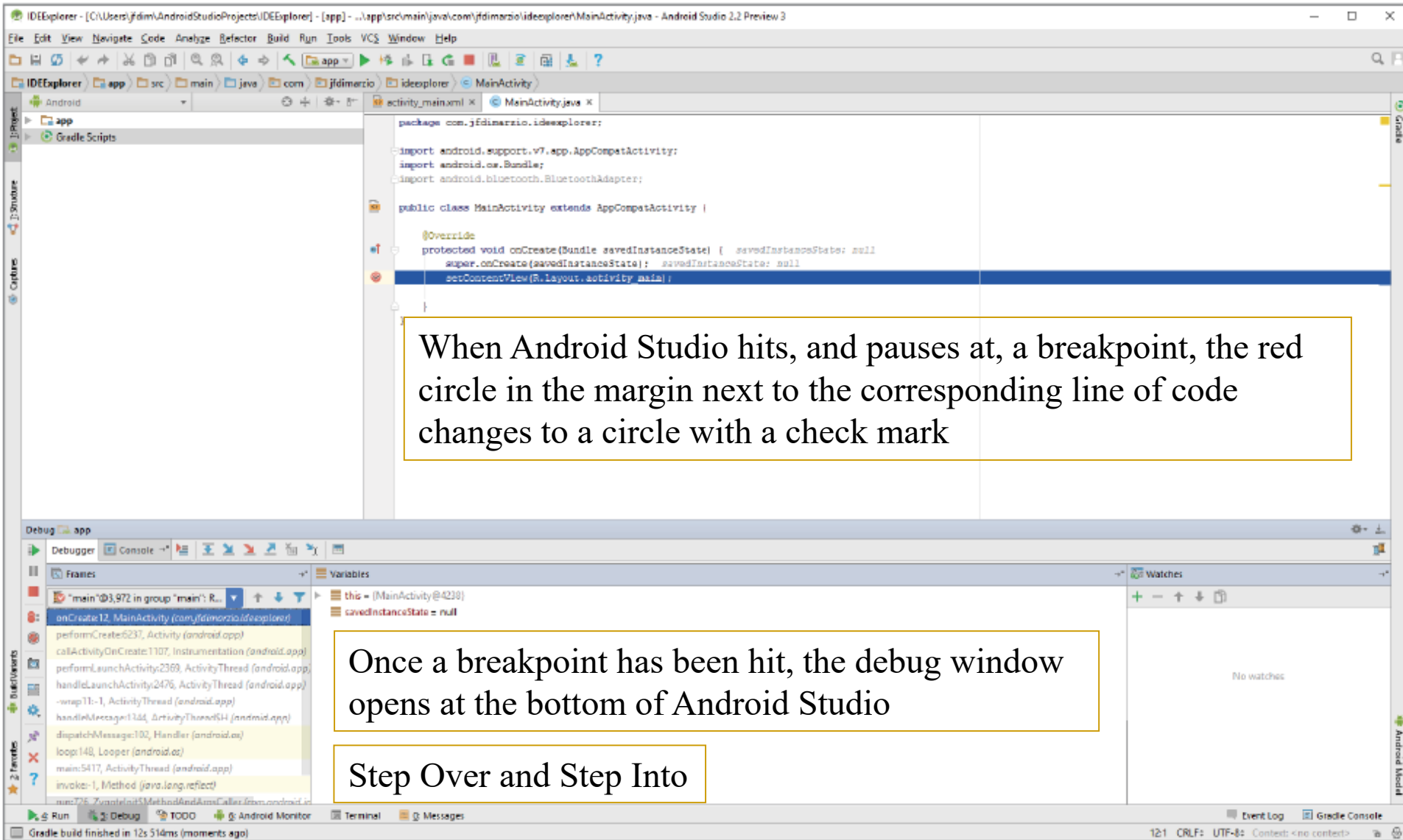
```
I/System.out: waiting for debugger to settle...
I/System.out: debugger has settled (1500)
W/System: ClassLoader referenced unknown path: /data/app/com.jfdimarzio.ideexplorer-1/11b/x86
W/System: ClassLoader referenced unknown path: /data/app/com.jfdimarzio.ideexplorer-1/11b/x86
W/art: Before Android 4.1, method android.graphics.PorterDuffColorFilter android.support.graphics.drawable.Drawable$1.setColorFilter(android.graphics.PorterDuffColorFilter, android.content.res.ColorStateList, android.graphics.PorterDuffColorFilter) is not supported
D/OpenGLES: Use EGL_SWAP_BEHAVIOR_PRESERVED: true
[ 07-10 08:40:31.055 3385: 3385 D/ ]
HostConnection::get() New Host Connection established 0xaa928ed0, tid 3385
[ 07-10 08:40:31.098 3385: 3390 D/ ]
HostConnection::get() New Host Connection established 0xaa929200, tid 3390
I/OpenGLES: Initialized EGL, version 1.4
Disconnected from the target VM, address: 'localhost:8601', transport: 'socket'
```

