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

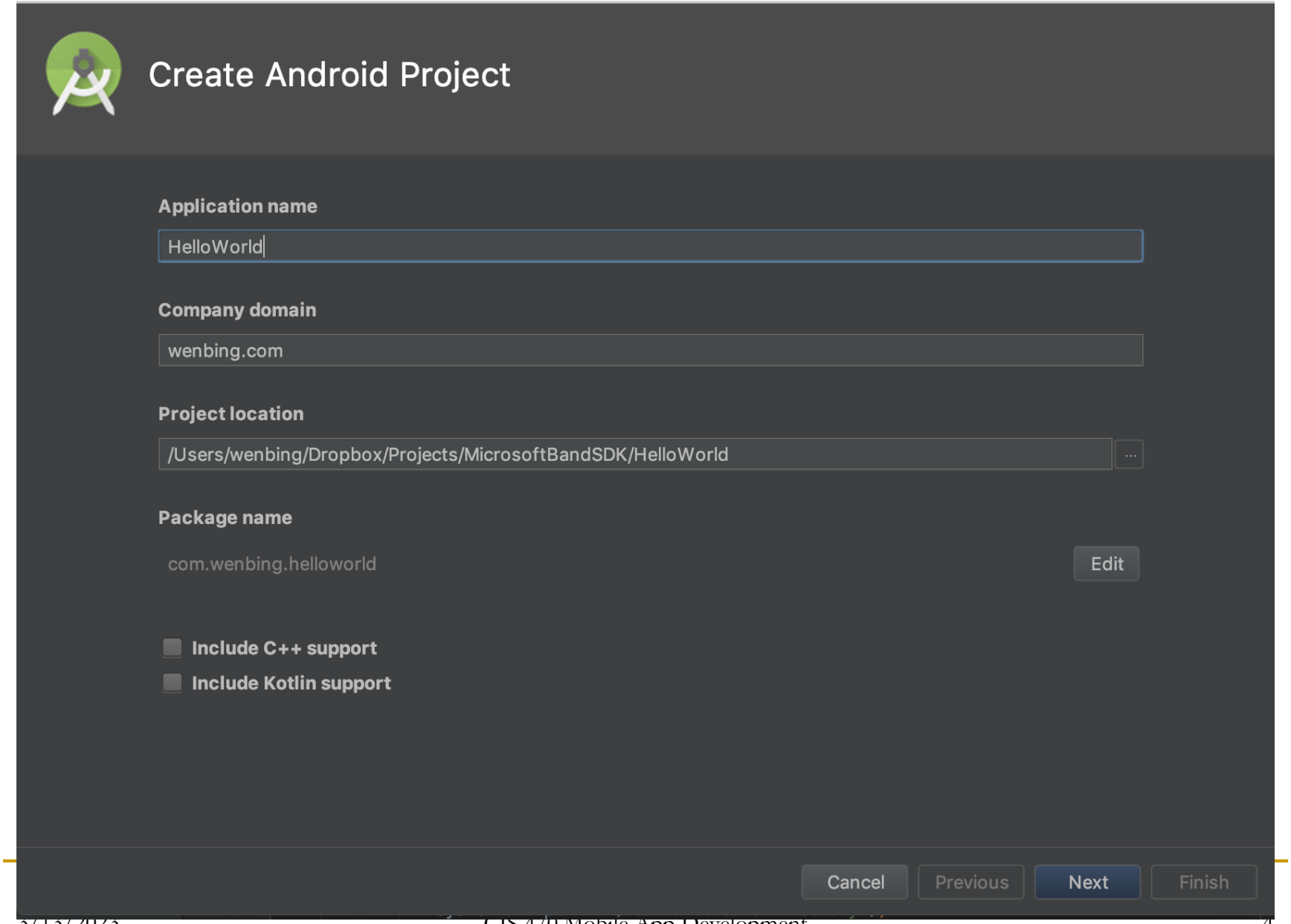
Android Application Development Using Android Studio

- 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



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.

- [x] **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**

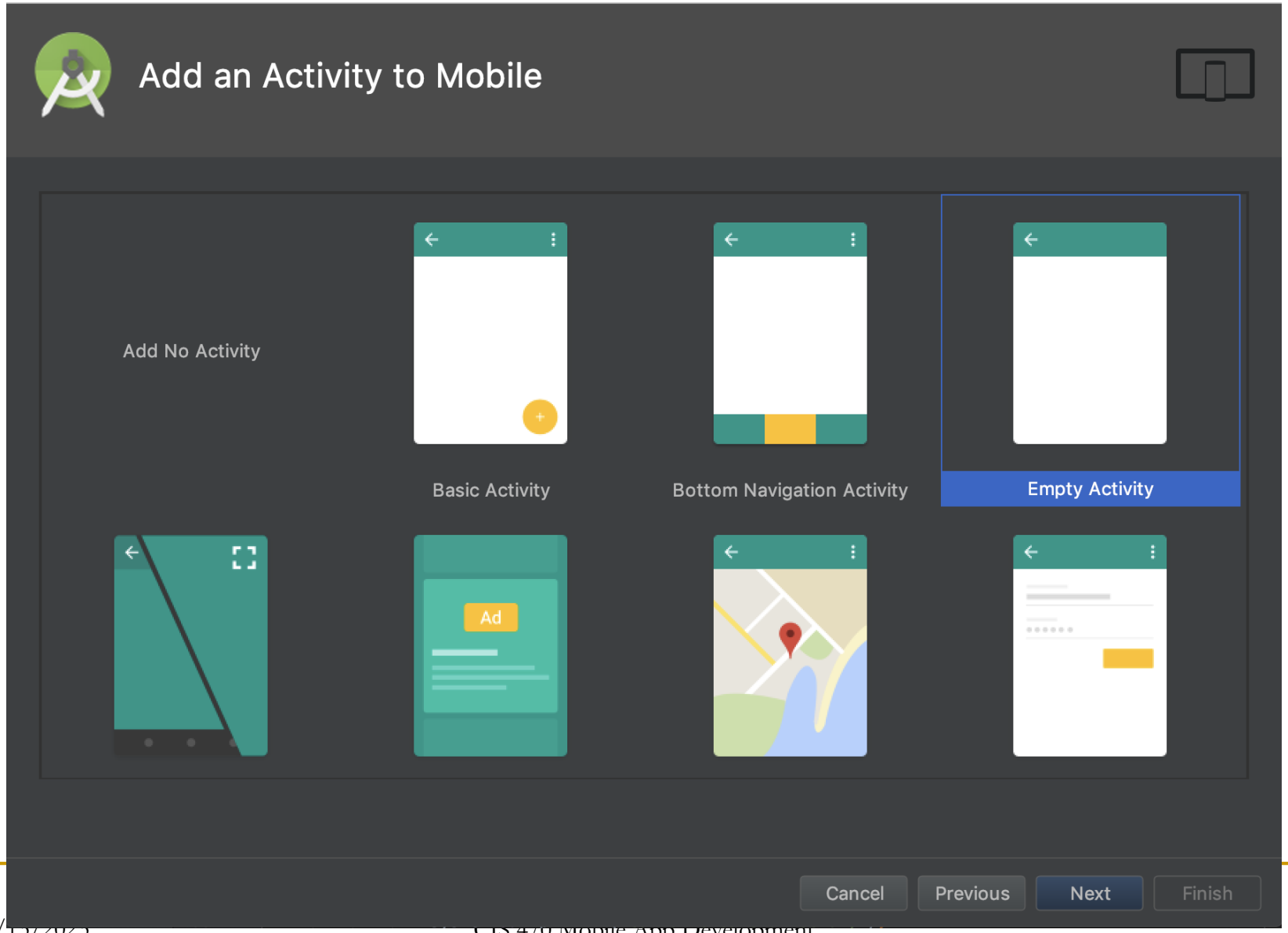
  API 21: Android 5.0 (Lollipop)

