Mobile Programming

Dr. Nader Mahmoud

Lecturer at Computer Science department

Course Content

Agenda

- **Chapter 2 - Using Android Studio for Android Development**
  - **Exploring the IDE**
  - **Using Code Completion**
  - **Debugging Your Application**
    - Setting Breakpoints
    - Navigating Paused Code
  - **Publishing Your Application**
    - Generating a Signed APK

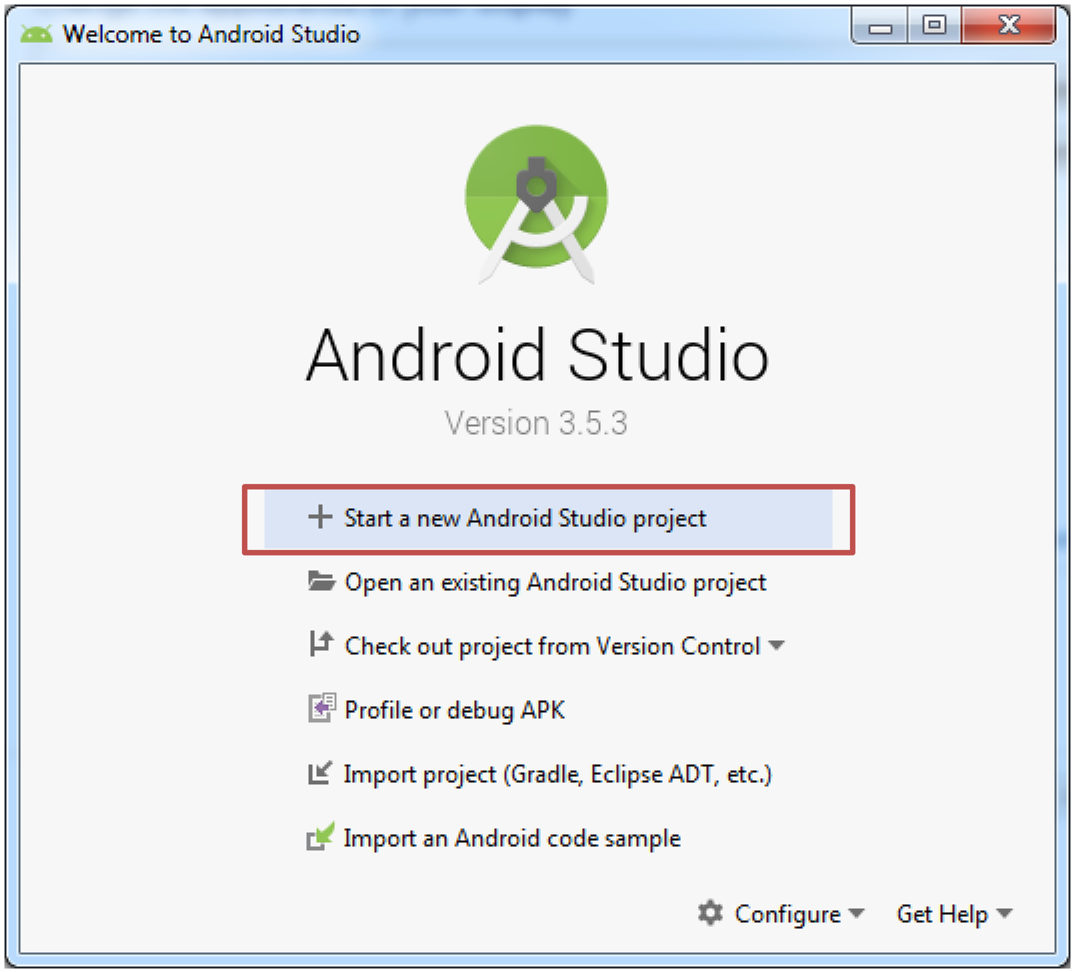
Exploring the IDE

- Android Studio contains various features to help everyone from beginners to professional developers.

- The Integrated Development Environment (IDE) is the interface between you and Android Studio.

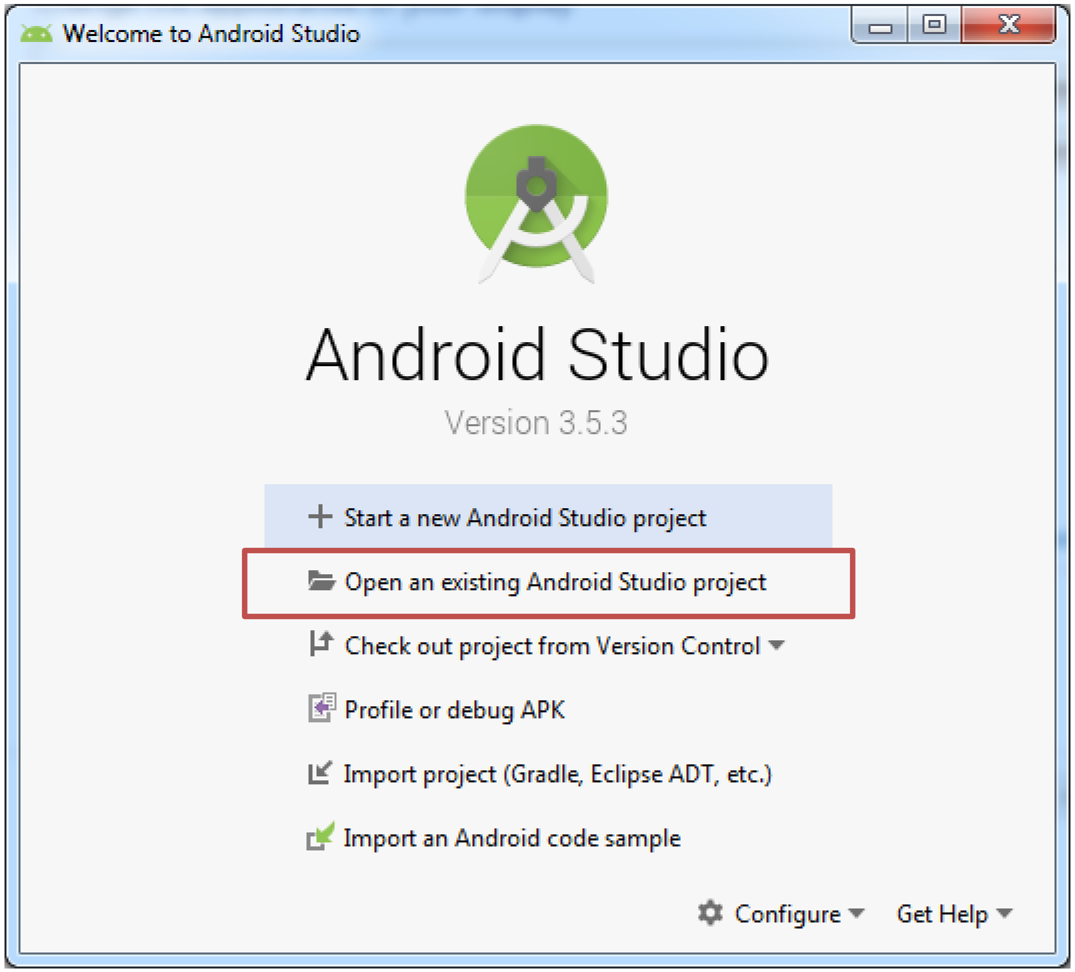
- We will examine each interface in android studio in order to be able to utilize all the capabilities of android studio

Exploring the IDE


Welcome to Android Studio
Android Studio
Version 3.5.3
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

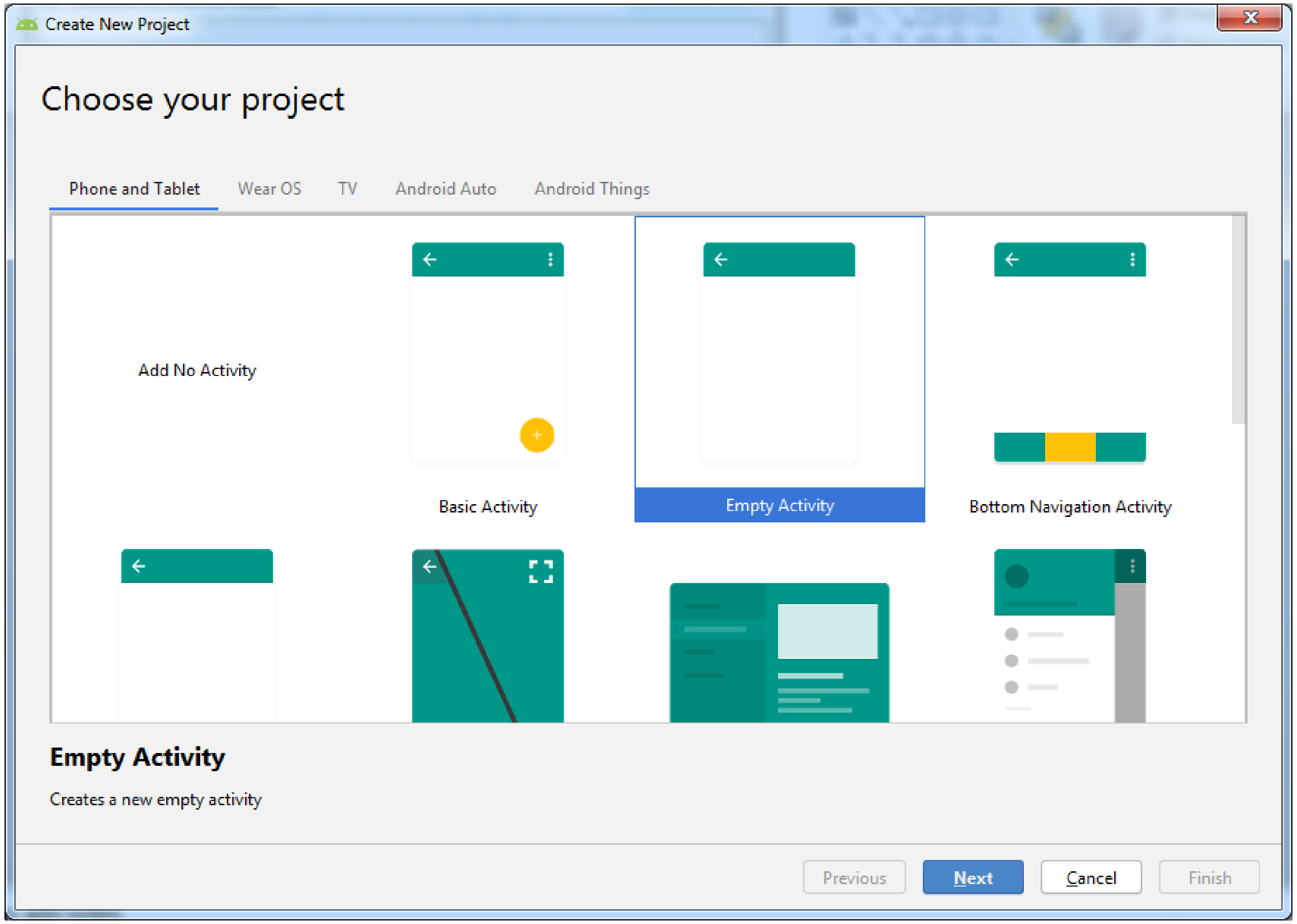
To start a new project

Exploring the IDE


Welcome to Android Studio
Android Studio
Version 3.5.3
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

To open an existing project

Exploring the IDE


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

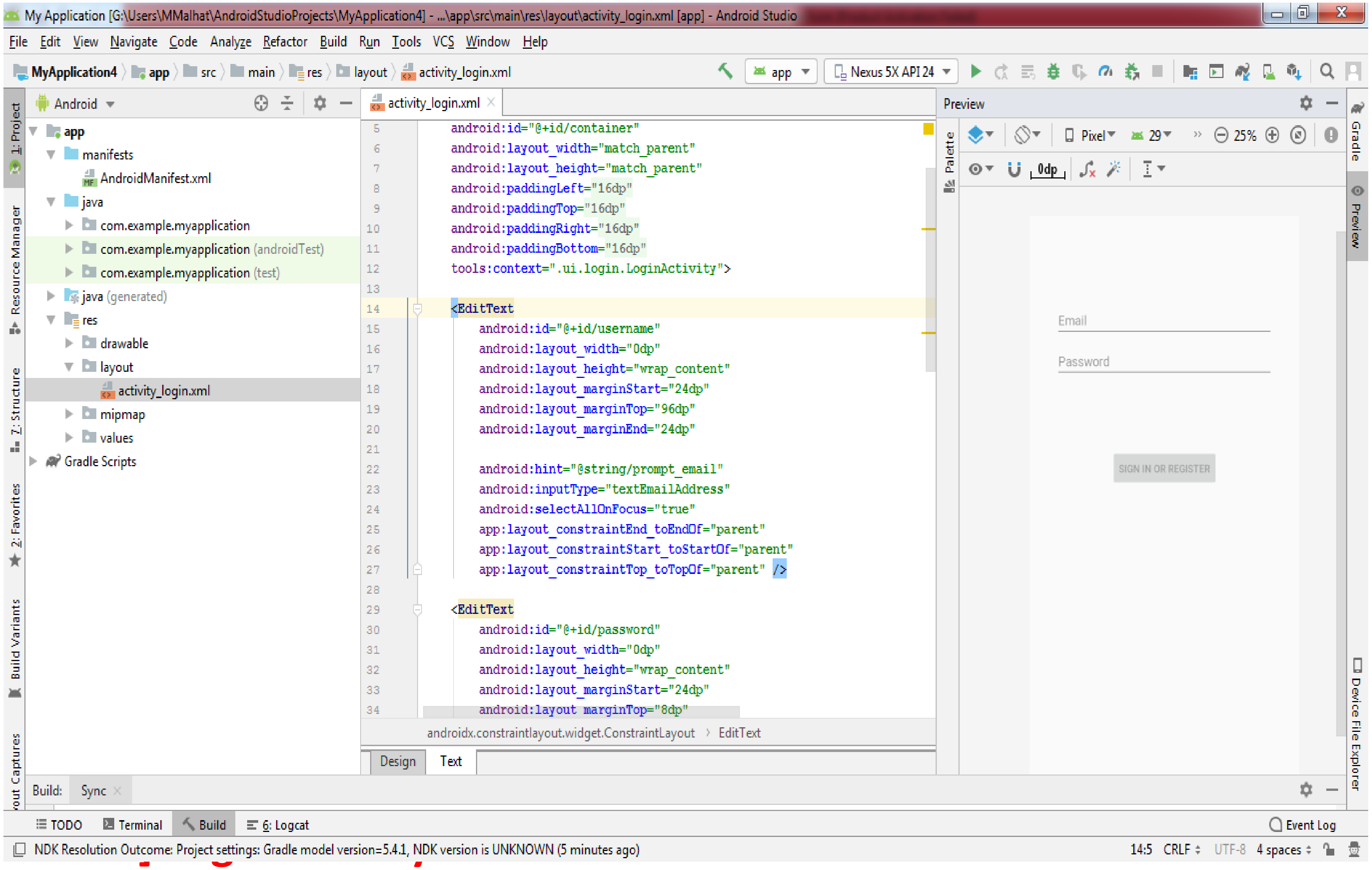
Empty Activity

Creates a new empty activity

Previous | Next | Cancel | Finish

The android studio supports you with some of the common activities that are needed for most applications