A condition breakpoint is a breakpoint at which Android Studio only pauses when specific conditions are met.

To set a conditional breakpoint, first set a simple breakpoint at the line of code you want to examine, then right-click the simple breakpoint to bring up the condition context menu

You would then set the condition in the breakpoint such as: `foo == true`

Navigating Paused Code



Publishing Your Application

- Involves deploying your application to Google Store for others to use or enjoy
- You must first generate Android Application Package (APK)
- APK is a compiled and executable version of your application
- Signing it identifies application developer to Google and users who will be installing the application

1. Generate a signed APK from your code by selecting Build ⇨ Generate Signed APK from the Menu bar to bring up the Generate Signed APK window as shown in Figure 2-17.

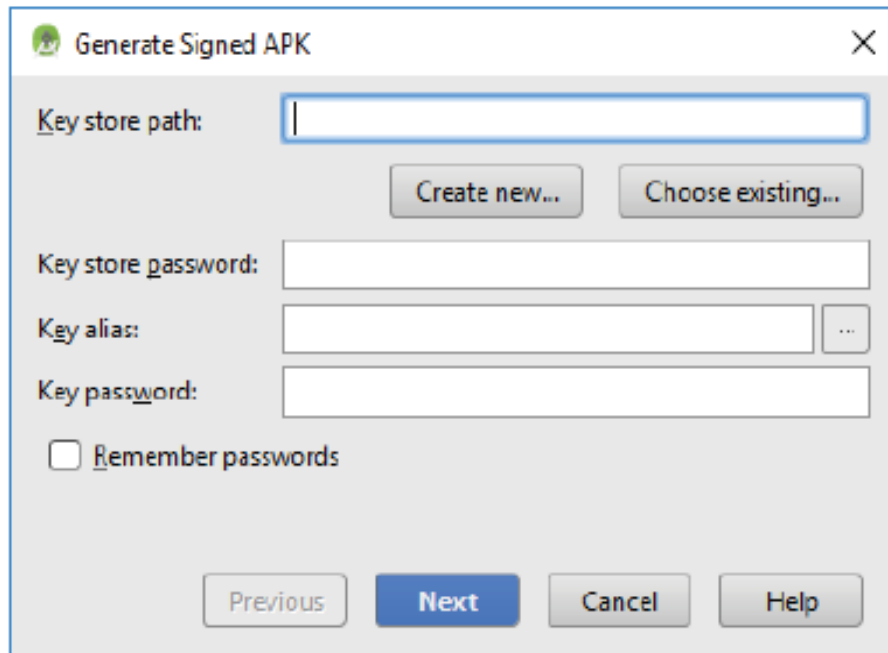


FIGURE 2-17

2. Assuming you have never published an application from Android Studio, you need to create a new key store. Click the Create New button to display the New Key Store window (see Figure 2-18).
3. Fill out all of the information on this form because it pertains to your entity and application. Notice that there are two places for a password. These are the passwords for your key store /