- [ ] **Android Auto**
- [ ] **Android Things**

  API 24: Android 7.0 (Nougat)

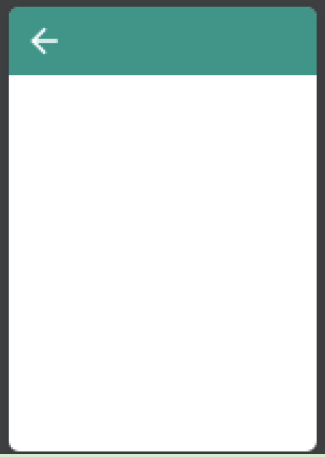
Cancel | Previous | Next | Finish

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

Cancel | Previous | Next | Finish

The Android Studio IDE

main > java > com > wenbing > helloworld > MainA

app

Android

- app
- Gradle Scripts

activity_main.xml | MainActivity.java

```
1  package com.wenbing.helloworld;
2
3  import ...
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
```

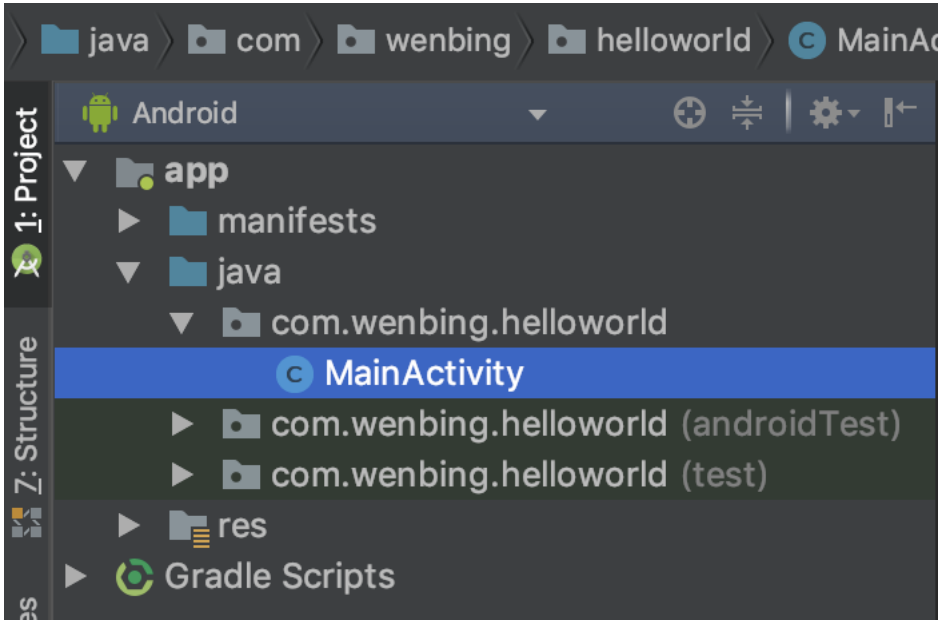
1: Project | 7: Structure | Captures | 2: Favorites | Gradle | Device File Explorer

Terminal | Build | 6: Logcat | TODO | Event Log

Gradle build finished in 8s 100ms

1:1 LF UTF-8 Context: <no context>

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

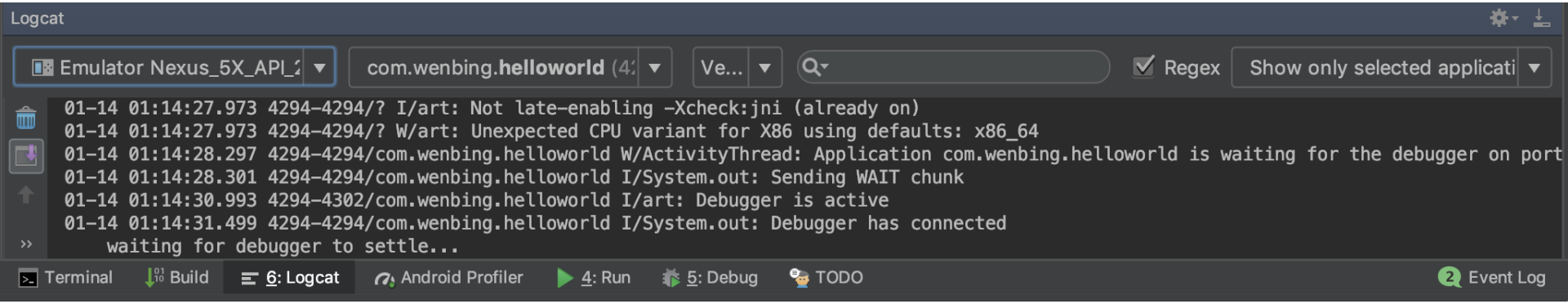


On the right side of the IDE are the Editor tabs. The Editor tabs are where you write and work with your code files.

