Android App development for the GFS6p forecast

By: Bas Retsios

Date: 25-November-2021

Goal

Develop an app for the Android operating system that displays the 10-day forecast graphs of the 6 parameters of the Global Forecast System (GFS).

Prerequisite

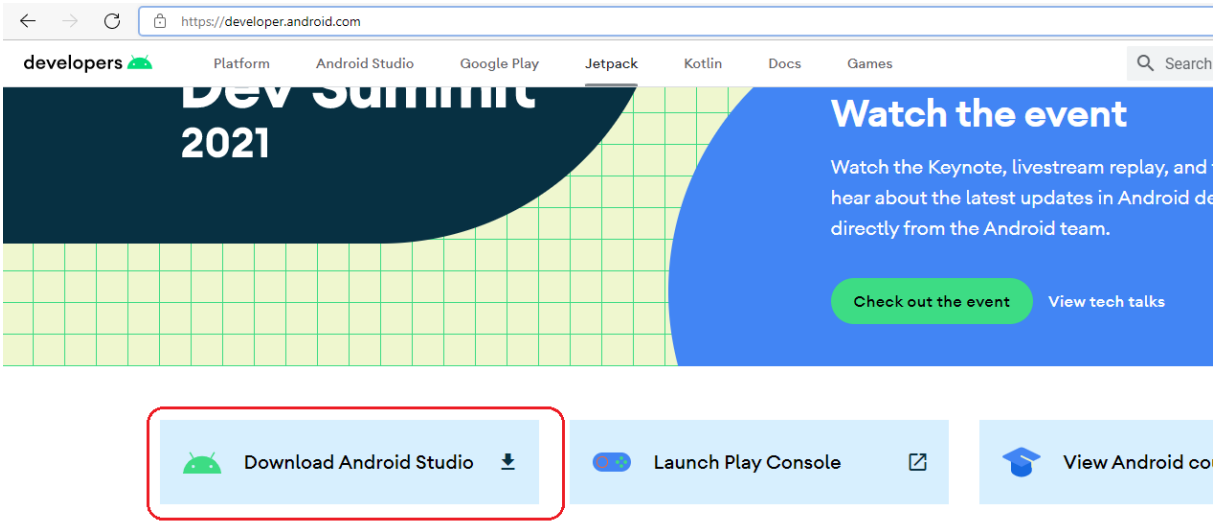
- A Web API service that provides the 10-day GFS forecast data for a single lat/lon location, instantly, and requiring minimal internet bandwidth.
- A device (Android telephone) with internet connectivity.

Download, Install and Configure Android Studio

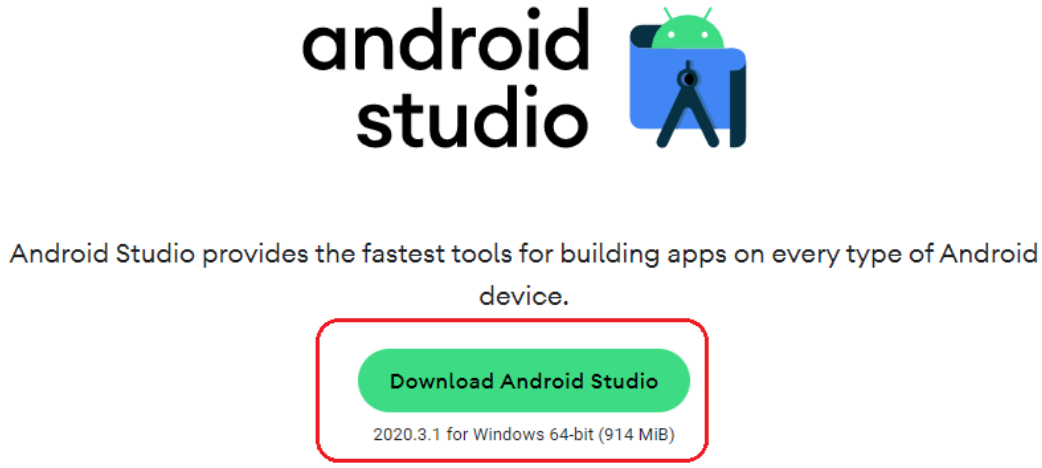
An app for the Android operating system is best developed with Android Studio, which is a free software environment provided exactly for this.

Navigate to the android developer website:

https://developer.android.com/



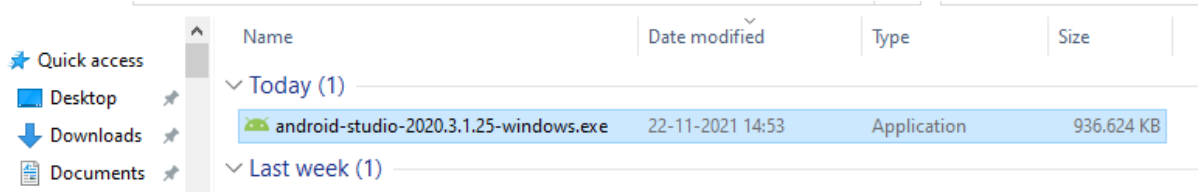
Click the Download Android Studio button.



Read and accept the license agreement and click the Download button at the end of the license.

Depending on the speed of your internet connection, downloading may take a few minutes.

The file will appear in your Downloads folder.

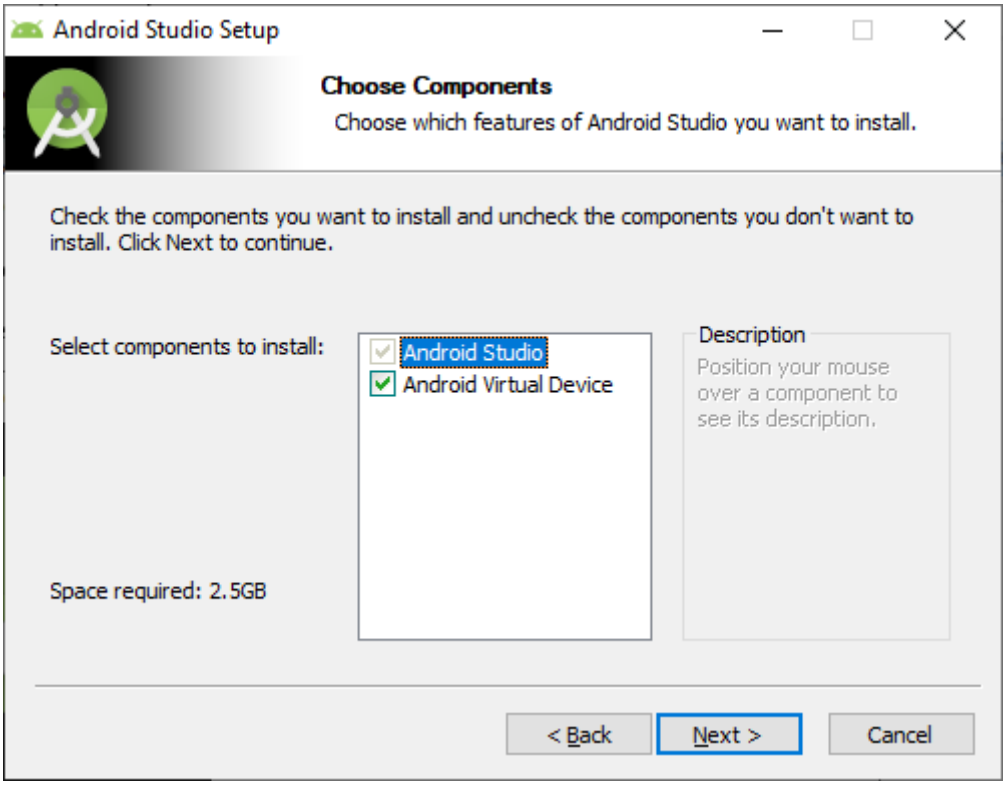


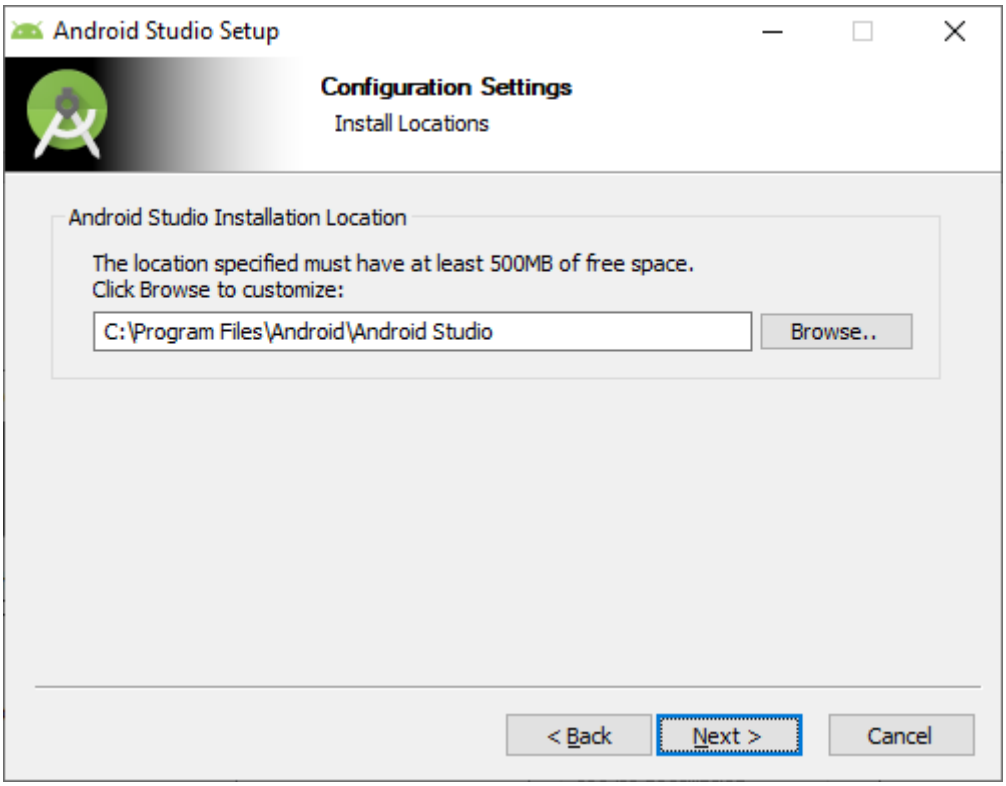
Double-click it to start the installation.

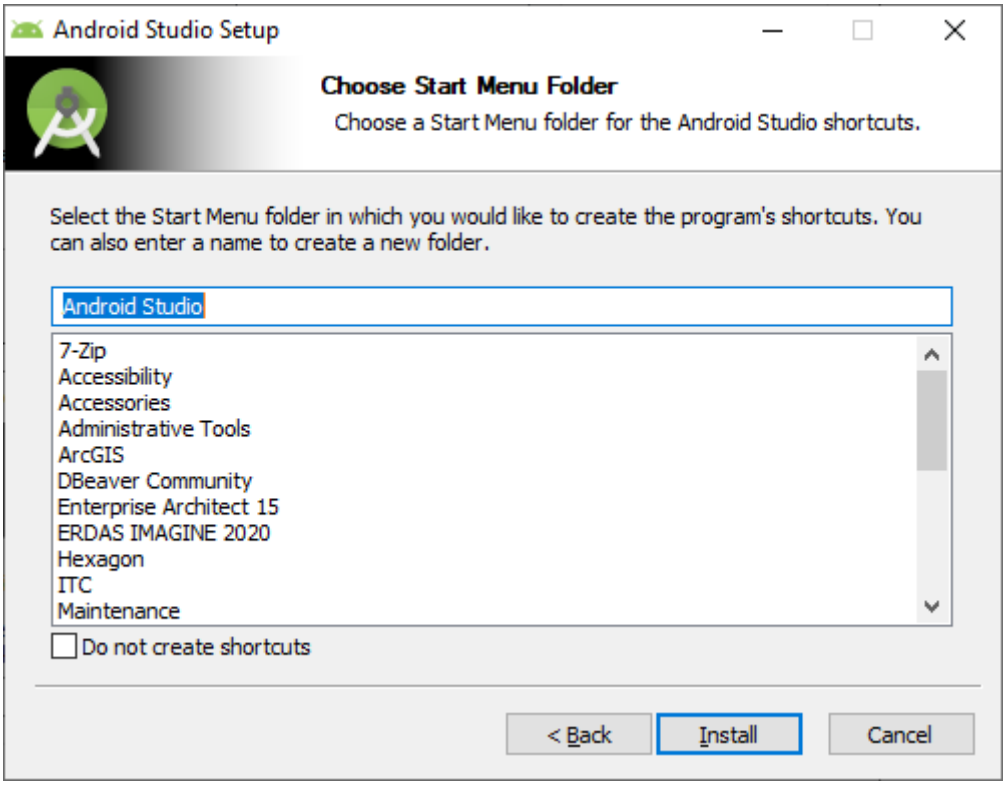
Answer Yes to the question “Do you want this app to make changes to your computer?”

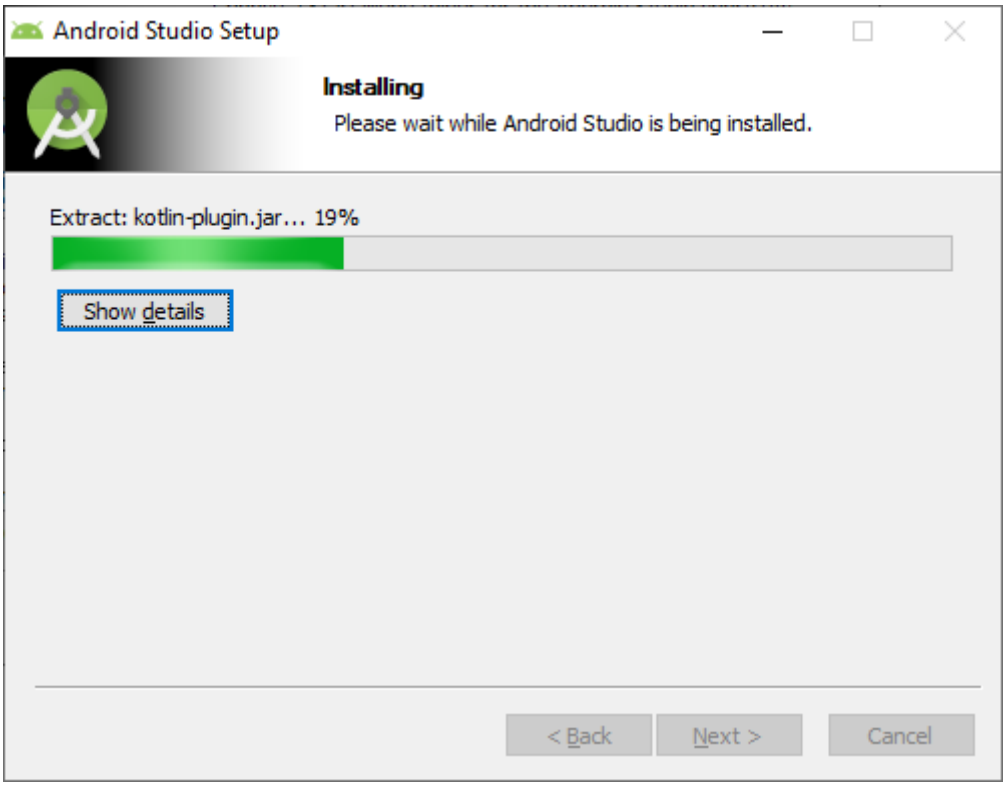
At every screen you may accept the default options, and Install.

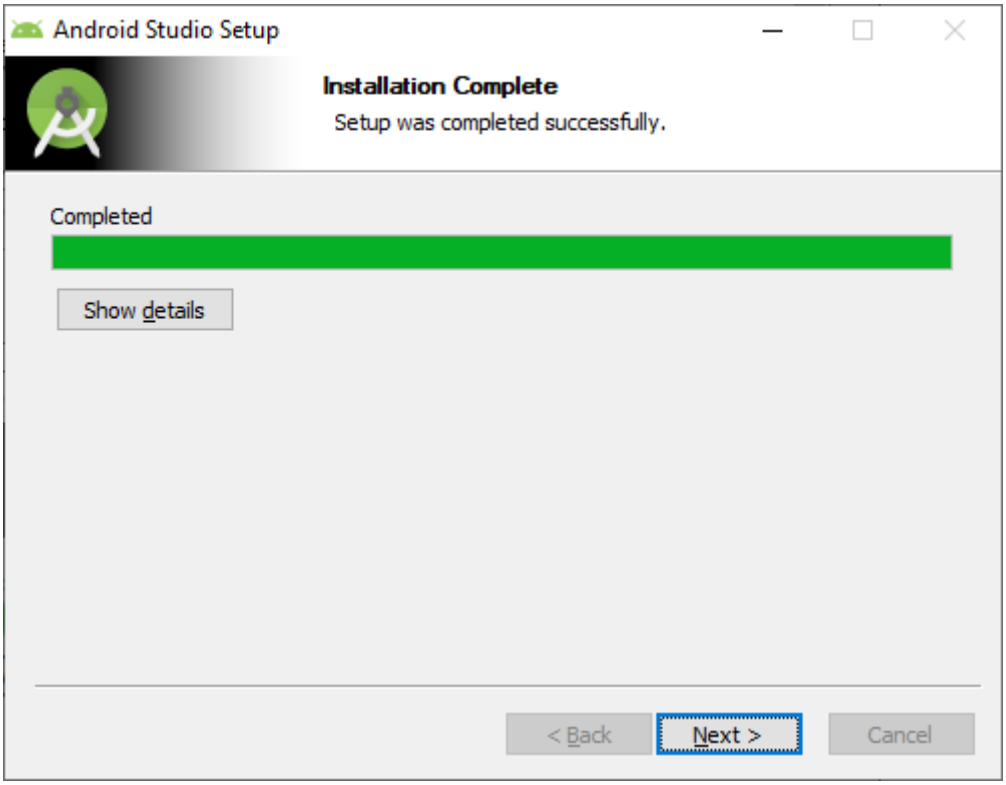

Android Studio Setup
Welcome to Android Studio Setup
Setup will guide you through the installation of Android Studio.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Android
Studio
< Back
Next >
Cancel

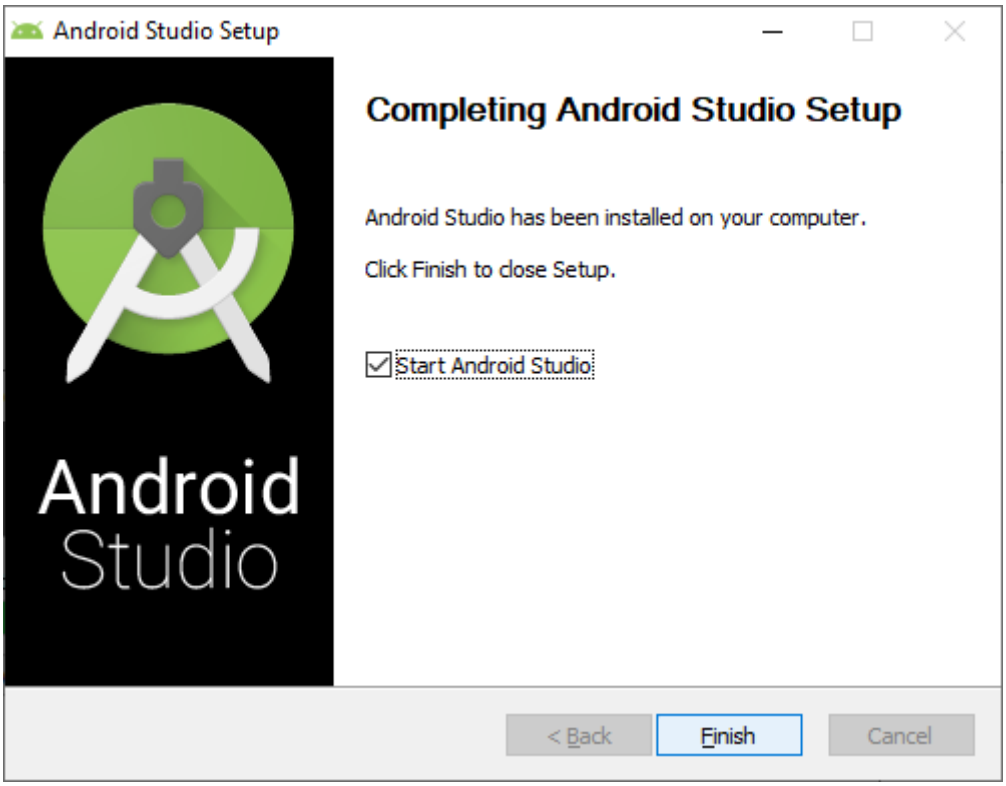

Android Studio Setup
Choose Components
Choose which features of Android Studio you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Android Studio
Android Virtual Device
Description
Position your mouse over a component to see its description.
Space required: 2.5GB
< Back
Next >
Cancel


Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse..
< Back
Next >
Cancel


Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Android Studio
7-Zip
Accessibility
Accessories
Administrative Tools
ArcGIS
DBeaver Community
Enterprise Architect 15
ERDAS IMAGINE 2020
Hexagon
ITC
Maintenance
Do not create shortcuts
< Back
Install
Cancel


Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: kotlin-plugin.jar... 19%
Show details
< Back
Next >
Cancel

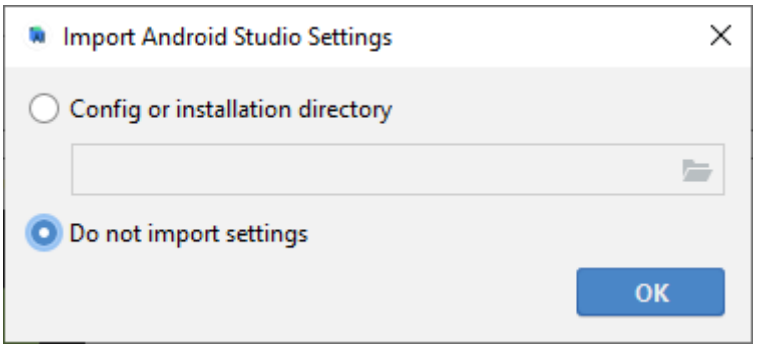

Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel

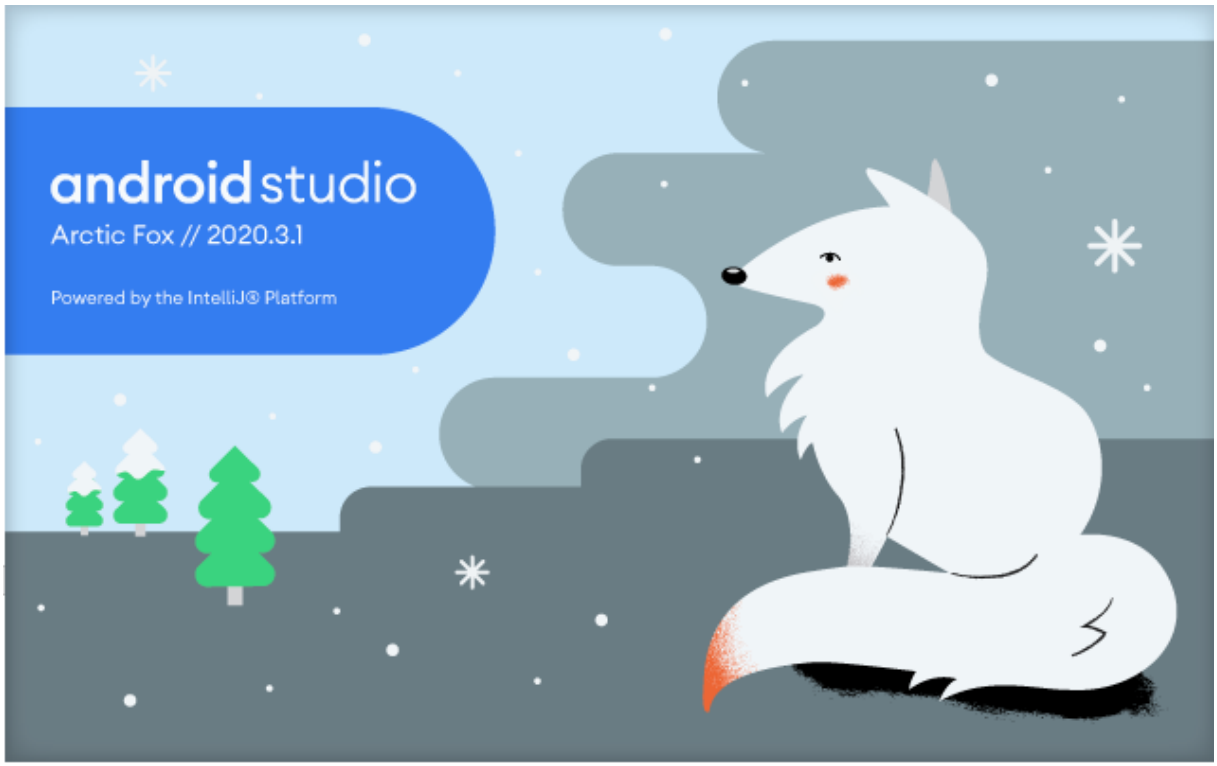


Start Android Studio if you haven't let the installer start it for you.

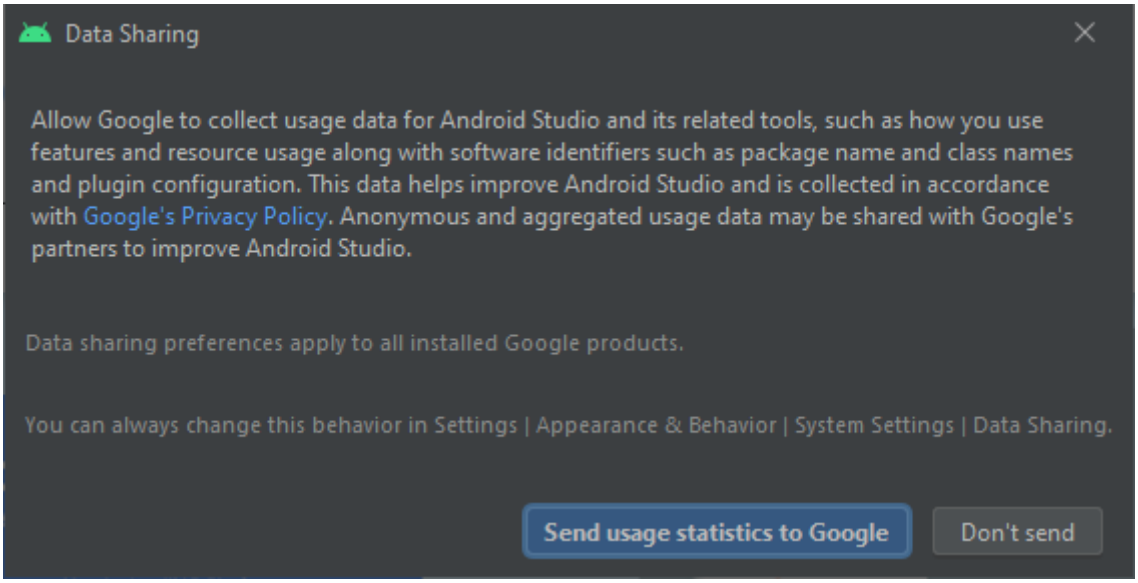
You will be offered the option to import settings from a previous installation / older version.

Assuming that you haven't used Android Studio previously, you can proceed with the option "Do not import settings".