Exploring the IDE

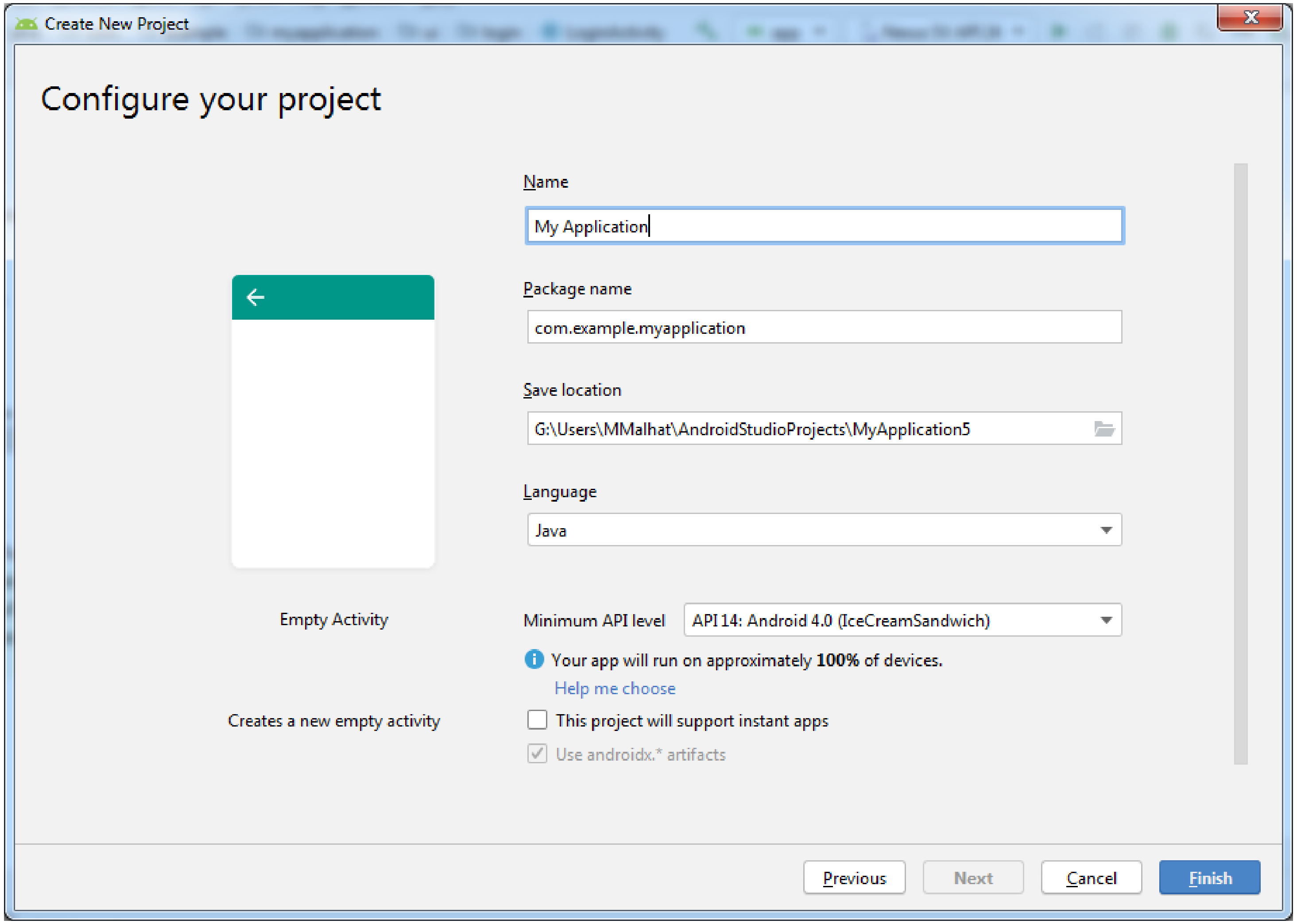

My Application [G:\Users\MMalhat\AndroidStudioProjects\MyApplication4] - ...\app\src\main\res\layout\activity_login.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MyApplication4 app src main res layout activity_login.xml
app
Nexus 5X API 24
Android
app
manifests
AndroidManifest.xml
java
com.example.myapplication
com.example.myapplication (androidTest)
com.example.myapplication (test)
java (generated)
res
drawable
layout
activity_login.xml
mipmap
values
Gradle Scripts
activity_login.xml
Preview
Pixel
29
25%
0dp
Email
Password
SIGN IN OR REGISTER
androidx.constraintlayout.widget.ConstraintLayout › EditText
Design
Text
Build: Sync
TODO
Terminal
Build
6: Logcat
Event Log
1: Project
Resource Manager
7: Structure
2: Favorites
Build Variants
Layout Captures
Palette
Gradle
Preview
Device File Explorer

```
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="16dp"
    tools:context=".ui.login.LoginActivity">

    <EditText
        android:id="@+id/username"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="96dp"
        android:layout_marginEnd="24dp"

        android:hint="@string/prompt_email"
        android:inputType="textEmailAddress"
        android:selectAllOnFocus="true"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/password"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="8dp"
```

NDK Resolution Outcome: Project settings: Gradle model version=5.4.1, NDK version is UNKNOWN (5 minutes ago)

14:5 CRLF UTF-8 4 spaces

Exploring the IDE



Create New Project

Configure your project

Name

My Application

Package name

com.example.myapplication

Save location

G:\Users\MMalhat\AndroidStudioProjects\MyApplication5

Language

Java

Minimum API level: API 14: Android 4.0 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

Help me choose

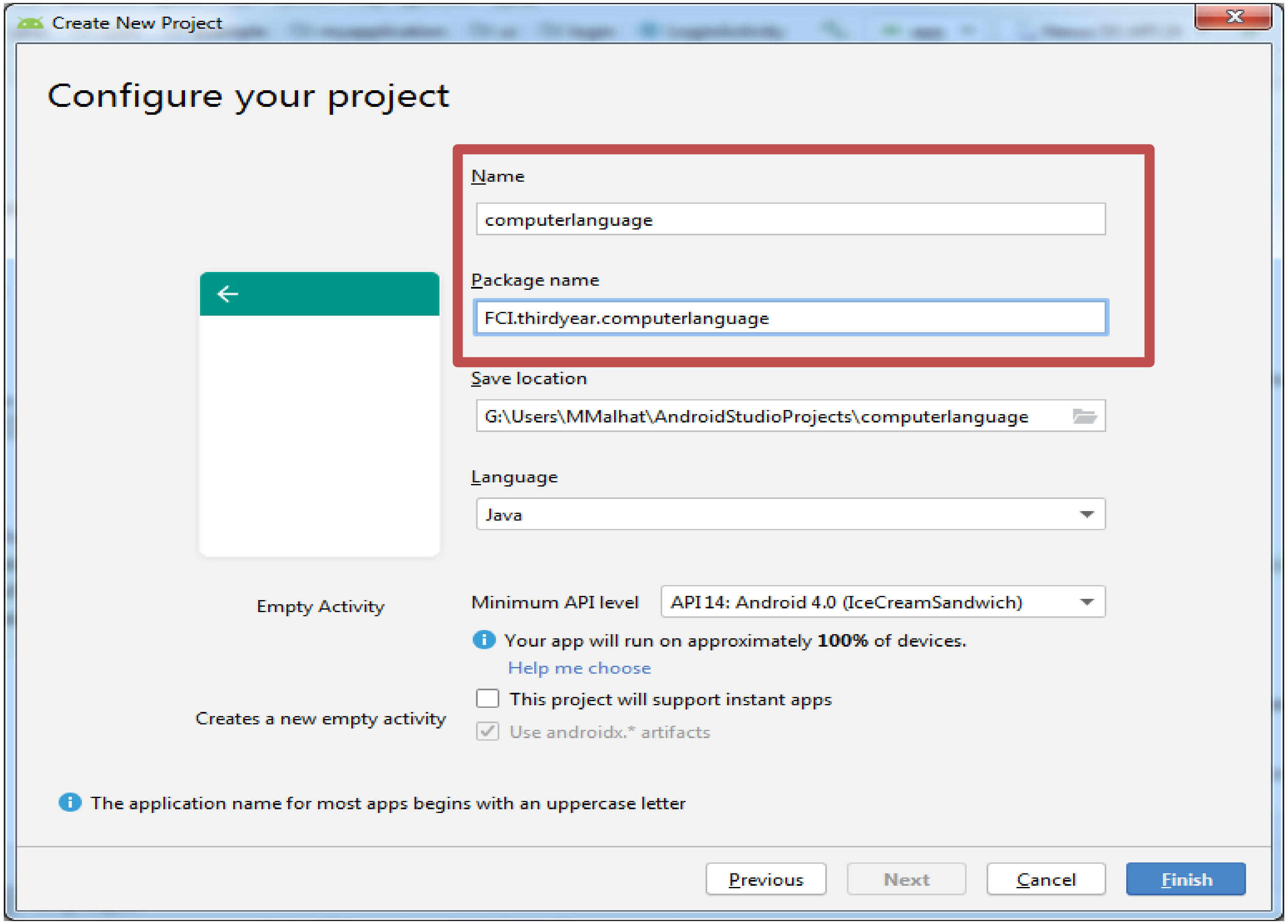
Empty Activity

Creates a new empty activity

- [ ] This project will support instant apps
- [x] Use androidx.* artifacts

Observe Name, Package name, and save location

Exploring the IDE



Create New Project

Configure your project

Name: computerlanguage

Package name: FCI.thirdyear.computerlanguage

Save location: G:\Users\MMalhat\AndroidStudioProjects\computerlanguage

Language: Java

Minimum API level: API 14: Android 4.0 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

Help me choose

[ ] This project will support instant apps

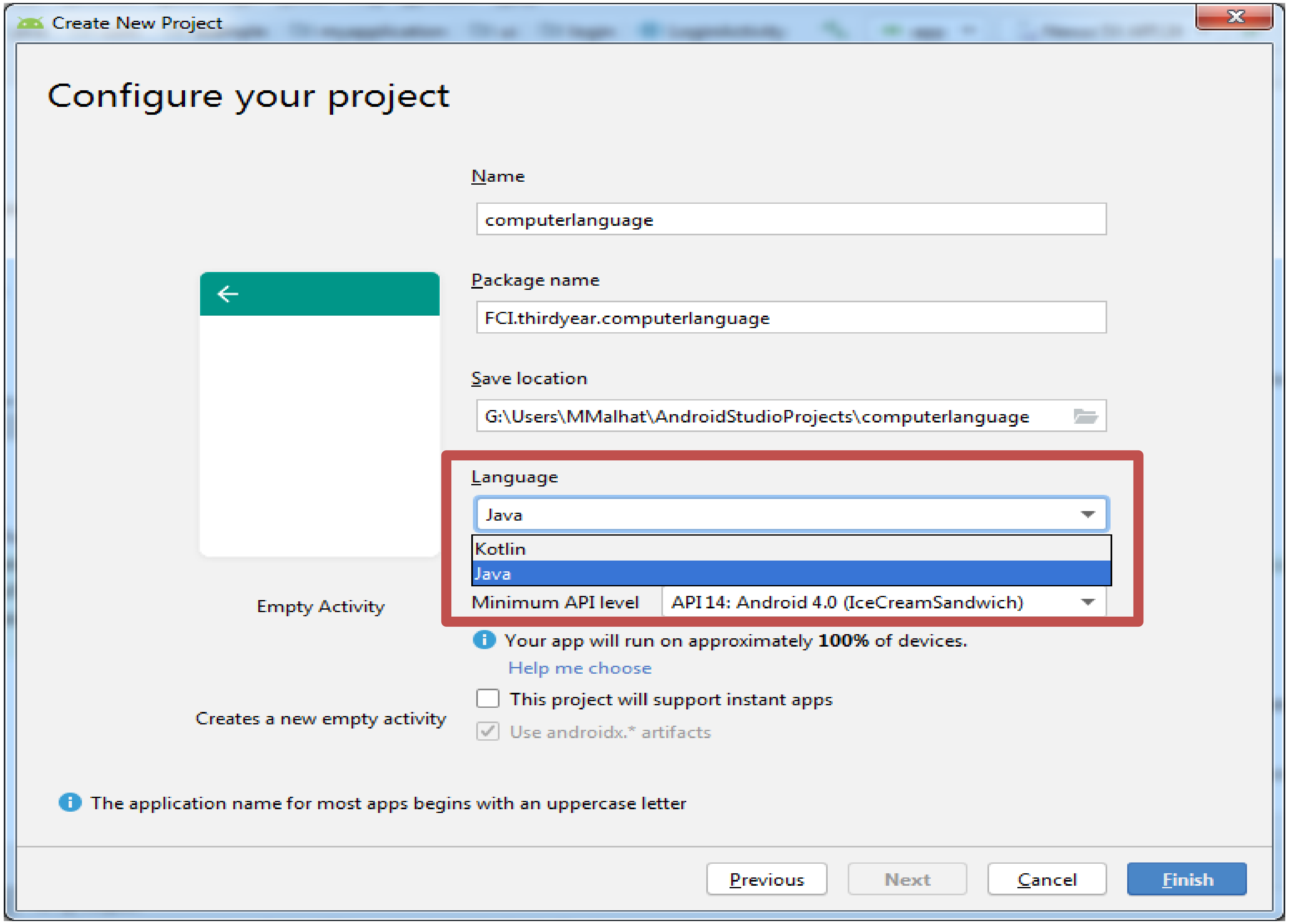
[✓] Use androidx.* artifacts

Empty Activity

Creates a new empty activity

The application name for most apps begins with an uppercase letter

Exploring the IDE



Create New Project

Configure your project

Name
computerlanguage

Package name
FCI.thirdyear.computerlanguage

Save location
G:\Users\MMalhat\AndroidStudioProjects\computerlanguage

Language
Java

Kotlin
Java

Minimum API level: API 14: Android 4.0 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.
Help me choose