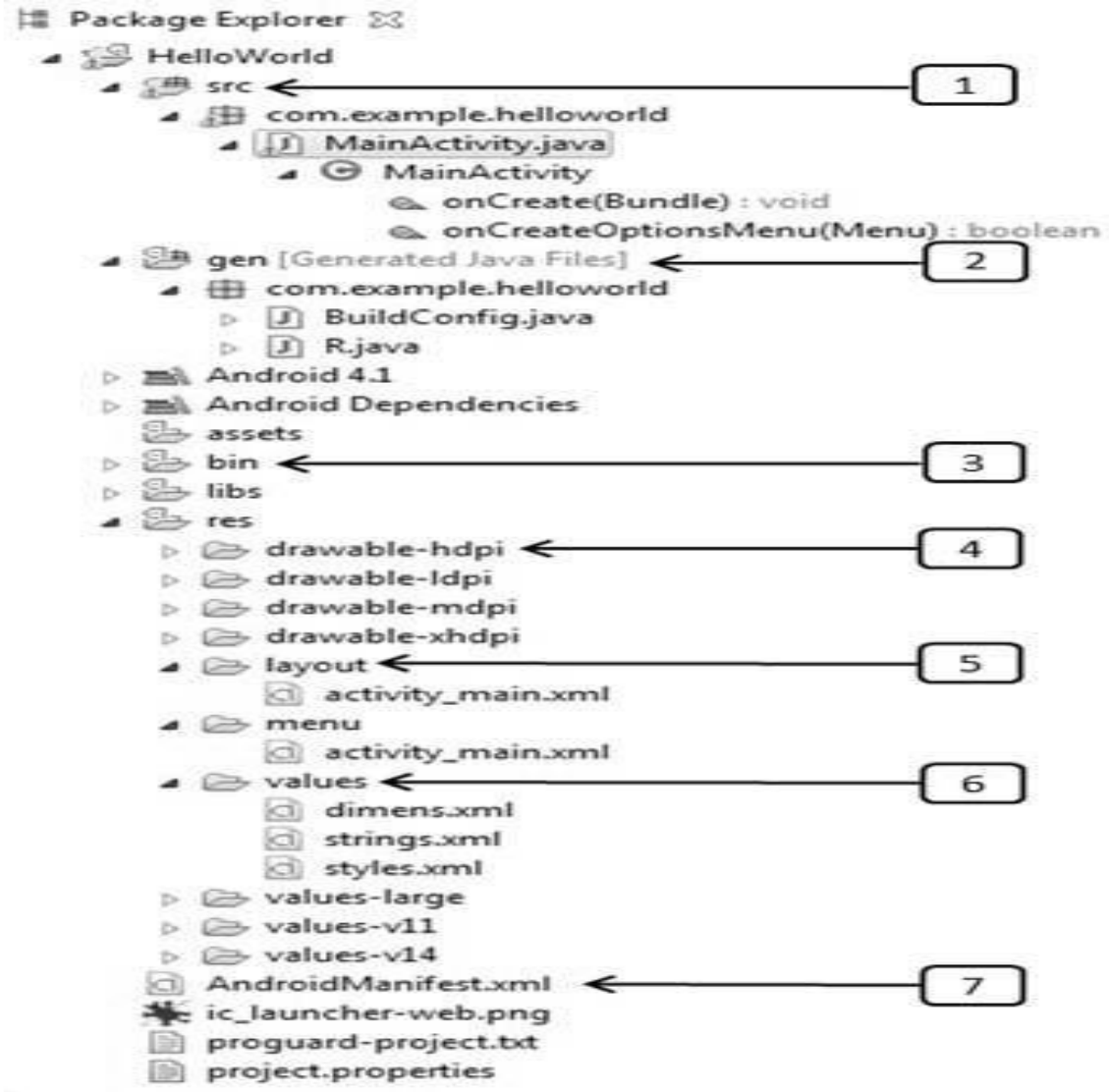
activity_main.xml | MainActivity.java

```java
package com.wenbing.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

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

- 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.



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 anMainActivity.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