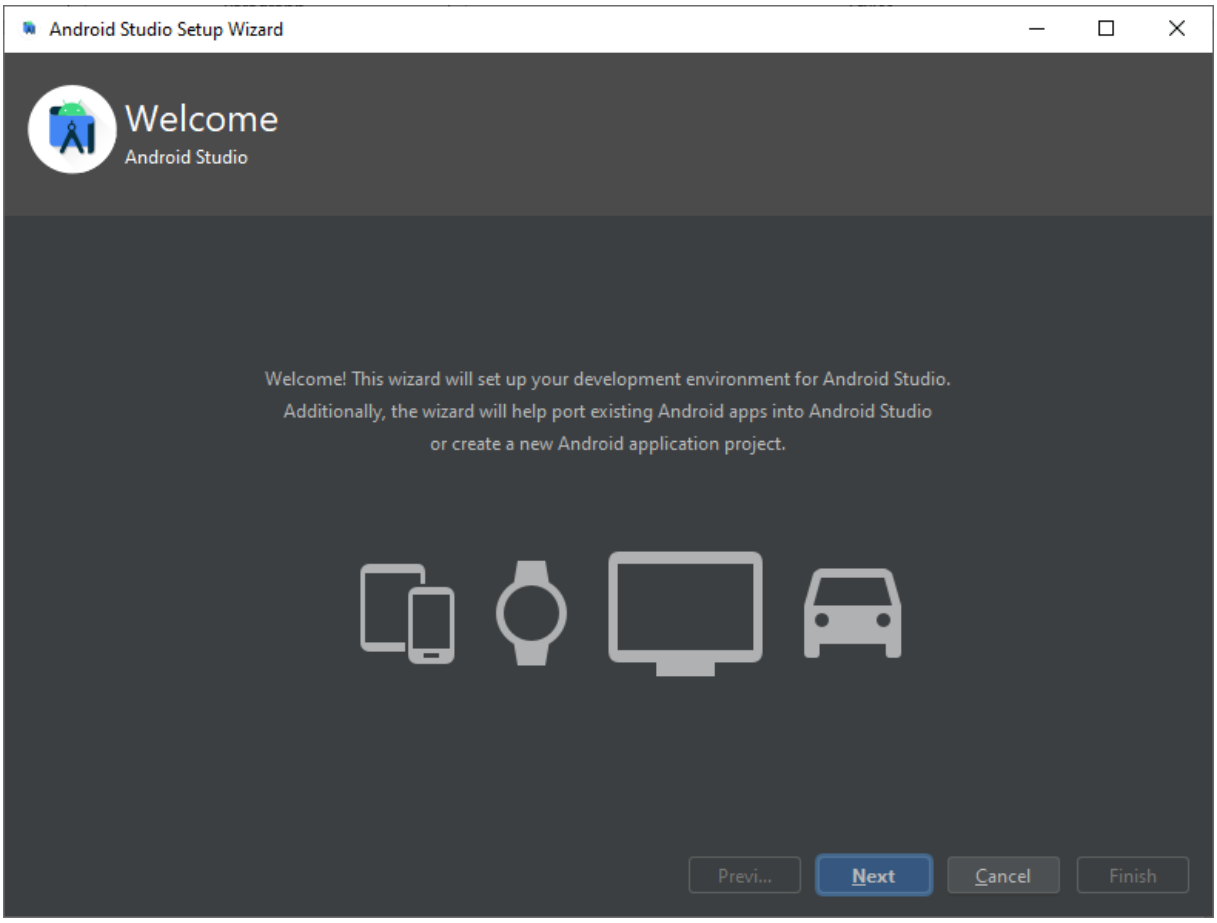




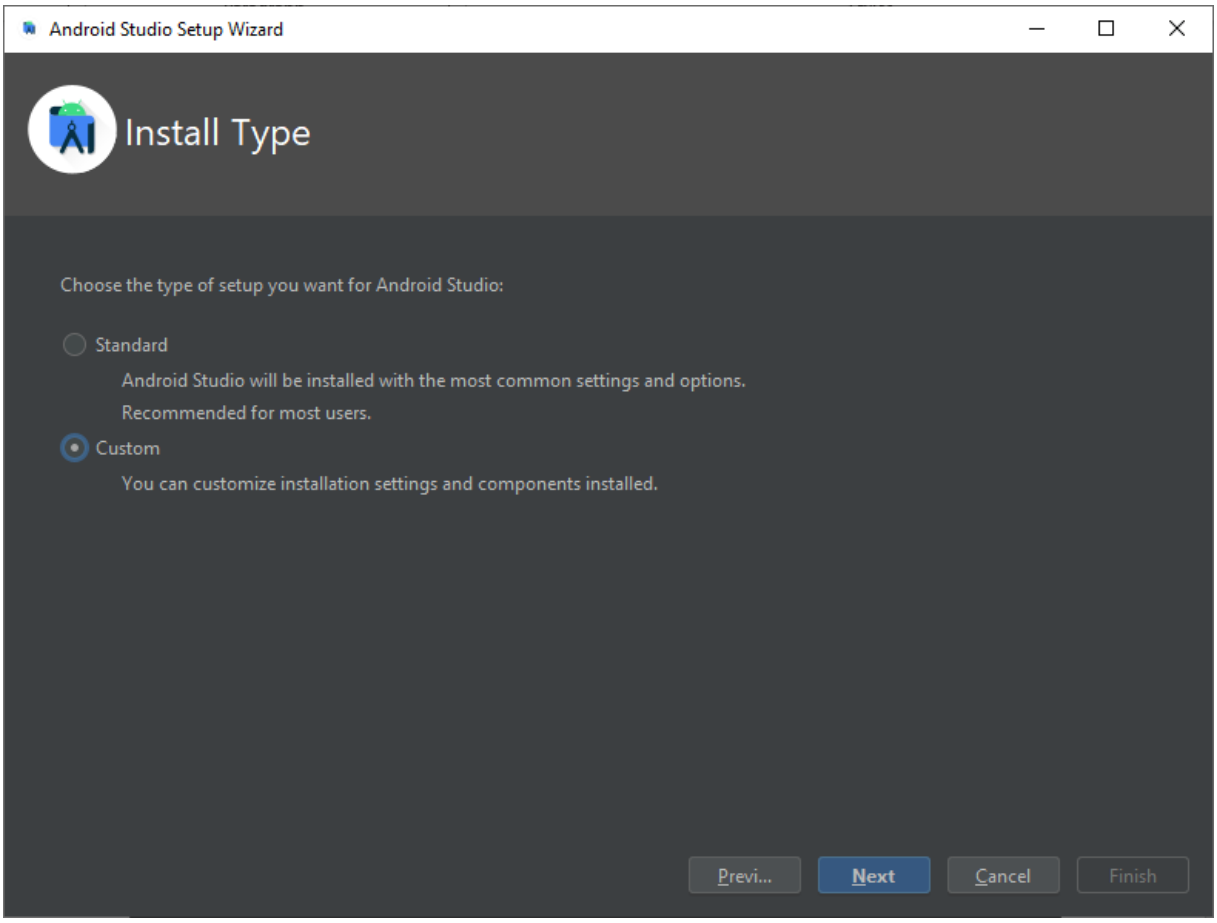
Read and decide by yourself about the Google data collection. It is for a good cause (which is to improve Android Studio) but you may not like to share usage data. Note the text that says that “Data sharing preferences apply to all installed Google products” so your decision here would obviously overrule a previously taken decision on this.

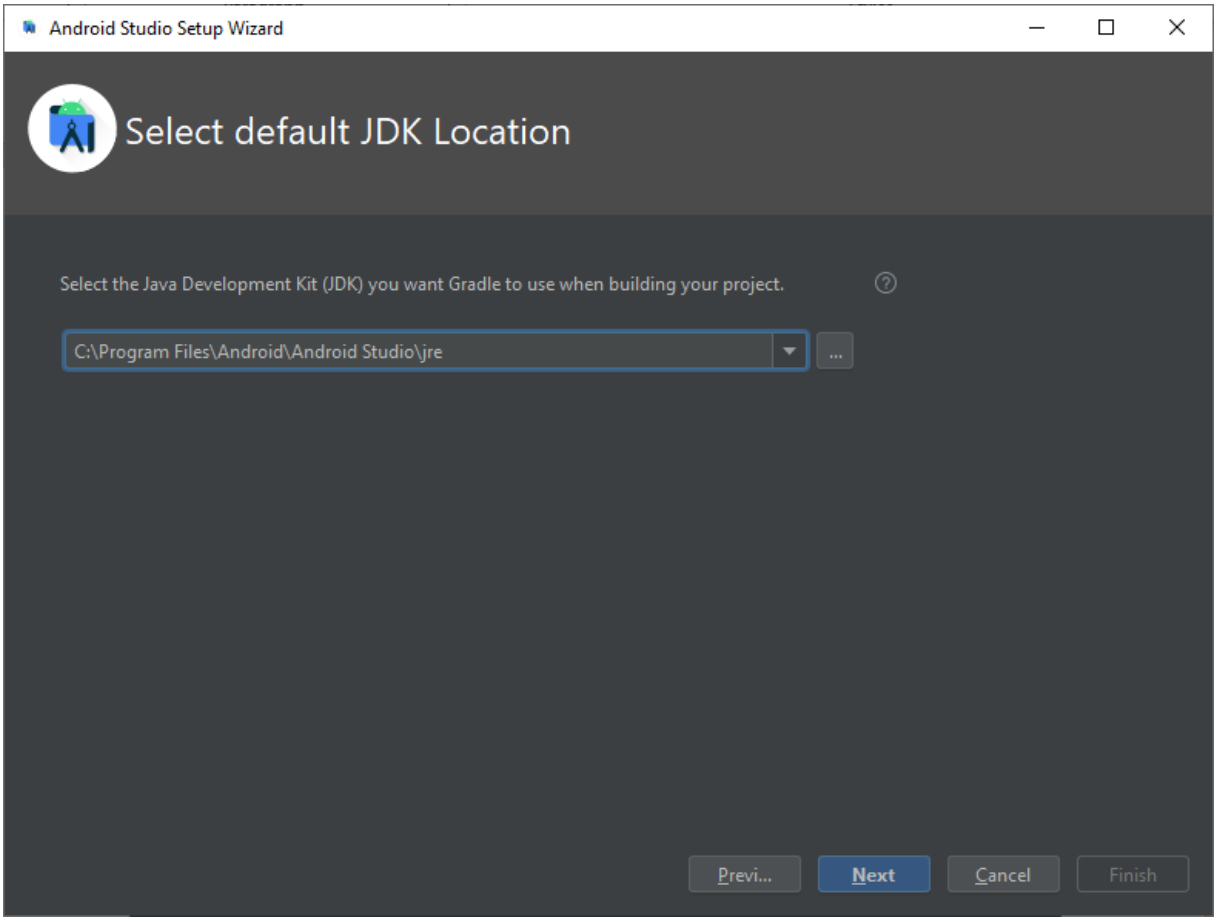


Note that Android Studio offers two computer languages for app development: Java and Kotlin. Java is the original language used for all Android Apps. Kotlin is recently added. This document focuses on app development using Java. In that sense you can ignore any Kotlin related offers, resources or available updates.

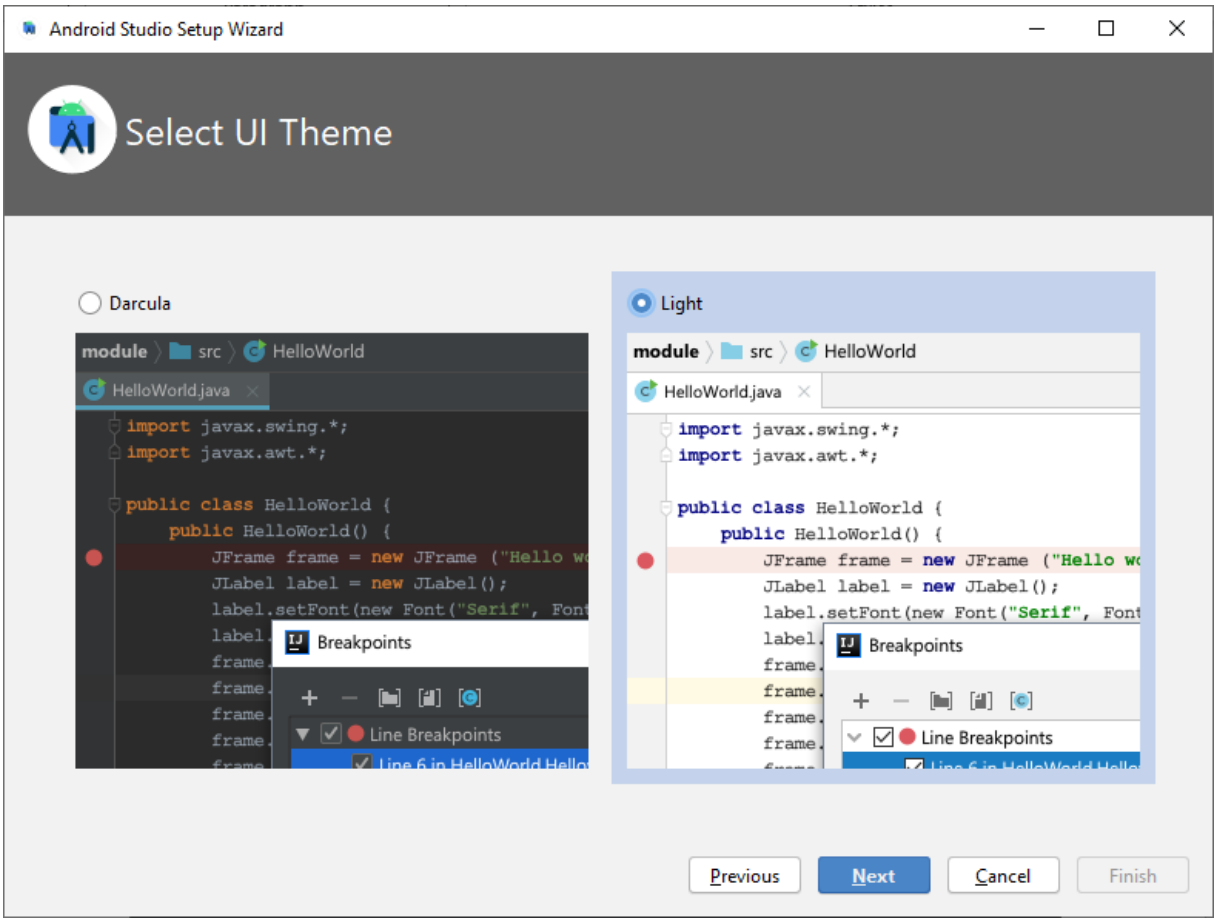


Choose “Custom” install.


Android Studio Setup Wizard
Install Type
Choose the type of setup you want for Android Studio:
Standard
Android Studio will be installed with the most common settings and options.
Recommended for most users.
Custom
You can customize installation settings and components installed.
Previ...
Next
Cancel
Finish


Android Studio Setup Wizard
Select default JDK Location
Select the Java Development Kit (JDK) you want Gradle to use when building your project.
C:\Program Files\Android\Android Studio\jre
...
Previ...
Next
Cancel
Finish

UI Theme: it is a personal preference. In this document we have chosen “light”.



Android Studio Setup Wizard

SDK Components Setup

Check the components you want to update/install. Click Next to continue.

- [x] Android SDK – (376 MB)
- [ ] Android SDK Platform
  - [x] API 31: Android 12.0 (S) – (152 MB)
- [x] Performance (Intel® HAXM) – (500 KB)
- [ ] Android Virtual Device – (installed)

The collection of Android platform APIs, tools and utilities that enables you to debug, profile, and compile your apps. The setup wizard will update your current Android SDK installation (if necessary) or install a new version.

Android SDK Location:

C:\Users\retsios\AppData\Local\Android\Sdk

Total download size: 529 MB

266 GB (drive)

Previous | Next | Cancel | Finish

Android Studio Setup Wizard

Emulator Settings

We have detected that your system can run the Android emulator in an accelerated performance mode.

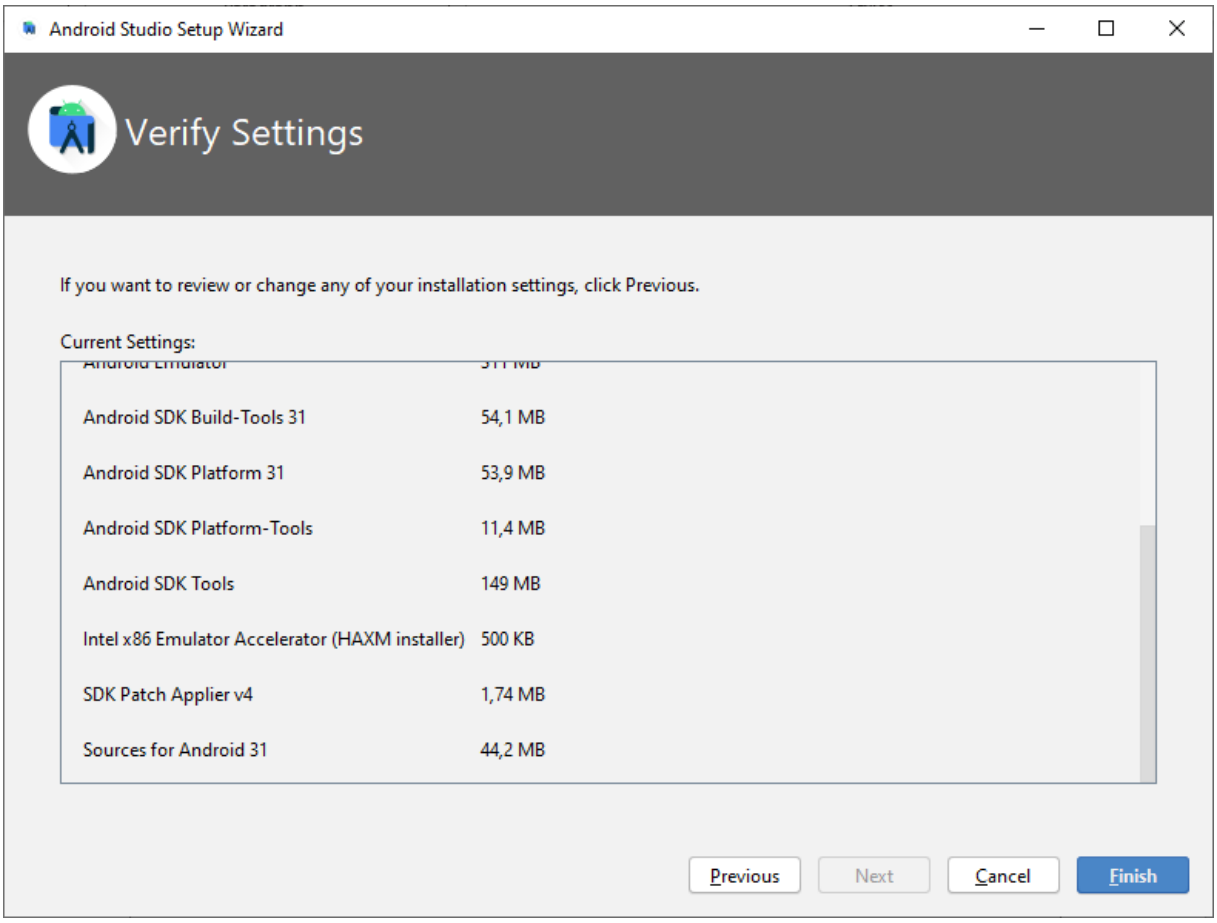
Set the maximum amount of RAM available for the Intel® Hardware Accelerated Execution Manager (HAXM) to use for all x86 emulato instances. You can change these settings at any time by running the Intel® HAXM installer.

Refer to the Intel® HAXM Documentation ↗ for more information.

512.0 MB | **2.0 GB (Recommended)** | 3.2 GB | 4.5 GB | 5.9 GB

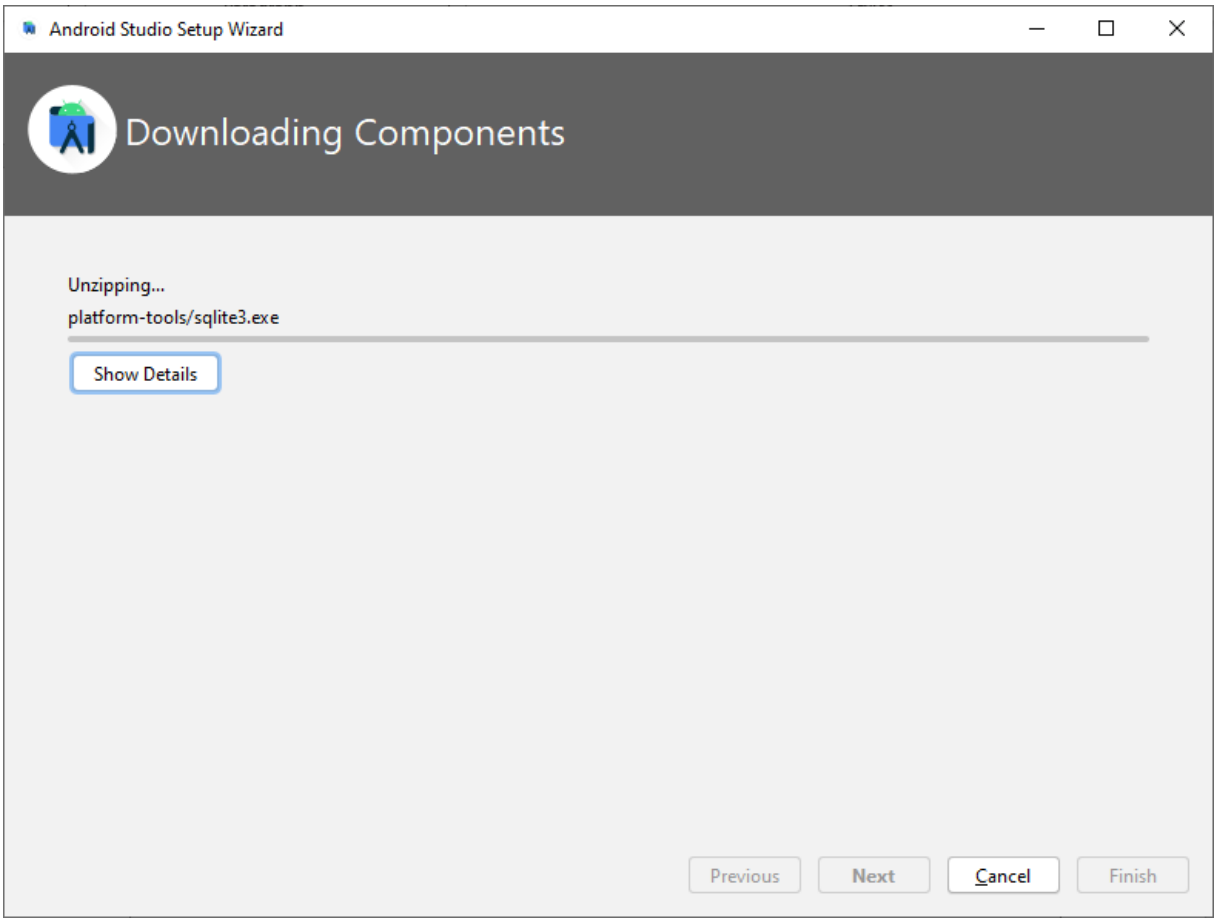
RAM allocation: 2.048 MiB Use recommended

Previous | Next | Cancel | Finish

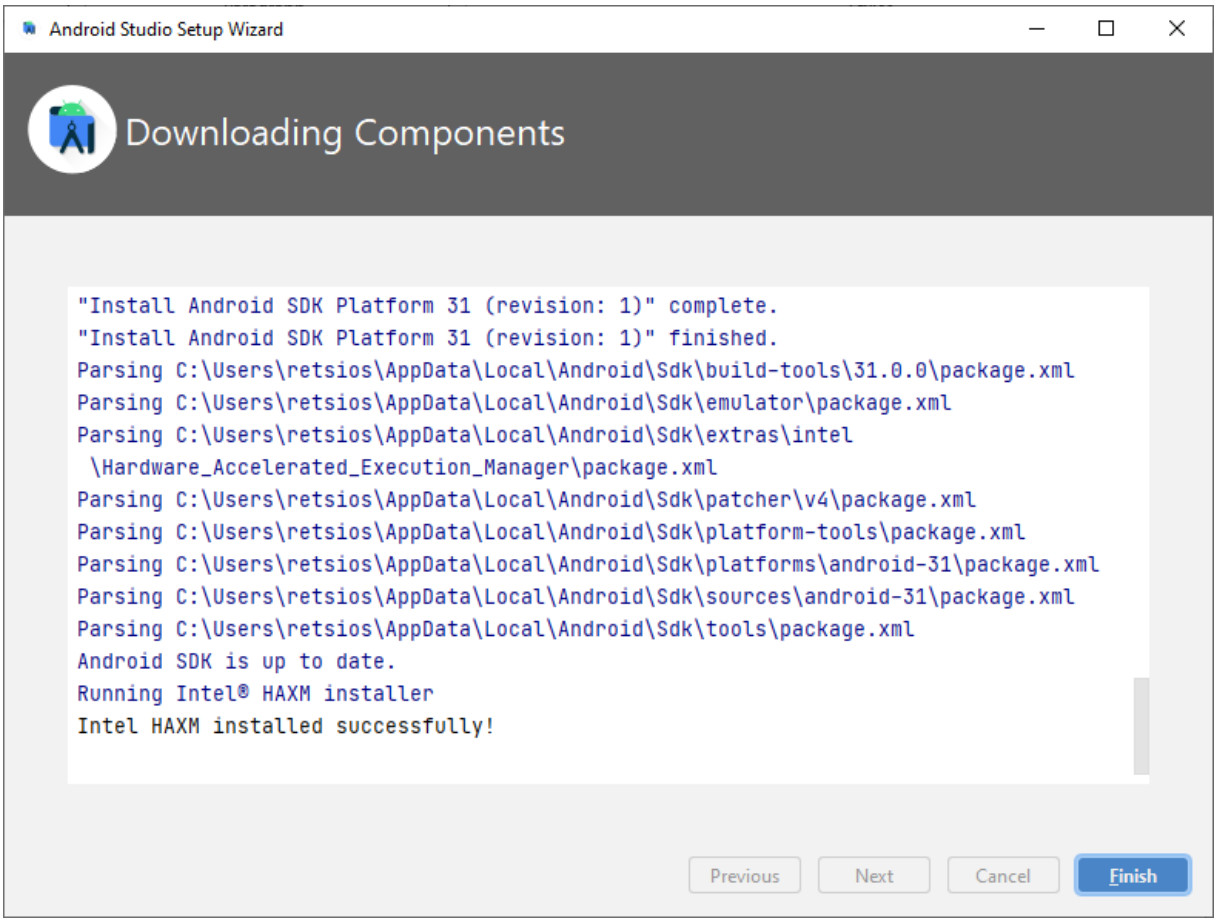


Click Finish to download the additional required components.

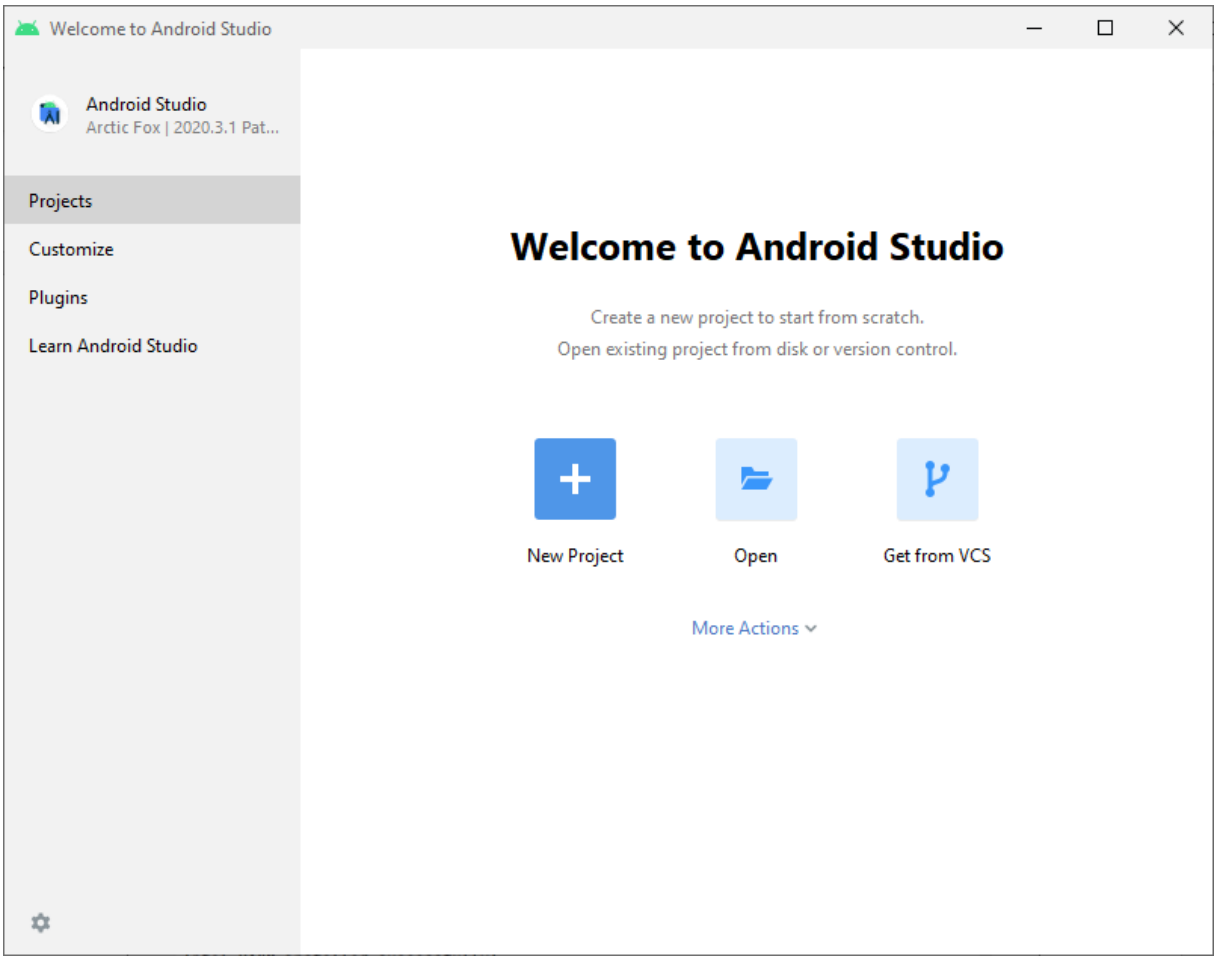
Downloading will start, and depending on your internet speed, this may take a few minutes.



When you receive a warning that Windows Command Processor wants to make changes to your computer, allow this.