- [ ] This project will support instant apps
- [x] Use androidx.* artifacts

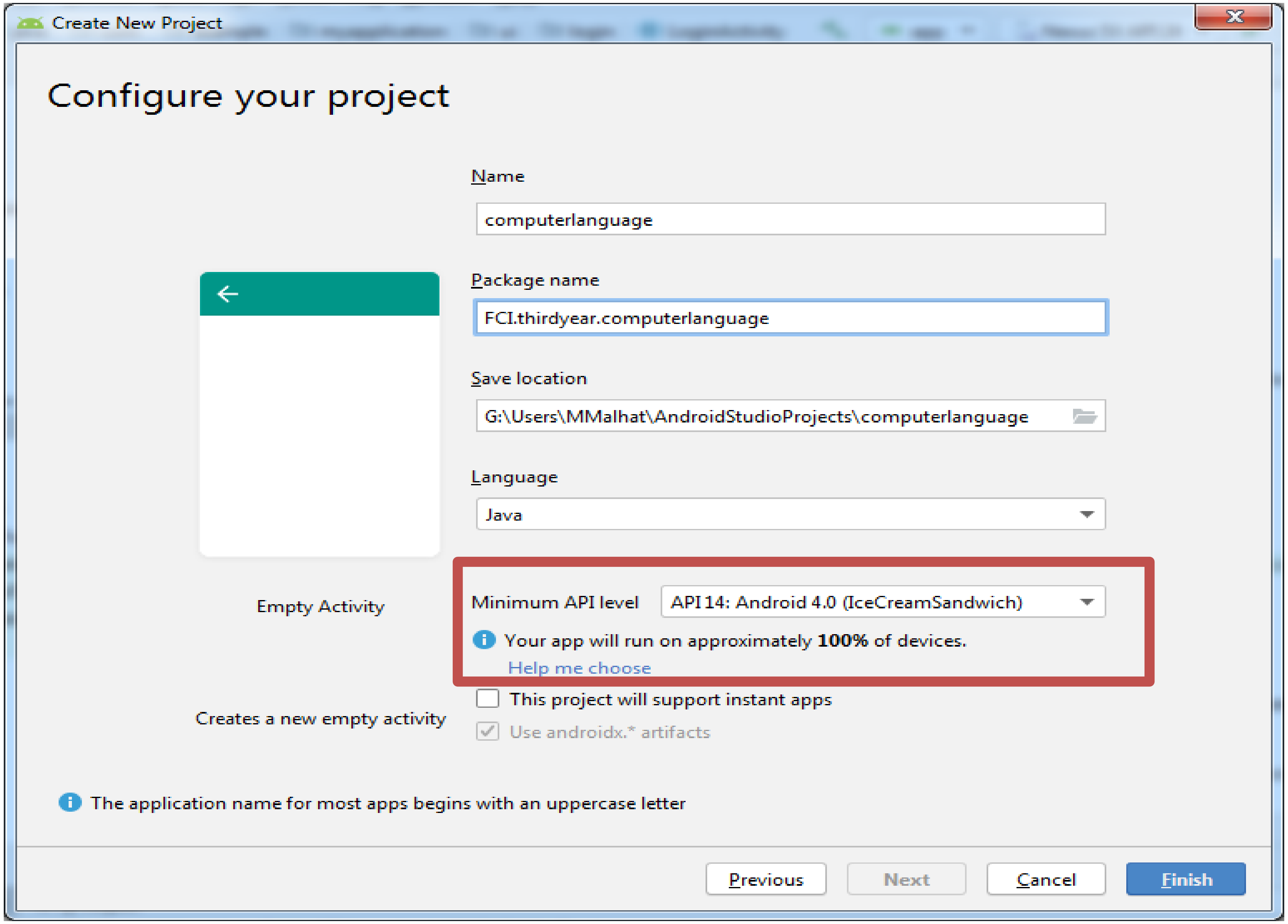
Empty Activity

Creates a new empty activity

The application name for most apps begins with an uppercase letter

Previous | Next | Cancel | Finish

Exploring the IDE



Create New Project

Configure your project

Name: computerlanguage

Package name: FCI.thirdyear.computerlanguage

Save location: G:\Users\MMalhat\AndroidStudioProjects\computerlanguage

Language: Java

Minimum API level: API 14: Android 4.0 (IceCreamSandwich)

Your app will run on approximately **100%** of devices.

Help me choose

☐ This project will support instant apps

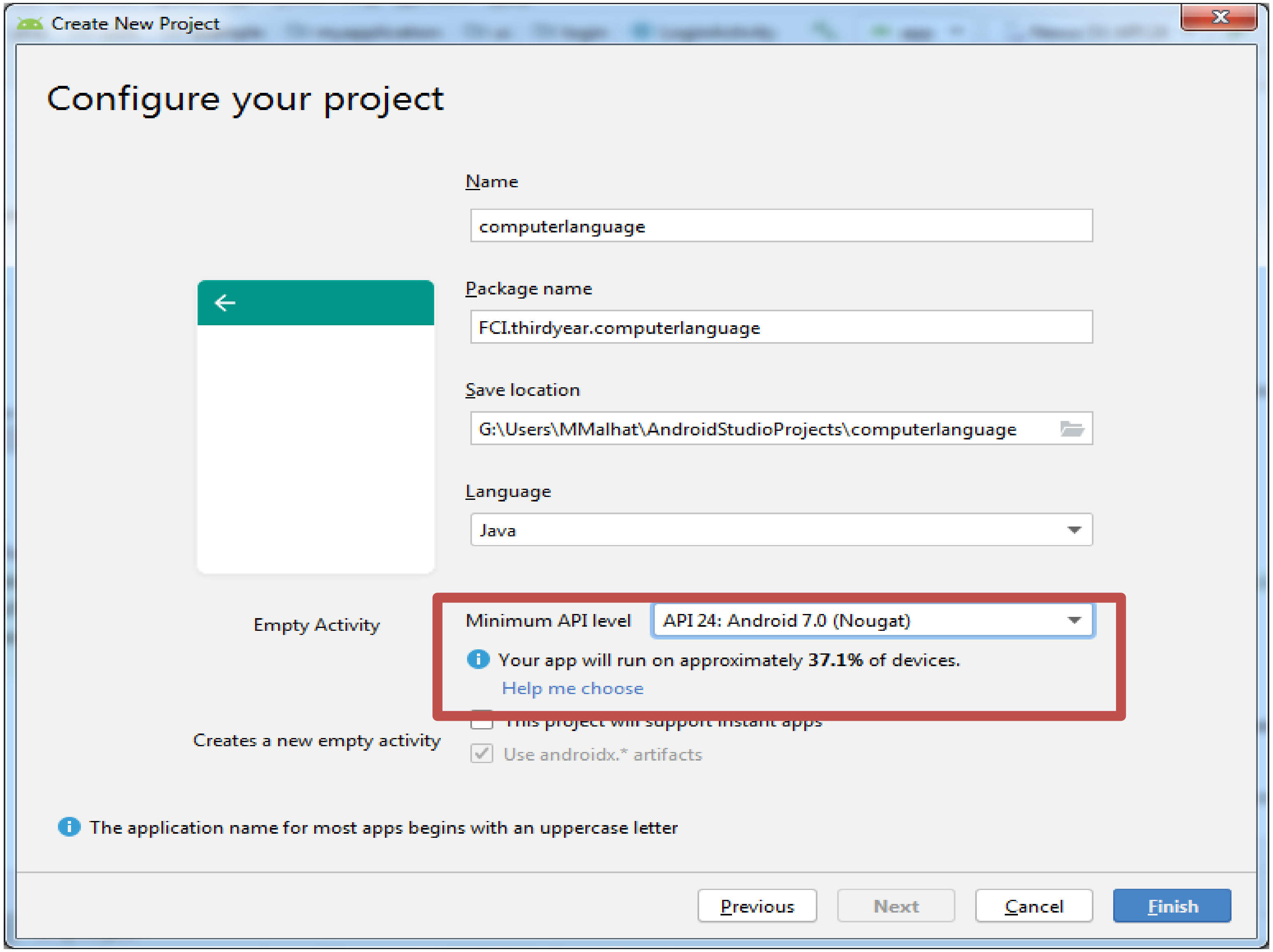
☑ Use androidx.* artifacts

Empty Activity

Creates a new empty activity

The application name for most apps begins with an uppercase letter

Exploring the IDE



Configure your project

Name: computerlanguage

Package name: FCI.thirdyear.computerlanguage

Save location: G:\Users\MMalhat\AndroidStudioProjects\computerlanguage

Language: Java

Minimum API level: API 24: Android 7.0 (Nougat)

Your app will run on approximately **37.1%** of devices.

Help me choose

This project will support instant apps

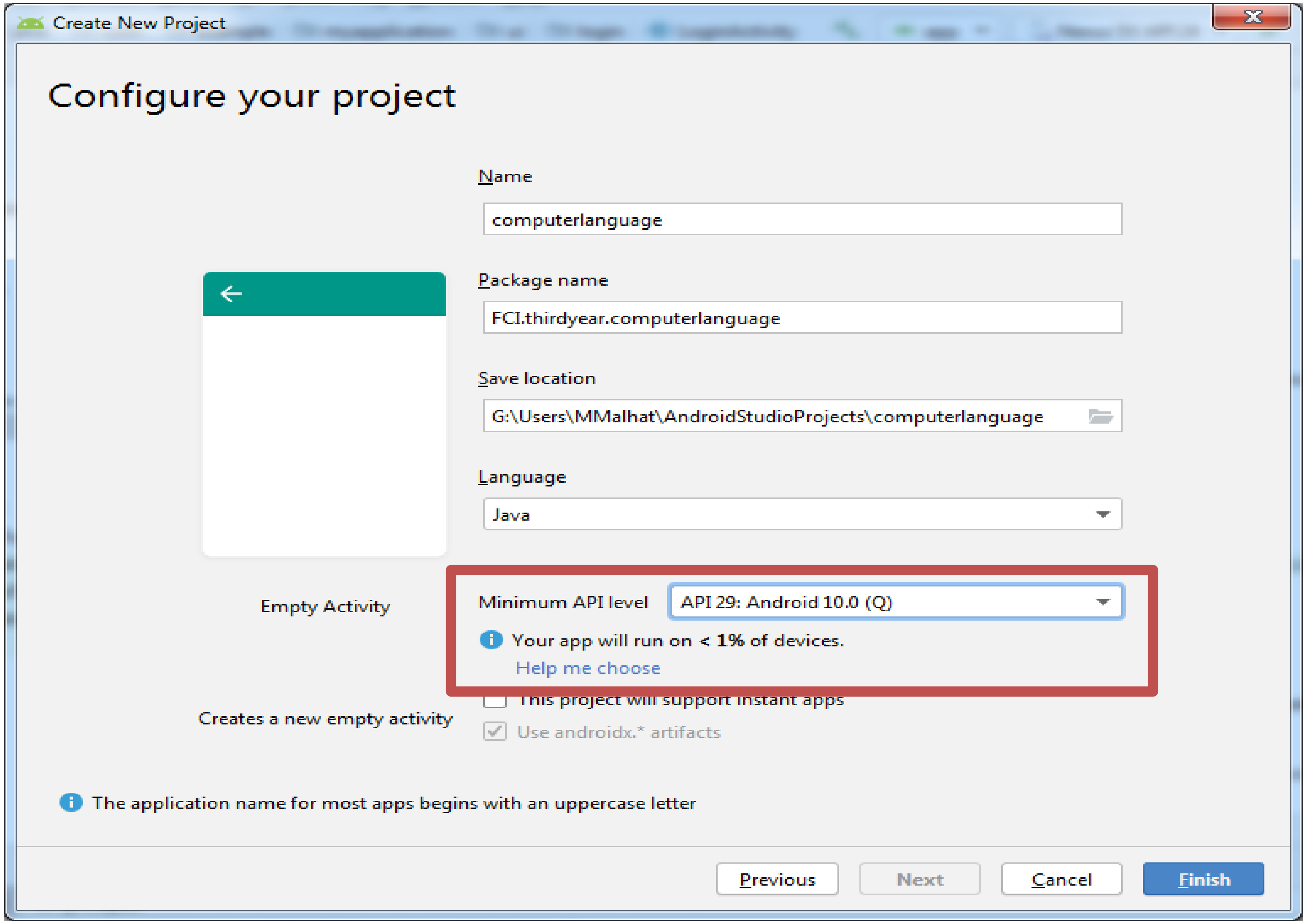
[✓] Use androidx.* artifacts

Empty Activity

Creates a new empty activity

The application name for most apps begins with an uppercase letter

Exploring the IDE



Configure your project

Name: computerlanguage

Package name: FCI.thirdyear.computerlanguage

Save location: G:\Users\MMalhat\AndroidStudioProjects\computerlanguage

Language: Java

Empty Activity

Creates a new empty activity

Minimum API level: API 29: Android 10.0 (Q)

Your app will run on < **1%** of devices.

Help me choose

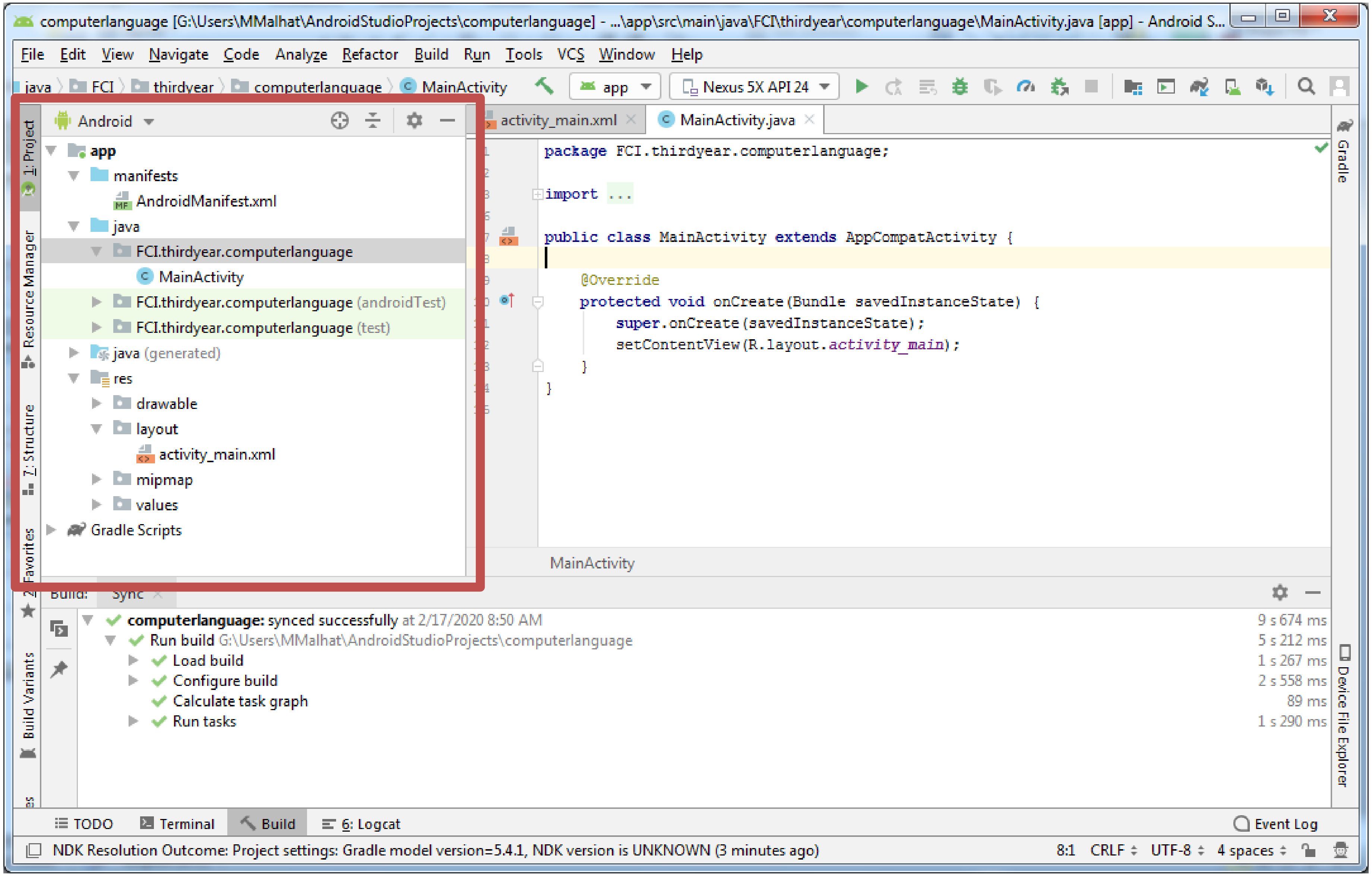
☐ This project will support instant apps

☑ Use androidx.* artifacts

The application name for most apps begins with an uppercase letter

You should select the minimum version of android that support the required features in the developed application