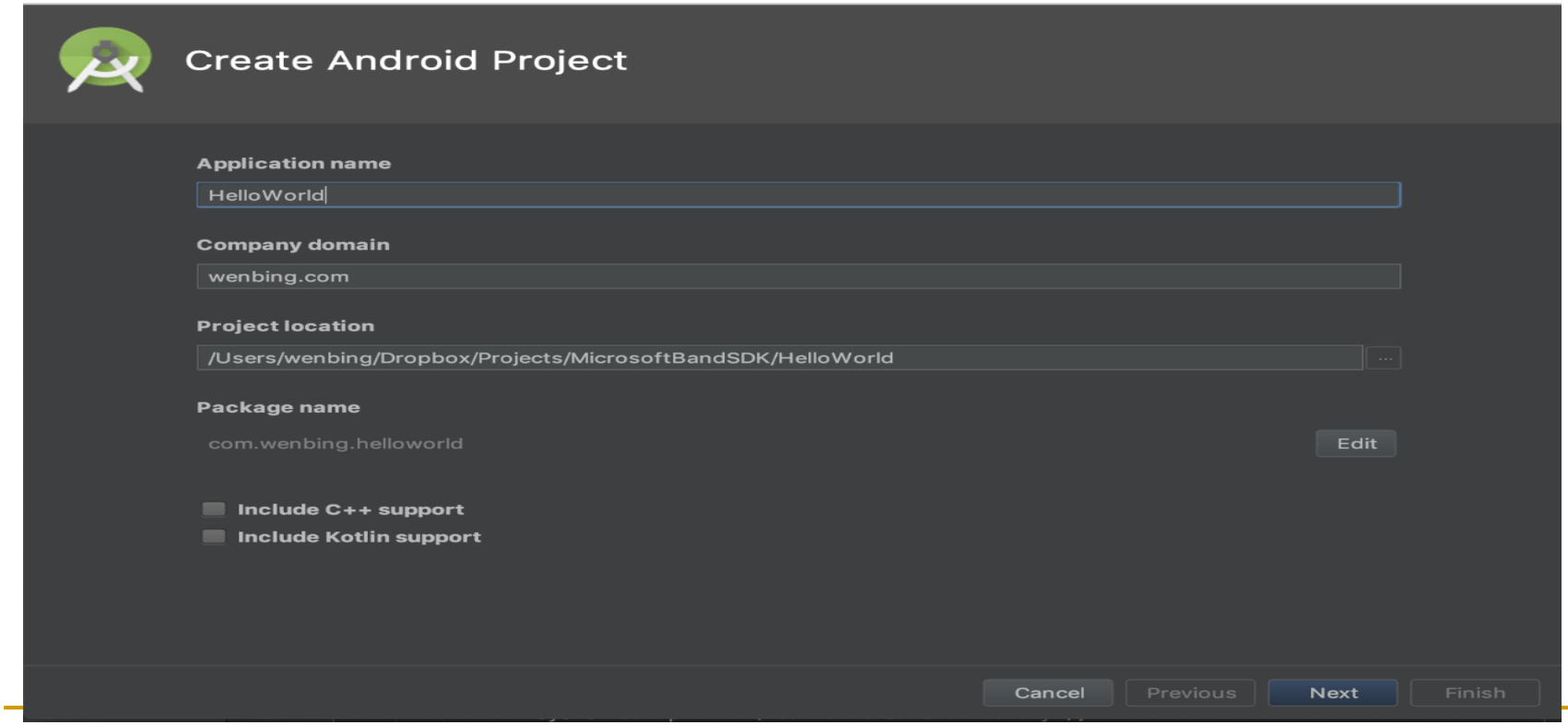
Important parts of Android App

Important parts of an Android application include the following:

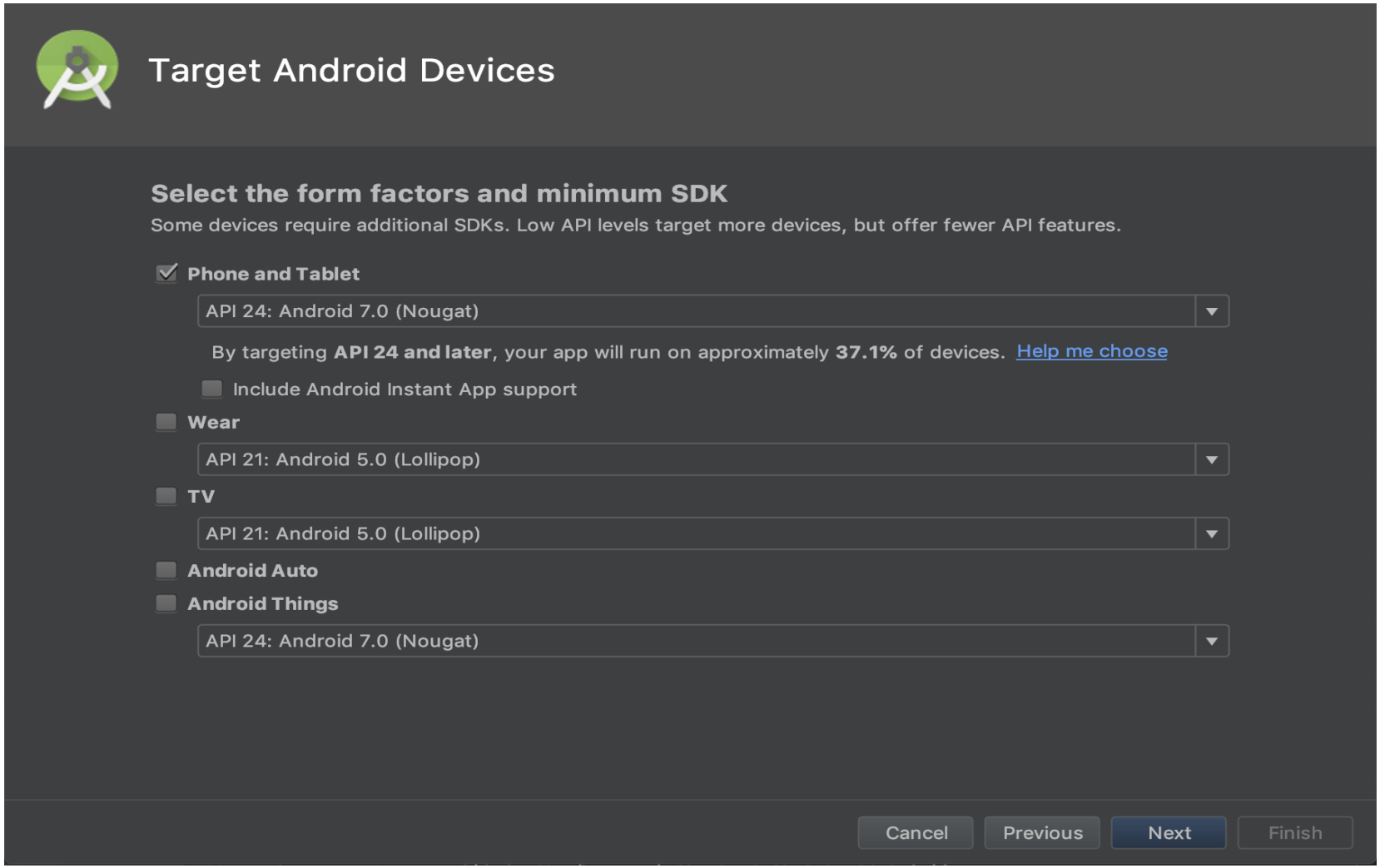
- **Activities**
  The Activities are the main Java classes, that contain the Android code with which we are going to develop, **what do we want the application to do**.