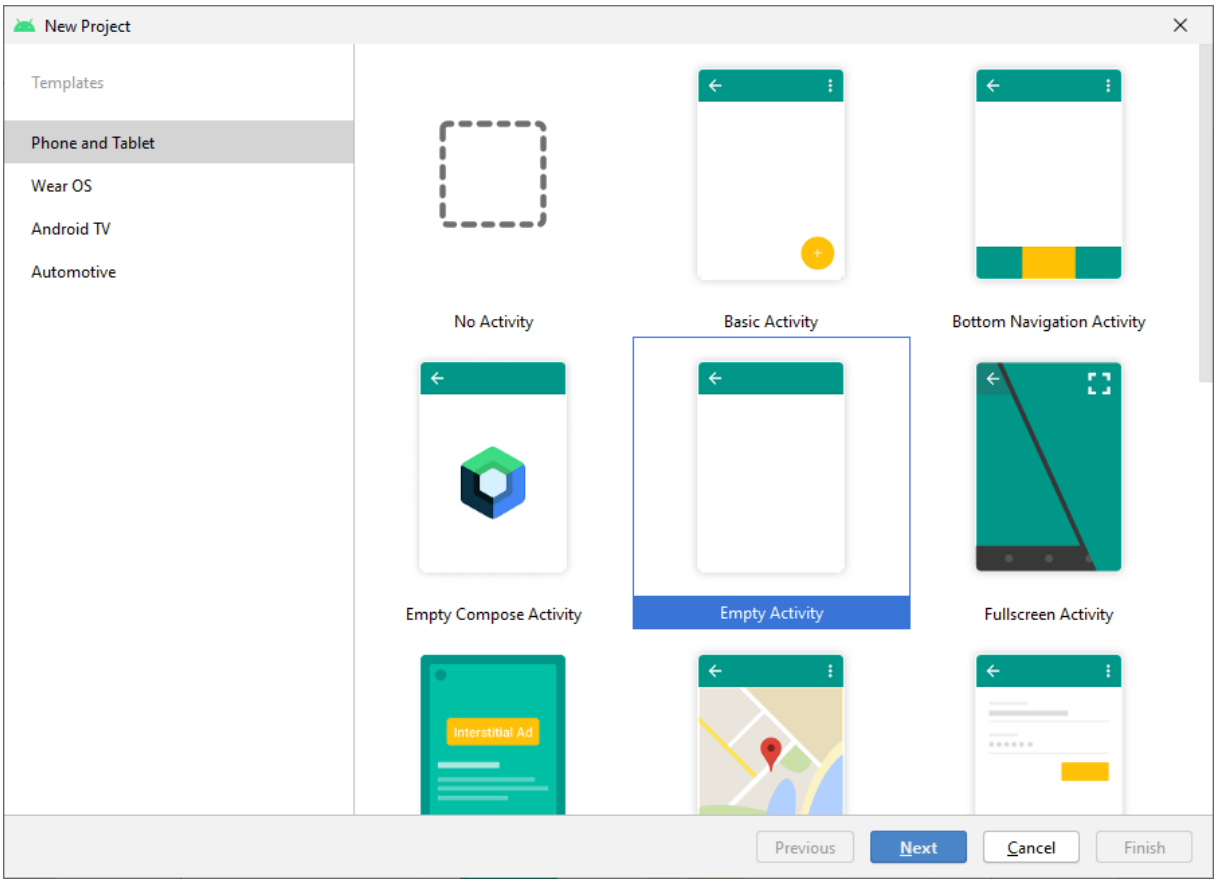

Android Studio Setup Wizard
Downloading Components
"Install Android SDK Platform 31 (revision: 1)" complete.
"Install Android SDK Platform 31 (revision: 1)" finished.
Parsing C:\Users\retsios\AppData\Local\Android\Sdk\build-tools\31.0.0\package.xml
Parsing C:\Users\retsios\AppData\Local\Android\Sdk\emulator\package.xml
Parsing C:\Users\retsios\AppData\Local\Android\Sdk\extras\intel
\Hardware_Accelerated_Execution_Manager\package.xml
Parsing C:\Users\retsios\AppData\Local\Android\Sdk\patcher\v4\package.xml
Parsing C:\Users\retsios\AppData\Local\Android\Sdk\platform-tools\package.xml
Parsing C:\Users\retsios\AppData\Local\Android\Sdk\platforms\android-31\package.xml
Parsing C:\Users\retsios\AppData\Local\Android\Sdk\sources\android-31\package.xml
Parsing C:\Users\retsios\AppData\Local\Android\Sdk\tools\package.xml
Android SDK is up to date.
Running Intel® HAXM installer
Intel HAXM installed successfully!
Previous
Next
Cancel
Finish

Creating your first app ('Hello World')

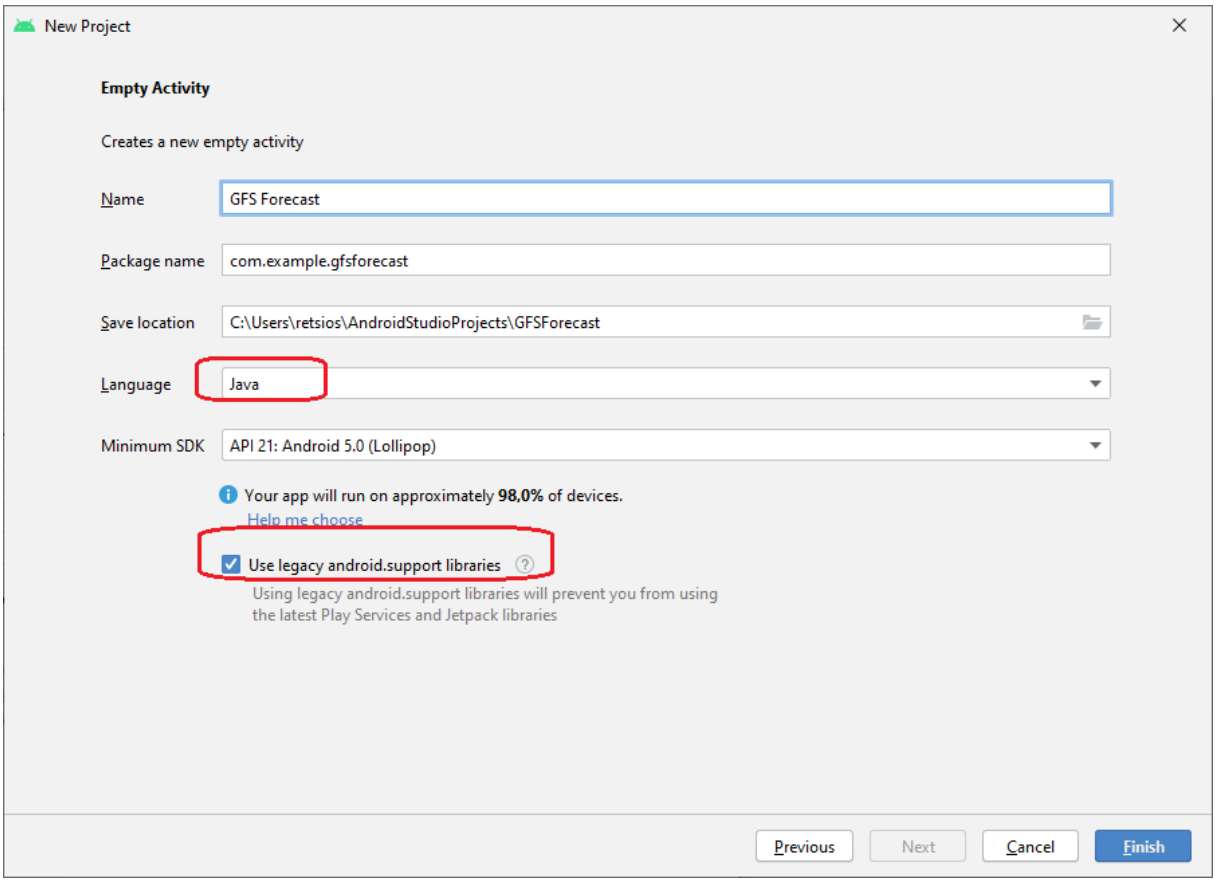


Click the '+' to create a New Project.

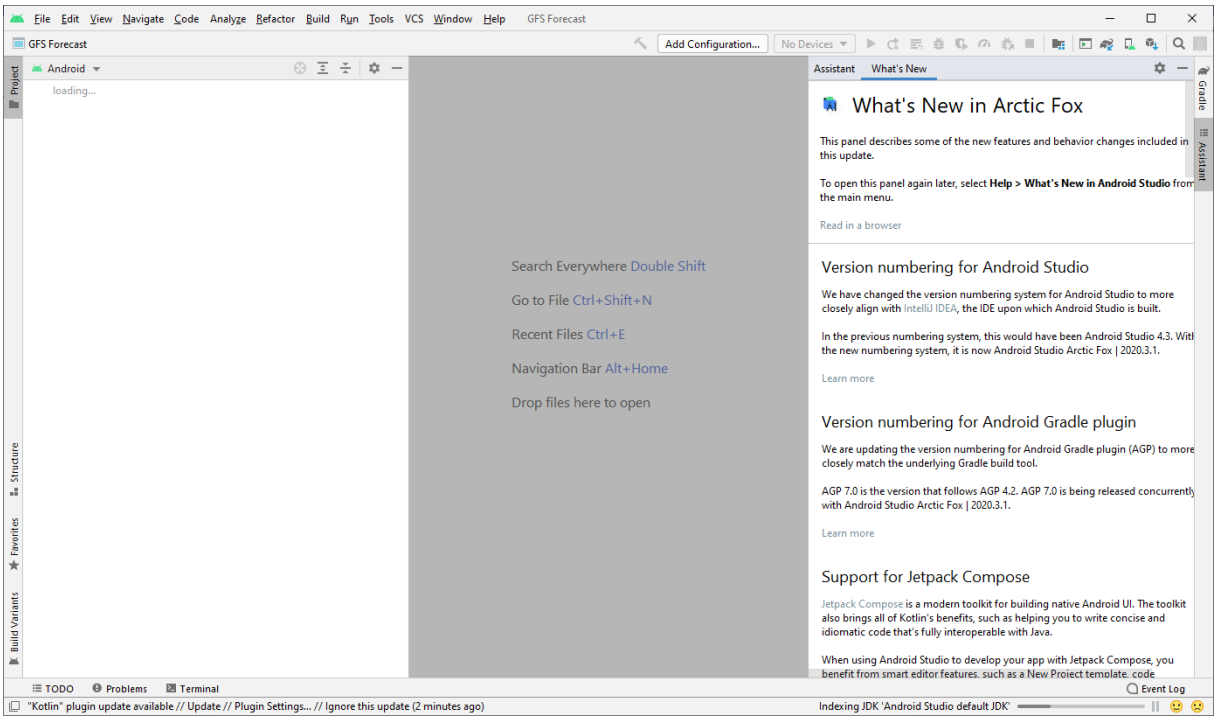
Select Empty Activity and click Next.



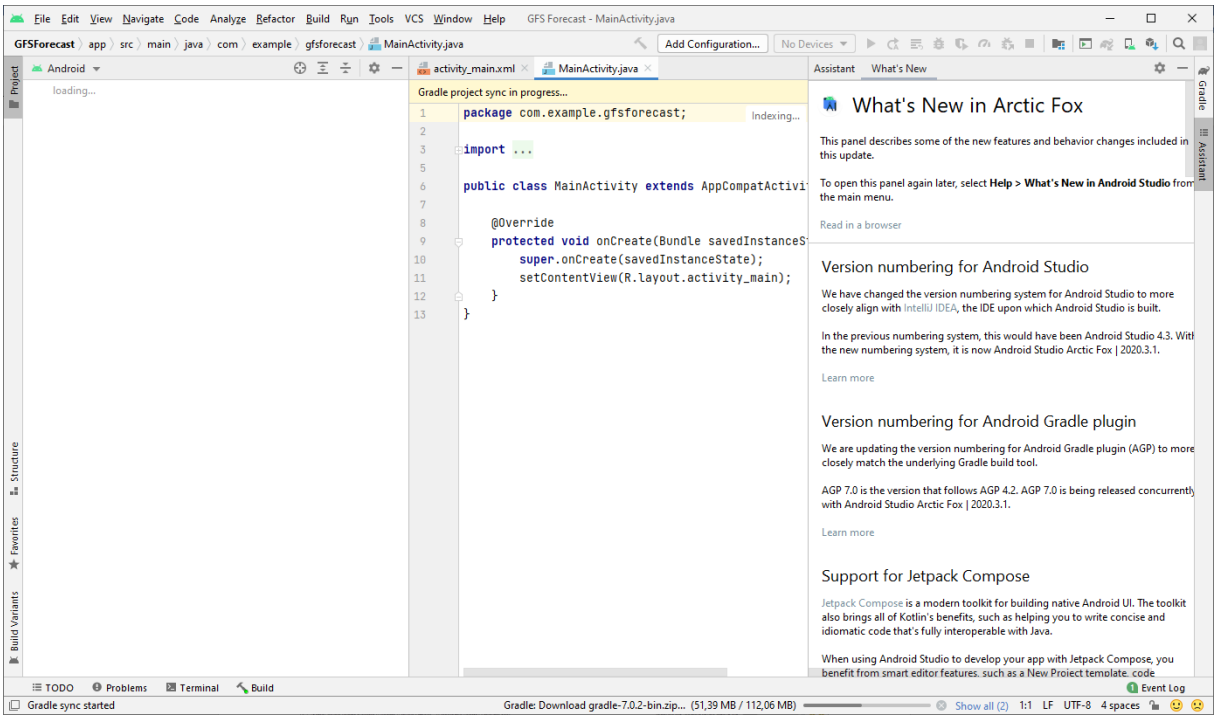
In the next screen, ensure to change “Kotlin” to “Java”. Also enable the legacy android.support libraries, in order to support older Android telephones or tablets. Leave the Minimum SDK to where it is (android 5.0 Lollipop), choose a nice name for your project, and click “Finish”.



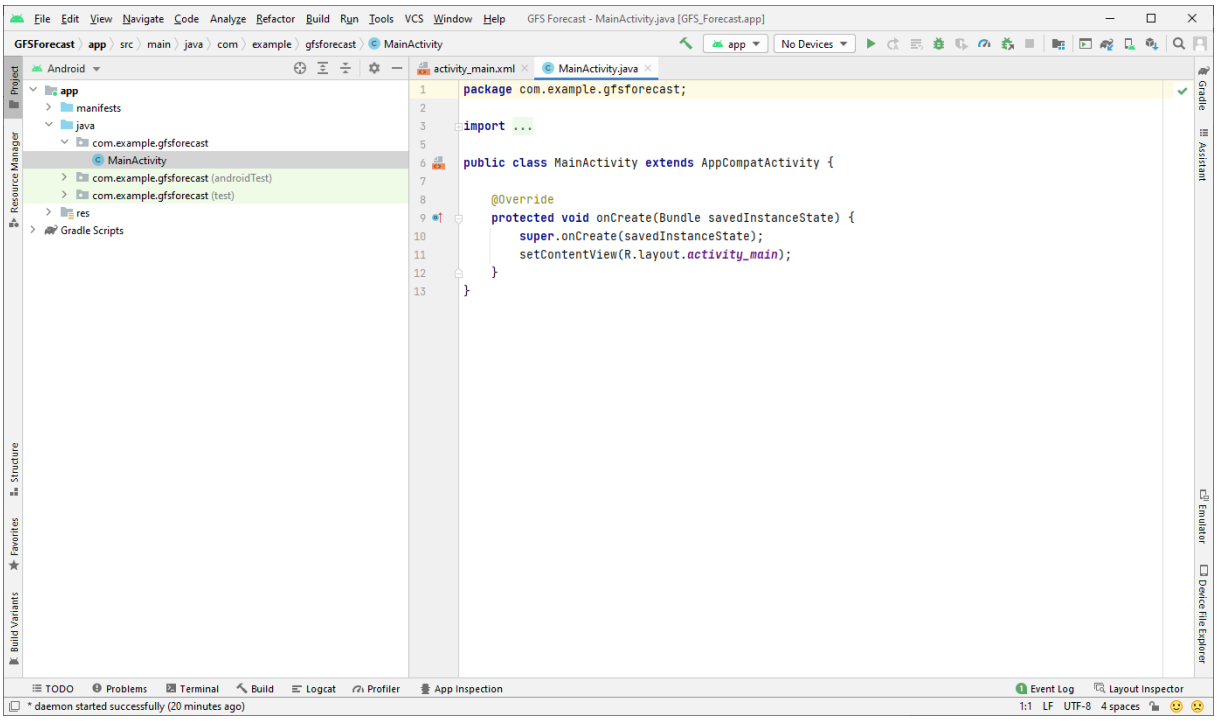
This will create the new project with the settings requested, and may take a few minutes.



After a few minutes:



You may Hide the screen “What’s new in Arctic Fox” (click the minus ‘-‘ sign).



Please inspect the files that were automatically created for this project:

manifests/AndroidManifest.xml (the global settings for this app)

AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="GFS Forecast"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.GFSForecast">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

java/com.example.fgsforecast/MainActivity.java (the main program that starts the app).

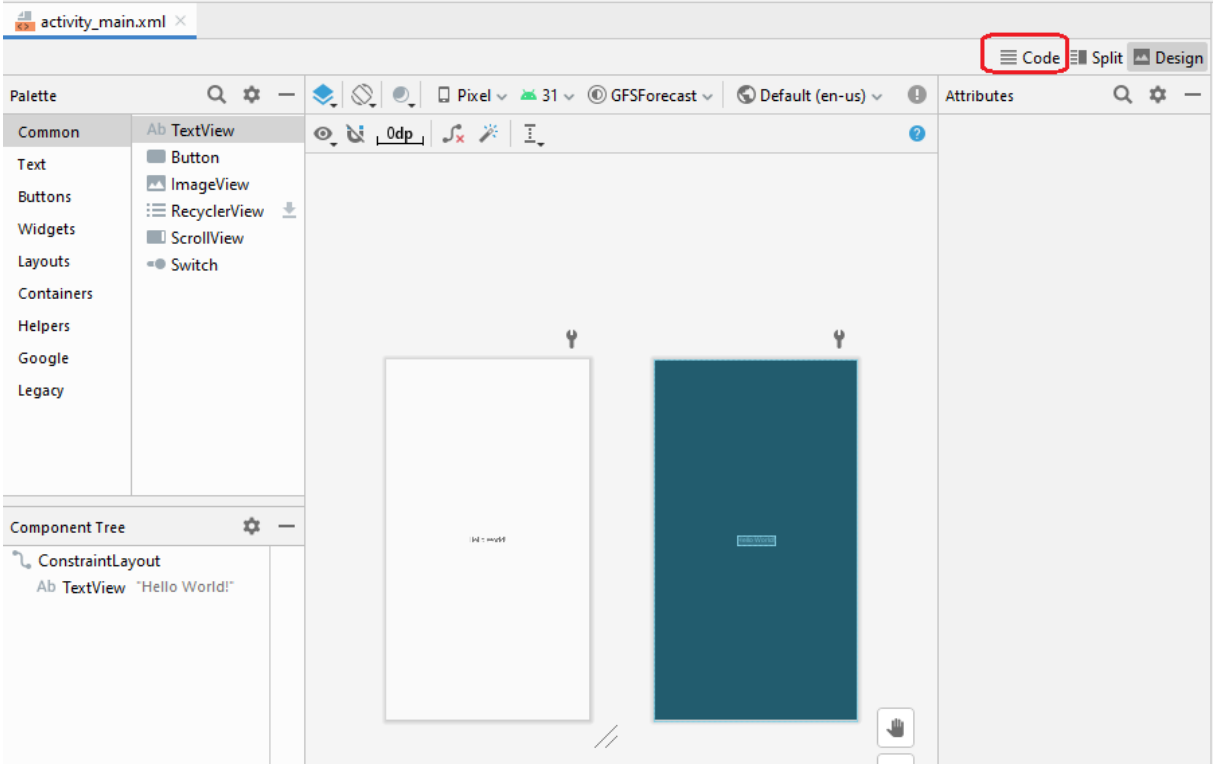
MainActivity.java

```java
package com.example.gfsforecast;

import android.support.v7.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);
    }
}
```

Note the line setContentView(R.Layout.activity_main);

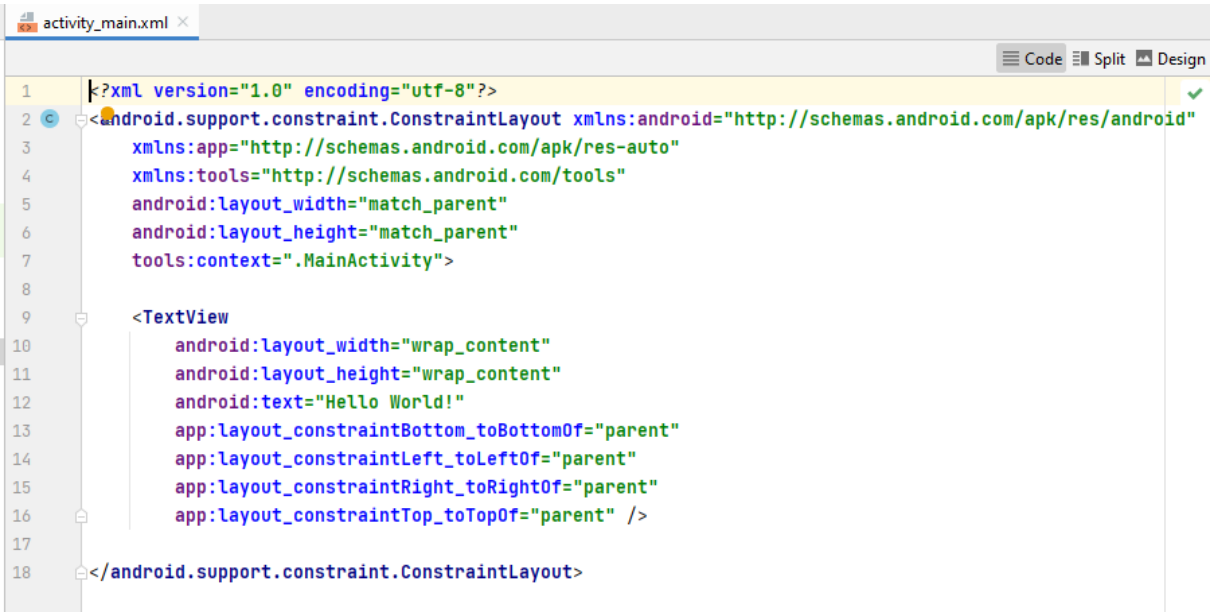
This line tells the program to display the layout from activity_main.xml .

res/layout/activity_main.xml (the layout specification of the main program).



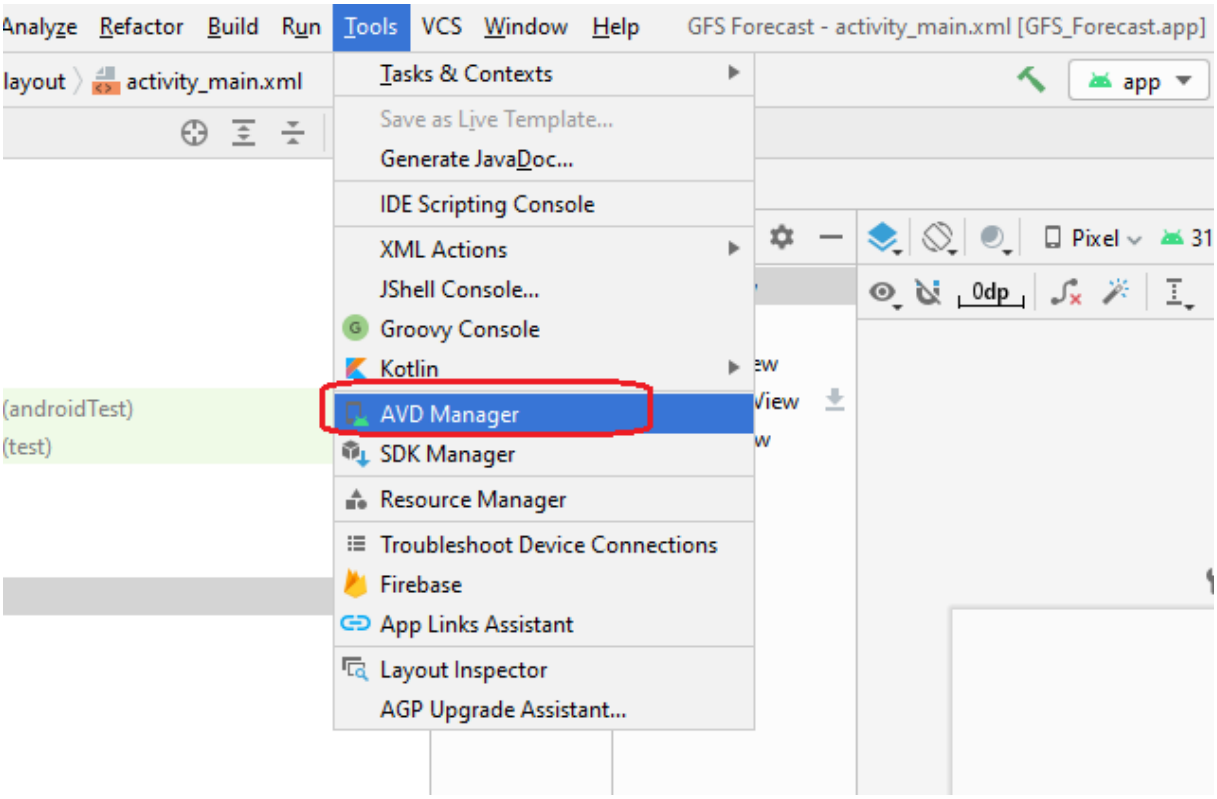
Click the Code button to view the layout programmatically. You can switch from Code to Design as often as you need. If you have enough space on the screen, you may even use the “Split” option to see both the code and the layout.

Note the text Hello World in the TextView element. This appears in the center of the app.

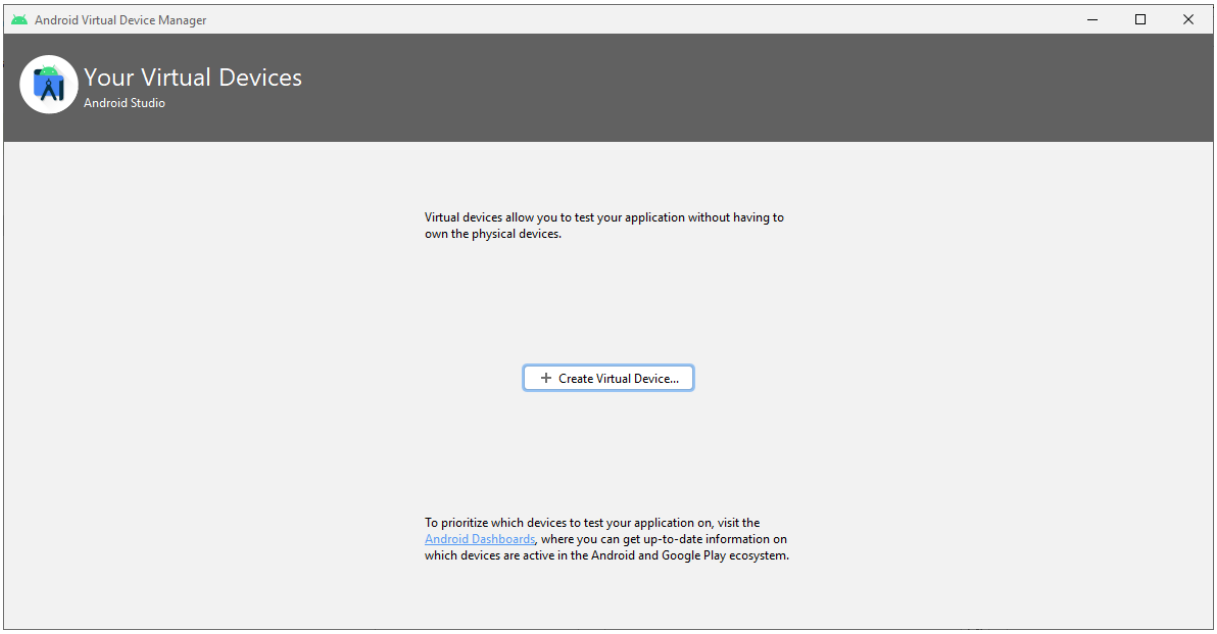


First run.

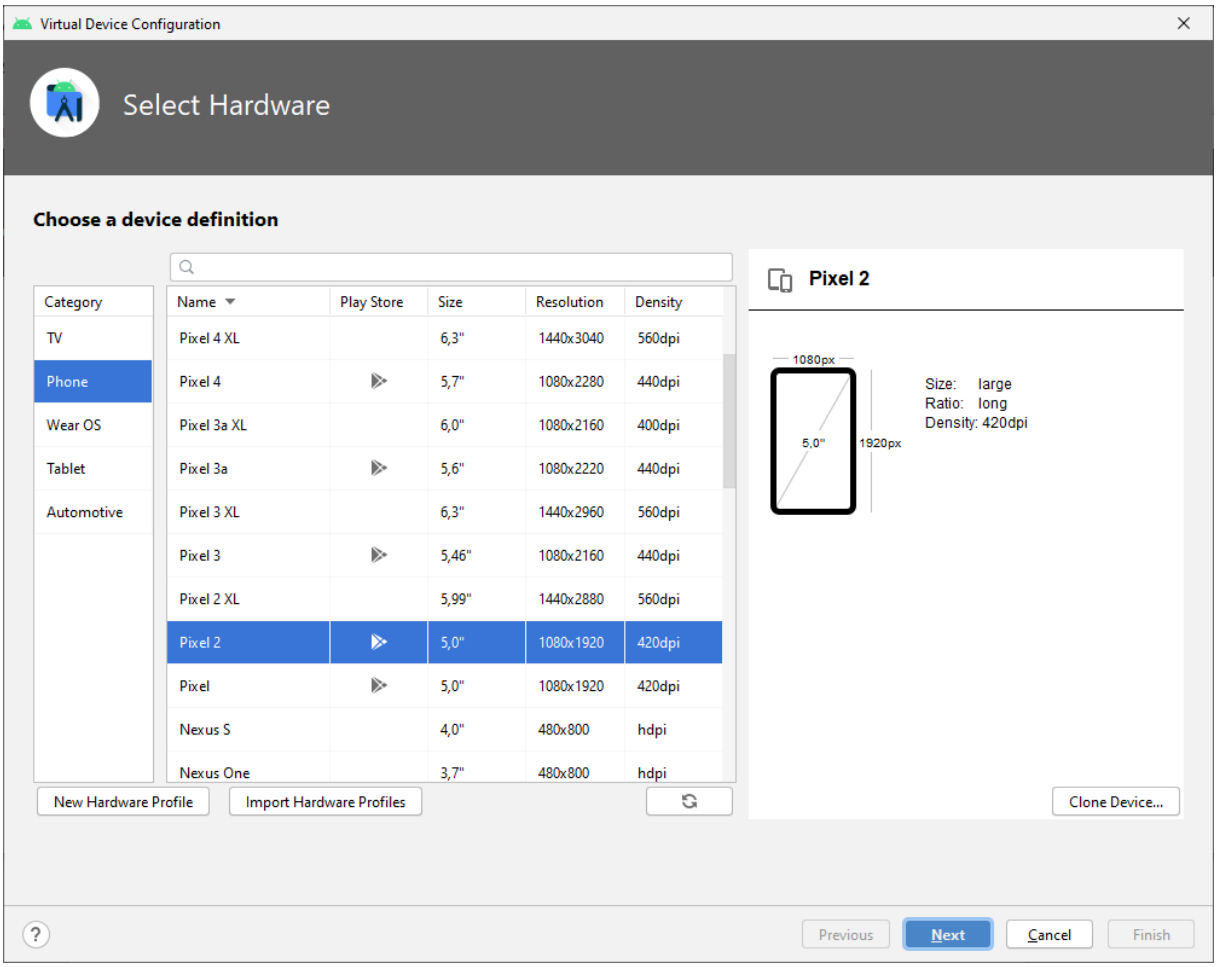
Go to Tools->AVD manager, to add a virtual device for emulating a phone to run your android app.