Exploring the IDE



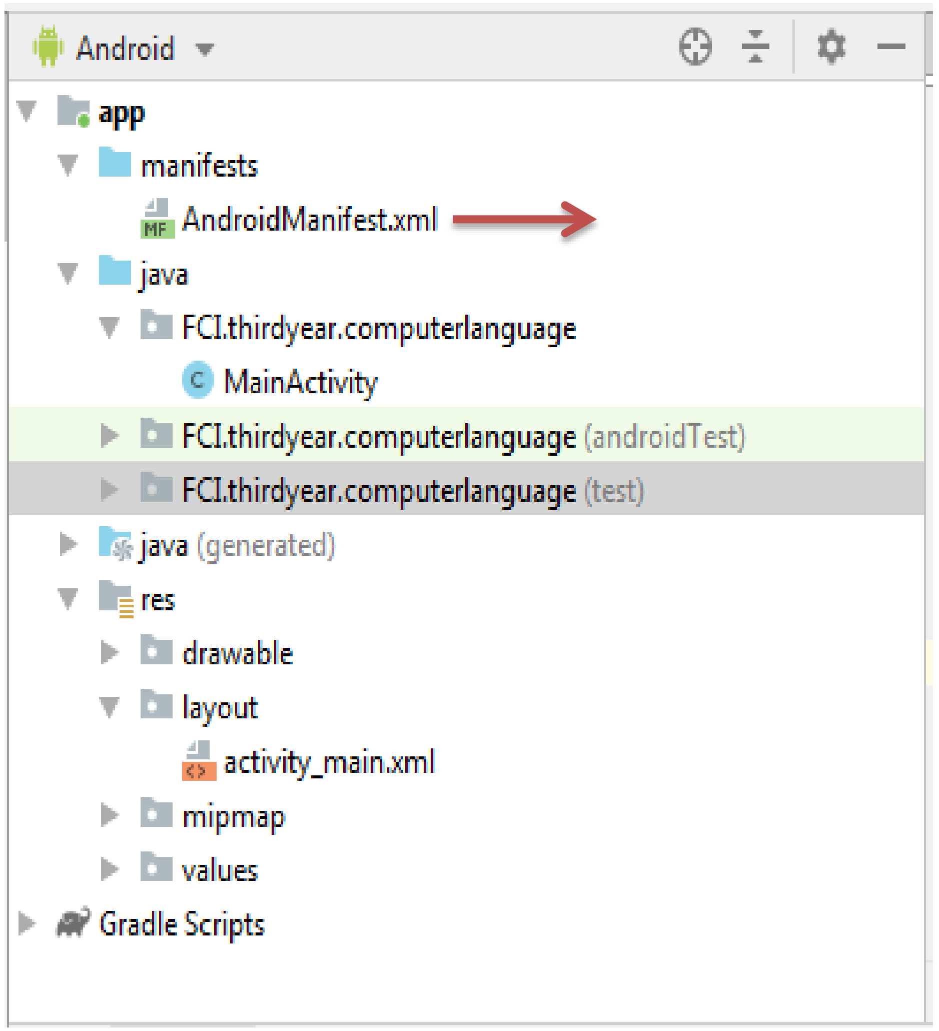
computerlanguage [G:\Users\MMalhat\AndroidStudioProjects\computerlanguage] - ...\app\src\main\java\FCI\thirdyear\computerlanguage\MainActivity.java [app] - Android S...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
java FCI thirdyear computerlanguage MainActivity
app
Nexus 5X API 24
Android
app
manifests
AndroidManifest.xml
java
FCI.thirdyear.computerlanguage
MainActivity
FCI.thirdyear.computerlanguage (androidTest)
FCI.thirdyear.computerlanguage (test)
java (generated)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
1: Project
Resource Manager
7: Structure
2: Favorites
Build Variants
Gradle
Device File Explorer
activity_main.xml
MainActivity.java
MainActivity
Build: Sync
TODO
Terminal
Build
6: Logcat
Event Log

```java
package FCI.thirdyear.computerlanguage;

import ...

public class MainActivity extends AppCompatActivity {

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

```
computerlanguage: synced successfully at 2/17/2020 8:50 AM                    9 s 674 ms
  Run build G:\Users\MMalhat\AndroidStudioProjects\computerlanguage            5 s 212 ms
    Load build                                                                 1 s 267 ms
    Configure build                                                            2 s 558 ms
    Calculate task graph                                                           89 ms
    Run tasks                                                                  1 s 290 ms
```

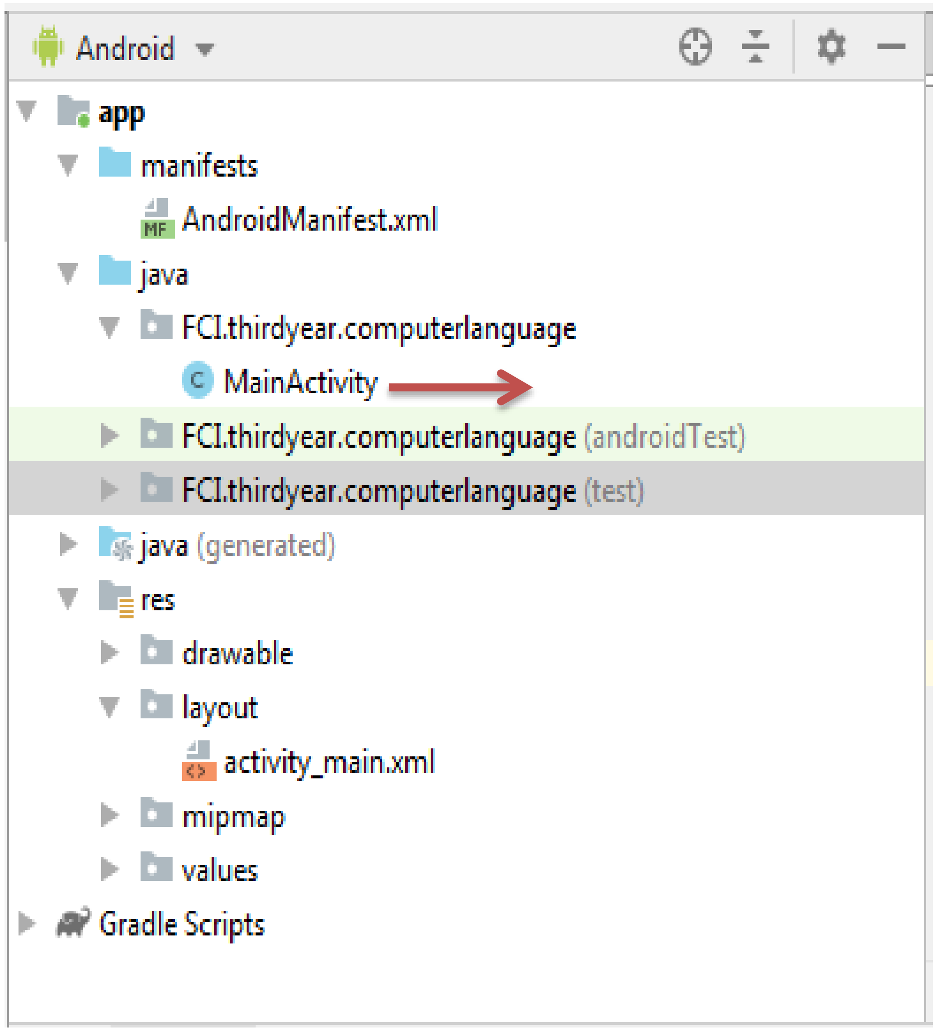
```
NDK Resolution Outcome: Project settings: Gradle model version=5.4.1, NDK version is UNKNOWN (3 minutes ago)    8:1  CRLF  UTF-8  4 spaces
```

Exploring the IDE



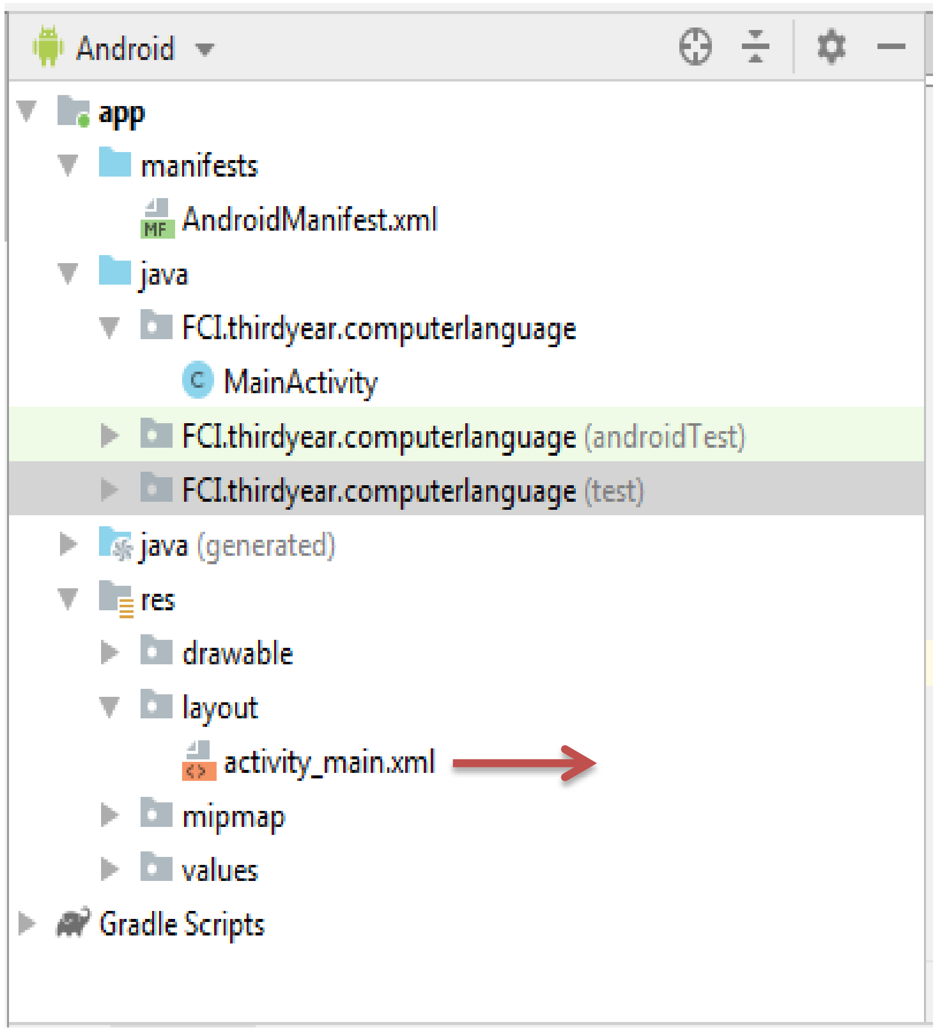
The manifest file describes the fundamental characteristics of the application (e.g., application name and Android version) and defines each of its components.

Exploring the IDE



This is the code for the first activity: the first screen of your app. It handles the logic of button presses and is where you'll write code to handle specific functions, like if you want to open camera or location service or validate data.

Exploring the IDE



This is the layout XML file, meaning it will handle the design and the appearance of your application. It's where we'll add buttons for instance

AndroidManifest.xml

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

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    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" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```

MainActivity.java

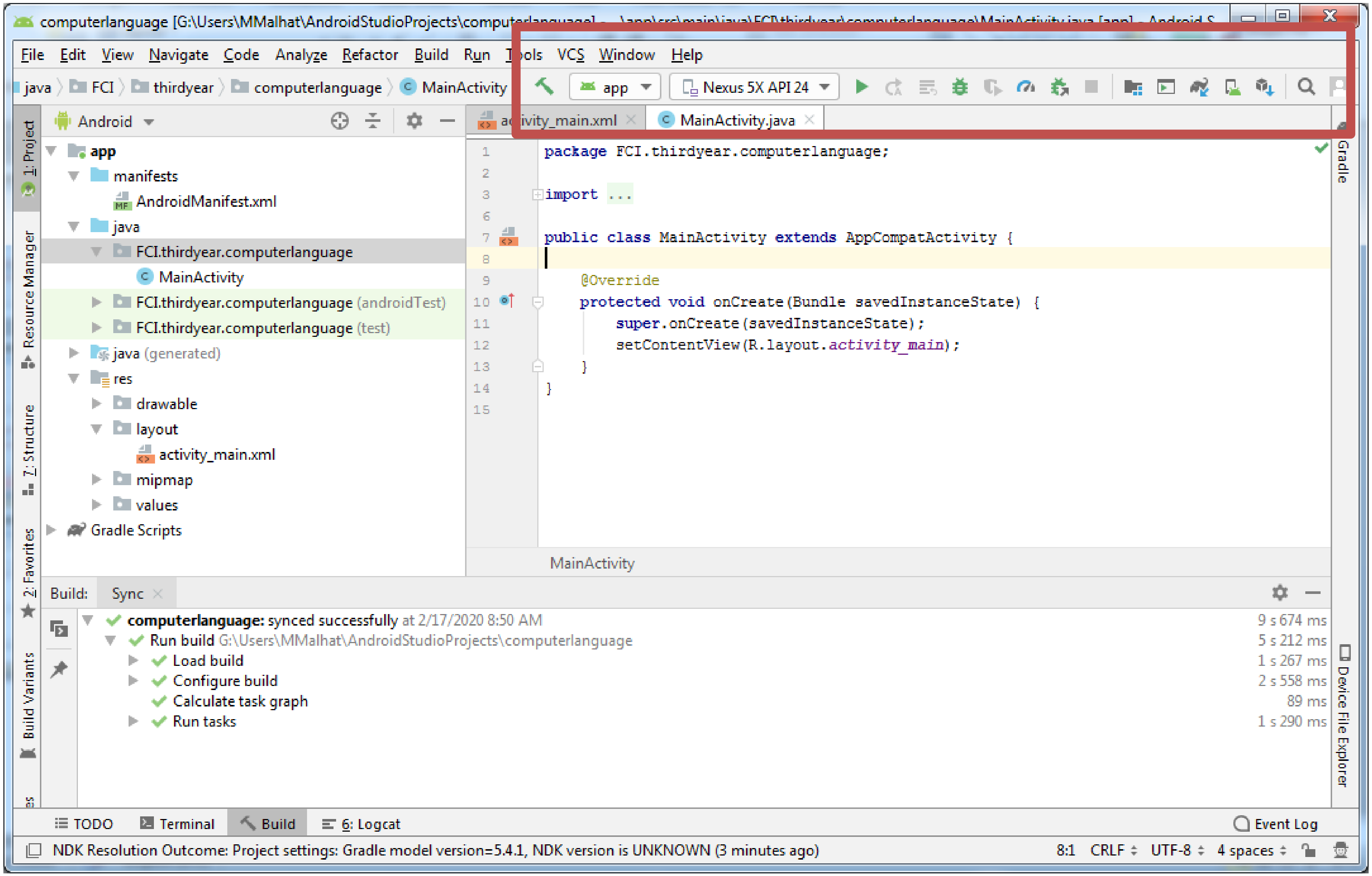
```
package FCI.thirdyear.computerlanguage;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">
  <TextView
    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"
    app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

an attribute that is defined in any root view and declares which activity or fragment the layout is associated with