- **Layouts**
  The Layouts are the main xml files, that contain the Android xml code with which we are going to develop, **how will the application views look like**.

Creating your first Android App

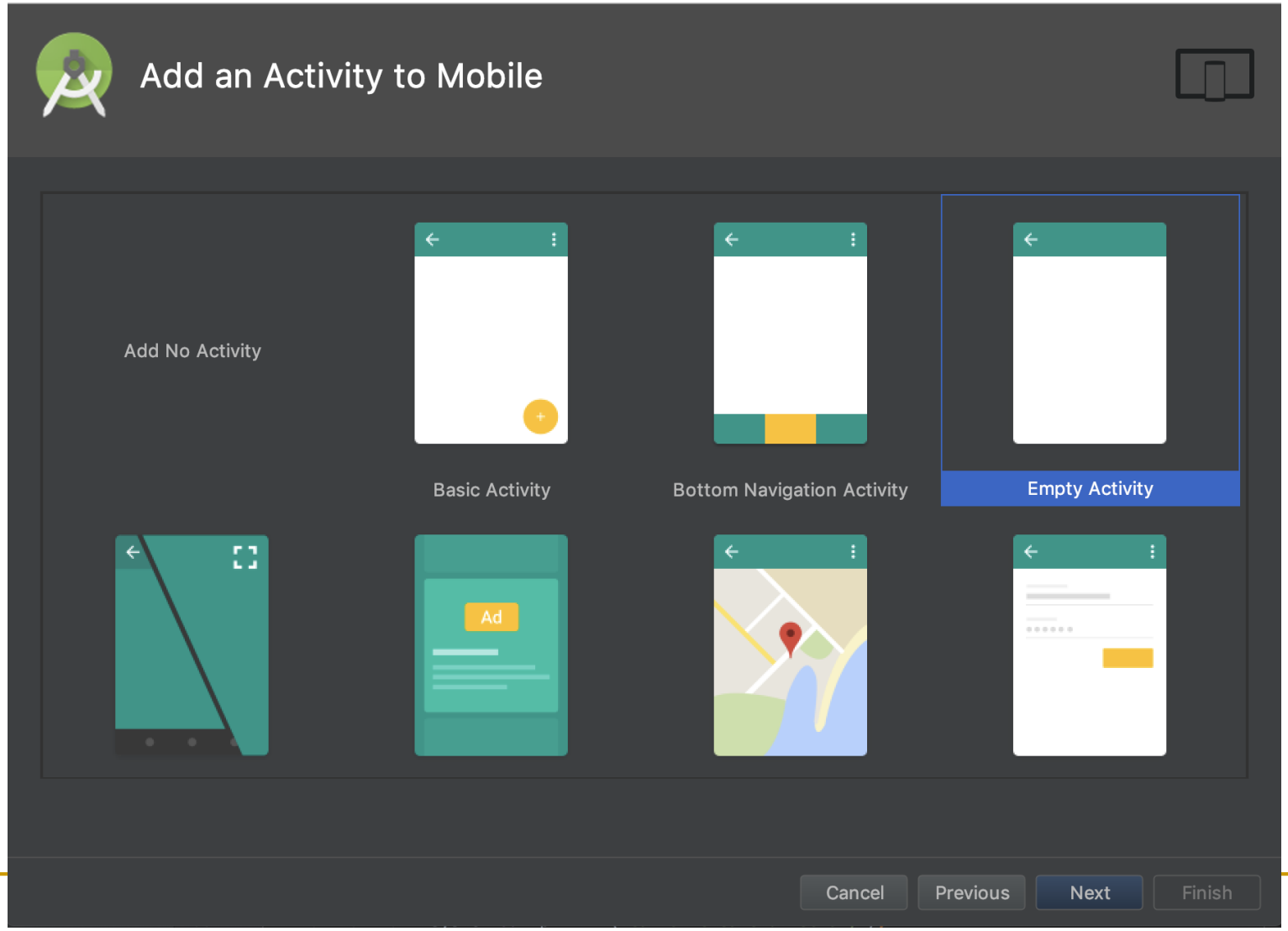
- File -> New->New Project
- Name the app: HelloWorld
- Then select default option for all remaining steps