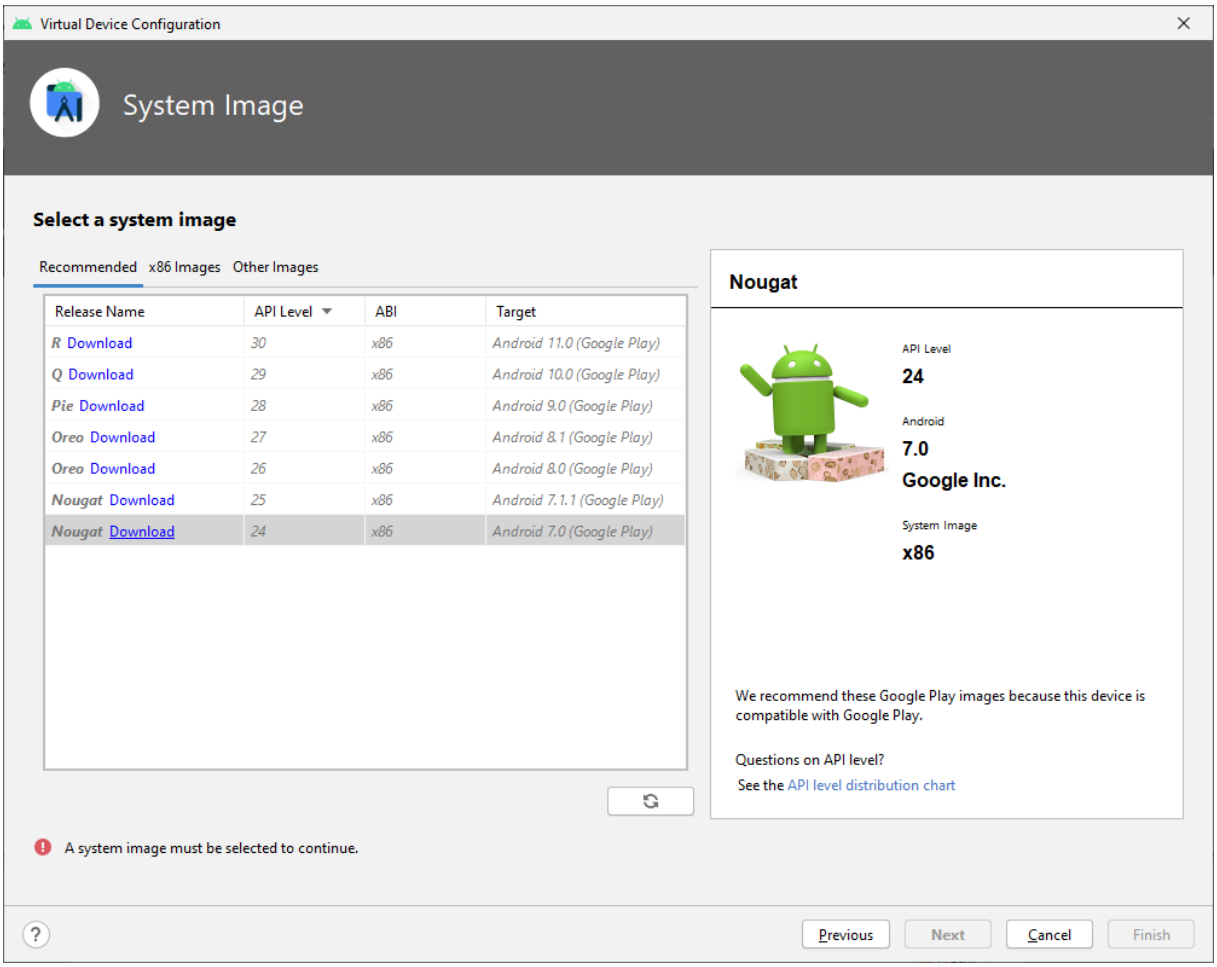
Click + Create Virtual Device…



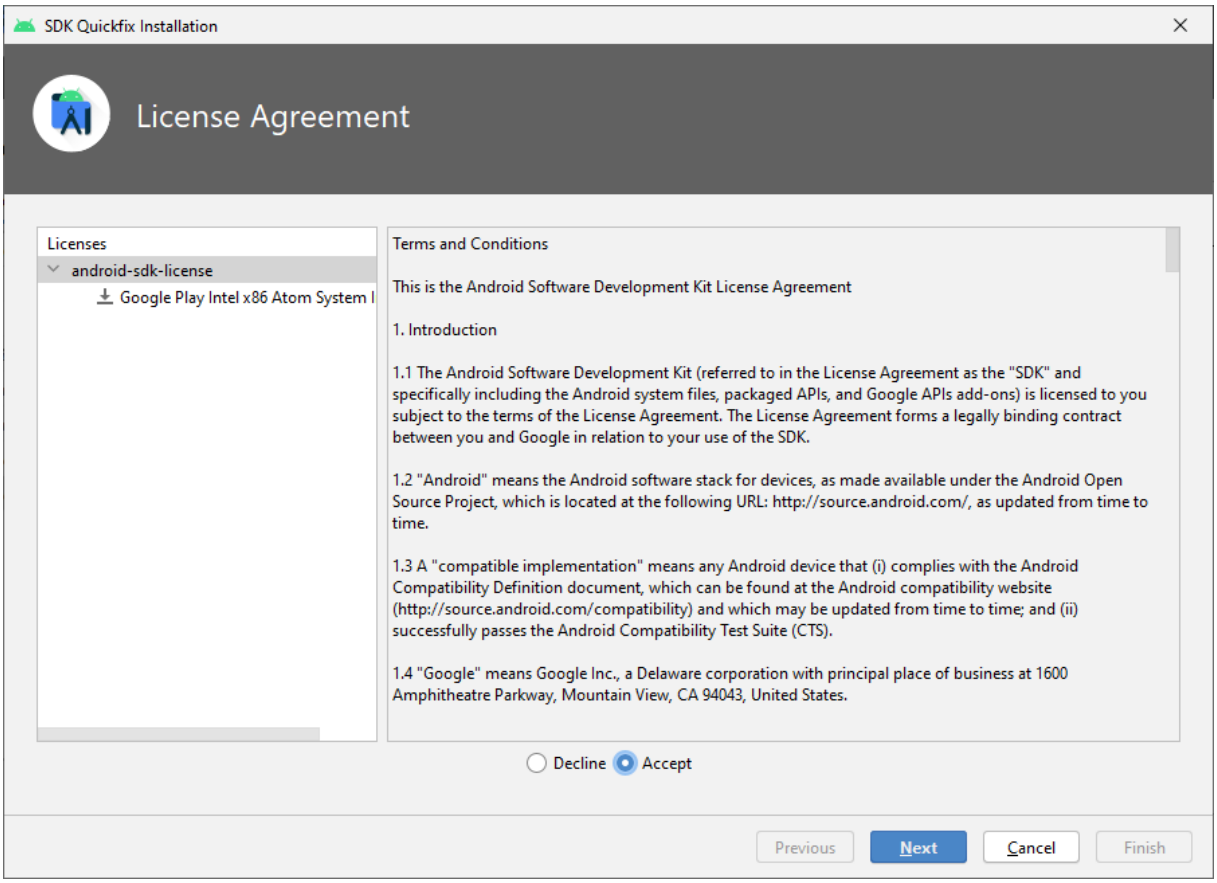
Select the “Pixel 2” from the phone templates, and click Next.



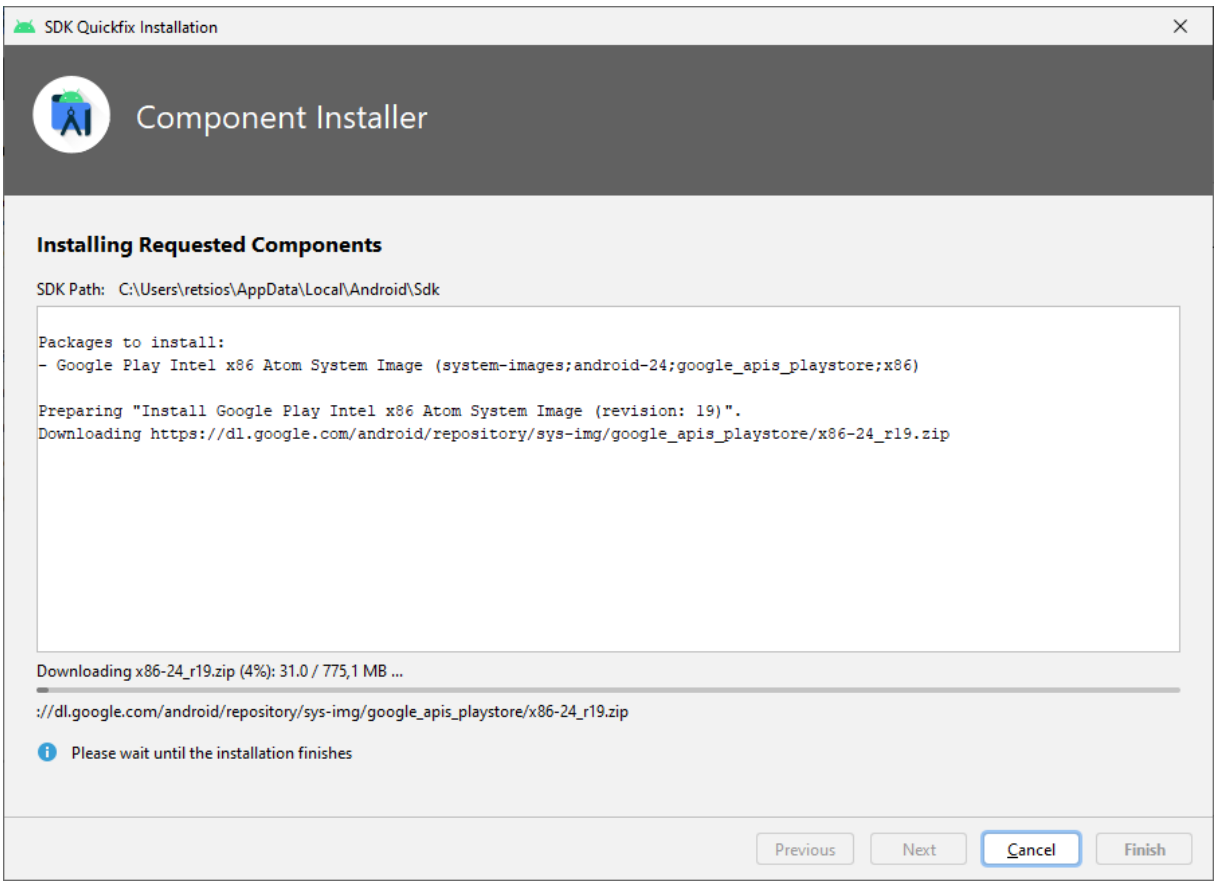
Choose an Android version that the “Pixel 2” should have, e.g. Android 7.0 Nougat (API level 24). Click “Download” to download it.

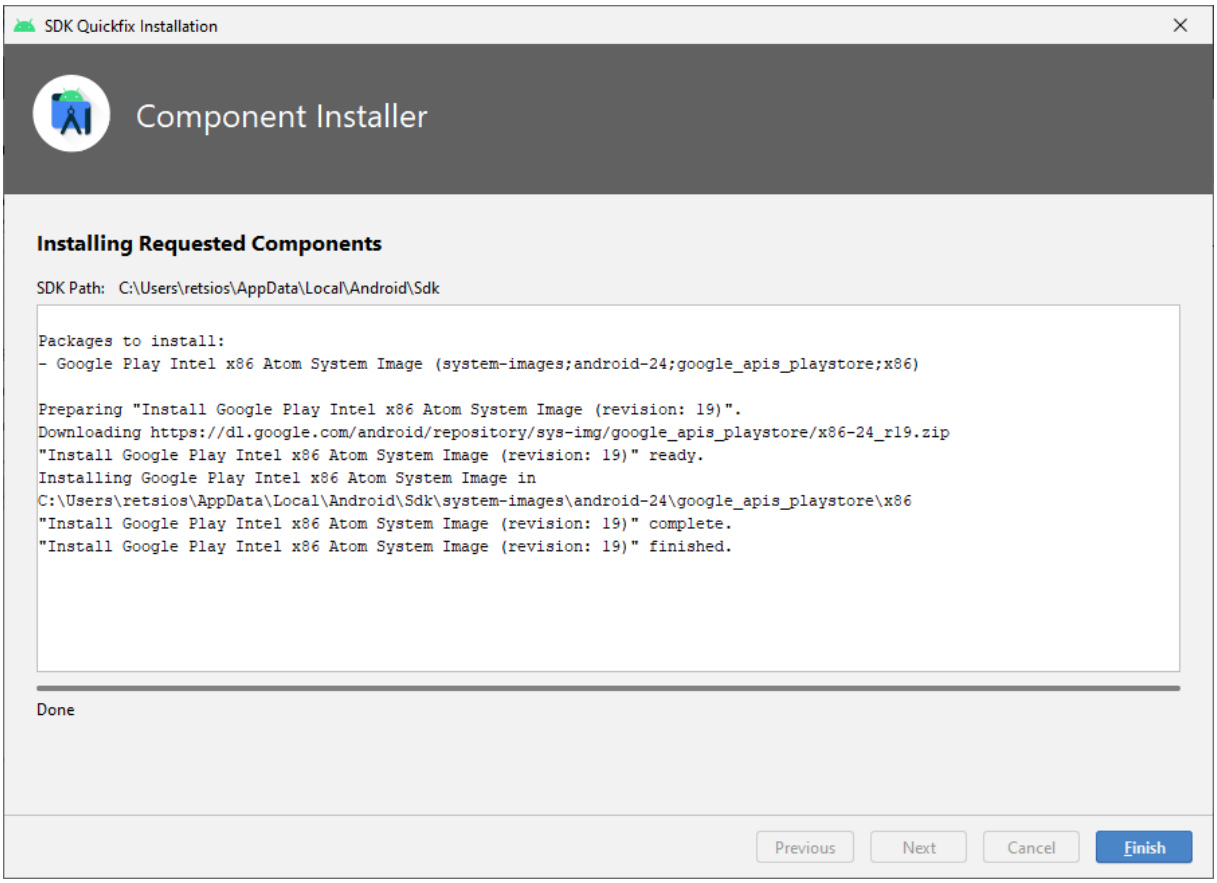


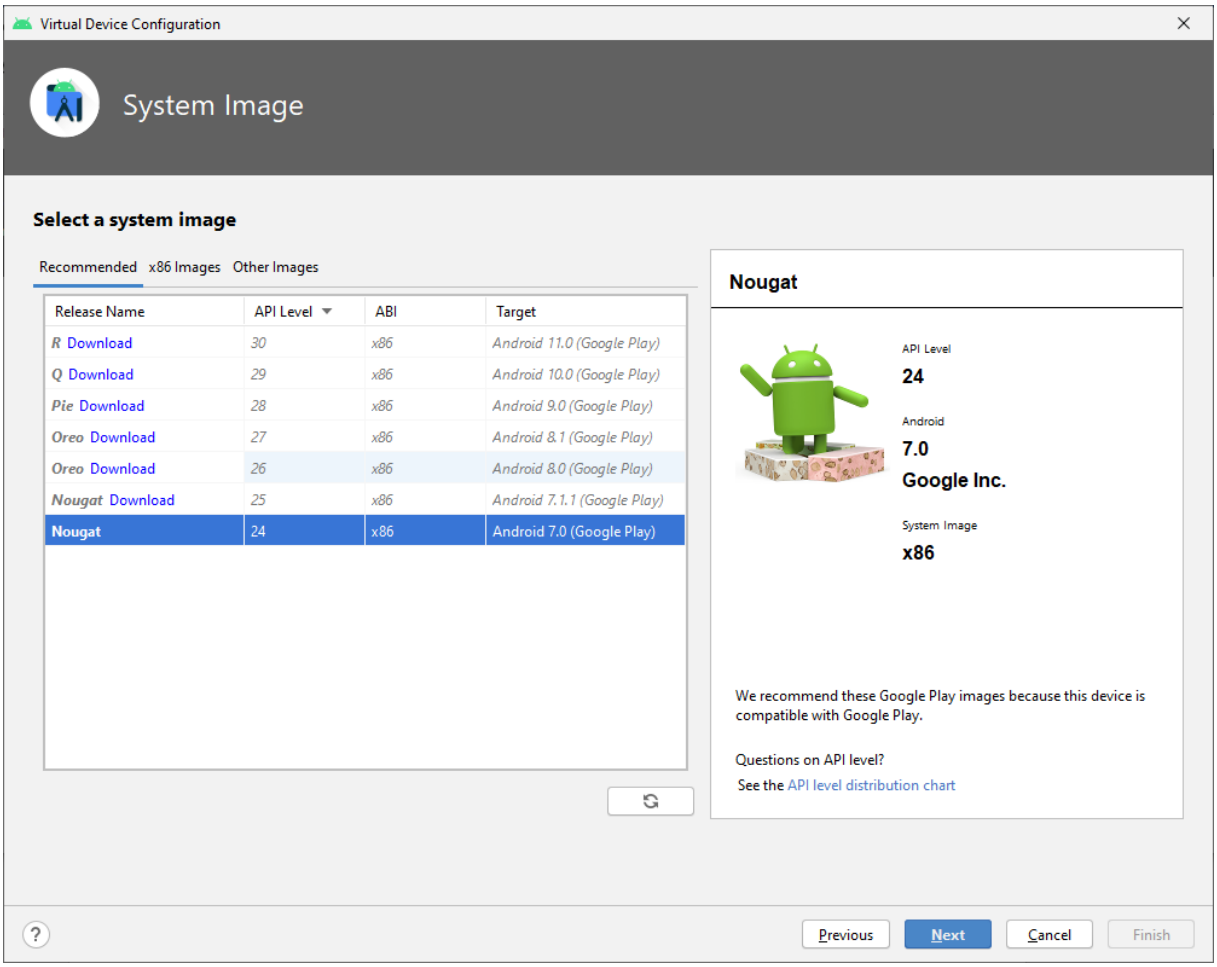
This will popup the Android SDK license agreement. Read, accept and continue.



The selected Android SDK will be downloaded and installed. This may take a few minutes.

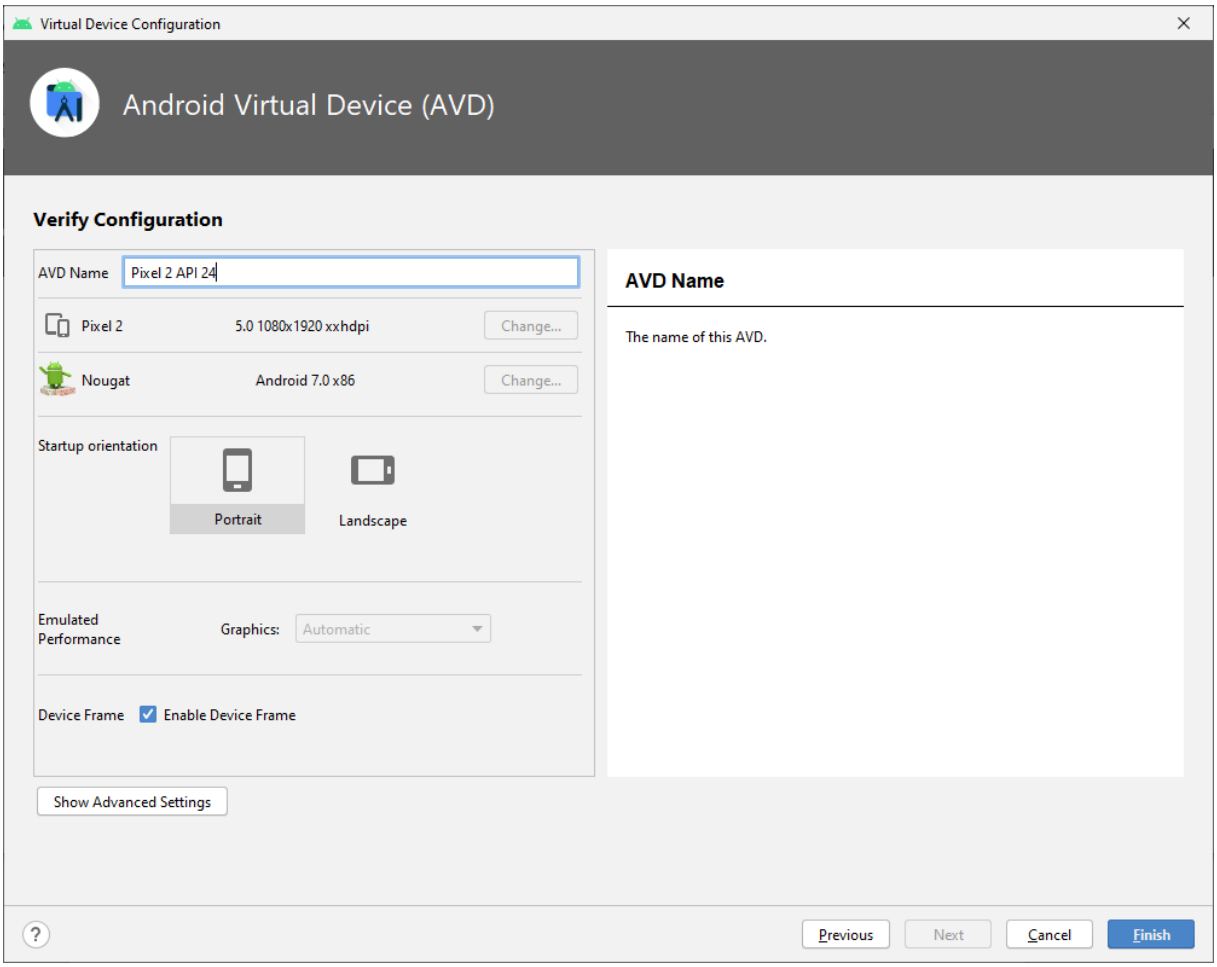



SDK Quickfix Installation
Component Installer
Installing Requested Components
SDK Path: C:\Users\retsios\AppData\Local\Android\Sdk
Packages to install:
- Google Play Intel x86 Atom System Image (system-images;android-24;google_apis_playstore;x86)
Preparing "Install Google Play Intel x86 Atom System Image (revision: 19)".
Downloading https://dl.google.com/android/repository/sys-img/google_apis_playstore/x86-24_r19.zip
"Install Google Play Intel x86 Atom System Image (revision: 19)" ready.
Installing Google Play Intel x86 Atom System Image in
C:\Users\retsios\AppData\Local\Android\Sdk\system-images\android-24\google_apis_playstore\x86
"Install Google Play Intel x86 Atom System Image (revision: 19)" complete.
"Install Google Play Intel x86 Atom System Image (revision: 19)" finished.
Done
Previous
Next
Cancel
Finish



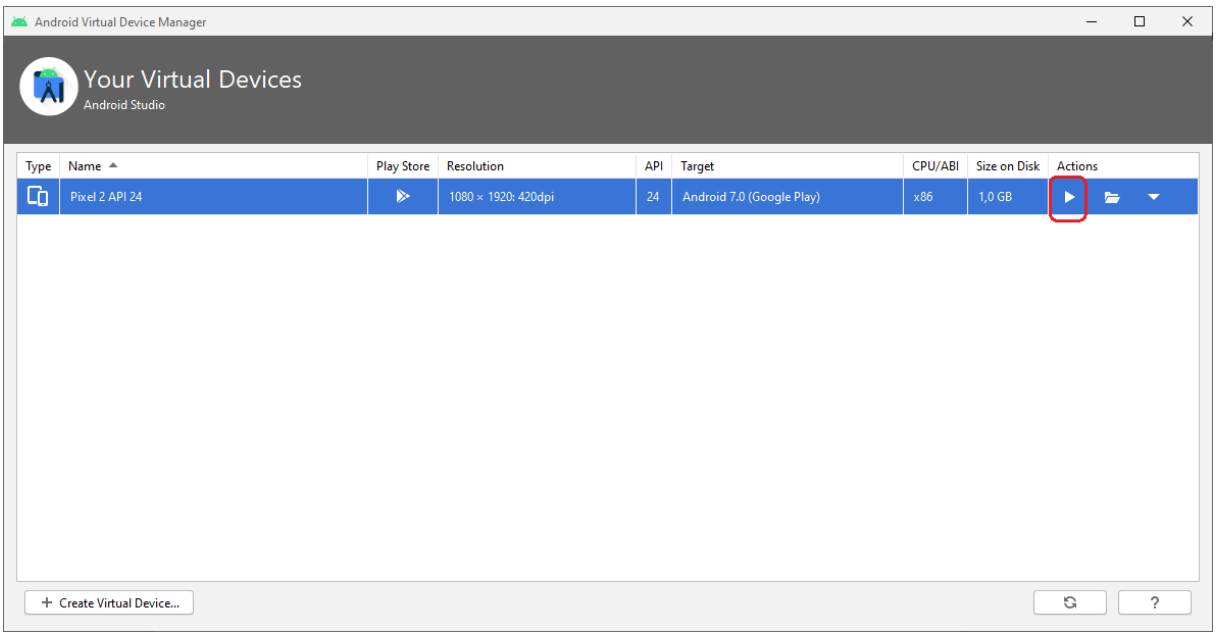
Now the option Nougat 7.0 is selecable. Select it and click Next.

Give the phone a logical name, and click Finish.

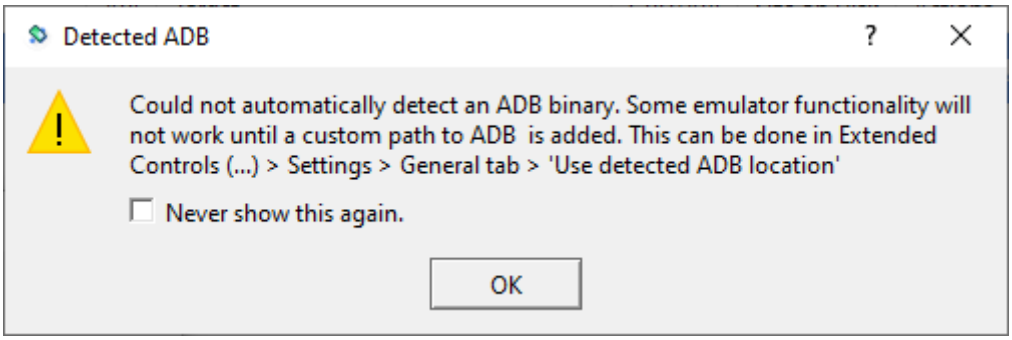


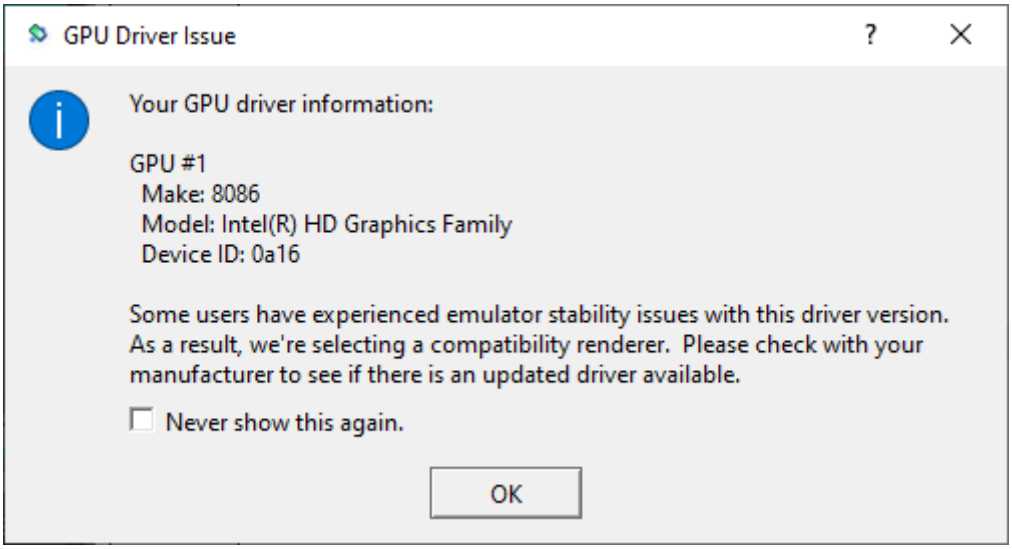
Now the “Pixel 2” phone appears in the list of available virtual devices.

Click the “Play” button under “Actions” to start it.