Exploring the IDE

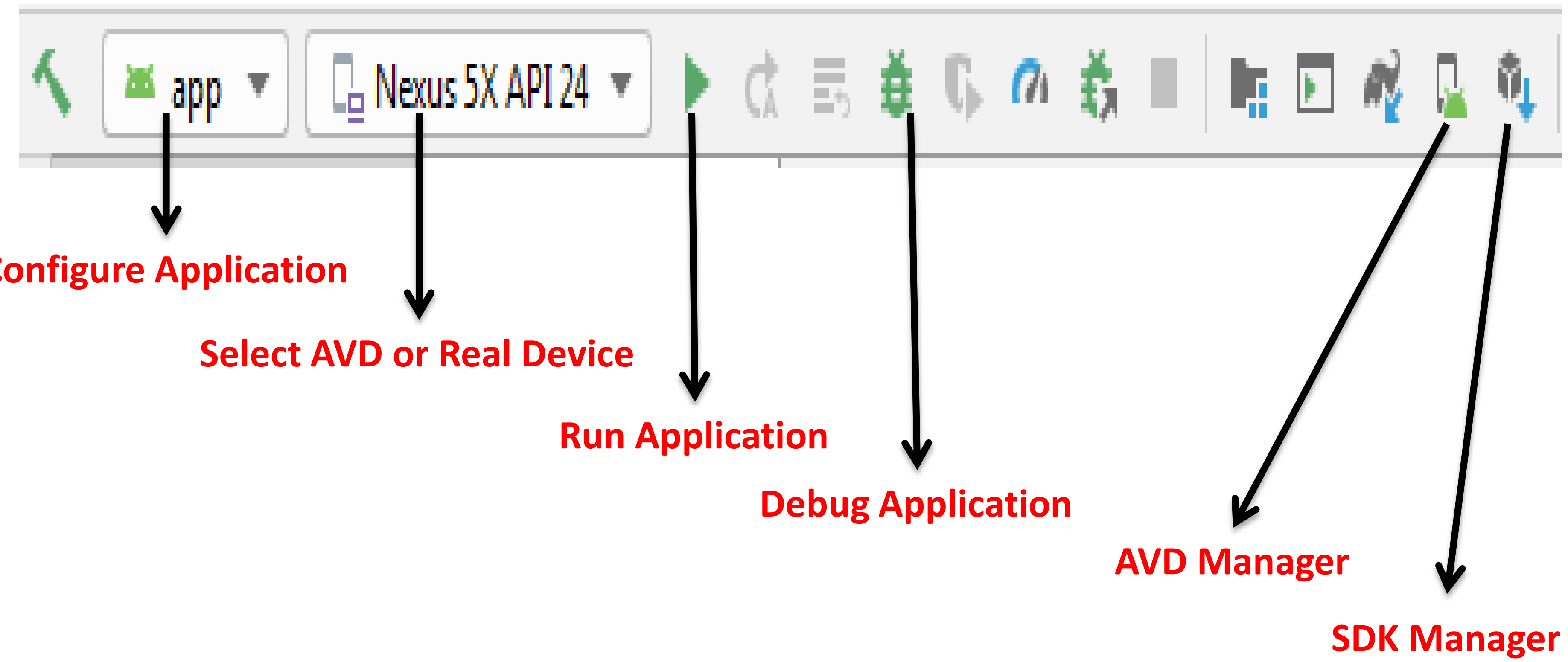


```
package FCI.thirdyear.computerlanguage;

import ...

public class MainActivity extends AppCompatActivity {

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

```

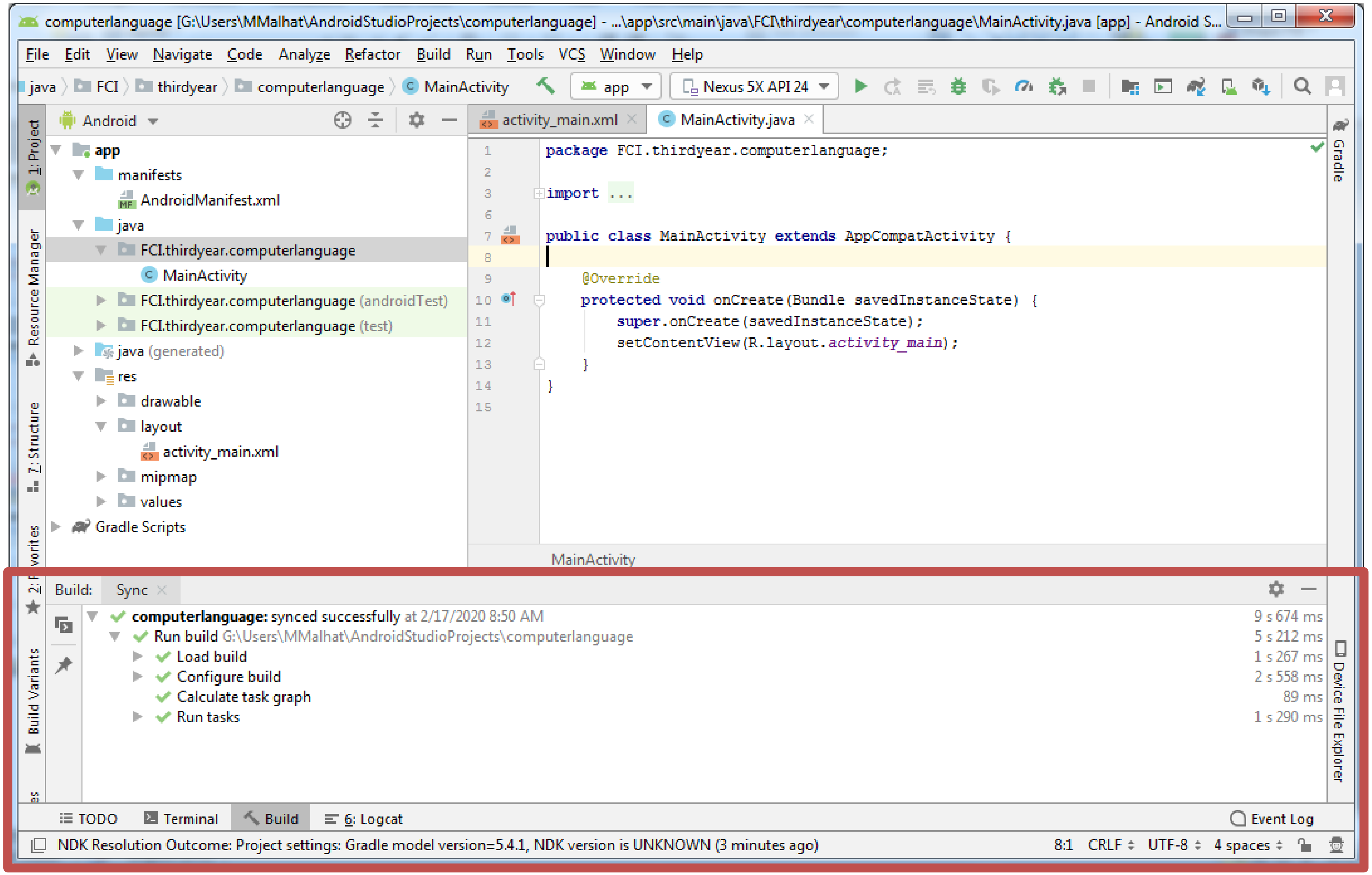
```
computerlanguage: synced successfully at 2/17/2020 8:50 AM          9 s 674 ms
  Run build G:\Users\MMalhat\AndroidStudioProjects\computerlanguage 5 s 212 ms
    Load build                                                       1 s 267 ms
    Configure build                                                  2 s 558 ms
    Calculate task graph                                                 89 ms
    Run tasks                                                        1 s 290 ms
```

```
NDK Resolution Outcome: Project settings: Gradle model version=5.4.1, NDK version is UNKNOWN (3 minutes ago)    8:1  CRLF  UTF-8  4 spaces
```

Exploring the IDE



Exploring the IDE


computerlanguage [G:\Users\MMalhat\AndroidStudioProjects\computerlanguage] - ...\app\src\main\java\FCI\thirdyear\computerlanguage\MainActivity.java [app] - Android S...

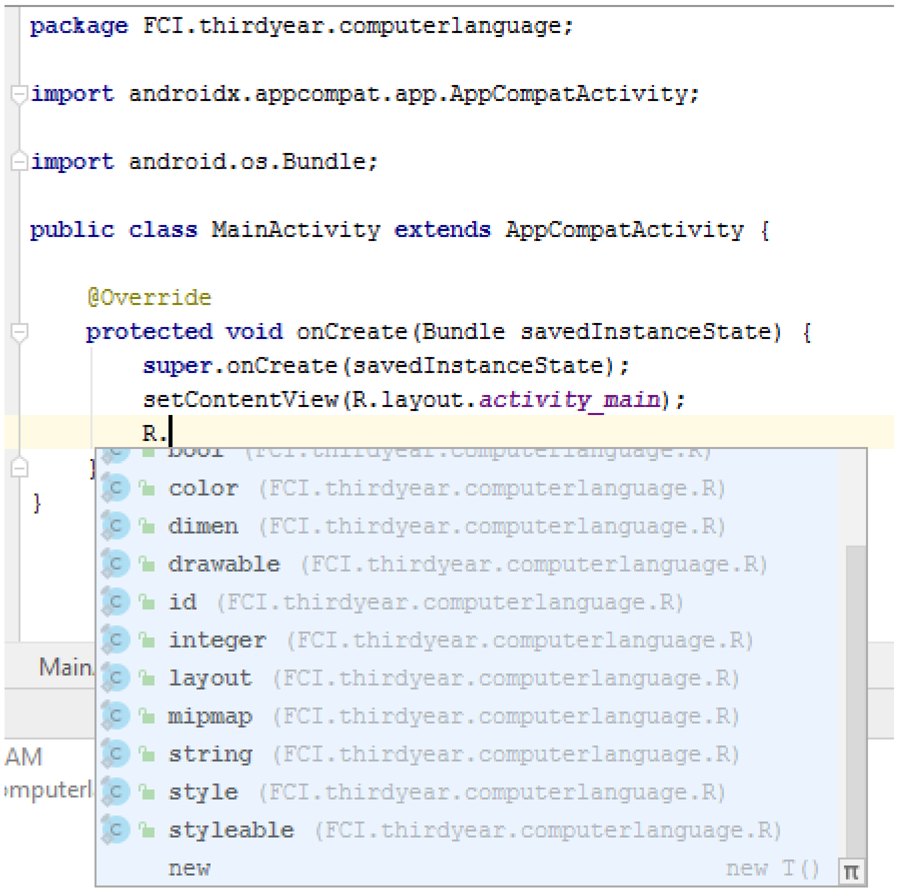
```
package FCI.thirdyear.computerlanguage;

import ...

public class MainActivity extends AppCompatActivity {

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

```
computerlanguage: synced successfully at 2/17/2020 8:50 AM          9 s 674 ms
  Run build G:\Users\MMalhat\AndroidStudioProjects\computerlanguage  5 s 212 ms
    Load build                                                      1 s 267 ms
    Configure build                                                 2 s 558 ms
    Calculate task graph                                                 89 ms
    Run tasks                                                       1 s 290 ms
```

NDK Resolution Outcome: Project settings: Gradle model version=5.4.1, NDK version is UNKNOWN (3 minutes ago)   8:1   CRLF   UTF-8   4 spaces

Using Code Completion

- Code completion is an invaluable tool that shows you contextual options for completing the piece of code that you are trying to write.
- **For example**, in the editor tab for the MainActivity.java file, type the letter R, and then type a period.

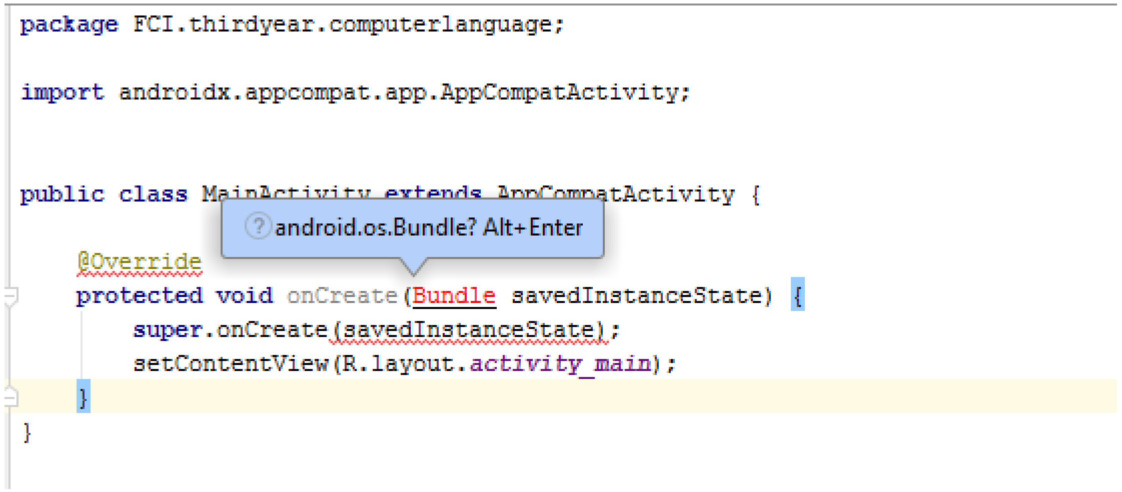


```
package FCI.thirdyear.computerlanguage;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

```
bool  (FCI.thirdyear.computerlanguage.R)
color  (FCI.thirdyear.computerlanguage.R)
dimen  (FCI.thirdyear.computerlanguage.R)
drawable  (FCI.thirdyear.computerlanguage.R)
id  (FCI.thirdyear.computerlanguage.R)
integer  (FCI.thirdyear.computerlanguage.R)
layout  (FCI.thirdyear.computerlanguage.R)
mipmap  (FCI.thirdyear.computerlanguage.R)
string  (FCI.thirdyear.computerlanguage.R)
style  (FCI.thirdyear.computerlanguage.R)
styleable  (FCI.thirdyear.computerlanguage.R)
new                                    new T()
```