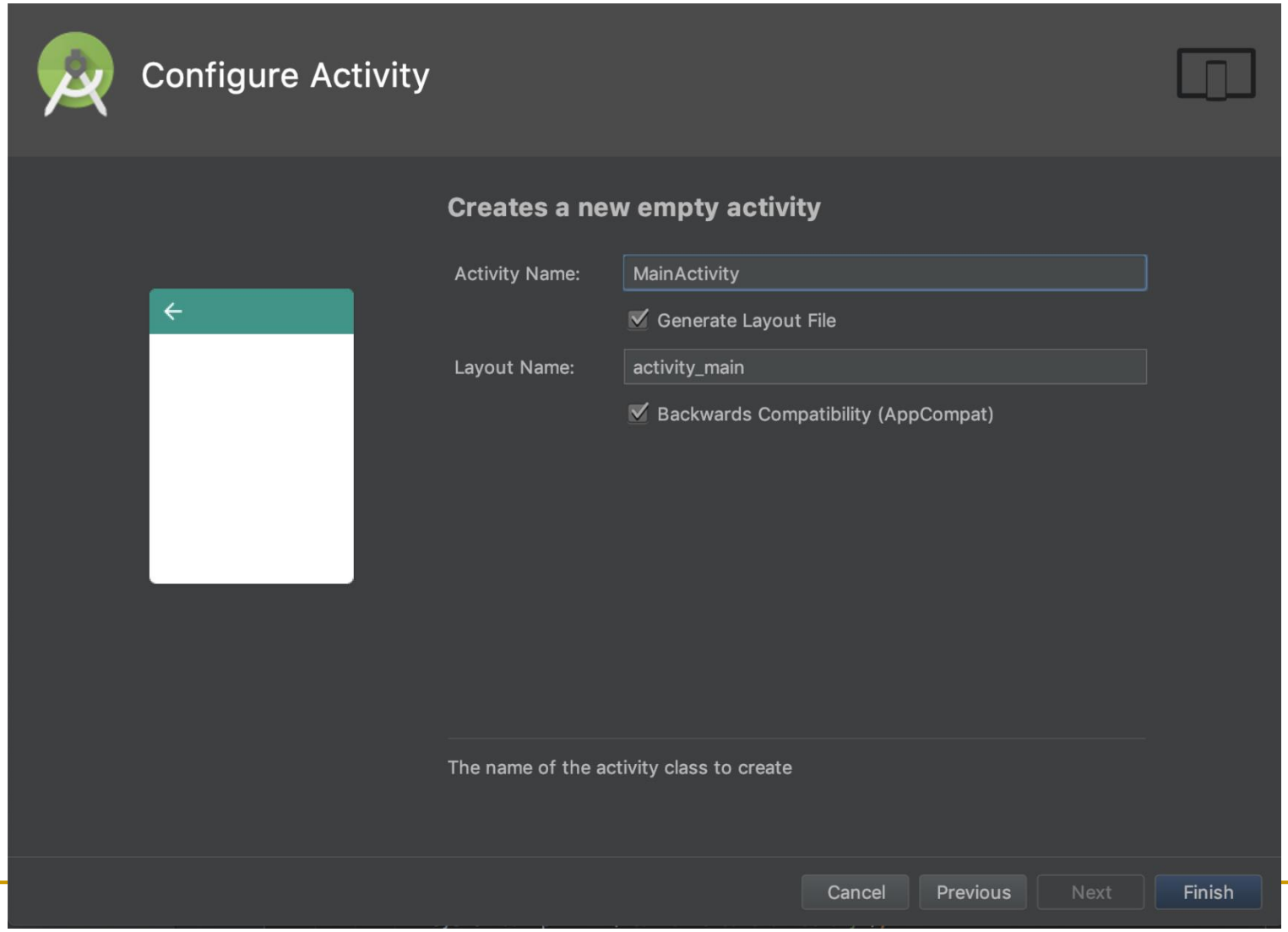
Creating your first Android App



Creating your first Android App



Creating your first Android App



Creating your first Android App

main > java > com > wenbing > helloworld > MainA

app

Android

- app
- Gradle Scripts

activity_main.xml | MainActivity.java

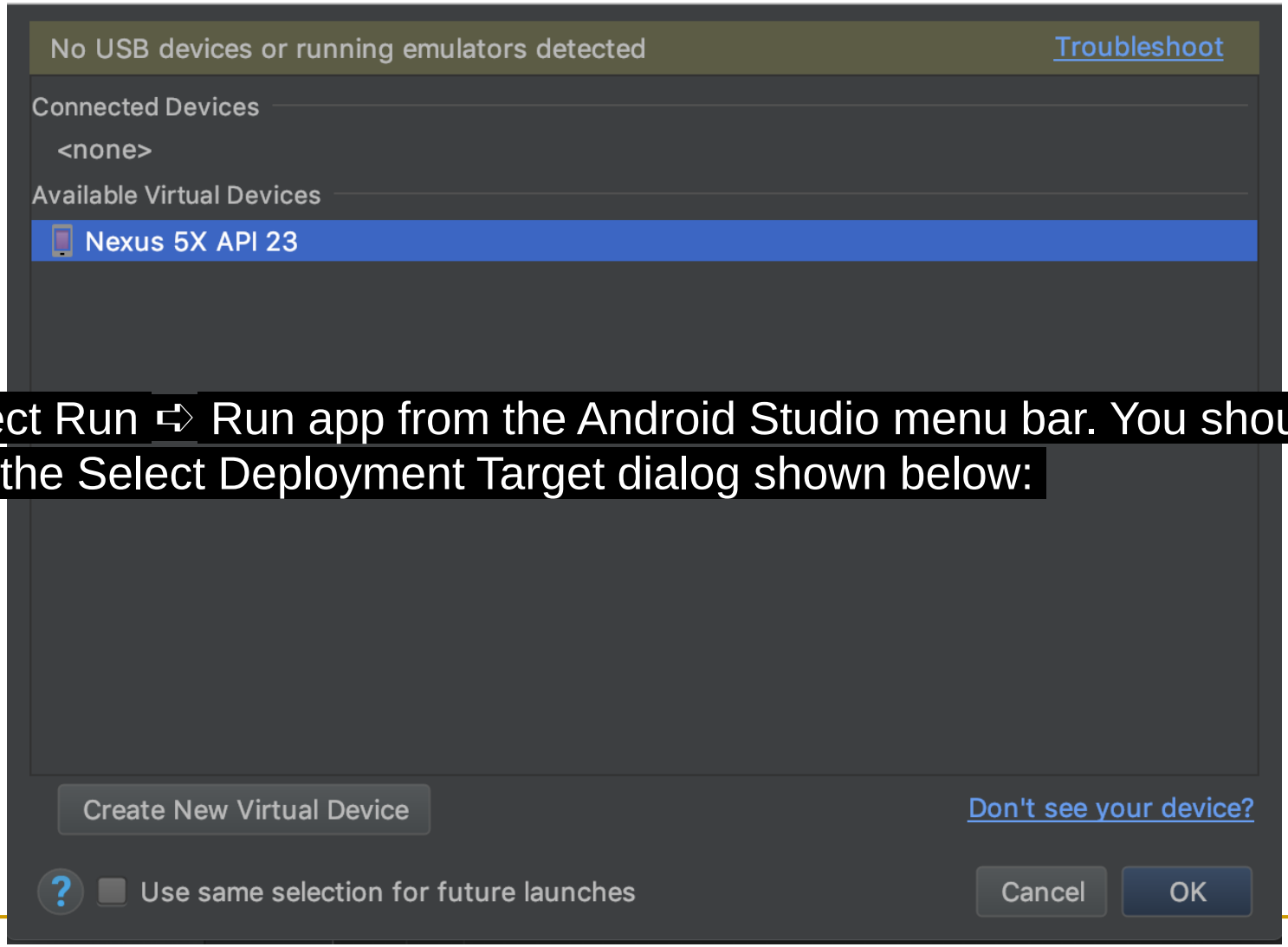
```java
package com.wenbing.helloworld;