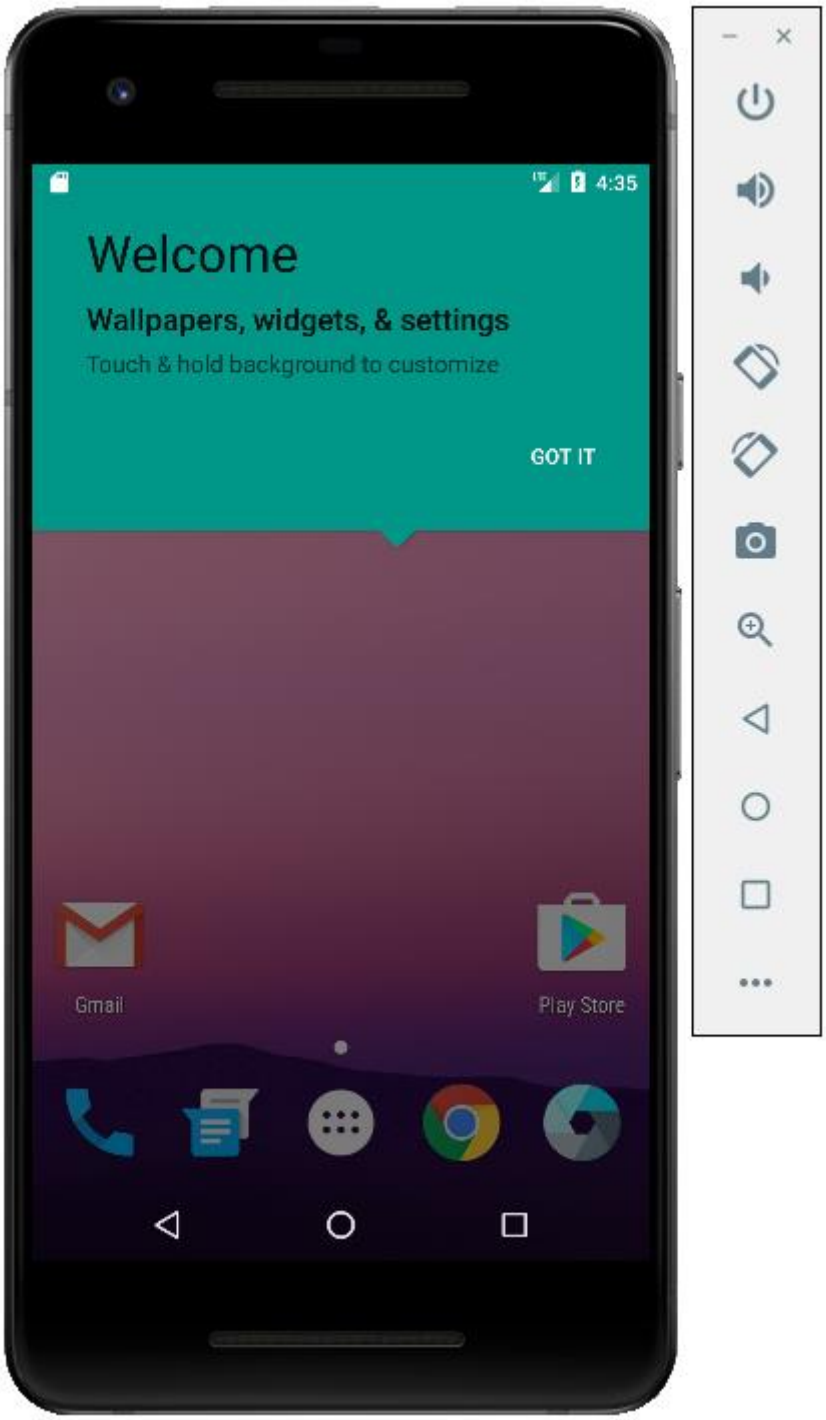


You may receive messages like the following, which you can safely ignore.





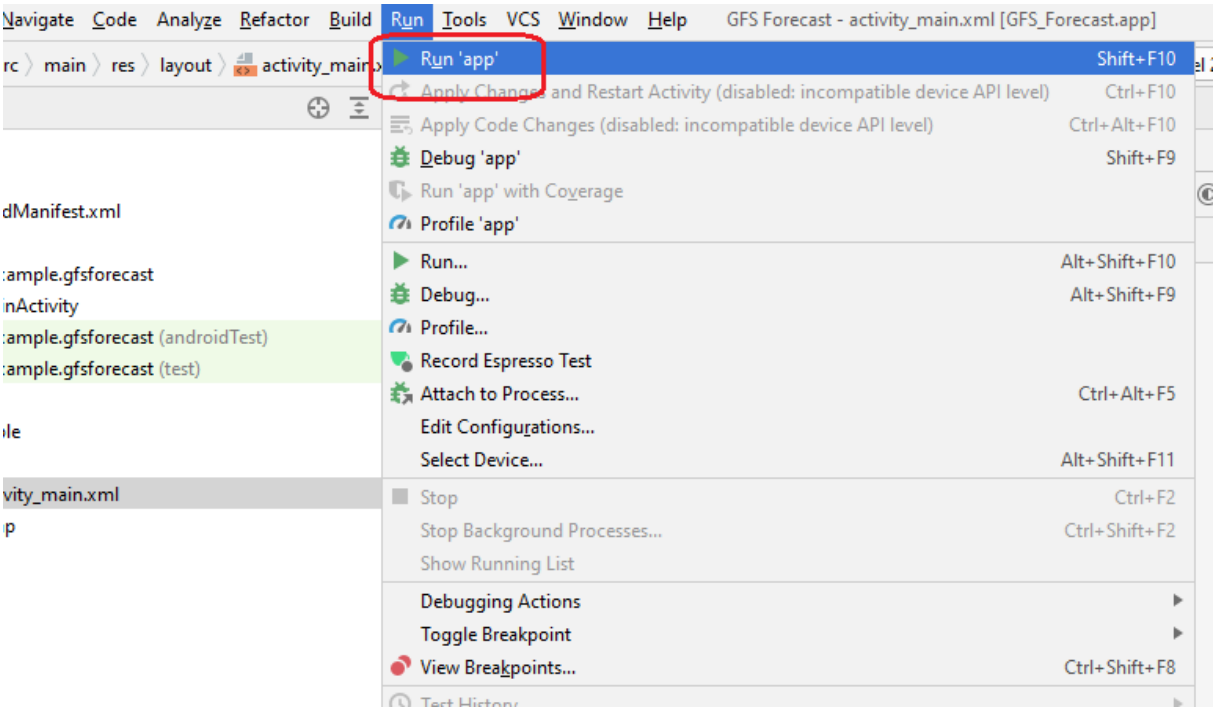
A successful start of the emulator looks like this:



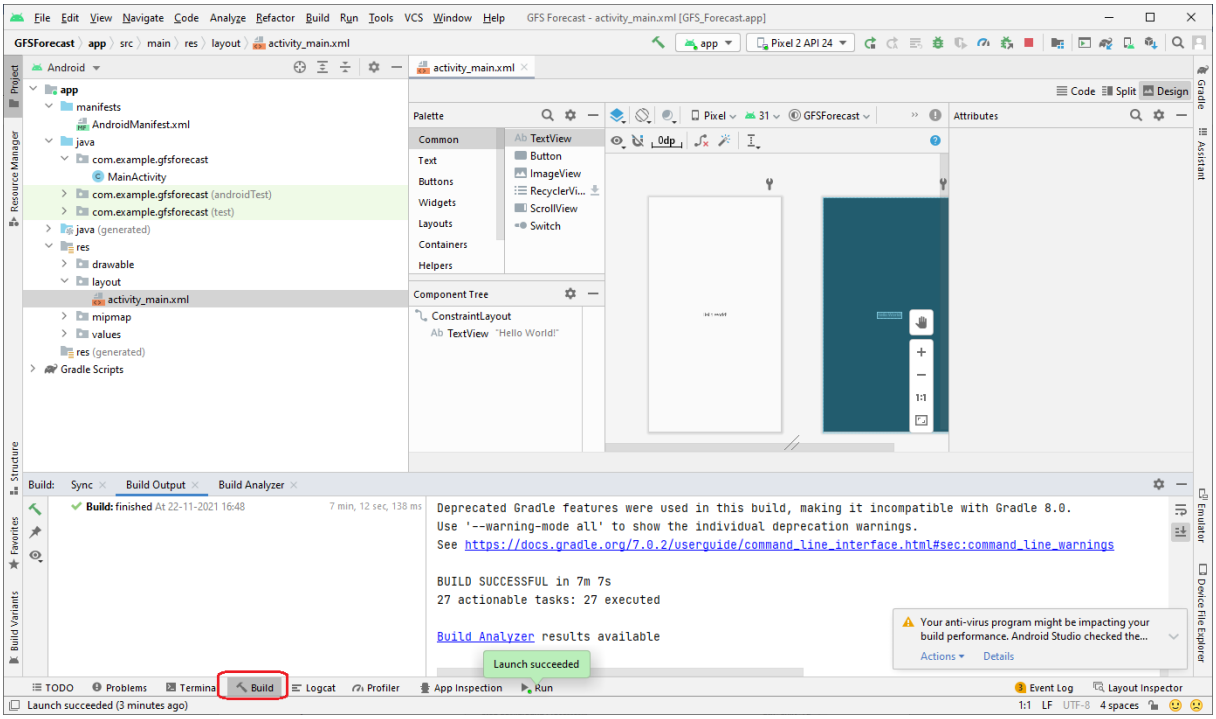
You can click away the android introductory text (click “GOT IT”).

Keep the emulator open. Close the Android Virtual Device Manager, and switch back to Android Studio. Only open the Android Virtual Device Manager if the emulator got stuck and you need to restart it.

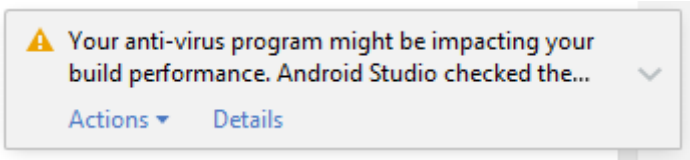
In Android Studio, select Run -> Run ‘app’ from the menu. This will build the Hello World app and run it as-is on the emulator that is running. The first time this may take a while.



You may turn on the “Build” view if you are wondering whether there is any progress.

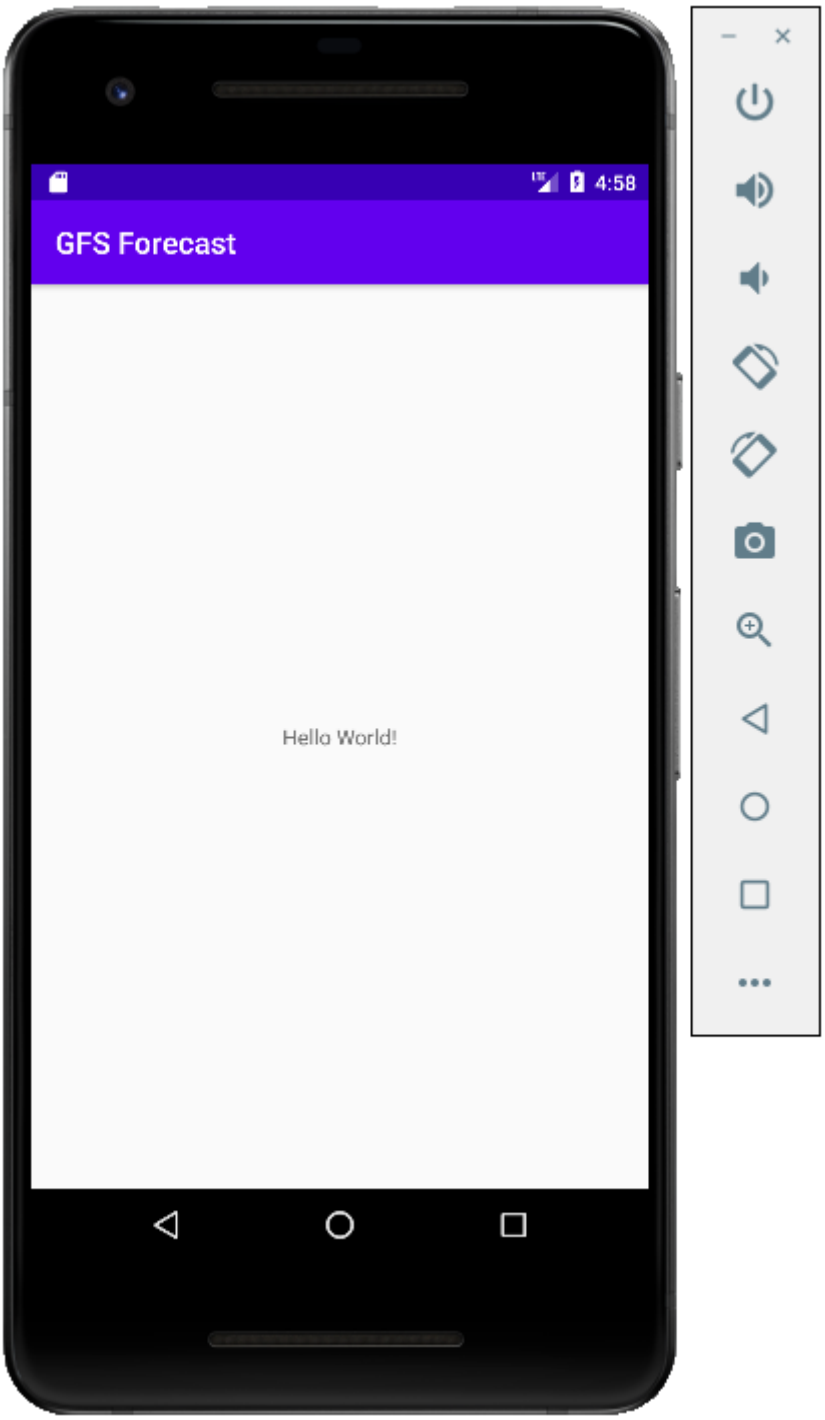


You may receive messages telling that the anti-virus program is slowing down Android Studio. If this does not bother you, you can ignore such messages.



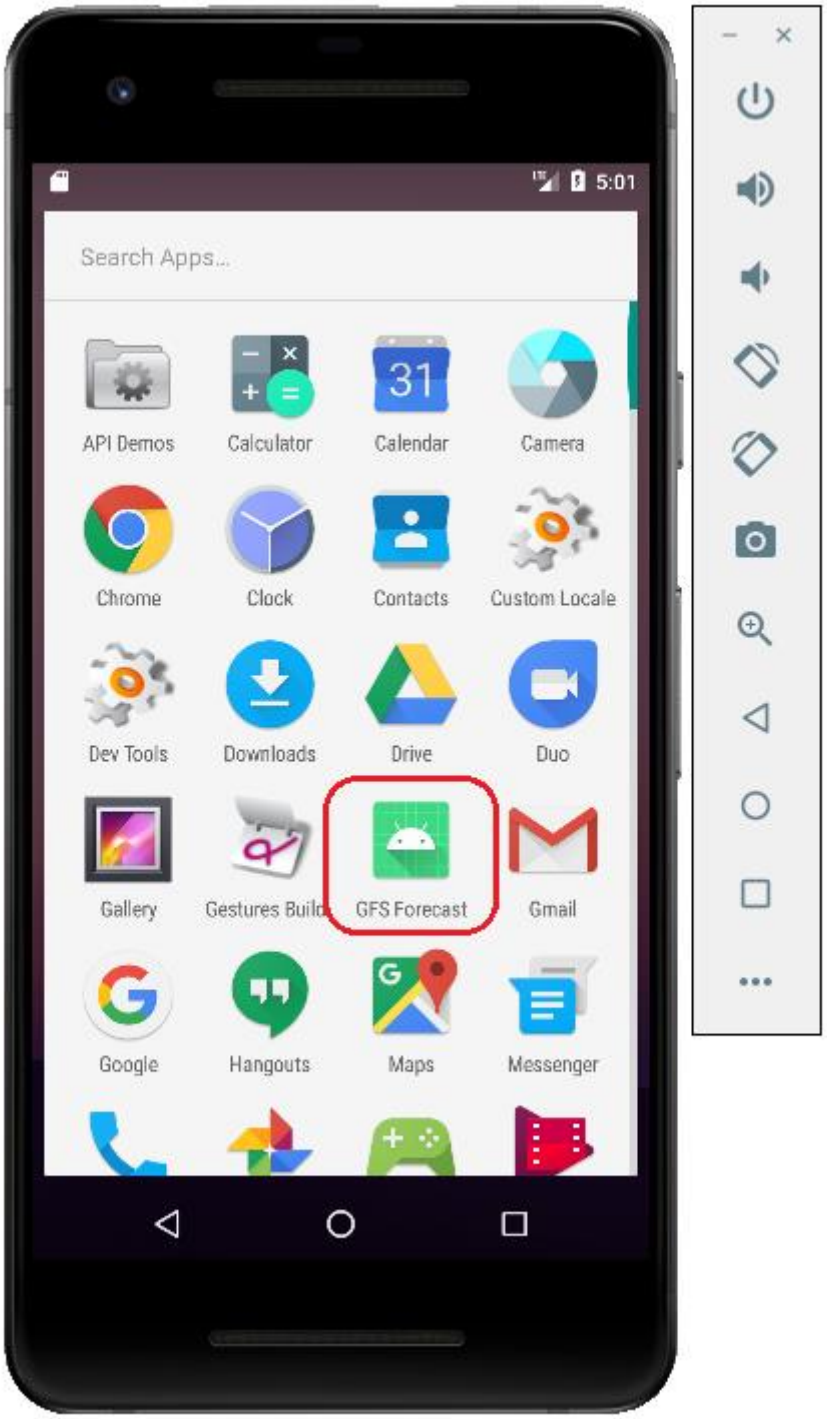
However If you want to attempt speeding up Android Studio with respect to the virus scanner, click Details. Specific instructions for Windows Defender (which is the default virus scanner that comes with Windows 10) are available here: https://docs.microsoft.com/en-us/windows/android/defender-settings

A successful first run of the Hello World app will look like this:



This app only displays the text “Hello World!”. It does nothing else. You can click the circle to hide the app.

If you inspect the installed apps list in Android (button ), you will notice that the app received a place in the list.



Extending the Hello World app to display Open Street Map

We will add the osmdroid library to the app.

Osmdroid is a free replacement of Google's Maps API, based on openstreetmap. It is preferred, as Google's Maps API requires registration keys and eventually a Credit Card for payment.

Add directives to the app builder script (gradle) to download and include the osmdroid binaries:

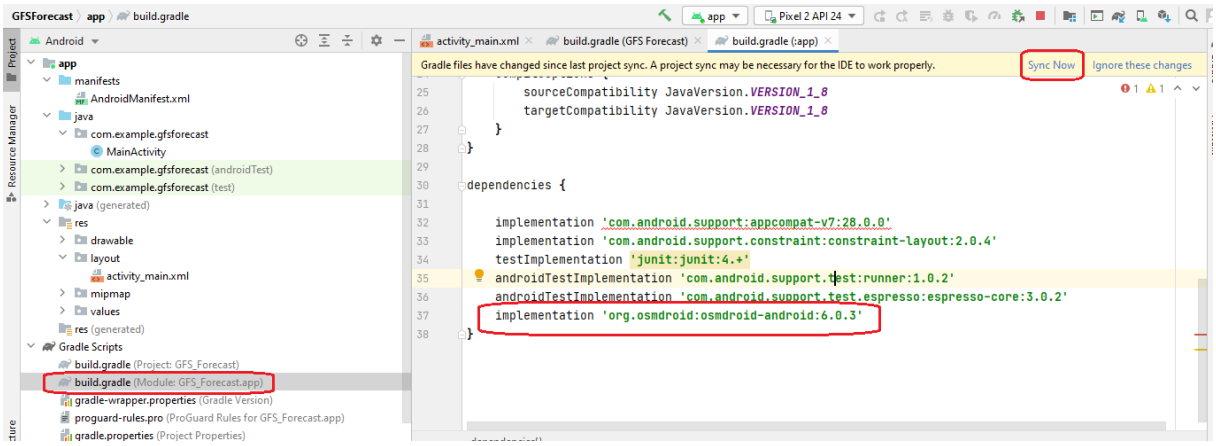
Open file Gradle Scripts / build.gradle (Module: GFS_Forecast.app) (which is the 2nd file named build.gradle).

At the end of the file, at dependencies, add:

```
implementation 'org.osmdroid:osmdroid-android:6.0.3'
```

If you copy/paste from this document, ensure that the quotes are simple single quotes (Microsoft Word may have changed them to decorated single quotes, which will not work).

Click Sync Now when done.



Add required app premissions to file AndroidManifest.xml

Edit file manifests / AndroidManifest.xml

At about line 10, that reads android:supportsRtl="true", add the following line:

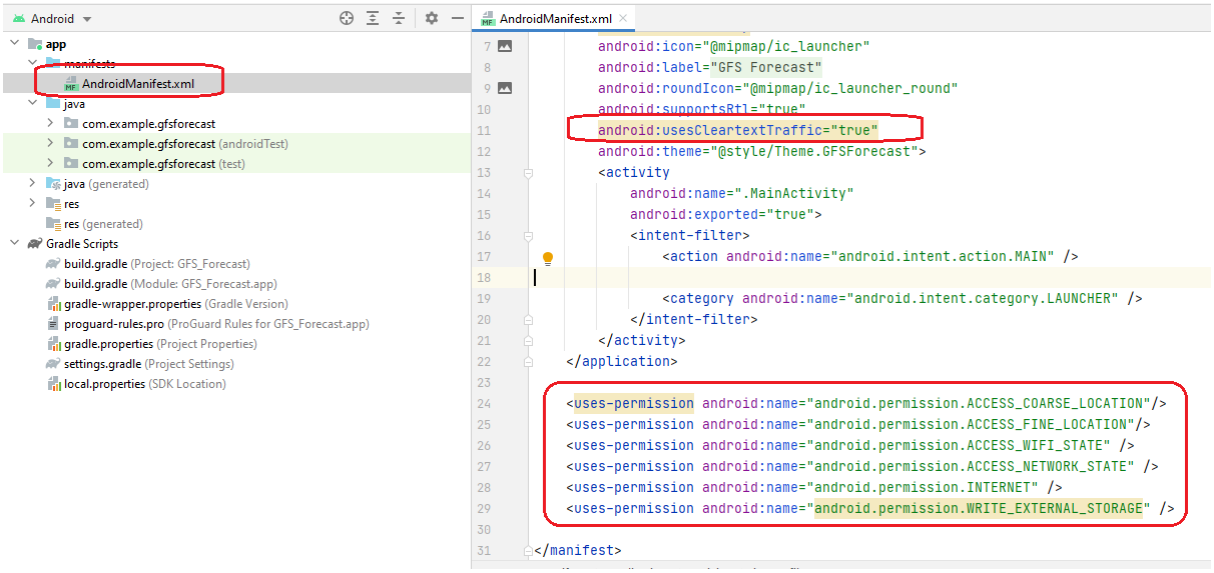
```
android:usesCleartextTraffic="true"
```

This allows the app to open URLs with http (instead of only https).

Also at the end of the file, between </application> and </manifest>, add the following 5 lines:

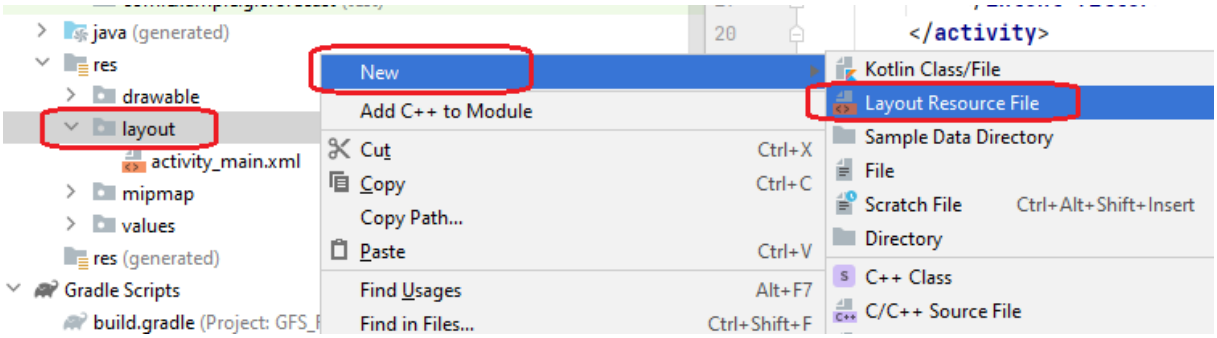
```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
```

This instructs android (when the app is installed) that this app requires permissions to sense the location of the phone (for the map to be able to center at the user's location), to download data from the internet (fetch the map tiles and the forecast data), and to write to external storage (a place to store the cached map tiles).

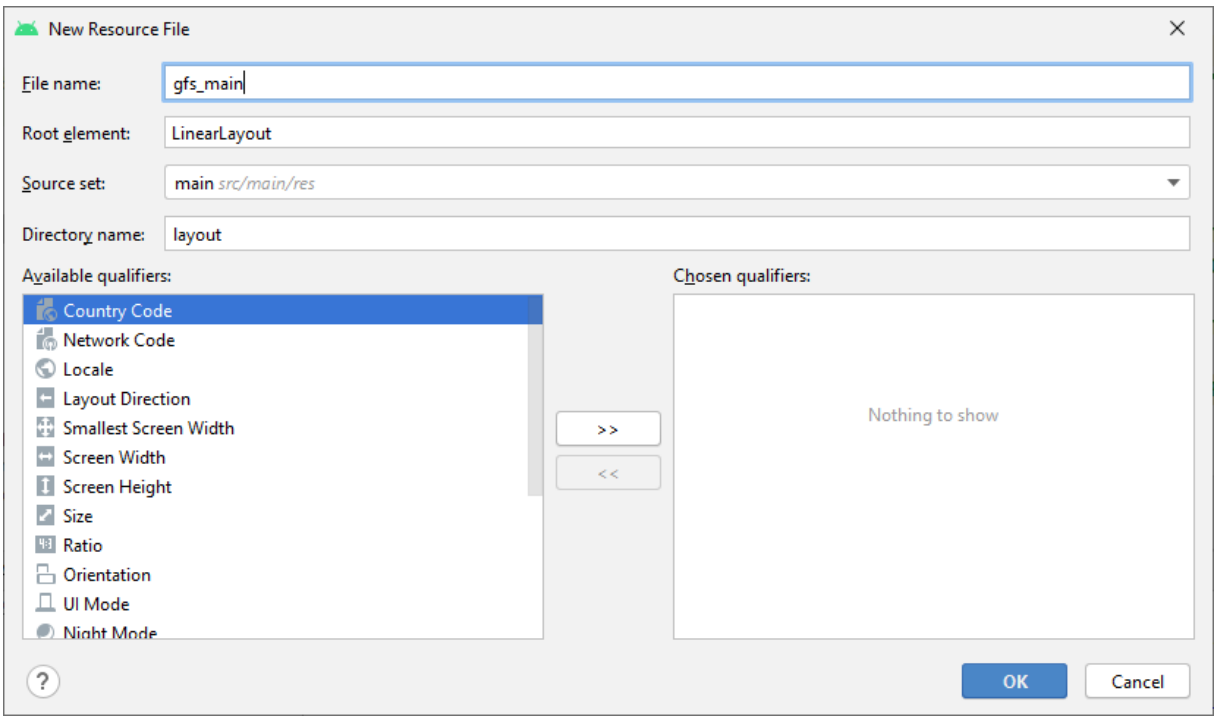


Add a layout containing a map.

Under res / layout, create a new Layout Resource file.

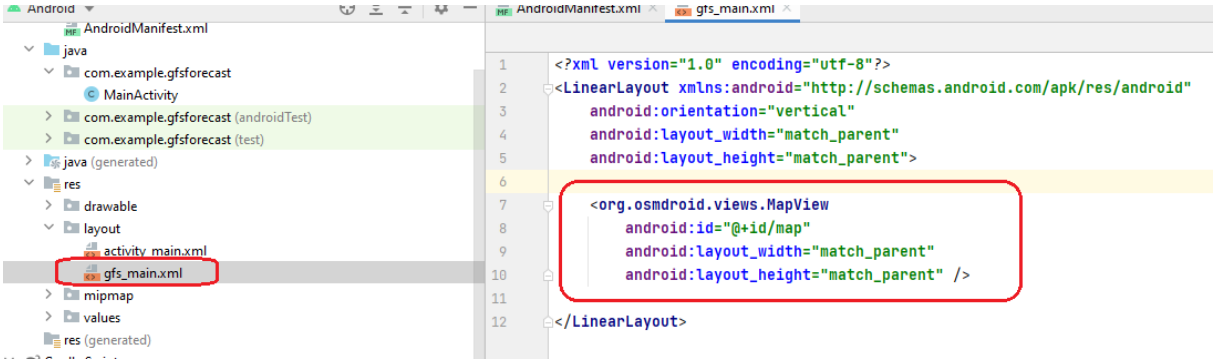


Name the file gfs_main, change the root element to LinearLayout and click OK.



Switch to Code view, and add the following element before the last line containing </LinearLayout>:

```
<org.osmdroid.views.MapView
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```



This adds a osmdroid MapView element inside a LinearLayout. In practice, this spans a map fullscreen in the app.

Now open file java / com.example.gfsforecast / MainActivity.java .

Replace the item R.layout.activity_main that is the parameter of the function-call setContentView() with the newly created R.layout.gfs_main . This will load the newly created gfs_main.xml layout as the main app content, instead of the previous hello world layout.

Also add the following code lines:

Under public class MainActivity …

```
private MapView map = null;
```

Note that the MapView is marked red. It is a still “unknown” keyword.

Moving the mouse over the word MapView will reveal a solution, which is to import the class so that it can be used.

```
public class MainActivity extends AppCompatActivity {

    private MapView map = null;

    @Override
    protected vo
        super.onCreate(savedInstanceState);
        setContentView(R.layout.gfs_main);
    }
}
```

Cannot resolve symbol 'MapView'

Import class Alt+Shift+Enter More actions... Alt+Enter

Click the option “Import class” that is offered. This will add the required import statement, and the red mark will disappear.

```
public class MainActivity extends AppCompatActivity {

    private MapView map = null;

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

Also add the following under super.onCreate(savedInstanceState);

```
//load/initialize the osmdroid configuration
Context ctx = getApplicationContext();
Configuration.getInstance().load(ctx,
PreferenceManager.getDefaultSharedPreferences(ctx));
```

Again, click the “import class” options that are offered (3x).