Using Code Completion

- You can also use code completion to import packages in the Android Studio.
  - For example, if you were to attempt to create a variable of a type that belongs to a package that you have not imported, Android Studio recognizes this and underlines the type with a red squiggle. Set the cursor to that line and press Alt+Enter to automatically import the package into a using statement at the top of your code file.



```
package FCI.thirdyear.computerlanguage;

import androidx.appcompat.app.AppCompatActivity;


public class MainActivity extends AppCompatActivity {

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

Debugging your Application

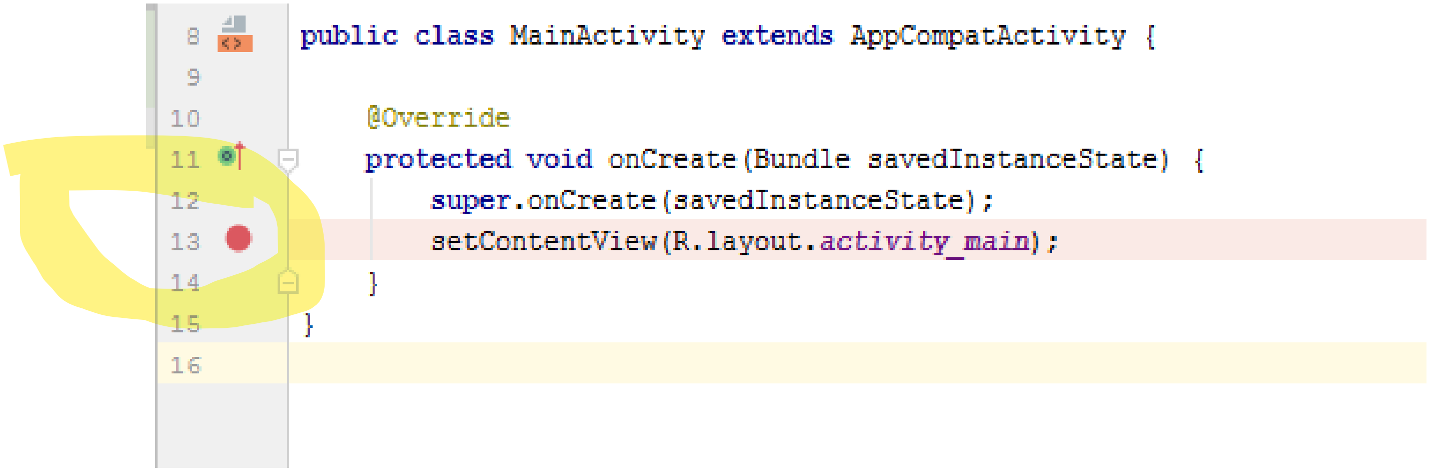
- After you have built an application, you need to be able to debug it and see what is going on inside your code.
- One of the handiest ways to be able to see inside your code it through the use of breakpoints.
- Breakpoints allow you to pause the execution of your code at specific locations and see what is going on (or what is going wrong).

Setting Breakpoints

- 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.
  - This means that you can check on the values of variables in your application while you are debugging it.
  - Also, you can check whether certain lines of code are being executed as expected—or at all.
- To tell Android Studio that you want to examine a specific line of code during debugging, you must set a breakpoint at that line.

Setting Breakpoints

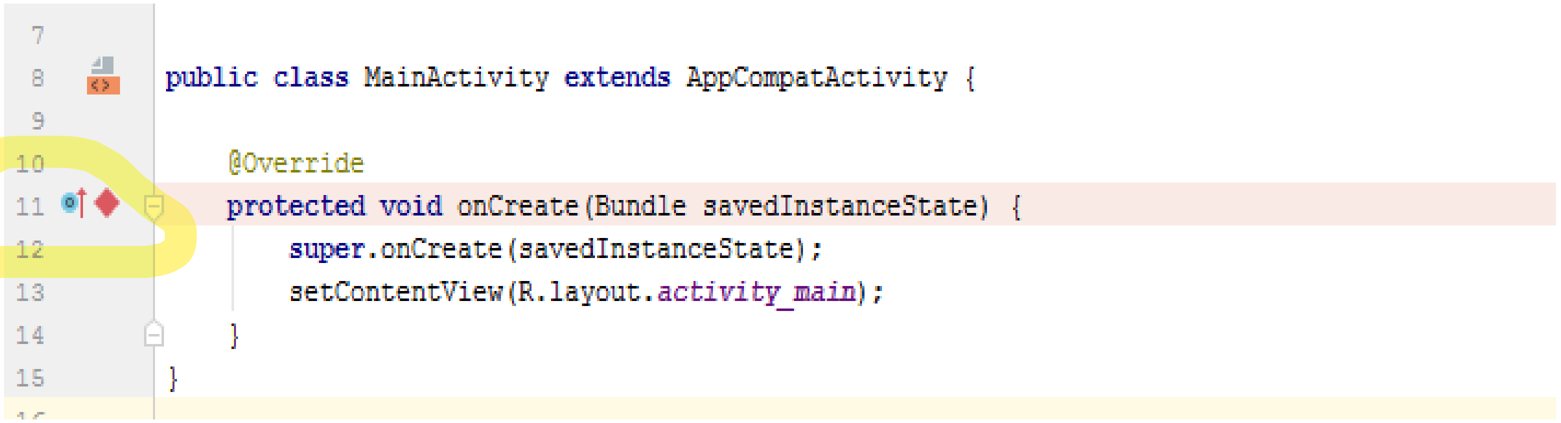
- You can set a breakpoint at specific line by two ways:
  - Click the margin of the editor tab next to line of code you want to break at, to set a breakpoint.
  - Placing your cursor in the line of code where you want it to break and clicking Run➤➪➤Toggle Line Breakpoint.
- A red circle is placed in the margin, and the corresponding line is highlighted in red



```
public class MainActivity extends AppCompatActivity {

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

Setting Breakpoints

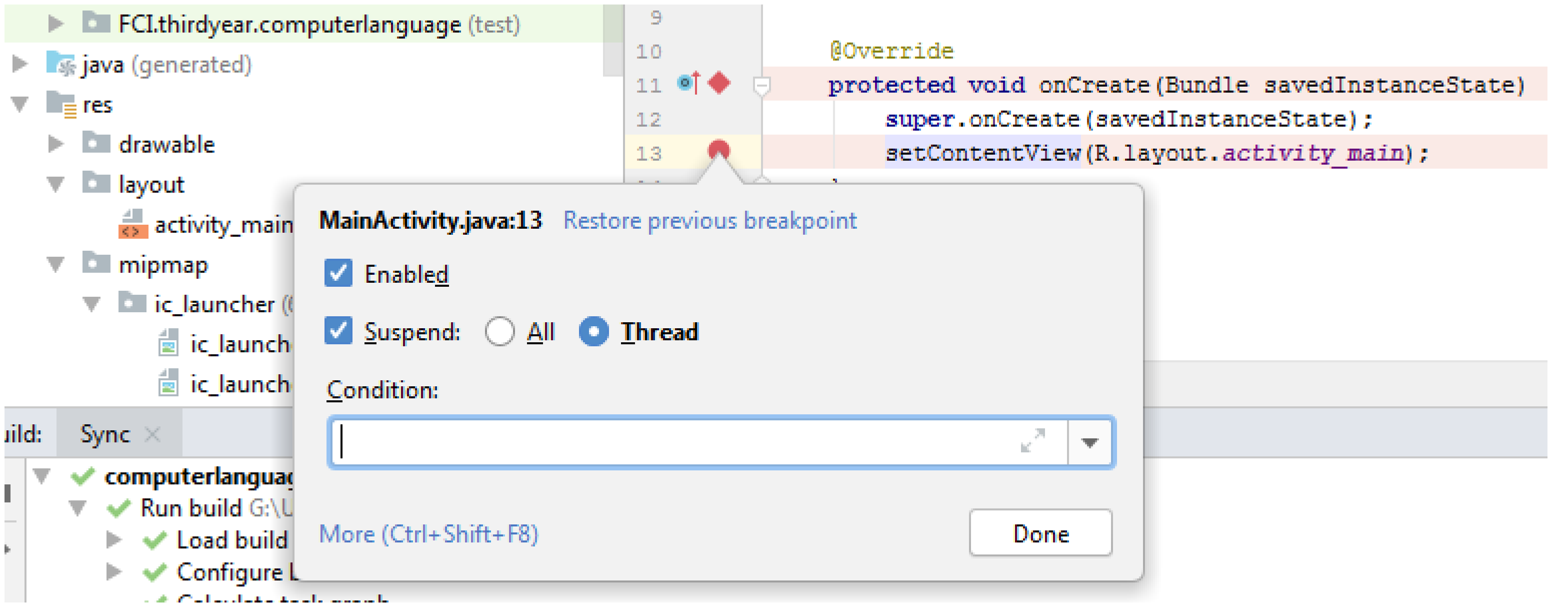
- Let's say that you do not know the exact line of code where you want the break to be. You might want to check on the condition of your code when a specific method is called.
- You can set a method breakpoint by selecting Run➤⇨➤Toggle Method Breakpoint. A method breakpoint is represented by a red diamond.



```
public class MainActivity extends AppCompatActivity {

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

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.



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

MainActivity.java:13 Restore previous breakpoint

[✓] Enabled

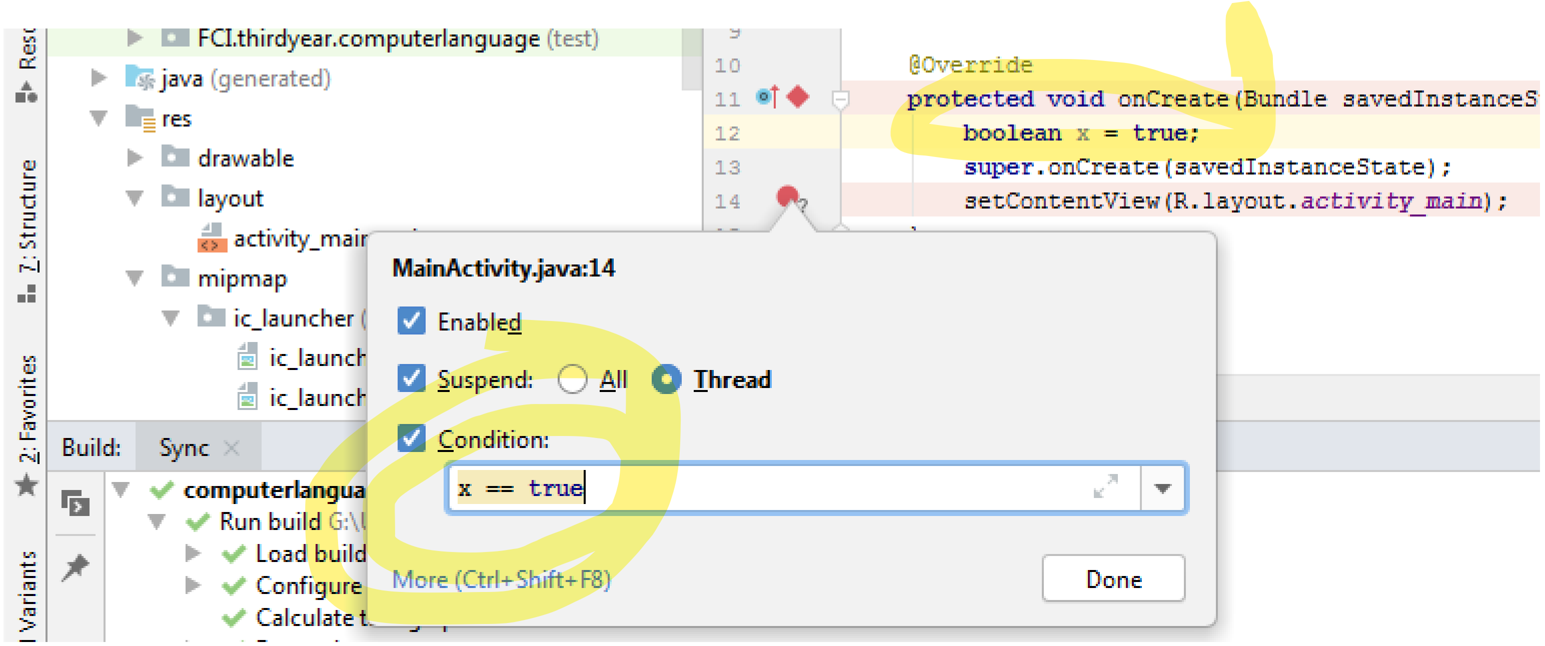
[✓] Suspend: ○ All [● Thread]

Condition:

More (Ctrl+Shift+F8) Done

Conditional Breakpoints

- From here you can set conditions that tell Android Studio when to pause at a breakpoint.
- For example, you can tell Android Studio to only pause at a line of code when your variable named x equals true.



```
@Override
protected void onCreate(Bundle savedInstanceS
    boolean x = true;
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

MainActivity.java:14

- [✓] Enabled
- [✓] Suspend: All [✓ Thread]
- [✓] Condition: `x == true`

Navigating Paused Code

- While in debug mode, Android Studio pauses at any breakpoint that you have set. That is, as long as a breakpoint has been set on a reachable line of code, Android Studio halts execution at that line until you tell it to continue.
- Once a breakpoint has been hit, the debug window opens at the bottom of Android Studio. The debug window contains many of the tools you use to navigate around your code.
- Notice the navigation buttons located in the menu bar of the debug window. The most commonly used are Step Over and Step Into.

In summary, Step Over allows you to skip the details of a method call and continue to the next line, while Step Into takes you inside the method call to analyze its behavior line-by-line. These features are valuable for debugging and understanding the flow of your code in Android Studio.

Navigating Paused Code

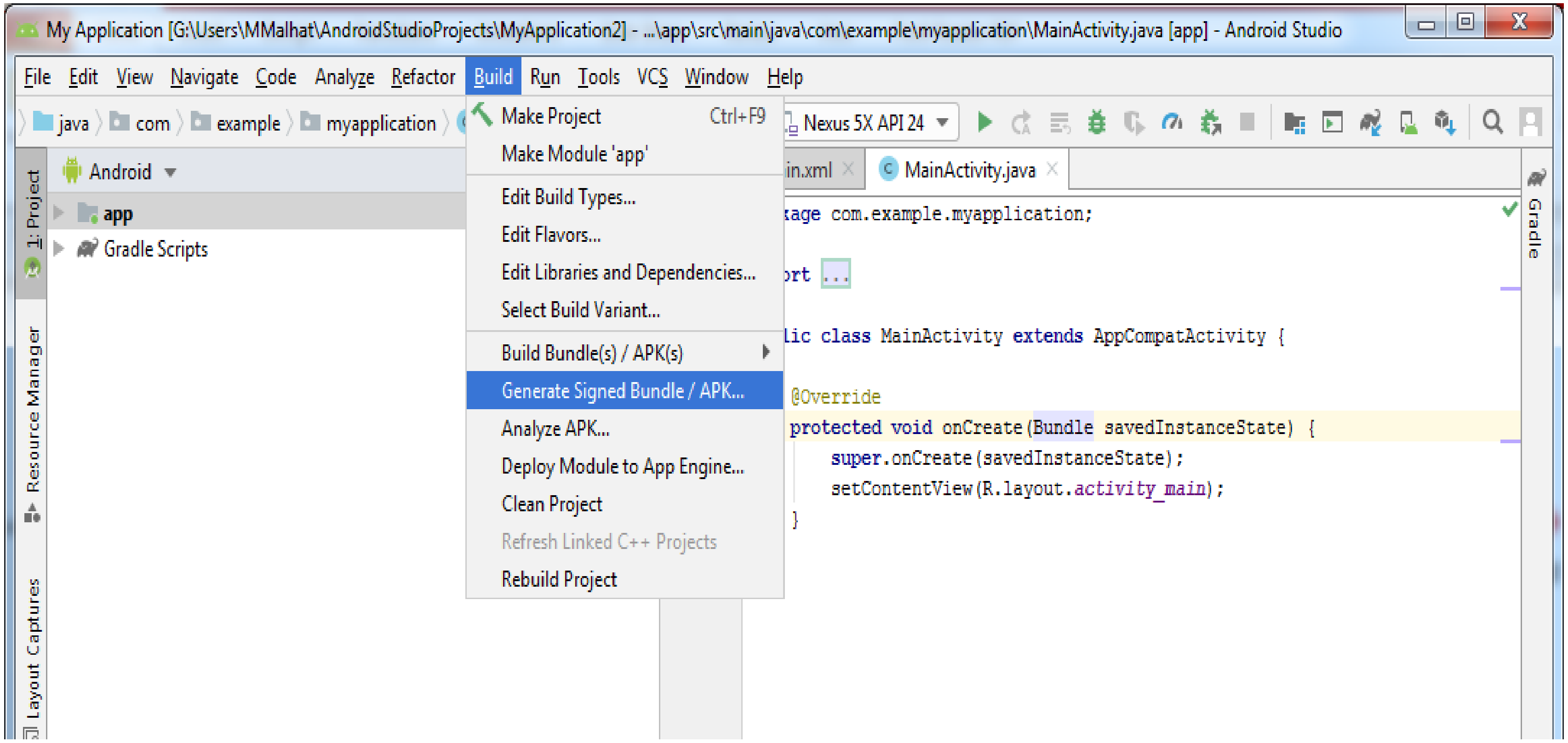
- Step Over advances you to the line of code that immediately follows the one at which you are currently paused. This means that if you are paused at a method call, and you press Step Over, Android Studio executes the method call without pausing and then pauses again when execution reached the next line. However, what if an exception happens in that method call and execution never reaches the next line of code? For these situations use Step Into.
- Step Into follows execution wherever it leads in the code. Therefore, if you are paused at a method call and click Step Into, Android Studio will shift the view to the method call and pause execution at the first line of code within that method. This allows you to then follow the execution of that method line-by-line before it returns to the calling block.