import ...

public class MainActivity extends AppCompatActivity {

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

Terminal | Build | 6: Logcat | TODO | Event Log

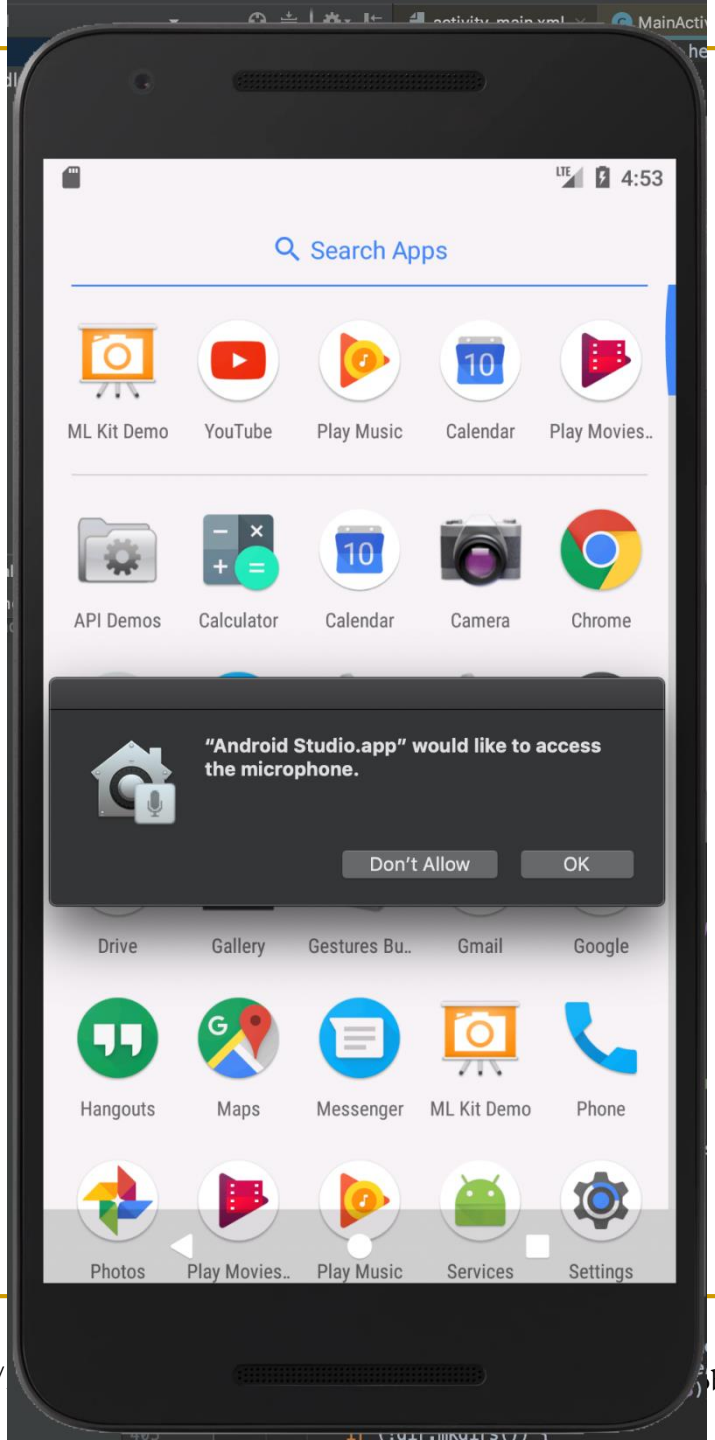
Gradle build finished in 8s 100ms

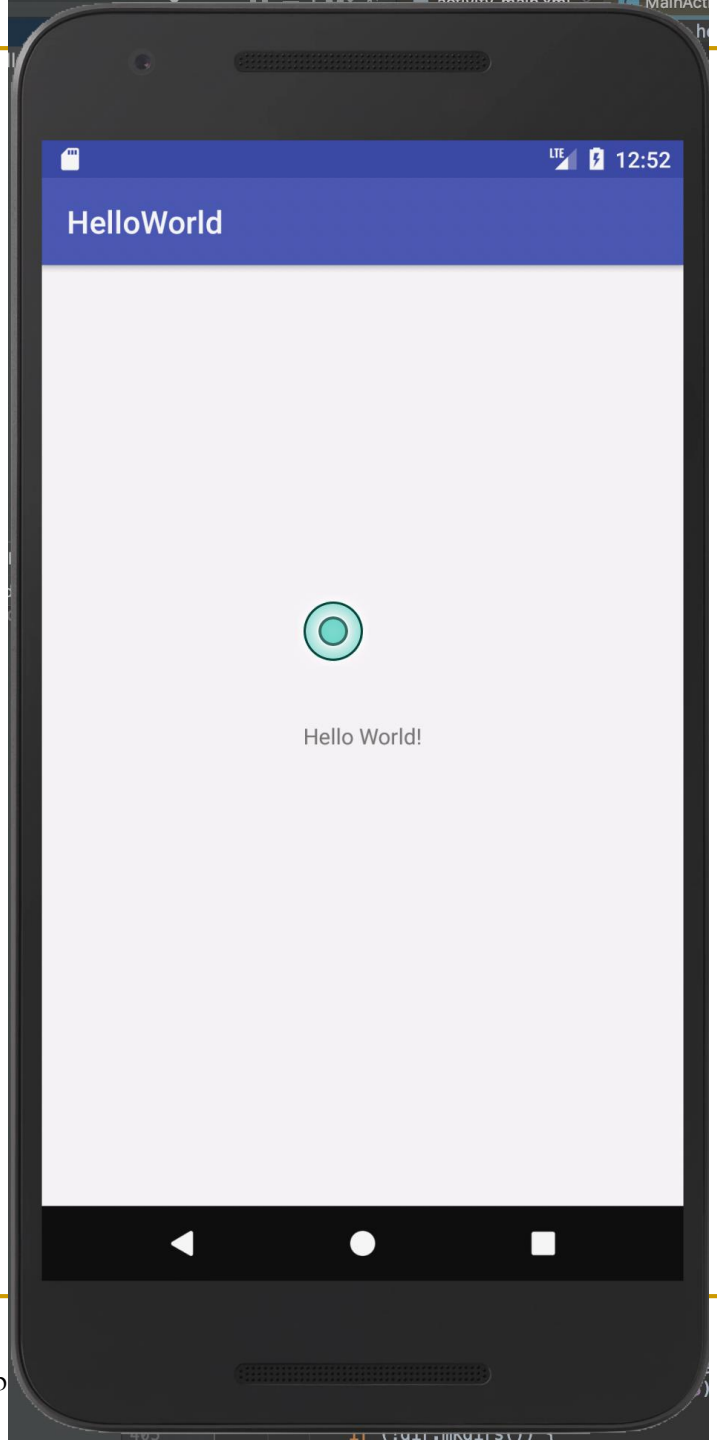
1:1 LF UTF-8 Context: <no context>

Launching your first Android App



Select Run ➪ Run app from the Android Studio menu bar. You should see the Select Deployment Target dialog shown below:


4:53
Search Apps
ML Kit Demo
YouTube
Play Music
Calendar
Play Movies..
API Demos
Calculator
Calendar
Camera
Chrome
"Android Studio.app" would like to access the microphone.
Don't Allow
OK
Drive
Gallery
Gestures Bu..
Gmail
Google
Hangouts
Maps
Messenger
ML Kit Demo
Phone
Photos
Play Movies..
Play Music
Services
Settings


12:52
HelloWorld
Hello World!