Under setContentView(…); add the following

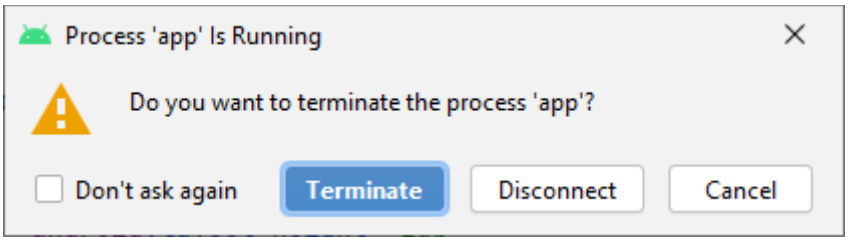
```
map = (MapView) findViewById(R.id.map);
map.setTileSource(TileSourceFactory.MAPNIK);
```

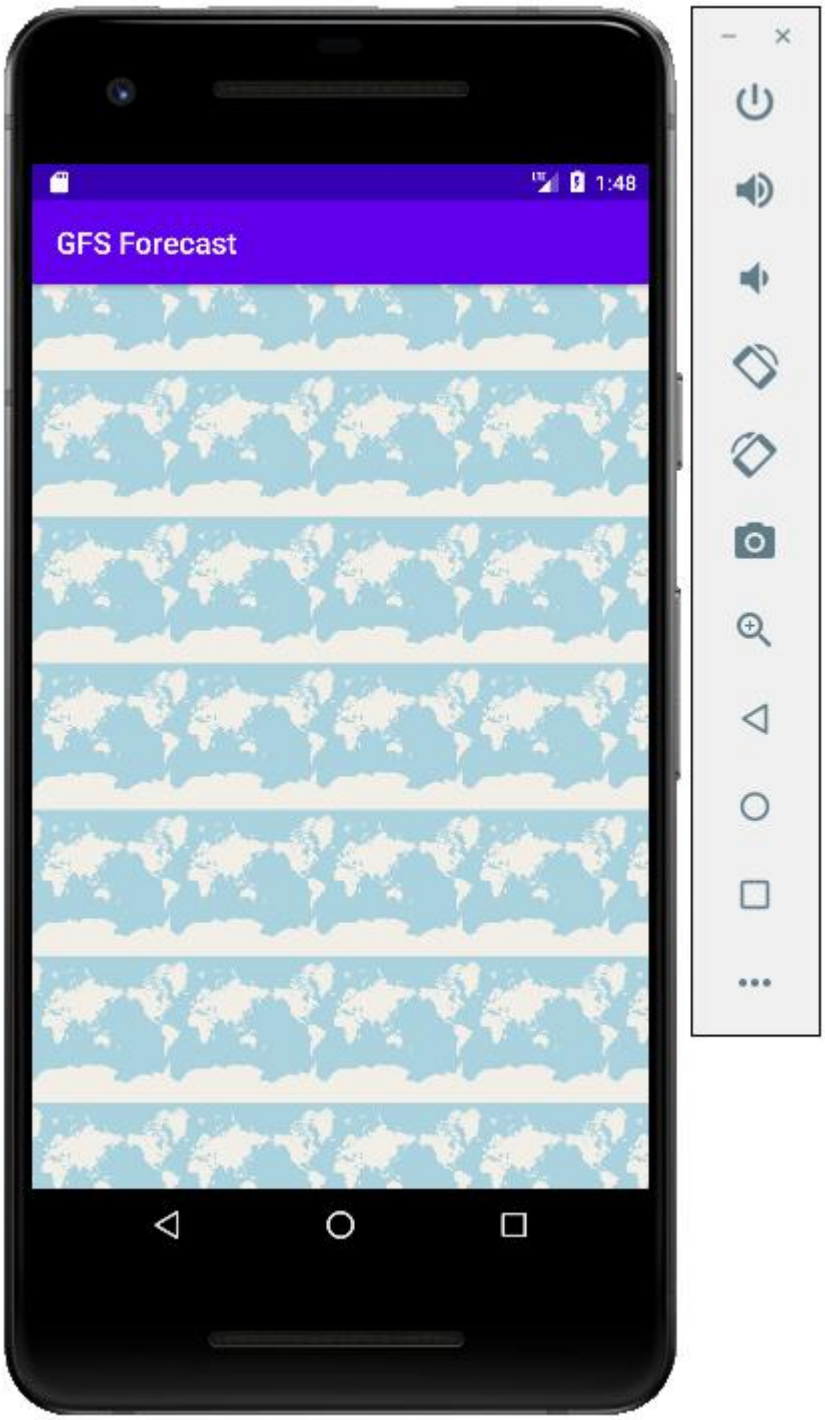
Click “import class” once again for the TileSourceFactory . This part of the code sets MAPNIK as the tile source for the map.

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    //load/initialize the osmdroid configuration, this can be done
    Context ctx = getApplicationContext();
    Configuration.getInstance().load(ctx, PreferenceManager.getDefaultSharedPreferences(ctx));

    setContentView(R.layout.gfs_main);

    map = (MapView) findViewById(R.id.map);
    map.setTileSource(TileSourceFactory.MAPNIK);
}
```

Save the file and test the app. Click Run -> Run ‘app’ to deploy the latest version into the Pixel 2 emulator.

You may get the following message that the previous version of the app is still running. Click Terminate to terminate it, and run the new version.





Click on the map to reveal the zoom + - buttons. Then use the zoom buttons to zoom the map.

Add space on the app's screen for a Graph

Our goal is to have an app whereby a map and a graph can be used together. The map allows selection of the location of interest, and the graph plots the data for that location of interest.

There are many ways to achieve a map/graph combo app: swipe left/right or top/bottom to change from map to graph and vice versa, double-tap the screen to switch between map and graph, etc.

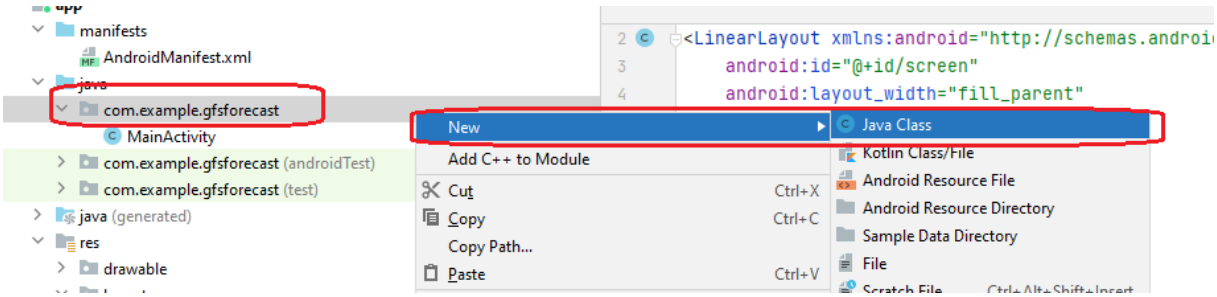
In our case we have chosen to split the screen in 2 parts: the graph at the upper part of the screen, and the map at the lower part of the screen. This allows simultaneous view of the map and the graph. Also we have chosen to be able to shrink the map, leaving more space for reading and analyzing the graph, and also the opposite: shrink the graph, leaving more space for the map, for a better location overview.

Proceed by adding two files to java / com.example.gfsforecast :

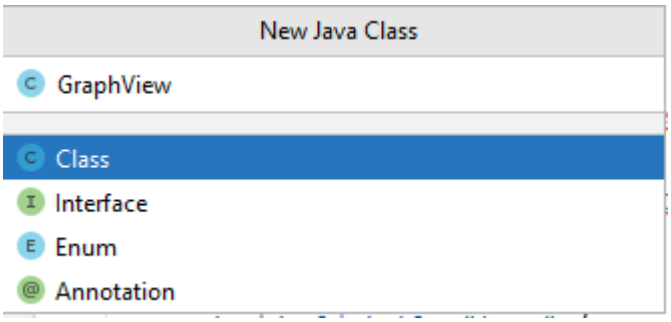
GraphView.java and DividerView.java

The GraphView is responsible for plotting the graphs and their legends, and the DividerView is a line that we can “catch” to resize the map or graph and re-divide the screen.

Proceed by right-clicking on com.example.gfsforecast, and selecting New -> Java Class.



Type the name GraphView.



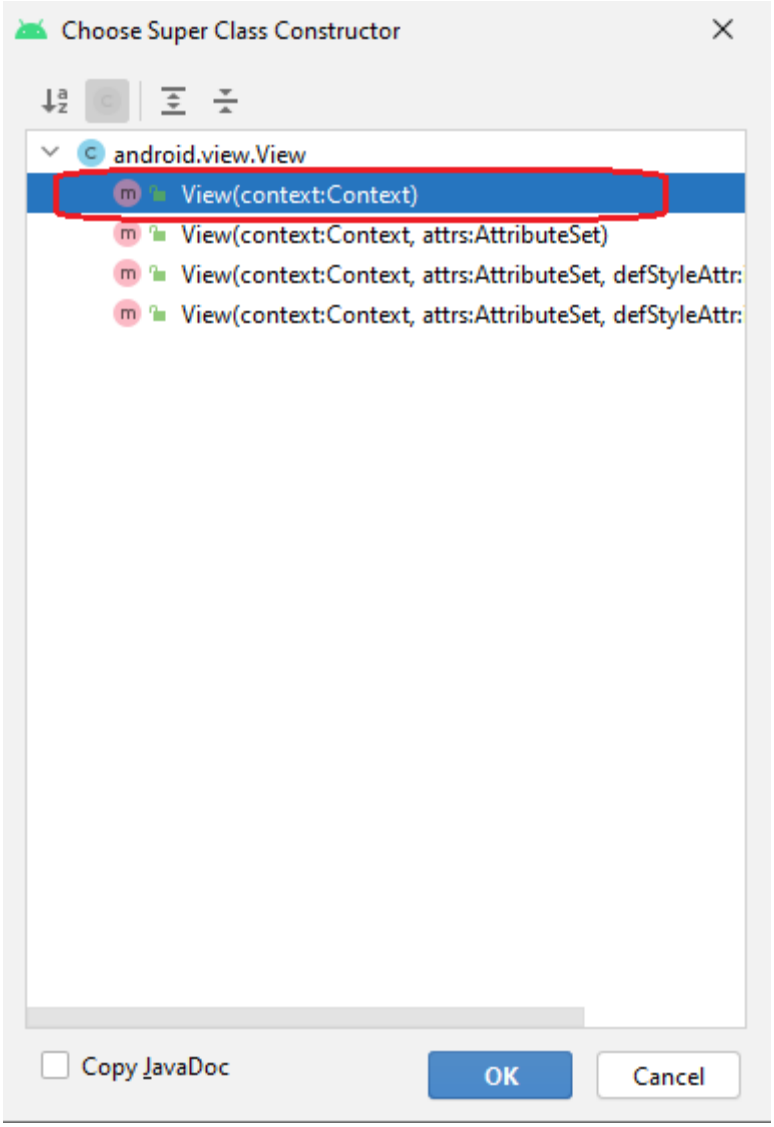
This will create the GraphView.java file.

Add the text “extends View” after public class GraphView.

```
public class GraphView extends View
```

Solve the missing import issue, by adding the import.

Then solve the missing constructor issue, by adding the first one from the list that appears.

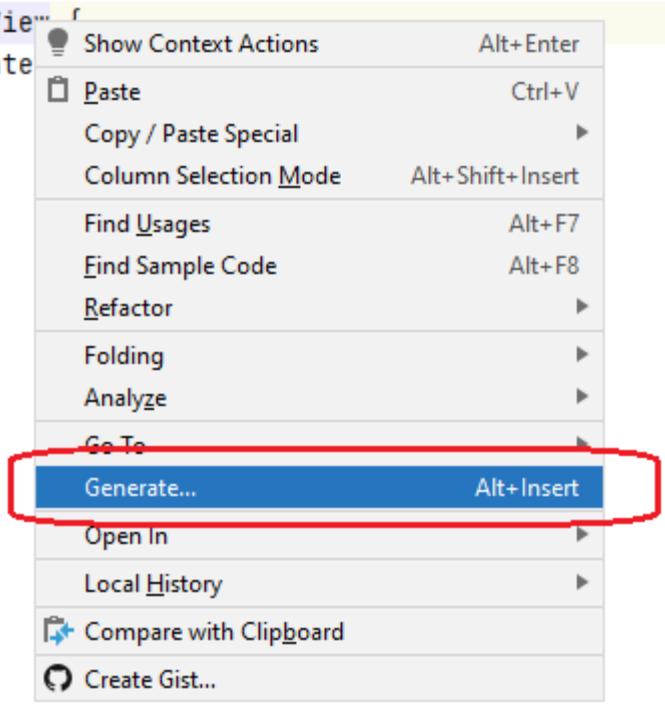


The resulting code looks like this:

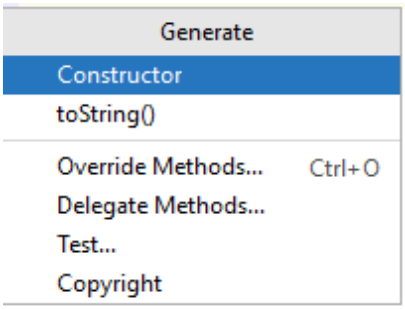
```
package com.example.gfsforecast;

import android.content.Context;
import android.view.View;

public class GraphView extends View {
    public GraphView(Context context) {
        super(context);
    }
}
```

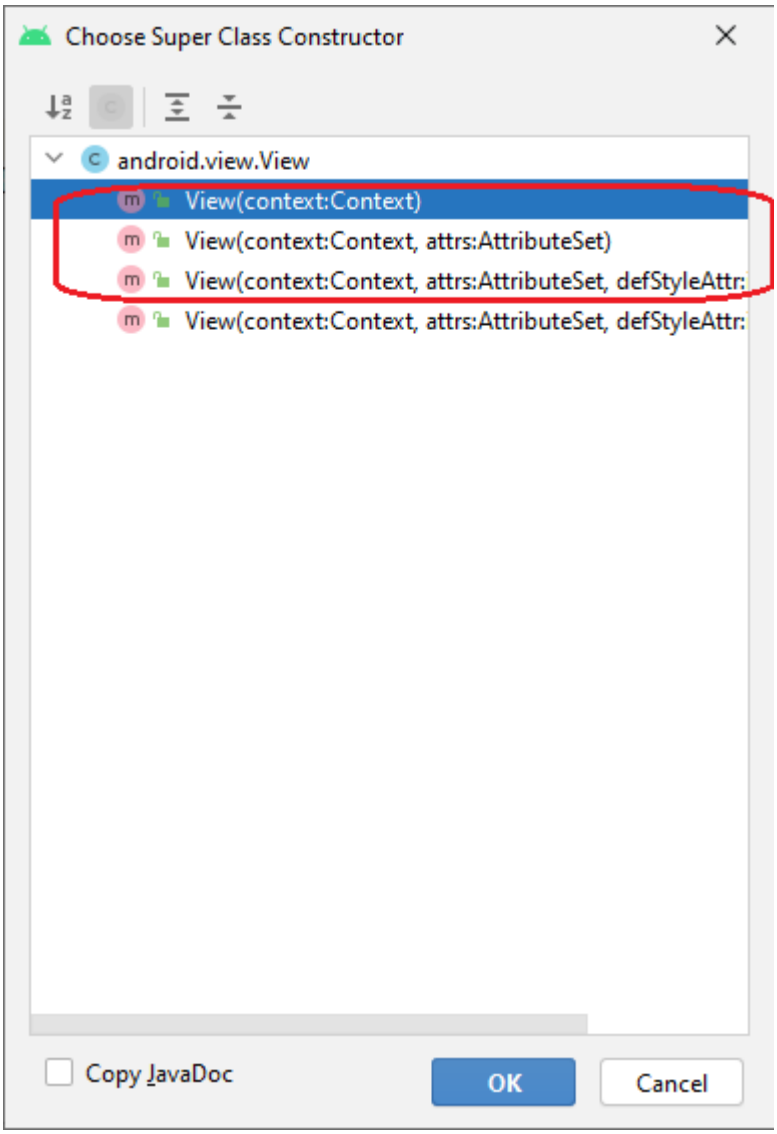
Right-click on View, and select Generate… Constructor to pop-up the same box and add the second and third constructor as well.

```java
package com.example.gfsforecast;

import android.content.Context;
import android.view.View;

public class GraphView extends View {
    public GraphView(Context conte
        super(context);
    }
}
```



`





After adding the first 3 that we require, it will look like this:

```java
package com.example.gfsforecast;

import android.content.Context;
import android.support.annotation.Nullable;
import android.util.AttributeSet;
import android.view.View;

public class GraphView extends View {
    public GraphView(Context context) {
        super(context);
    }

    public GraphView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
    }

    public GraphView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }
}
```

Do the same for class DividerView:

New -> Java Class -> DividerView extends View, solve all issues, and in the same way add the first 3 constructors from the list.

The result (DividerView.java) looks like this:

```java
package com.example.gfsforecast;

import android.content.Context;
import android.support.annotation.Nullable;
import android.util.AttributeSet;
import android.view.View;

public class DividerView extends View {
    public DividerView(Context context) {
        super(context);
    }

    public DividerView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
    }

    public DividerView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }
}
```

Note that you do not have to use the menus to add the constructors. You can just type the code that you see in this document. The result is exactly the same.

As a last step, replace the entire content of file res / layout / gfs_main.xml with the following:

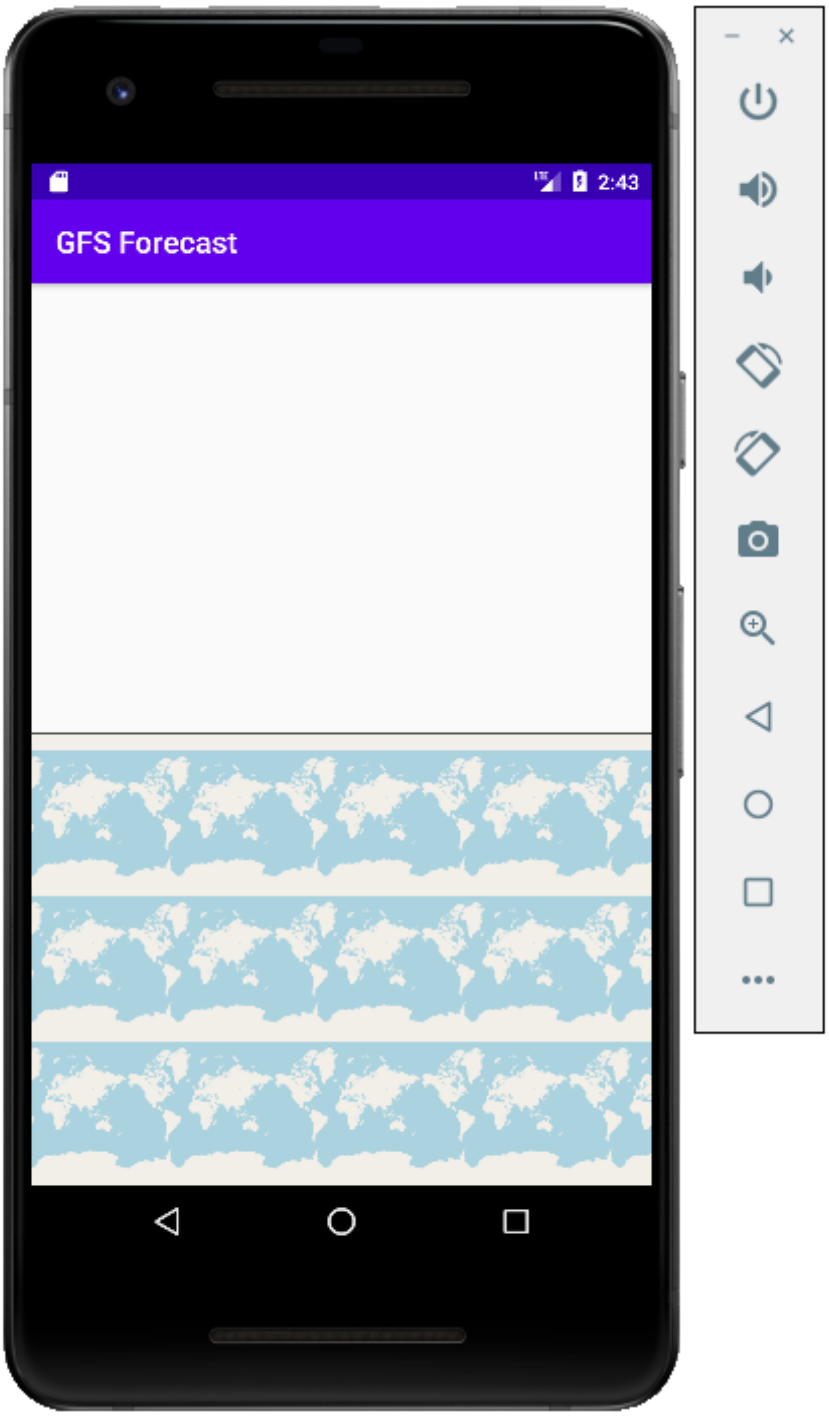
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/screen"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:clickable="true"
    android:orientation="vertical">

    <com.example.gfsforecast.GraphView
        android:id="@+id/graph"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="50" />

    <com.example.gfsforecast.DividerView
        android:id="@+id/divider"
        android:layout_width="wrap_content"
        android:layout_height="1dp"
        android:background="#222222"
        android:clickable="true" />

    <org.osmdroid.views.MapView
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="50" />
</LinearLayout>
```

This does the following:

1) The two new elements, GraphView and DividerView, are added to the app screen.
2) The GraphView and MapView elements have no height (0 height) but are instructed to fill the screen based on the weight (initially 50% each). Programmatically we can change the weights, to increase or decrease the size of each (map or graph).

Re-run the app to confirm the changes (click Terminate to terminate the previous running app if asked).

Now the app is 50-50 divided between map and graph. The map works, the graph only displays white, and the divider element can't be moved.



Make the app react to user's gestures to resize the map/graph space

The next step is to program the divider element, so that we can use it to resize the screen elements.

We add gesture detection, so that the app senses swipe movements on the screen. Horizontal swipes do nothing, but vertical swipes close to the divider will "catch" it and resize the map and the graph, giving the impression to the user that he moves the divider.

Open file java / com.example.gfsforecast / MainActivity.java for editing.

Under private MapView map = null; add the following, and then solve all missing imports:

```
private GestureDetector screenGestureDetector;
private boolean scrollInActionBar = false;
private boolean catchSplitter = false;
```

Solve the import for GestureDetector.

Under map.setTileSource(TileSourceFactory.MAPNIK); add the following lines.

```
LinearLayout screen = (LinearLayout) findViewById(R.id.screen);
GraphView graph = (GraphView) findViewById(R.id.graph);
DividerView divider = (DividerView) findViewById(R.id.divider);

final GestureDetector dividerGestureDetector = new GestureDetector(ctx, new
GestureDetector.SimpleOnGestureListener() {

    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float
distanceX, float distanceY) {
        int[] location = new int[2];
        screen.getLocationOnScreen(location);
        int x = location[0];
        int y = location[1];
        float perc = Math.max(0, Math.min(100, 100 * (e2.getRawY() - y) /
screen.getMeasuredHeight()));
        LinearLayout.LayoutParams lpm = (LinearLayout.LayoutParams)
(map.getLayoutParams());
        lpm.weight = 100 - perc;
        LinearLayout.LayoutParams lpg = (LinearLayout.LayoutParams)
(graph.getLayoutParams());
        lpg.weight = perc;
        map.setLayoutParams(lpm);
        graph.setLayoutParams(lpg);
        return super.onScroll(e1, e2, distanceX, distanceY);
    }
});

screenGestureDetector = new GestureDetector(ctx, new
GestureDetector.SimpleOnGestureListener() {

    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float
distanceX, float distanceY) {
        if (!catchSplitter) {
            int[] location = new int[2];
            divider.getLocationOnScreen(location);
            int yd = location[1];
            LinearLayout.LayoutParams lpg = (LinearLayout.LayoutParams)
(graph.getLayoutParams());
            catchSplitter = (e2.getRawY() >= yd) || ((lpg.weight > 99) &&
(e2.getRawY() >= yd - 50));
        } else {
            int[] location = new int[2];
            screen.getLocationOnScreen(location);
            int x = location[0];
            int y = location[1];
            float perc = Math.max(0, Math.min(100, 100 * (e2.getRawY() - y)
/ screen.getMeasuredHeight()));
            LinearLayout.LayoutParams lpm = (LinearLayout.LayoutParams)
```

```
(map.getLayoutParams());
            lpm.weight = 100 - perc;
            LinearLayout.LayoutParams lpg = (LinearLayout.LayoutParams)
(graph.getLayoutParams());
            lpg.weight = perc;
            map.setLayoutParams(lpm);
            graph.setLayoutParams(lpg);
        }
        return super.onScroll(e1, e2, distanceX, distanceY);
    }
});

divider.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if(motionEvent.getAction() == MotionEvent.ACTION_UP) {
        }
        return dividerGestureDetector.onTouchEvent(motionEvent);
    }
});

screen.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if(motionEvent.getAction() == MotionEvent.ACTION_UP) {
            catchSplitter = false;
        }
        return screenGestureDetector.onTouchEvent(motionEvent);
    }
});

screen.addOnLayoutChangeListener(new View.OnLayoutChangeListener() {
    @Override
    public void onLayoutChange(View v, int left, int top, int right, int
bottom, int oldLeft, int oldTop, int oldRight, int oldBottom) {
        int screenHeight = screen.getMeasuredHeight();
        screen.removeOnLayoutChangeListener(this);
    }
});
```

Towards the end of the file, after the end of function onCreate(), insert the following code:

```
@Override
public boolean dispatchTouchEvent(MotionEvent ev) {
    if (!scrollInActionBar) {
        int[] location = new int[2];
        LinearLayout screen = (LinearLayout) findViewById(R.id.screen);
        screen.getLocationOnScreen(location);
        int x = location[0];
        int y = location[1];
        scrollInActionBar = ev.getRawY() < y;
    } else {
        screenGestureDetector.onTouchEvent(ev);
        if(ev.getAction() == MotionEvent.ACTION_UP) {
            scrollInActionBar = false;
            catchSplitter = false;
        }

    }
```

```
    return super.dispatchTouchEvent(ev);
}
```

You do not need to understand the code line by line here, though you can briefly have a look. All code is related to: 1) provide the ability to the user to move the divider; 2) be able to bring the divider back into the screen if the user has pushed it out of the screen, all the way up or down. Note that the user does not have to be that precise in catching the divider: placing the mouse (the finger) slightly above it will also work. However placing it on the map will not work (as this movement is already reserved for scrolling the map).

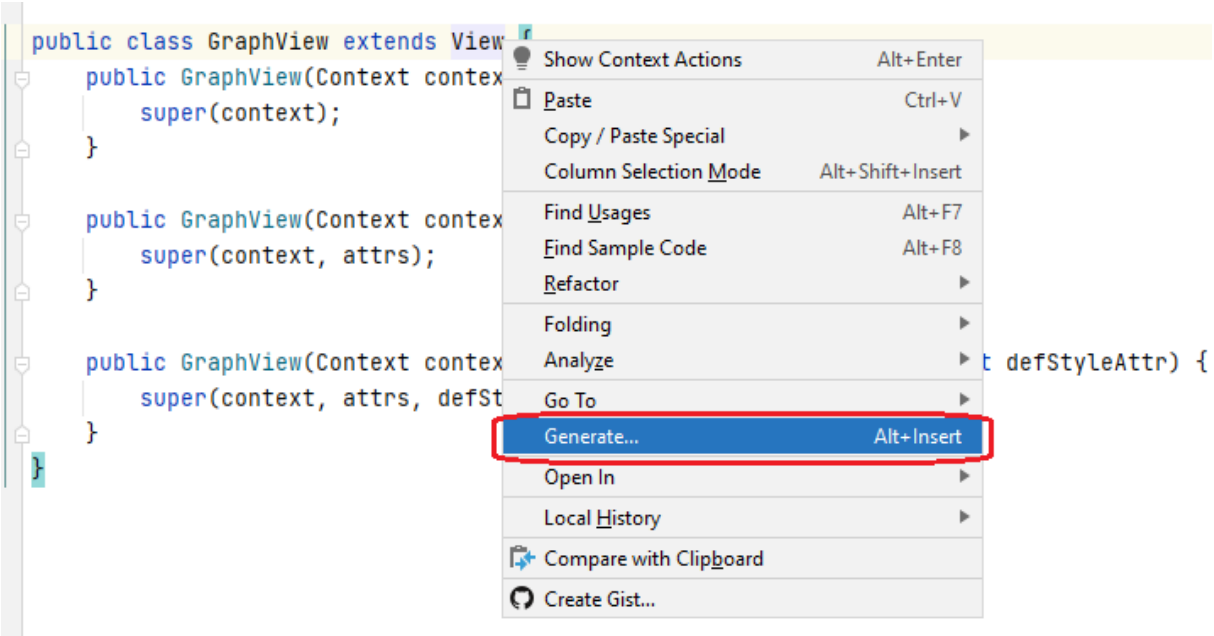
Run the app to see how it works (click Terminate to terminate the previous running app). Try to enlarge or shrink the graph.

Reserve space for drawing the graphs

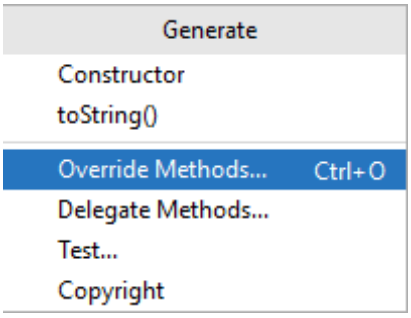
Now we will draw a black rectangle which will be enclosing the graphs. Above the rectangle we will leave some space for the legends, and on the other 3 sides we leave space for labels (left y-axis, right y-axis and x-axis).

Open file java / com.example.gfsforecast / GraphView.java for editing.