Publishing Your Application

- After you have created, and fully debugged, your application, you might want to deploy it to the Google Store for others to enjoy.
- To **publish** your finished application on the **Google Play** Store, you must generate a **signed APK** (i.e., Android Application Package).
- The APK is the compiled, executable version of your application.
- Signing it is much like signing your name to a document.
- The signature identifies the app's developer to Google and the users who install your application.

Generating a Signed APK

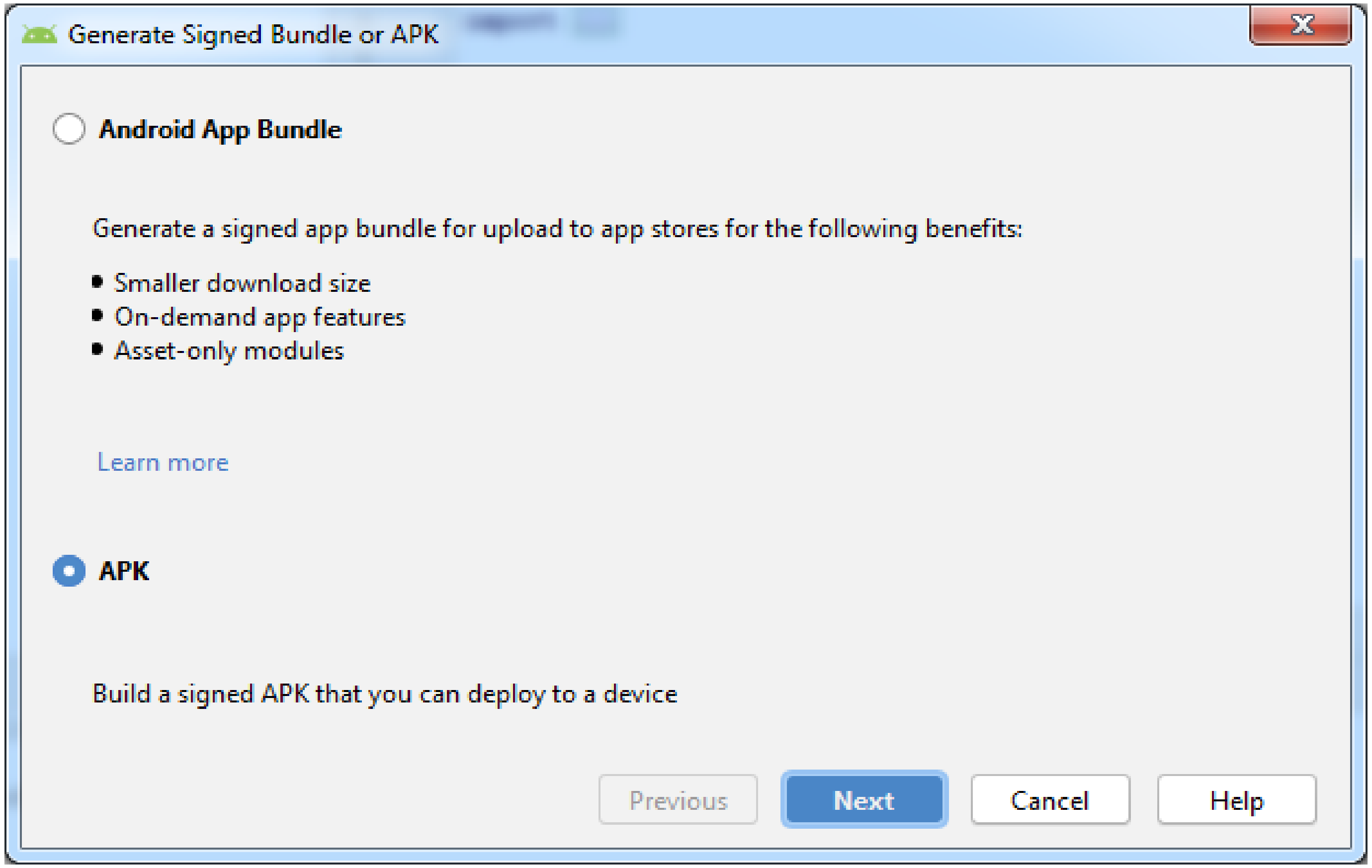
- Use the following steps to generate a signed APK:
  1. Select **Build → Generate Signed APK** from the Menu bar to bring up the Generate Signed APK window.


My Application [G:\Users\MMalhat\AndroidStudioProjects\MyApplication2] - ...\app\src\main\java\com\example\myapplication\MainActivity.java [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
java com example myapplication
Nexus 5X API 24
Android
app
Gradle Scripts
Make Project Ctrl+F9
Make Module 'app'
Edit Build Types...
Edit Flavors...
Edit Libraries and Dependencies...
Select Build Variant...
Build Bundle(s) / APK(s)
Generate Signed Bundle / APK...
Analyze APK...
Deploy Module to App Engine...
Clean Project
Refresh Linked C++ Projects
Rebuild Project
in.xml
MainActivity.java
1: Project
Resource Manager
Layout Captures
Gradle

```
kage com.example.myapplication;

ort ...

lic class MainActivity extends AppCompatActivity {

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

Generating a Signed APK

- Use the following steps to generate a signed APK:
  2. Choose **APK** and click next.



Generate Signed Bundle or APK

Android App Bundle

Generate a signed app bundle for upload to app stores for the following benefits:

- Smaller download size
- On-demand app features
- Asset-only modules

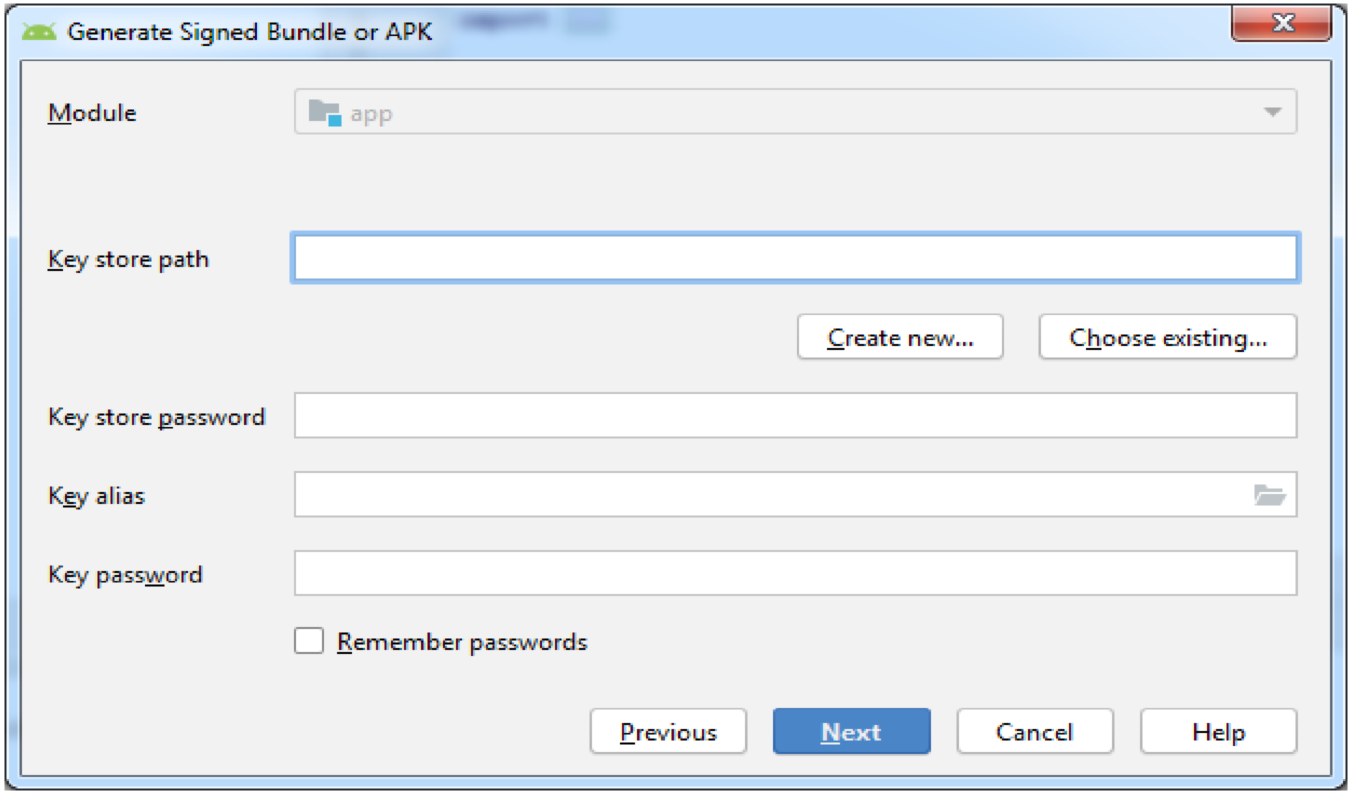
Learn more

[✓ APK]

Build a signed APK that you can deploy to a device

Generating a Signed APK

- Use the following steps to generate a signed APK:
   3. 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



Generate Signed Bundle or APK

Module: app

Key store path:

Key store password:

Key alias:

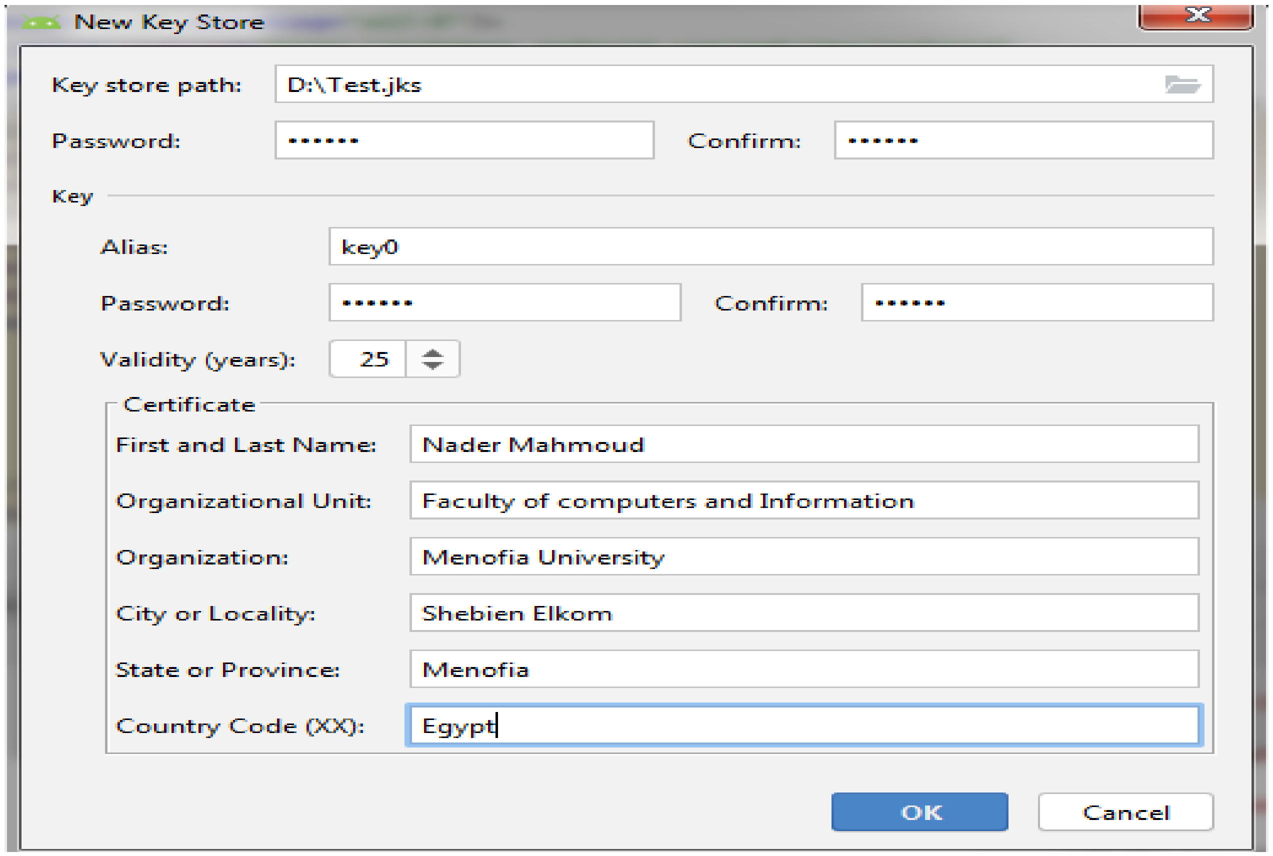
Key password:

Generating a Signed APK

- Signing an app first requires creating keystores.
- A keystore is a storage mechanism for security certificates
- A public key certificate is used to sign an APK before deployment to services like the Google Play Store
- Assigning the APK in this fashion allows Google to provide a high level of certainty that future updates to your APK of the same app come from you and not some malicious third party

Generating a Signed APK

- Use the following steps to generate a signed APK:
  4. Fill out all of the information on this form because it pertains to your entity and application.


New Key Store

Key store path:	D:\Test.jks		
Password:	••••••	Confirm:	••••••

Key

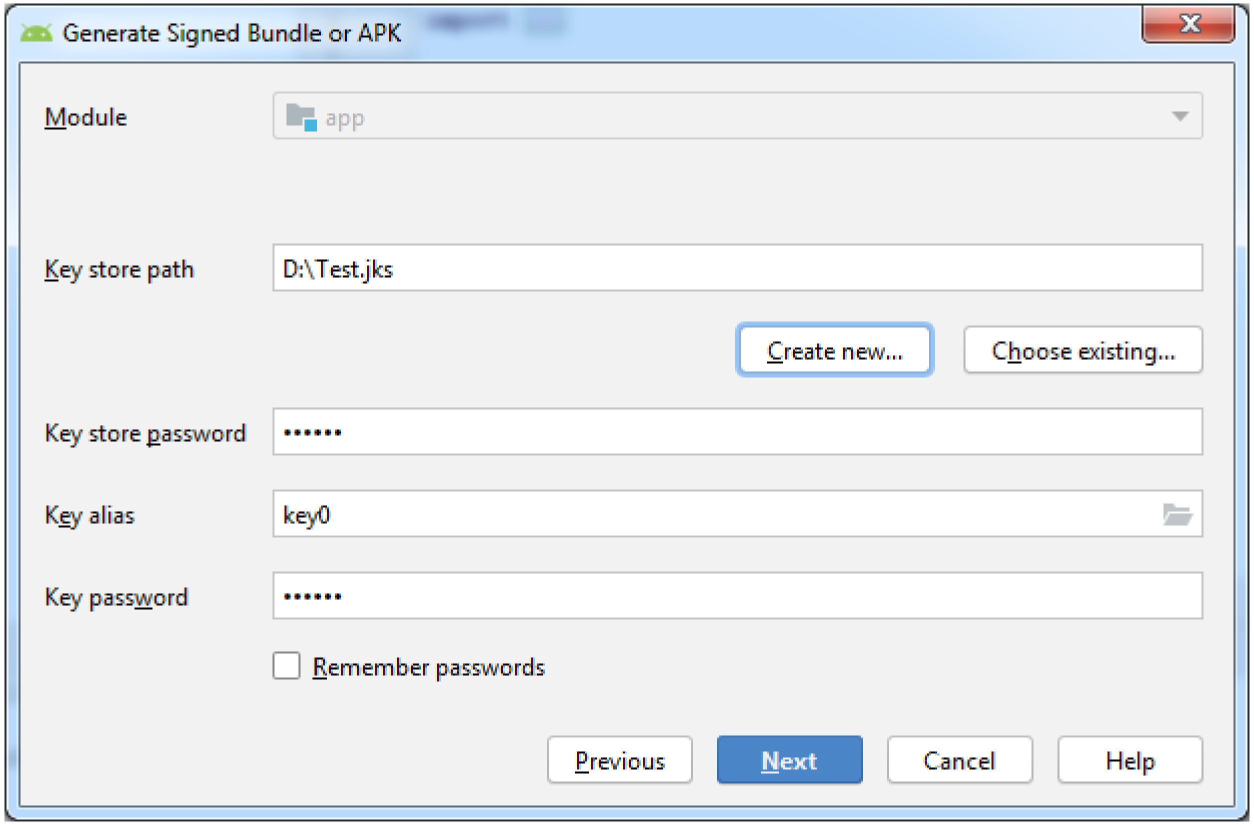
Alias:	key0		
Password:	••••••	Confirm:	••••••
Validity (years):	25		

Certificate

First and Last Name:	Nader Mahmoud
Organizational Unit:	Faculty of computers and Information
Organization:	Menofia University
City or Locality:	Shebien Elkom
State or Province:	Menofia
Country Code (XX):	Egypt

Generating a Signed APK

- Use the following steps to generate a signed APK:
  5. click Next to review and finish the process.



Module: app

Key store path: D:\Test.jks

Key store password: ••••••

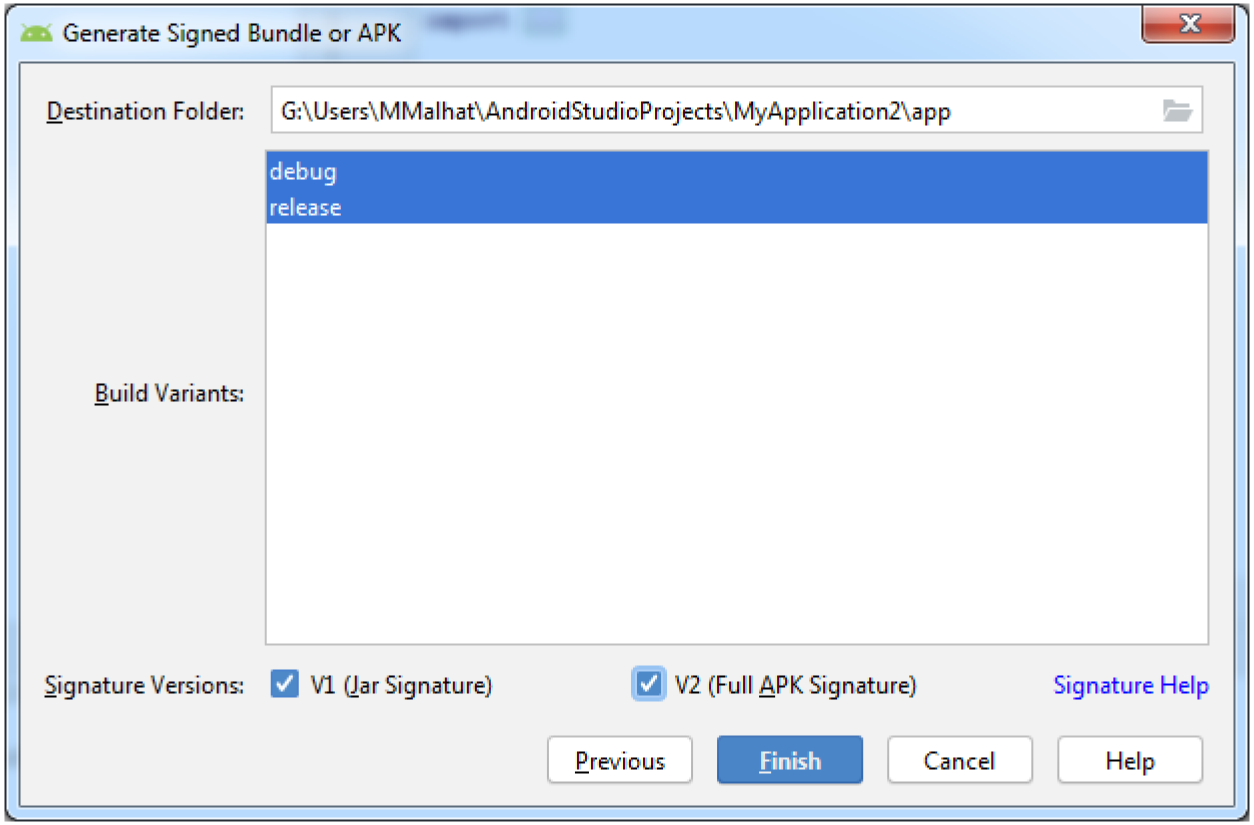
Key alias: key0

Key password: ••••••

☐ Remember passwords

Generating a Signed APK

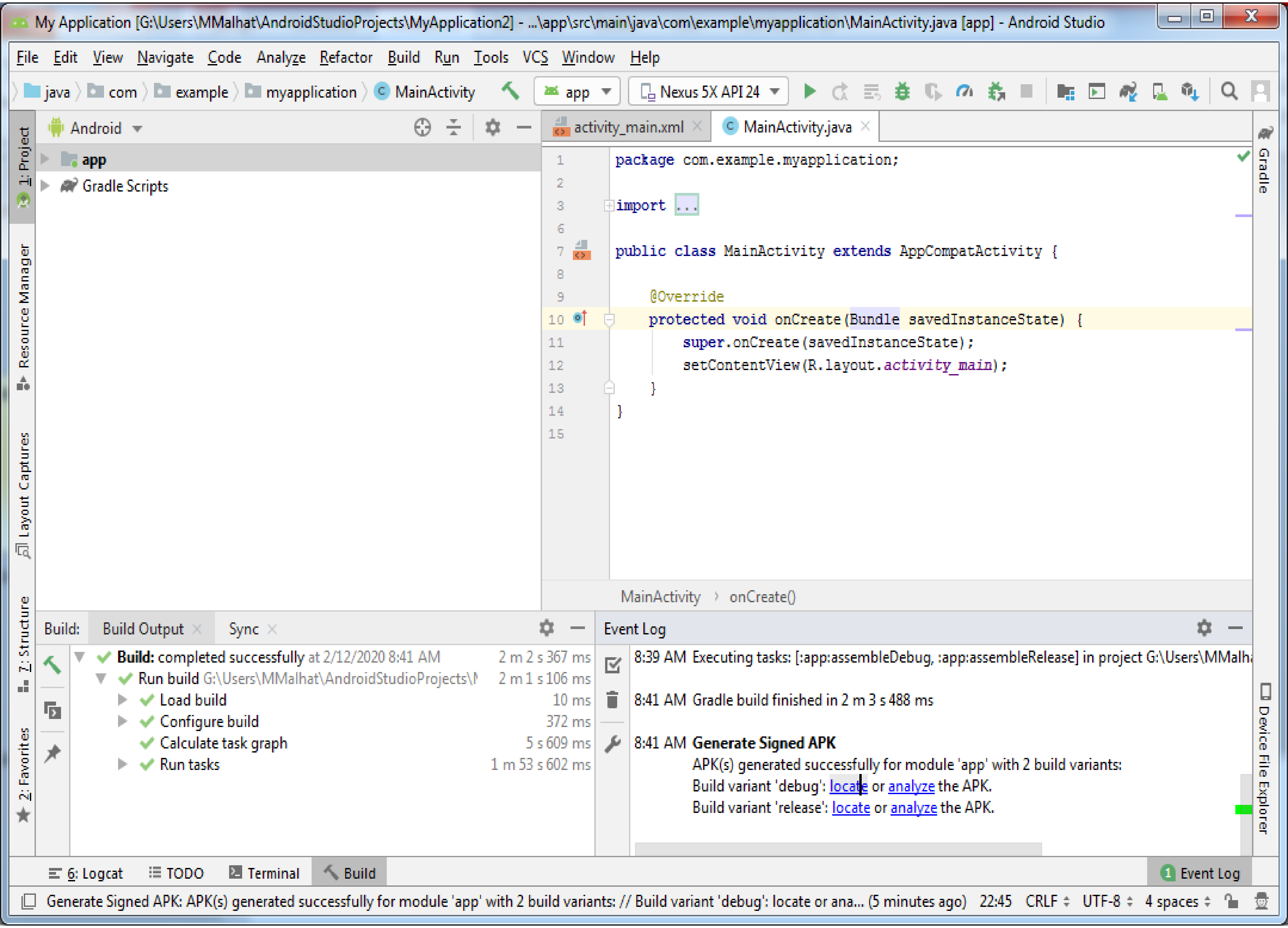
- Use the following steps to generate a signed APK:
  6. Select Build Variants and Signature versions, then click finish.



Destination Folder: G:\Users\MMalhat\AndroidStudioProjects\MyApplication2\app

Build Variants: debug, release

Signature Versions: [✓ V1 (Jar Signature)] [✓ V2 (Full APK Signature)]



My Application [G:\Users\MMalhat\AndroidStudioProjects\MyApplication2] - ...\app\src\main\java\com\example\myapplication\MainActivity.java [app] - Android Studio

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
java › com › example › myapplication › MainActivity
app
Nexus 5X API 24
Android
app
Gradle Scripts
activity_main.xml
MainActivity.java