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

```
activity_main.xml   MainActivity.java
1   package com.wenbing.helloworld;
2
3   import ...
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)
    mipmap (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)

activity_main.xml | MainActivity.java

```
1   package com.wenbing.helloworld;
2
3   import ...
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
```

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

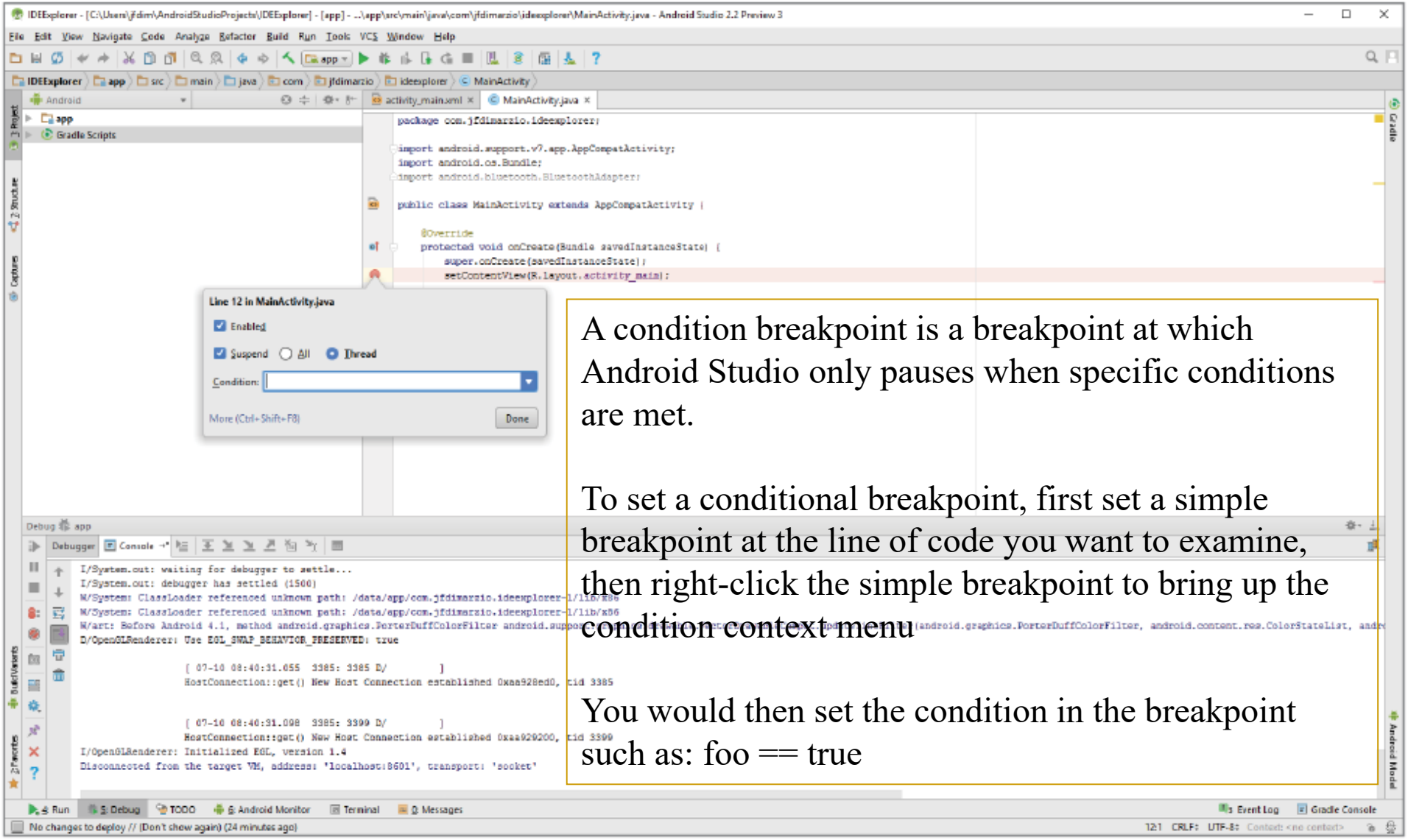
Temporary Breakpoints

Useful in a loop

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.

Android Studio only stops at this breakpoint the first time your code enters it

Conditional Breakpoints



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

When Android Studio hits, and pauses at, a breakpoint, the red circle in the margin next to the corresponding line of code changes to a circle with a check mark

Once a breakpoint has been hit, the debug window opens at the bottom of Android Studio

Step Over and Step Into

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.

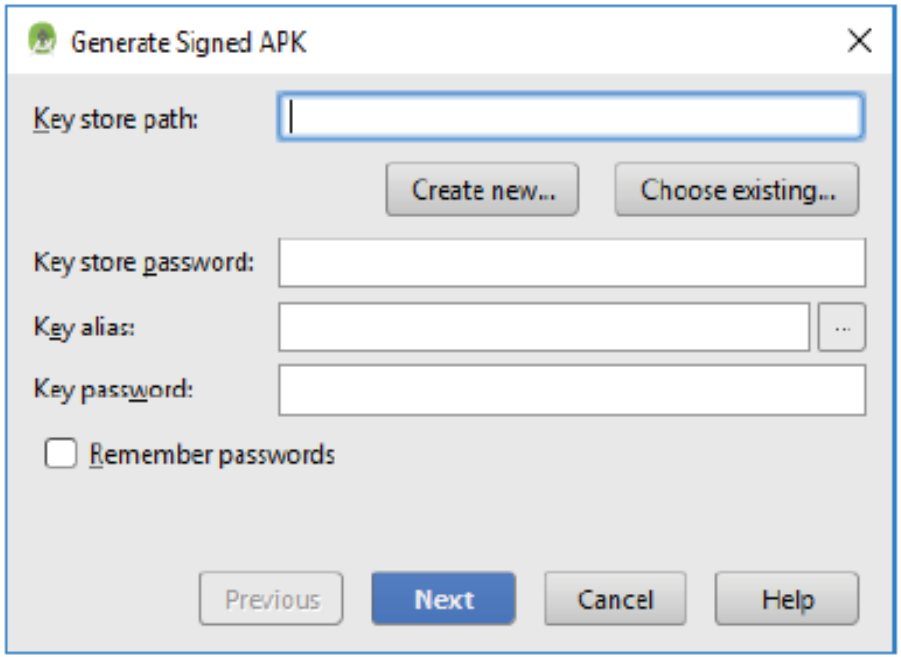


FIGURE 2-17

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/