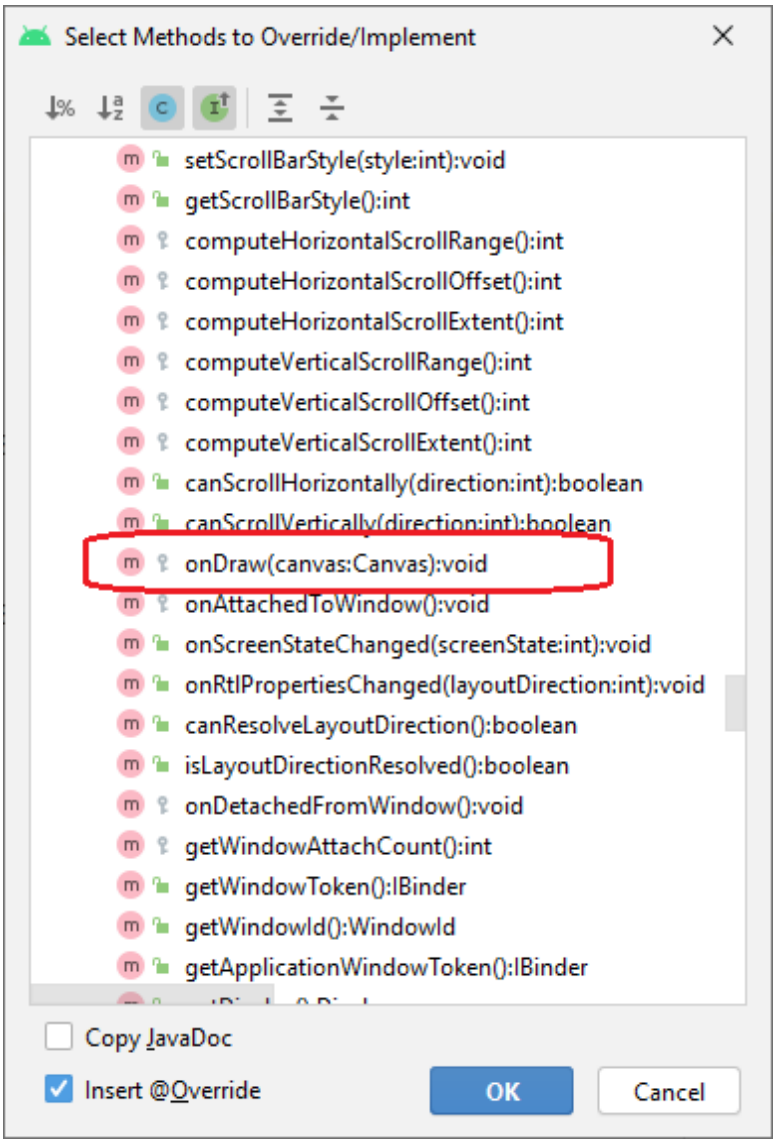
Right-click on View and click Generate…



Select Override Methods…



From the large list that appears, find and select the function named onDraw(), and click OK.



The code now looks like this:

```java
public class GraphView extends View {
    public GraphView(Context context) {
        super(context);
    }

    public GraphView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
    }

    public GraphView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
    }
}
```

The onDraw() function allows drawing on the canvas that belongs to the GraphView part of the app. The initial implementation does nothing, that is why the GraphView is empty / white.

Below super.onDraw() add the line:

```
drawGraphRect(canvas);
```

This calls the function drawGraphRect() which we still need to create.

Choose the appropriate action to let the editor auto-create the function for you.

```java
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    drawGraphRect(canvas);
}
```

Cannot resolve method 'drawGraphRect' in 'GraphView'

Create method 'drawGraphRect' in 'GraphView' Alt+Shift+Enter More actions... Alt+Enter

Result:

```
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        drawGraphRect(canvas);
    }

    private void drawGraphRect(Canvas canvas) {
    }
}
```

Add the following as the implementation of function drawGraphRect():

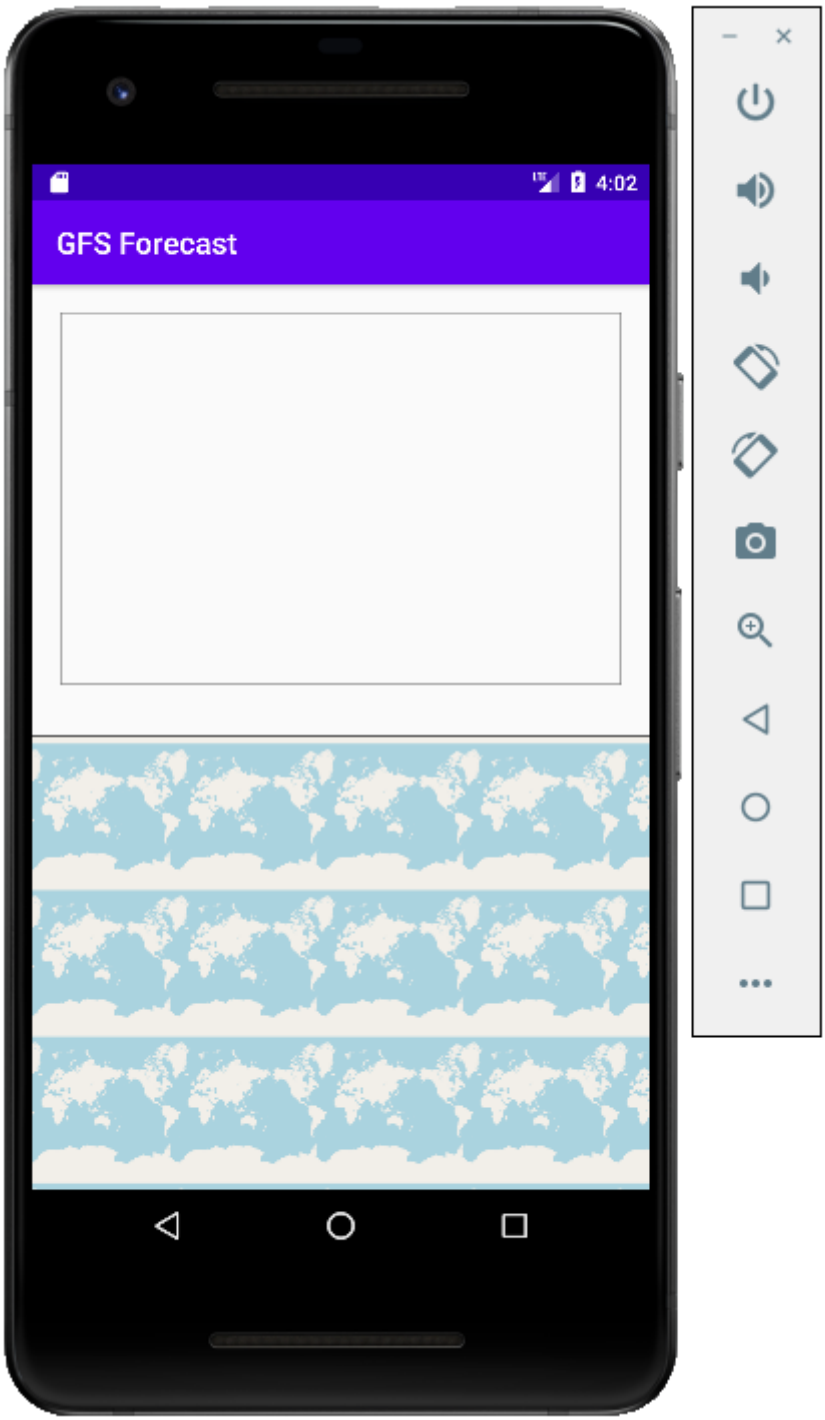
```
Paint paint = new Paint();
paint.setAntiAlias(true);
paint.setStyle(Paint.Style.STROKE);
canvas.drawRect(50, 50, 1030, 700, paint);
```

Result:

```
private void drawGraphRect(Canvas canvas) {
    Paint paint = new Paint();
    paint.setAntiAlias(true);
    paint.setStyle(Paint.Style.STROKE);
    canvas.drawRect( left: 50,  top: 50,  right: 1030,  bottom: 700, paint);
}
```

Explanation: AntiAlias is enabled, in order to get crisp lines, the line-style is set to stroke, and a rectangle with very specific coordinates is drawn.

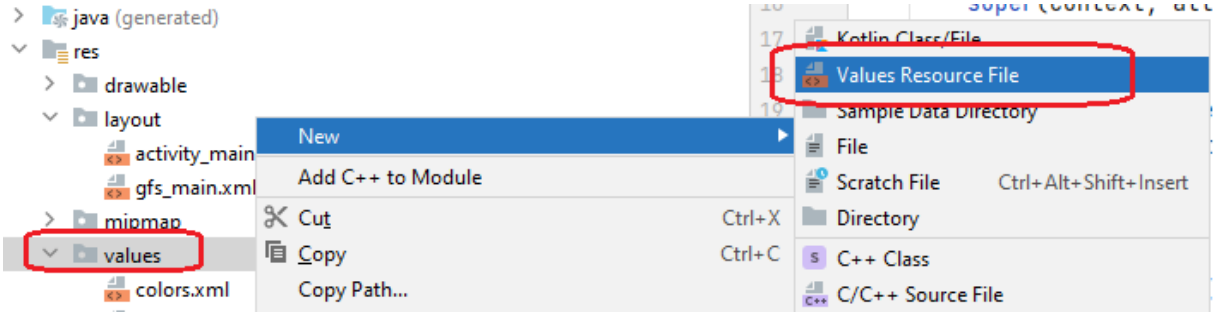
Run -> Run ‘app’ for testing this (Terminate the previous app).



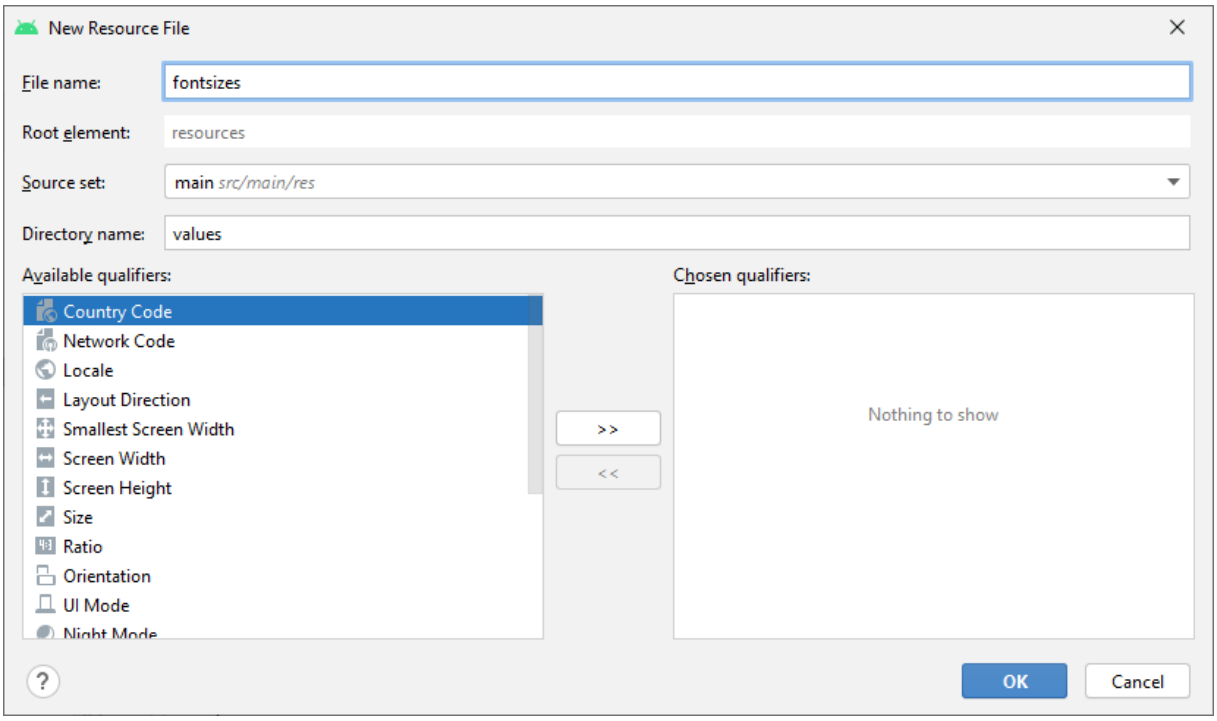
This code has several disadvantages. 1) The coordinates that are used do not depend on the device screen (e.g. on a phone with a smaller screen the rectangle would not fit on the screen). 2) The rectangle does not respect the size of the GraphView, when the user has moved the divider (you can try it: the rectangle always keeps the same size). 3) A new Paint() object is created every time the screen is re-drawn. This can be quite often. E.g. when resizing with the divider, the screen is redrawn multiple times per second. The Android framework will ensure that the unused Paint() objects will be “recycled”, but actually we don’t want to leave this to the auto-cleanup system when we can do something about it.

Before proceeding, we will add a fontSize element, in a way that it is easy to change if we want a different font size later on.

Right-click on values, and select New -> Values Resource File



Name the file fontsizes and click OK.



In the file that is created, add the following text between <resources> and </resources>

```
<dimen name="fontSizeRegular">10sp</dimen>
```

Result:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="fontSizeRegular">10sp</dimen>
</resources>
```

Note the unit of the font size being 10sp. The directive SP stands for Scalable Pixels. This scales correctly with higher or lower density screens.

Now edit the file java / com.example.gfsforecast / GraphView.java for editing.

Add the following line after public class GraphView extends View {

```
    private final Paint paint = new Paint();
```

Then add the following lines 3x, in each of the 3 constructors, after the super(contect); statement:

```
int fontSize =
getResources().getDimensionPixelSize(R.dimen.fontSizeRegular);
paint.setTextSize(fontSize);
```

This fetches the specified font size from the newly created fontsizes file, and assigns it to the paint object, which is created only once.

Result:

```
public class GraphView extends View {

    private final Paint paint = new Paint();

    public GraphView(Context context) {
        super(context);
        int fontSize = getResources().getDimensionPixelSize(R.dimen.fontSizeRegular);
        paint.setTextSize(fontSize);
    }

    public GraphView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        int fontSize = getResources().getDimensionPixelSize(R.dimen.fontSizeRegular);
        paint.setTextSize(fontSize);
    }

    public GraphView(Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        int fontSize = getResources().getDimensionPixelSize(R.dimen.fontSizeRegular);
        paint.setTextSize(fontSize);
    }
```

Add the following function after drawGraphRect():

```
private float getFontHeight(Paint p) {
    Paint.FontMetrics fontMetrics = p.getFontMetrics();
    return Math.abs(fontMetrics.top - fontMetrics.bottom);
}
```

This function computes the font height in pixels for the current screen. The screen-independent 10sp size is converted to a screen-specific size in pixels.

Add the following lines at the top of the file, just under private final Paint paint = …

```
private int left = 0;
private int topFull = 0;
private int topHalf = 0;
private int right = 0;
private int bottom = 0;
```

```
private int marginTop = 0;
private int marginLeft = 0;
private int marginBottom = 0;
private int marginRight = 0;
```

Those are placeholders for parts of the screen dimensions, that will ensure plotting the rectangle at the correct position.

Add the following function before onDraw():

```
@Override
protected void onLayout(boolean changed, int left, int top, int right, int
bottom) {
    super.onLayout(changed, left, top, right, bottom);
    this.left = left;
    this.topFull = top;
    this.right = right;
    this.bottom = bottom;
    int fontHeight = (int) getFontHeight(paint);
    this.marginLeft = 2 * fontHeight;
    this.marginRight = 2 * fontHeight;
    this.marginTop = 2 * fontHeight;
    this.topHalf = 4 * fontHeight;
    this.marginBottom = 2 * fontHeight;
}
```

This function is called everytime the GraphView's dimensions change, e.g. when the user resizes it by swiping the divider. It captures and stores the new dimensions.

Change onDraw() as follows:

```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    paint.setAntiAlias(true); // crisp graphs and lines
    paint.setStyle(Paint.Style.FILL);
    paint.setARGB(255, 255, 255, 255);
    canvas.drawRect(left, topFull, right, bottom, paint); // clear the
background
    paint.setStyle(Paint.Style.STROKE);
    if (bottom - marginBottom > topHalf + marginTop) {
        drawGraphRect(canvas);
    }
}
```

Now onDraw() will erase the background (by plotting a filled white rectangle) before drawing anything else. Also the stroke style is set here, since it will be used multiple times.

Change the implementation of drawGraphRect as follows:

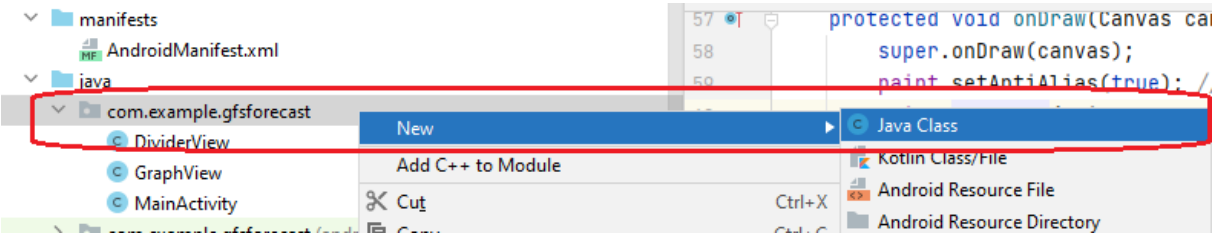
```
private void drawGraphRect(Canvas canvas) {
    paint.setARGB(255, 0, 0, 0);
    paint.setStrokeWidth(3);
    canvas.drawRect(left + marginLeft, topHalf + marginTop, right -
marginRight, bottom - marginBottom, paint);
}
```

This will draw the graph rectangle in such a way that its size and position depends on the size available within the GraphView, leaving space around for legends and labels.

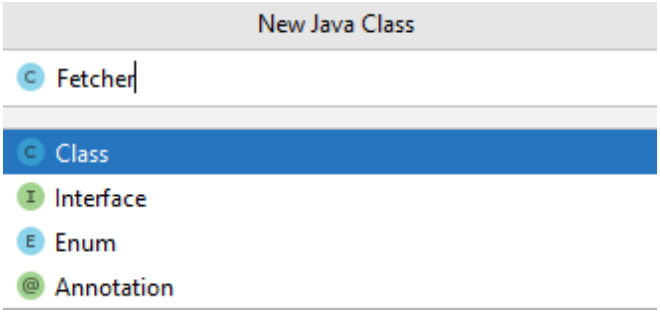
Retrieving data from the Web API server

Add a class that will be responsible for downloading data from the Web API server (given a latitude and longitude), and store it in its internal memory for easier use. Name the class “Fetcher”.

Right-click on java / com.example.gfsforecast and select New -> Java Class.



Name: Fetcher



Use the following code for the implementation of class Fetcher. Then resolve all required “import class” for the classes that come from libraries (Vector, HashMap, Date, DateFormat, SimpleDateFormat). Ensure to use java.util.Date (multiple choices are available), java.text.DateFormat, java.text.SimpleDateFormat and java.util.TimeZone.

```
public class Fetcher {
    private Vector<HashMap> values = new Vector<HashMap>();
    private double minVal;
    private double maxVal;
    private Date minDate;
    private Date maxDate;
    private final DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-
dd");

    public Fetcher() {
        dateFormat.setTimeZone(TimeZone.getTimeZone("UTC"));
    }

    public Vector<HashMap> getValues() {
        return values;
    }

    public double minVal() {
        return minVal;
    }

    public double maxVal() {
        return maxVal;
    }
```

```
    public Date minDate() {
        return minDate;
    }

    public Date maxDate() {
        return maxDate;
    }
}
```

The class has several placeholders for keeping track of the minimum date available in the data returned from the Map API server, the maximum date (for the x-axis of the graphs), the minimum value, the maximum value (for y-stretching of the graphs), and the actual values as date-value pairs (HashMap is the Java equivalent of a Python dictionary). An equal number of functions is available to get the values.

Now add a function fetchValues() that is responsible to actually download the data. Solve all "import class" issues.

```
public void fetchValues(double lat, double lon) {
    String serverUrl =
"http://rsgportal.itc.utwente.nl/gfs/get_forecast_ext.py";
    URL url = new URL(serverUrl + "?lon=" + lon + "&lat=" + lat);
    HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
    connection.setRequestMethod("GET");
    connection.connect();
    InputStream is = connection.getInputStream();
    values.clear();
}
```

This code has a reference to the URL of the server. If you want to try using the service that was previously created on your own laptop, you can (temporarily) replace the URL with your laptop's URL. Note that http://localhost/fetch_forecast.py will probably not work from within the app, as the concept "localhost" would refer to the emulated android device (Pixel 2) and not to the laptop. Use the IP address of the laptop instead (discover the IP address with ipconfig in a Command Prompt).

The code also appends the latitude and longitude to the URL (the lat and lon are parameters of the fetchValues() function), opens the connection, and empties the previously fetched values.

Append the following lines after values.clear();

```
boolean firstValue = true;
boolean firstDate = true;
JsonReader reader = new JsonReader(new InputStreamReader(is));
reader.beginArray();
while (reader.hasNext()) {
    HashMap<String, Object> parameter = new HashMap<String, Object>();
    reader.beginObject();
    while (reader.hasNext()) {
        String key = reader.nextName();
        if (key.equals("series")) {
            Vector<Pair<Double, Date>> series = new Vector<Pair<Double,
Date>>();
            reader.beginArray();
            while (reader.hasNext()) {
                HashMap<String, Object> item = new HashMap<String,
```

```
Object>();
                reader.beginObject();
                while (reader.hasNext()) {
                    String innerKey = reader.nextName();
                    if (innerKey.equals("date")) {
                        String value = reader.nextString();
                        try {
                            Date date = dateFormat.parse(value);
                            item.put(innerKey, date);
                            if (firstDate) {
                                minDate = date;
                                maxDate = date;
                                firstDate = false;
                            } else {
                                minDate = (minDate.compareTo(date) < 0) ?
minDate : date;
                                maxDate = (maxDate.compareTo(date) > 0) ?
maxDate : date;
                            }
                        } catch (ParseException e) {
                            e.printStackTrace();
                        } finally {

                        }
                    } else if (innerKey.equals("value")) {
                        double value = reader.nextDouble();
                        item.put(innerKey, value);
                        if (parameter.get("axisLeft").equals("true")) {
                            if (firstValue) {
                                minVal = value;
                                maxVal = value;
                                firstValue = false;
                            } else {
                                minVal = Math.min(minVal, value);
                                maxVal = Math.max(maxVal, value);
                            }
                        }
                    }
                }
                reader.endObject();
                series.add(new Pair<Double, Date>((Double)
item.get("value"), (Date) item.get("date")));
            }
            reader.endArray();
            parameter.put(key, series);
        } else {
            String value = reader.nextString();
            parameter.put(key, value);
            if (key.equals("parametername")) {
                if (value.equals("RH")) {
                    parameter.put("axisLeft", "false");
                } else {
                    parameter.put("axisLeft", "true");
                }
            }
        }
    }
    reader.endObject();
    values.add(parameter);
}
reader.endArray();
```

Solve the missing imports (ensure to use android.util.Pair and java.text.ParseException).

Also note the need to protect the code against Exceptions (that would normally crash the program).

Add the try keyword after the first line of the function (String serverUrl = …)

```
try {
```

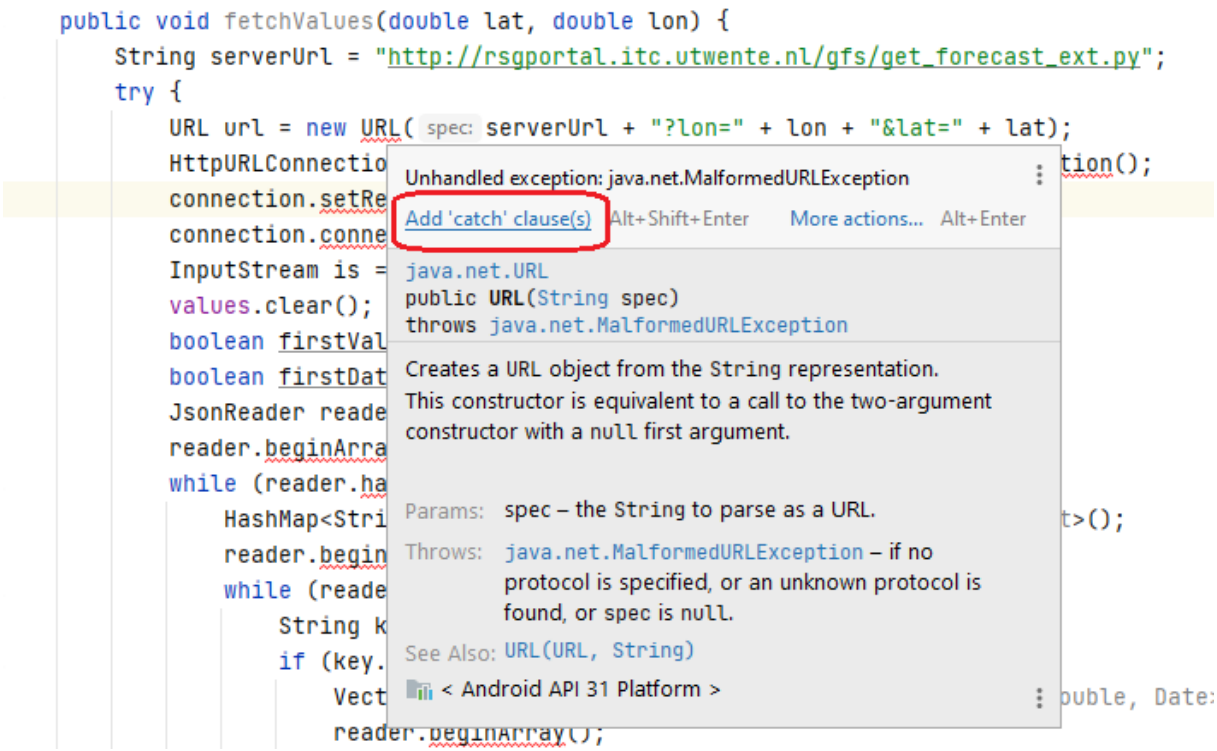
At the end of the function, add a closing bracket on a new line.

```
}
```

The editor will react and indent the code. Now it looks like this:

```
public void fetchValues(double lat, double lon) {
    String serverUrl = "http://rsgportal.itc.utwente.nl/gfs/get_forecast_ext.py";
    try {
        URL url = new URL( spec: serverUrl + "?lon=" + lon + "&lat=" + lat);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");
        connection.connect();
```

Now move over the red-underlined word URL, and select the option “add ‘catch’ clauses”:



At the bottom of the file, the code now looks like this:

```
                }
            }
            reader.endObject();
            values.add(parameter);
        }
        reader.endArray();
    } catch (ProtocolException e) {
        e.printStackTrace();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
  }
}
```

This means that the program will not crash in the event that e.g. the URL is malformed or that the internet was disconnected while reading data, but will be returned to a controllable state.

The new code reads-in the JSON text that was received from the Web API server, keeps track of the minimum and maximum dates and values, inserts the attributes for each parameter (its parameter-name, its timeseries and a property that indicates whether the parameter belongs to the left y-axis or to the right one) into the values. Note that all parameters are set to the left y-axis, except RH (Relative Humidity) for which a right y-axis is created. This is because the units of RH are in percentage, which ranges from 0 to 100, and deviates too much from the values of all other parameters, that range from approximately 0 to 30.

Edit file java / com.example.gfsforecast / GraphView.java

Add the following in class GraphView, above the private final Paint pain = new Paint();

```
private Vector<HashMap> values = new Vector<HashMap>();
private double minVal;
private double maxVal;
private Date minDate;
private Date maxDate;
private final Vector<Path> paths = new Vector<Path>();
```

Solve the imports (ensure you select java.util.Date and android.graphics.Path).

This is a local copy of the values, the minimum and maximum date and value (for stretching), and a list of paths. A path is an object that contains line-data for plotting lines.

After function onLayout() add function redrawPaths():

```
private void redrawPaths() {
    Matrix matrixLeft = new Matrix();
    matrixLeft.reset();
    Matrix matrixRight = new Matrix();
    matrixRight.reset();
    if (minDate != null && maxDate != null && minDate.compareTo(maxDate) <
0) {
        matrixLeft.preTranslate(-minDate.getTime(), -(float) minVal);
```

```
        matrixLeft.postScale((float) (right - left - marginLeft -
marginRight) / (float) (maxDate.getTime() - minDate.getTime()), (float)
((topHalf + marginTop + marginBottom -bottom) / (maxVal - minVal))); //
stretch it to fill the View
        matrixLeft.postTranslate(marginLeft, bottom - marginBottom);
        matrixRight.preTranslate(-minDate.getTime(), 0);
        matrixRight.postScale((float) (right - left - marginLeft -
marginRight) / (float) (maxDate.getTime() - minDate.getTime()), (float)
((topHalf + marginTop + marginBottom -bottom) / 100)); // stretch it to
fill the View
        matrixRight.postTranslate(marginLeft, bottom - marginBottom);
    }
    paths.setSize(values.size());
    for (int i = 0; i < paths.size(); ++i) {
        if (paths.get(i) != null)
            paths.get(i).rewind();
        else
            paths.set(i, new Path());
        HashMap<String, Object> parameter = values.get(i);
        Vector<Pair<Double, Date>> series = (Vector<Pair<Double, Date>>)
parameter.get("series");
        if ((series.size() > 1) && (right > left) && (Math.abs(topHalf -
bottom) > 0)) {
            long x = series.get(0).second.getTime();
            float y = series.get(0).first.floatValue();
            paths.get(i).moveTo(series.get(0).second.getTime(),
series.get(0).first.floatValue()); // first item
            for (int j = 1; j < series.size(); ++j) {
                paths.get(i).lineTo(series.get(j).second.getTime(),
series.get(j).first.floatValue()); // all other items
            }
            paths.get(i).transform(parameter.get("axisLeft").equals("true")
? matrixLeft : matrixRight);
        }
    }
}
```

Resolve the imports (android.graphics.Matrix, android.util.Pair).

The function redrawPaths() creates the paths using the series-data from the parameters, as a concatenation of moveTo() and lineTo() commands. Each path is scaled, so that it fills the available space (horizontally it fills the date-range, and vertically the value-range). The Relative Humidity is vertically scaled from 0 to 100. Scaling is done with the Matrix objects (matrixLeft for the x-axis and the left y-axis, and matrixRight for the x-axis and the right y-axis).

After function redrawPaths() add function setValues():