```
package com.example.myapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

```
Build: completed successfully at 2/12/2020 8:41 AM            2 m 2 s 367 ms
  Run build G:\Users\MMalhat\AndroidStudioProjects\M          2 m 1 s 106 ms
    Load build                                                10 ms
    Configure build                                           372 ms
    Calculate task graph                                      5 s 609 ms
    Run tasks                                                 1 m 53 s 602 ms
```

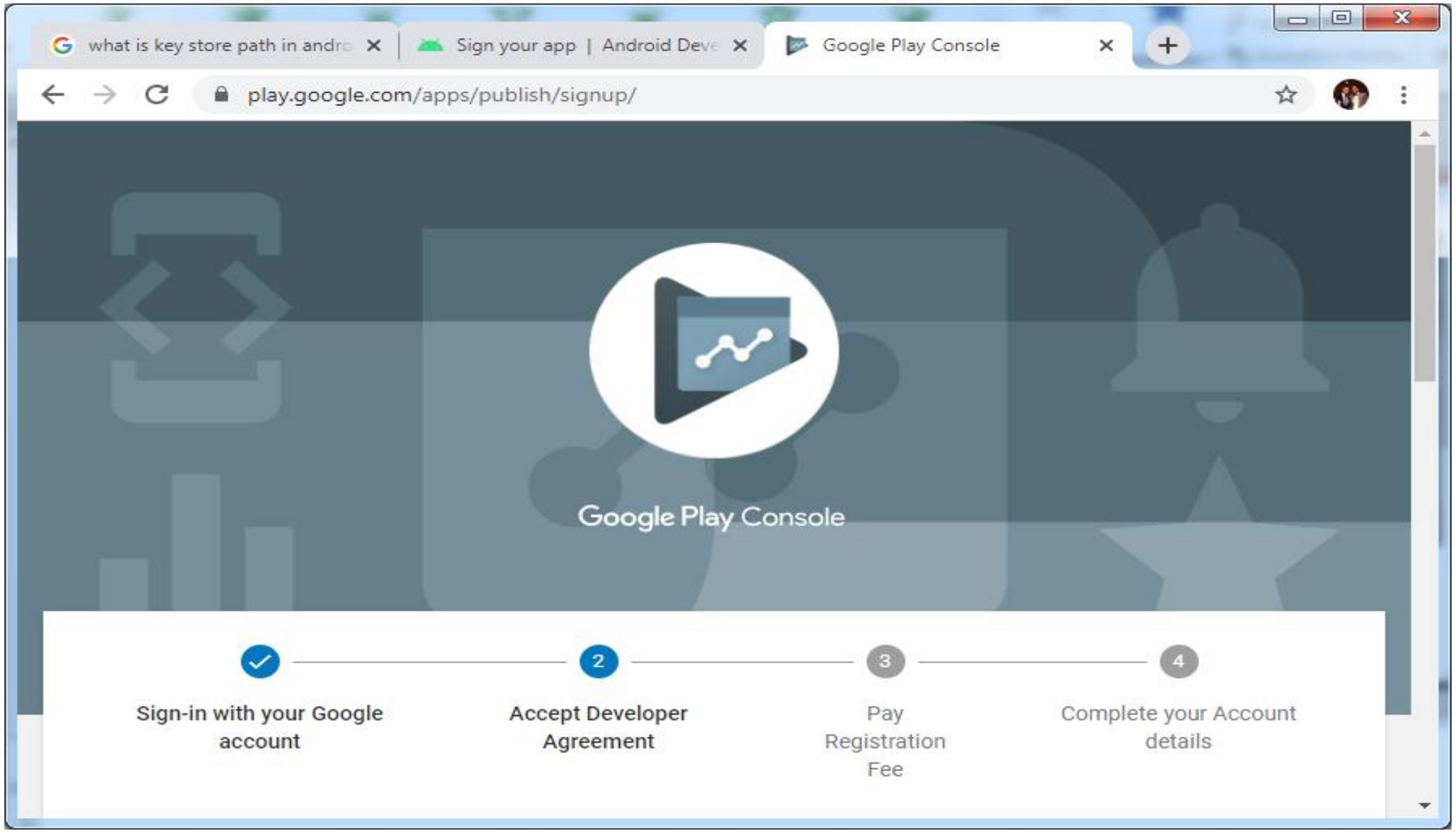
Event Log

```
8:39 AM Executing tasks: [:app:assembleDebug, :app:assembleRelease] in project G:\Users\MMalh

8:41 AM Gradle build finished in 2 m 3 s 488 ms

8:41 AM Generate Signed APK
        APK(s) generated successfully for module 'app' with 2 build variants:
        Build variant 'debug': locate or analyze the APK.
        Build variant 'release': locate or analyze the APK.
```

Generate Signed APK: APK(s) generated successfully for module 'app' with 2 build variants: // Build variant 'debug': locate or ana... (5 minutes ago) 22:45 CRLF UTF-8 4 spaces

Generating a Signed APK

- Now that you have a signed APK, you can upload it to the Google Play Store using the developer console at https://play.google.com/apps/publish/


what is key store path in andro
Sign your app | Android Deve
Google Play Console
play.google.com/apps/publish/signup/
Google Play Console
Sign-in with your Google account
2
Accept Developer Agreement
3
Pay Registration Fee
4
Complete your Account details

End of Lecture