```
public void setValues(Vector<HashMap> values, double minVal, double maxVal,
Date minDate, Date maxDate) {
    this.values = values;
    this.minVal = minVal;
    this.maxVal = maxVal;
    this.minDate = minDate;
    this.maxDate = maxDate;
    redrawPaths();
}
```

This will receive the values, keep a local copy, and re-create the paths.

Add two more functions: setARGB() and drawGraphs():

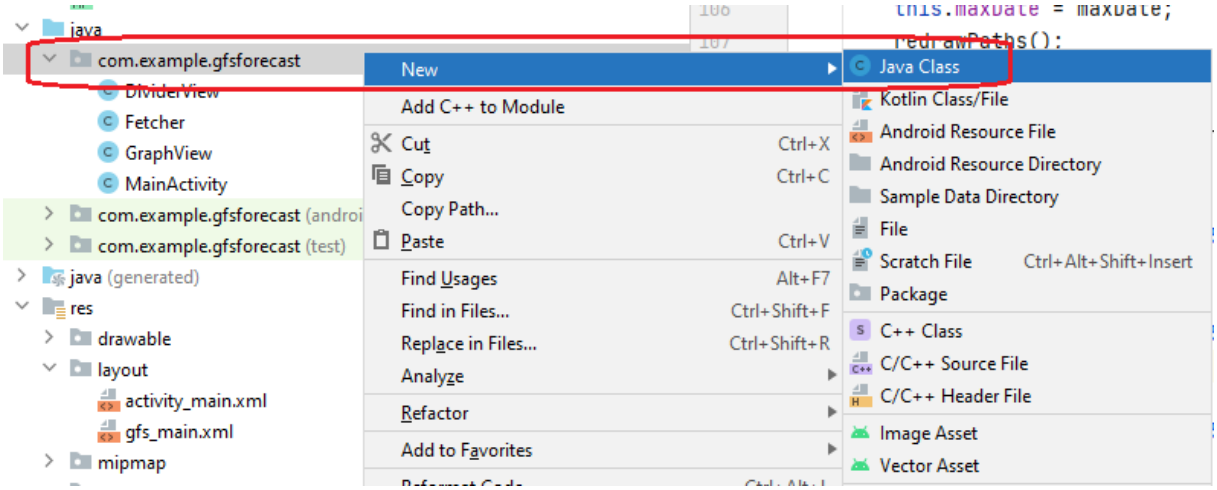
```
private void setARGB(String color) {
    switch (color) {
        case "orange":
            paint.setARGB(255, 255, 166, 2);
            break;
        case "green":
            paint.setARGB(255, 4, 129, 4);
            break;
        case "blue":
            paint.setARGB(255, 2, 2, 255);
            break;
        case "gray":
            paint.setARGB(255,129, 129, 129);
            break;
        case "red":
            paint.setARGB(255, 255, 2, 2);
            break;
        case "cyan":
            paint.setARGB(255, 4, 255, 255);
            break;
        default:
            paint.setARGB(255, 0, 0, 0);
    }
}

private void drawGraphs(Canvas canvas) {
    paint.setARGB(255, 1, 185, 255);
    paint.setStrokeWidth(3);
    for (int i = 0; i < paths.size(); ++i) {
        HashMap<String, Object> parameter = values.get(i);
        String color = (String)parameter.get("color");
        setARGB(color);
        canvas.drawPath(paths.get(i), paint);
    }
}
```

Then in onDraw() append a call to drawGraphs(), immediately after drawGraphRect():

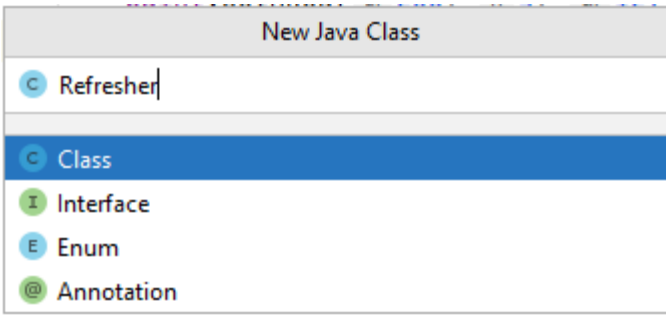
```
drawGraphs(canvas);
```

At this stage, GraphView.java is prepared to actually plot graphs (but there is still no grid or labels or legends). To proceed, a call to its setValues() must be triggered. For this we first add one more class, responsible for fetching data from the Web API server, and sending the values to the GraphView.

Right-click on com.example.gfsforecast, and select New -> Java Class .



Name it Refresher:



Add the following as the implementation of Refresher:

```
private GraphView graphview;
private GeoPoint location;
private boolean refresh = false;
private boolean inThread = false;

public Refresher(GraphView graphview) {
    this.graphview = graphview;
    this.location = new GeoPoint(0.0, 0.0);
}

public void setLocation(GeoPoint pt) {
    this.location = pt;
    this.refresh = true;
}

private void getData() {
    double lat = location.getLatitude();
    double lon = location.getLongitude();
    Fetcher fetcher = new Fetcher();
    fetcher.fetchValues(lat, lon);
    Vector<HashMap> values = fetcher.getValues();
    double minVal = fetcher.minVal();
    double maxVal = fetcher.maxVal();
    Date minDate = fetcher.minDate();
    Date maxDate = fetcher.maxDate();
    graphview.setValues(values, minVal, maxVal, minDate, maxDate);
    graphview.postInvalidate();
}
```

```java
public void refresh() {
    if (inThread)
        return;
    new Thread(new Runnable() {
        @Override
        public void run() {
            inThread = true;
            while (refresh) {
                refresh = false;
                getData();
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                }
            }
            inThread = false;
        }
    }).start();
}
```

Resolve the missing imports.

This class has an important function: refresh(). This function will fetch the data and re-plot the graphs. Note that this work is done in a separate thread (multithreading application). This is so that the user does not experience any delays in the app's user-interface while the download is ongoing, as perhaps the internet is slow (and the app would otherwise freeze while waiting for the download to complete). While downloading, the app will have a responsive user-interface.

Note that the refresh is protected to happen at most every 1 second (to prevent overloading the Web API server, and also to prevent the phone from unintentionally over-using the mobile-data, after which the user is charged).

The Refresher must be initiated from the MainActivity. Therefore open file java / com.example.gfsforecast / MainActivity.java for editing.

At the top of the file, after the line with private boolean catchSplitter = false; add:

```java
private Refresher refresher;
```

At the end of function onCreate(), add:

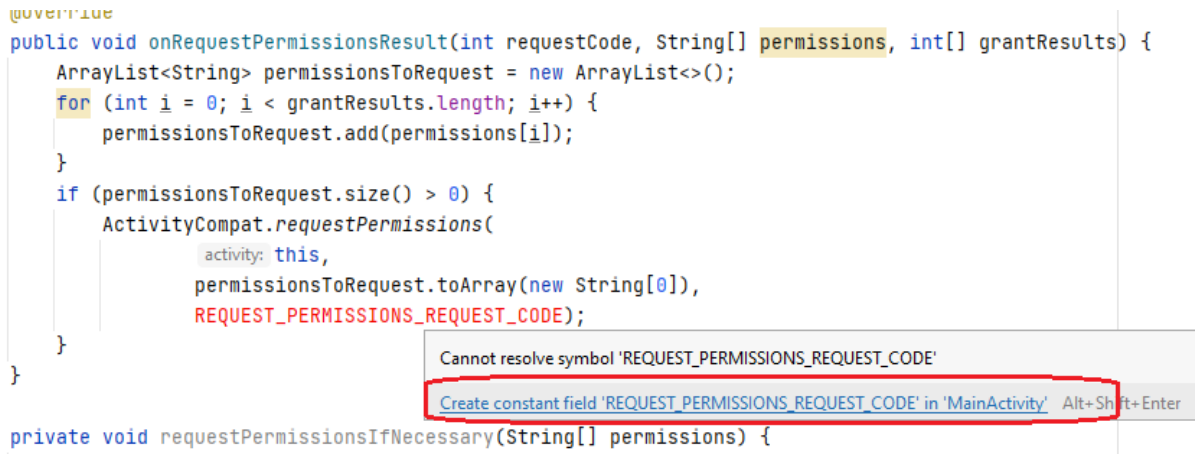
```java
refresher = new Refresher(graph);
```

Append the following four functions as the last functions of class MainActivity (after dispatchTouchEvent()):

```java
@Override
public void onResume() {
    super.onResume();
    map.onResume(); //needed for compass, my location overlays
}

@Override
public void onPause() {
    super.onPause();
    map.onPause();  //needed for compass, my location overlays
```

```
}

@Override
public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
    ArrayList<String> permissionsToRequest = new ArrayList<>();
    for (int i = 0; i < grantResults.length; i++) {
        permissionsToRequest.add(permissions[i]);
    }
    if (permissionsToRequest.size() > 0) {
        ActivityCompat.requestPermissions(
                this,
                permissionsToRequest.toArray(new String[0]),
                REQUEST_PERMISSIONS_REQUEST_CODE);
    }
}

private void requestPermissionsIfNecessary(String[] permissions) {
    ArrayList<String> permissionsToRequest = new ArrayList<>();
    for (String permission : permissions) {
        if (ContextCompat.checkSelfPermission(this, permission)
                != PackageManager.PERMISSION_GRANTED) {
            // Permission is not granted
            permissionsToRequest.add(permission);
        }
    }
    if (permissionsToRequest.size() > 0) {
        ActivityCompat.requestPermissions(
                this,
                permissionsToRequest.toArray(new String[0]),
                REQUEST_PERMISSIONS_REQUEST_CODE);
    }
}
```

Resolve the missing imports. For the REQUEST_PERMISSIONS_REQUEST_CODE, select the option to create a constant field in MainActivity.



Select type = int:

```
private static final int REQUEST_PERMISSIONS_REQUEST_CODE = ;
private MapView map
private GestureDetec
private boolean scro
private boolean catc
private Refresher re
```

int
byte
char
short
Press Enter or Tab to replace

Initialize the value to 1. The result looks like this:

```
private static final int REQUEST_PERMISSIONS_REQUEST_CODE = 1;
```

The onPause() and onResume() functions are for the map to behave properly when the user switches to another app.
The permissions functions are to assist asking the user for permissions required by this app (e.g. this app wants to read your location).

Finally, add the following code near the top of the file, between map.setTileSource() and LinearLayout screen = …

```
map.setMultiTouchControls(true);
MyLocationNewOverlay mMyLocationOverlay = new MyLocationNewOverlay(new
GpsMyLocationProvider(this), map);
IMapController mapController = map.getController();
mMyLocationOverlay.enableMyLocation();
mMyLocationOverlay.enableFollowLocation();
mMyLocationOverlay.setEnabled(true);
mMyLocationOverlay.setDrawAccuracyEnabled(true);
Marker mapCenter = new Marker(map);
mapCenter.setPosition((GeoPoint) map.getMapCenter());
mapCenter.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM);
mapCenter.setOnMarkerClickListener(new Marker.OnMarkerClickListener() {
    @Override
    public boolean onMarkerClick(Marker marker, MapView mapView) {
        mapController.setZoom(12.0);
        mapController.setCenter(mMyLocationOverlay.getMyLocation());
        mapController.animateTo(mMyLocationOverlay.getMyLocation());
        mapCenter.setPosition(mMyLocationOverlay.getMyLocation());
        return false;
    }
});
map.getOverlays().add(mapCenter);
Handler mHandler = new Handler(Looper.getMainLooper());
mMyLocationOverlay.runOnFirstFix(new Runnable() {
    @Override
    public void run() {
        mHandler.post(new Runnable() {
            public void run() {
                mapController.setZoom(12.0);

mapController.setCenter(mMyLocationOverlay.getMyLocation());
```

```
mapController.animateTo(mMyLocationOverlay.getMyLocation());
                mapCenter.setPosition(mMyLocationOverlay.getMyLocation());
            }
        });
    }
});
map.getOverlays().add(mMyLocationOverlay);

map.addMapListener(new MapListener() {
    @Override
    public boolean onScroll(ScrollEvent scrollEvent) {
        GeoPoint pt = (GeoPoint) map.getMapCenter();
        mapCenter.setPosition(pt);
        refresher.setLocation(pt);
        refresher.refresh();
        return false;
    }
    @Override
    public boolean onZoom(ZoomEvent zoomEvent) {
        GeoPoint pt = (GeoPoint) map.getMapCenter();
        mapCenter.setPosition(pt);
        refresher.setLocation(pt);
        refresher.refresh();
        return false;
    }
});

requestPermissionsIfNecessary(new String[] {
        // ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION are needed to
show the current location
        Manifest.permission.ACCESS_COARSE_LOCATION,
        Manifest.permission.ACCESS_FINE_LOCATION,
        // WRITE_EXTERNAL_STORAGE is required in order to show the map
        Manifest.permission.WRITE_EXTERNAL_STORAGE
});
```

Resolve the missing imports (at Handler select android.os.Handler).

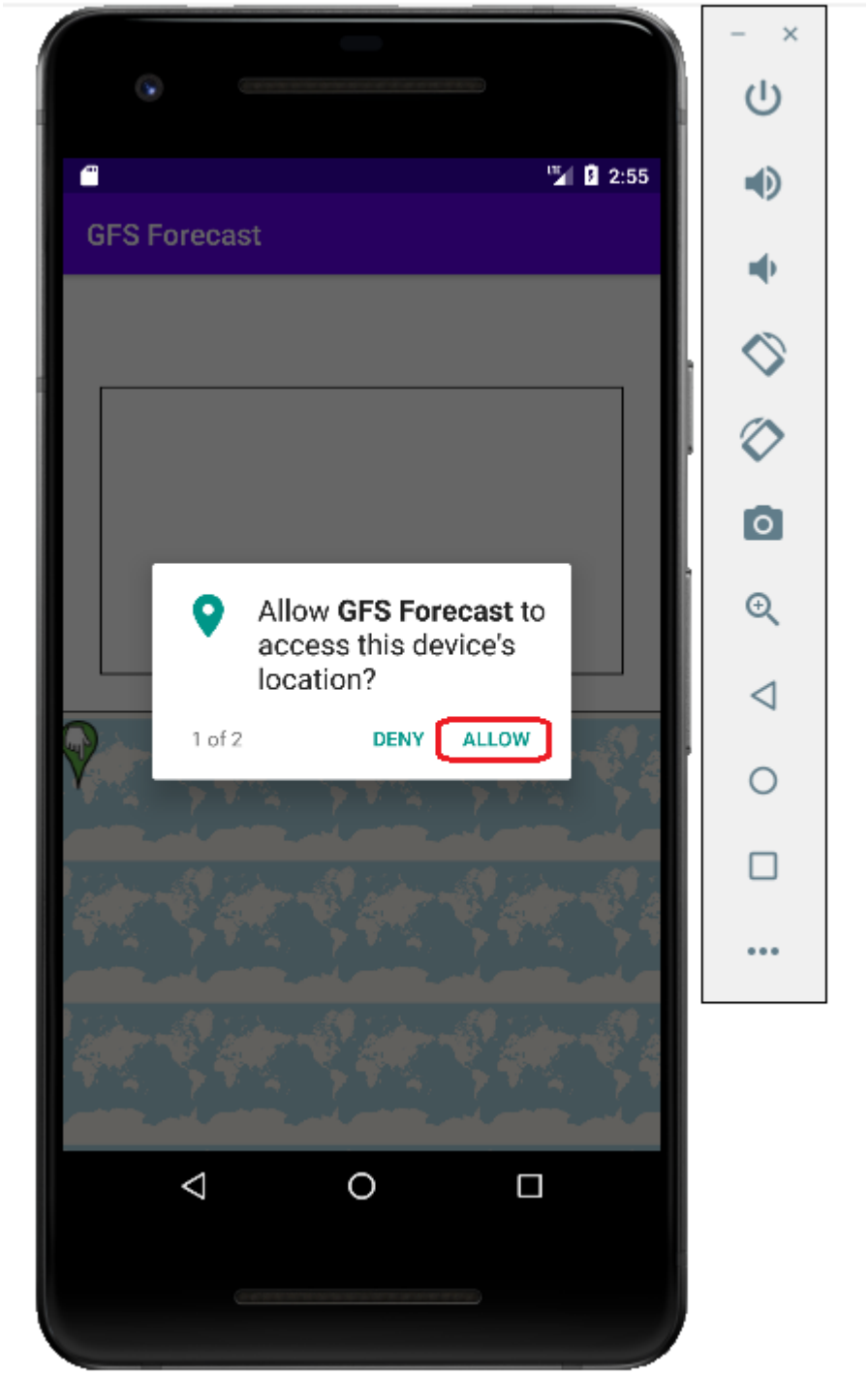
The new code asks the user for the required permissions (location and storage). It also adds two markers on the screen:

-  the current location of the device / phone
-  the center of the map, which determines the location for the data / graphs. Tapping on the center of the map marker moves the center of the map to the current location ("find me").

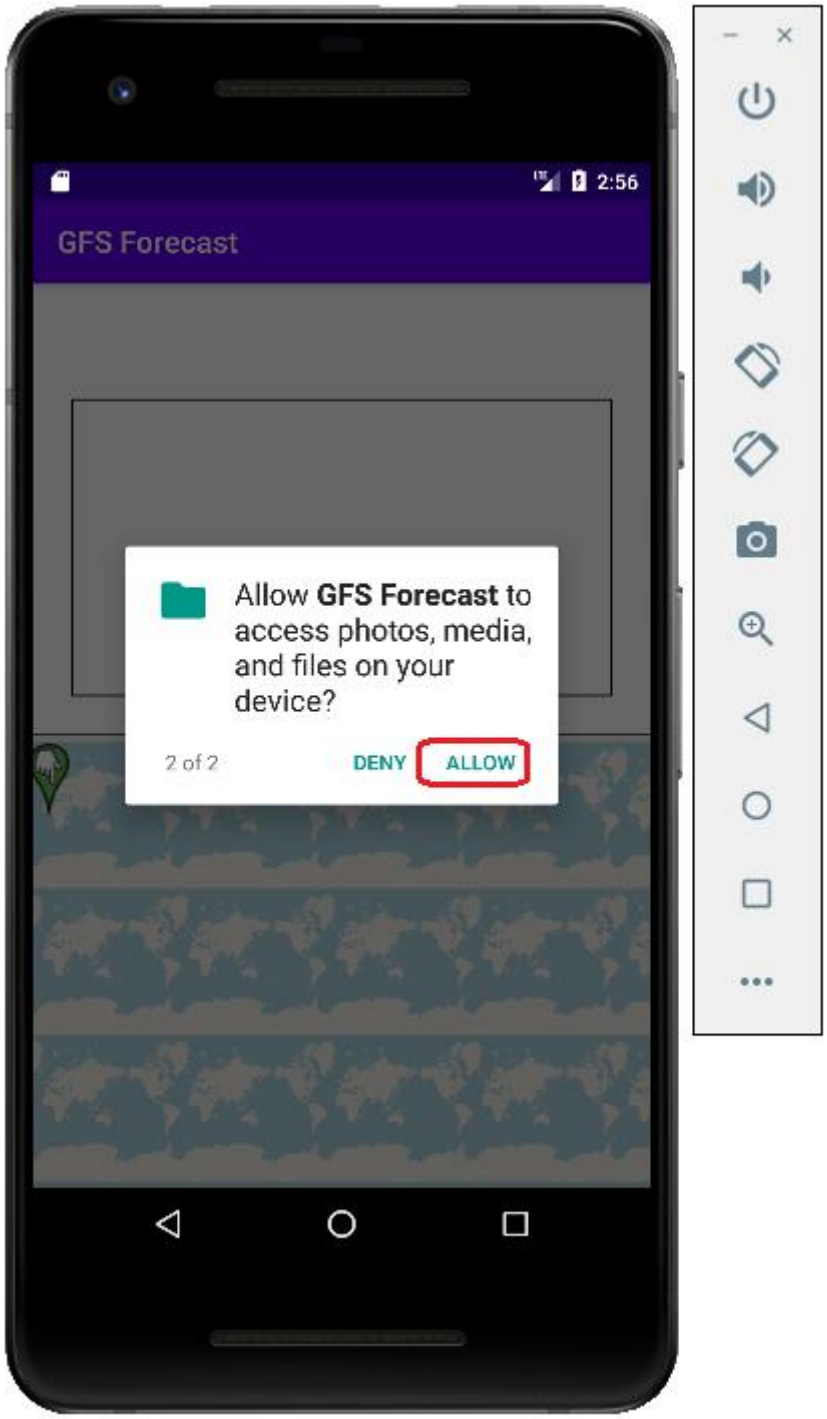
Start the app (Run -> Run 'app' to test the new version).

Answer Yes to both permissions that are asked.

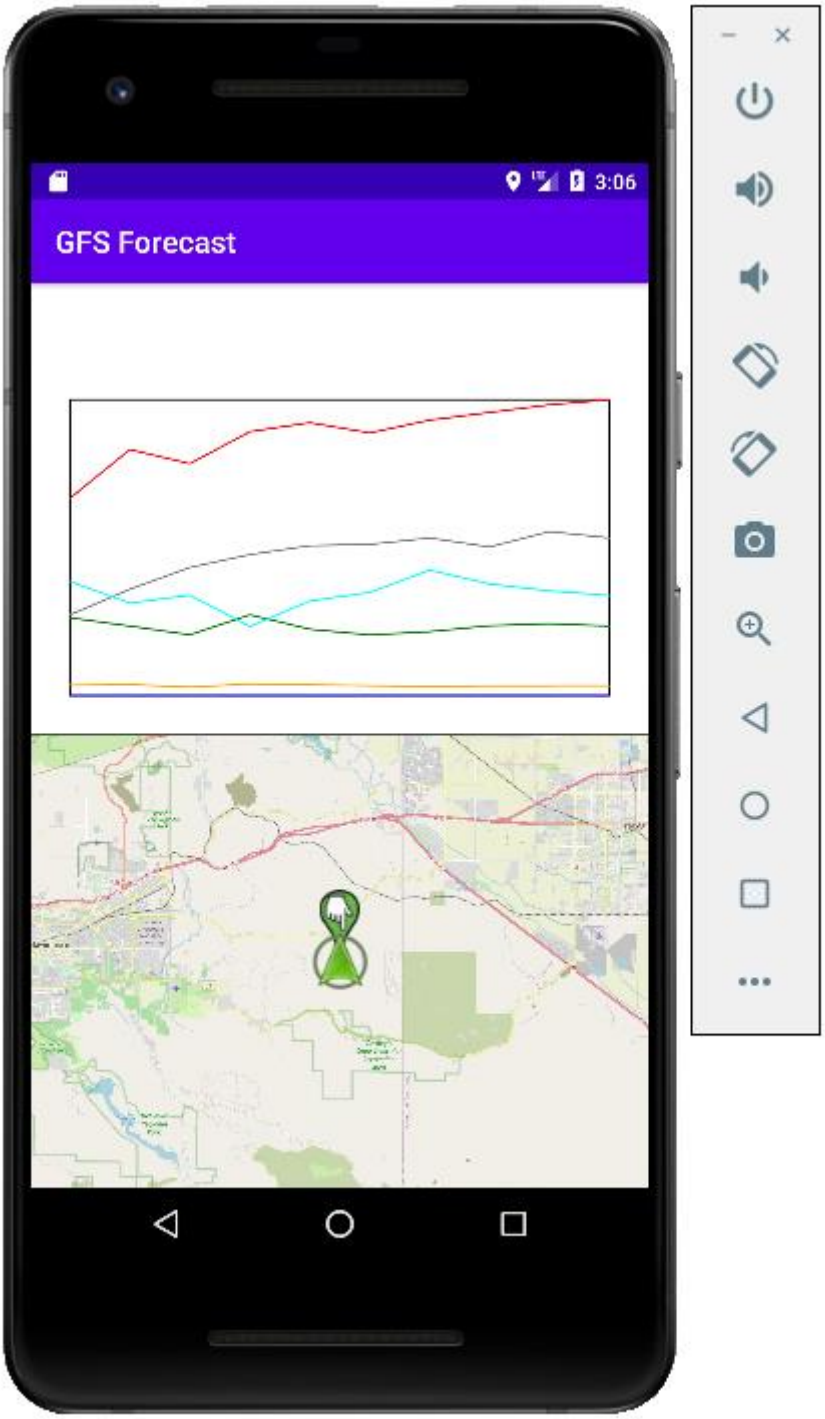
Location permission:



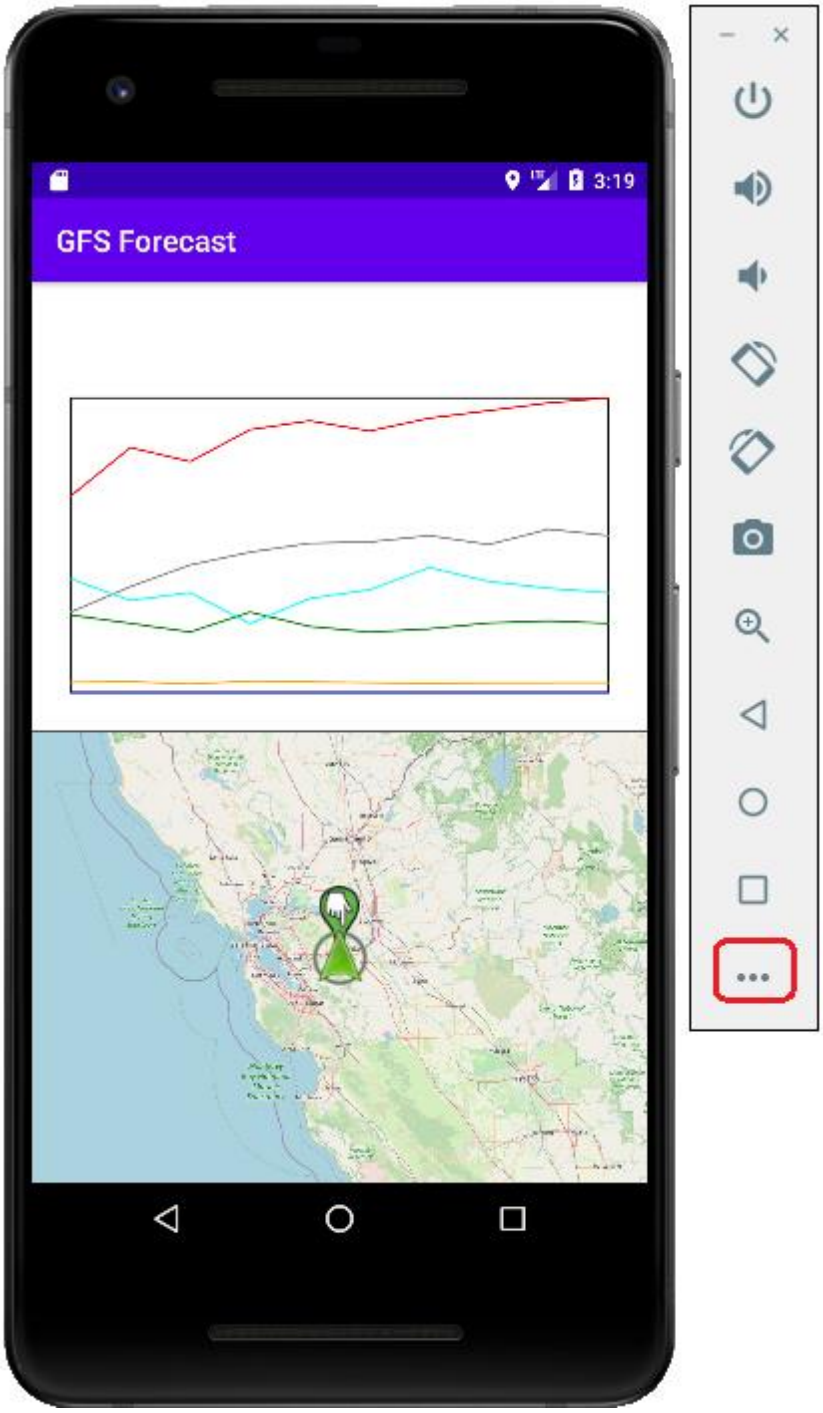
Permission to the device's storage. As you should be able to confirm, the app is not interested to access any of the user's photos. However requesting space for caching the map tiles pops the same message to the user.



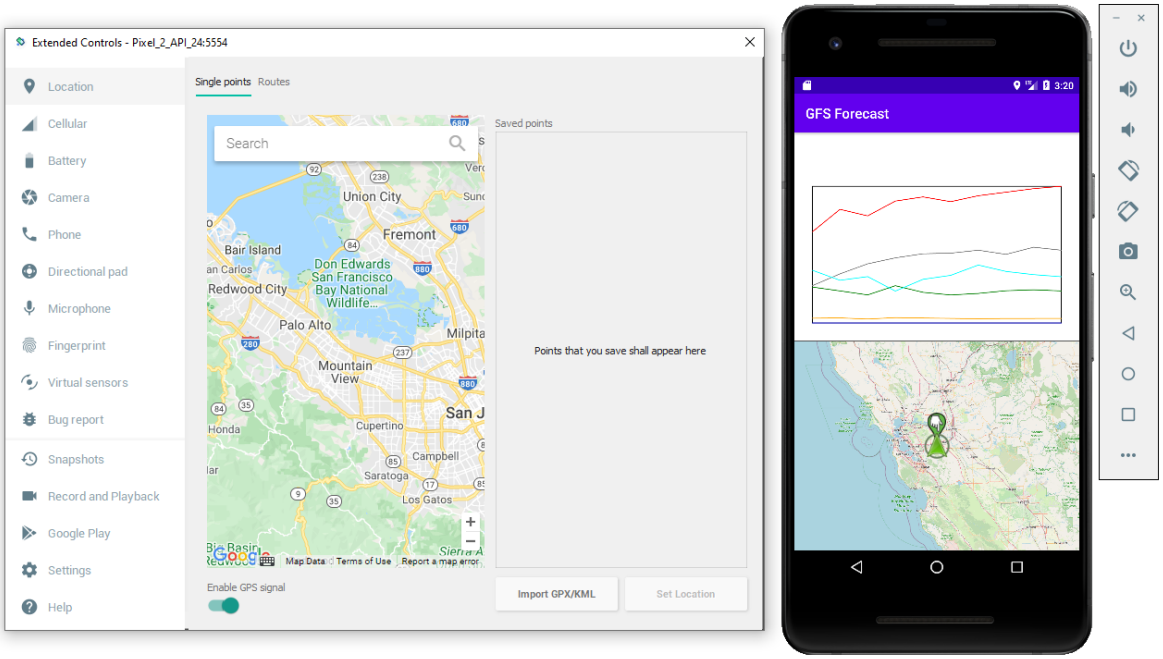
The app starts, thinking it is somewhere in California, in the United States.



You can, however, manipulate the GPS of the emulated device. Click on the three dots (…) at the bottom of the device's menu.

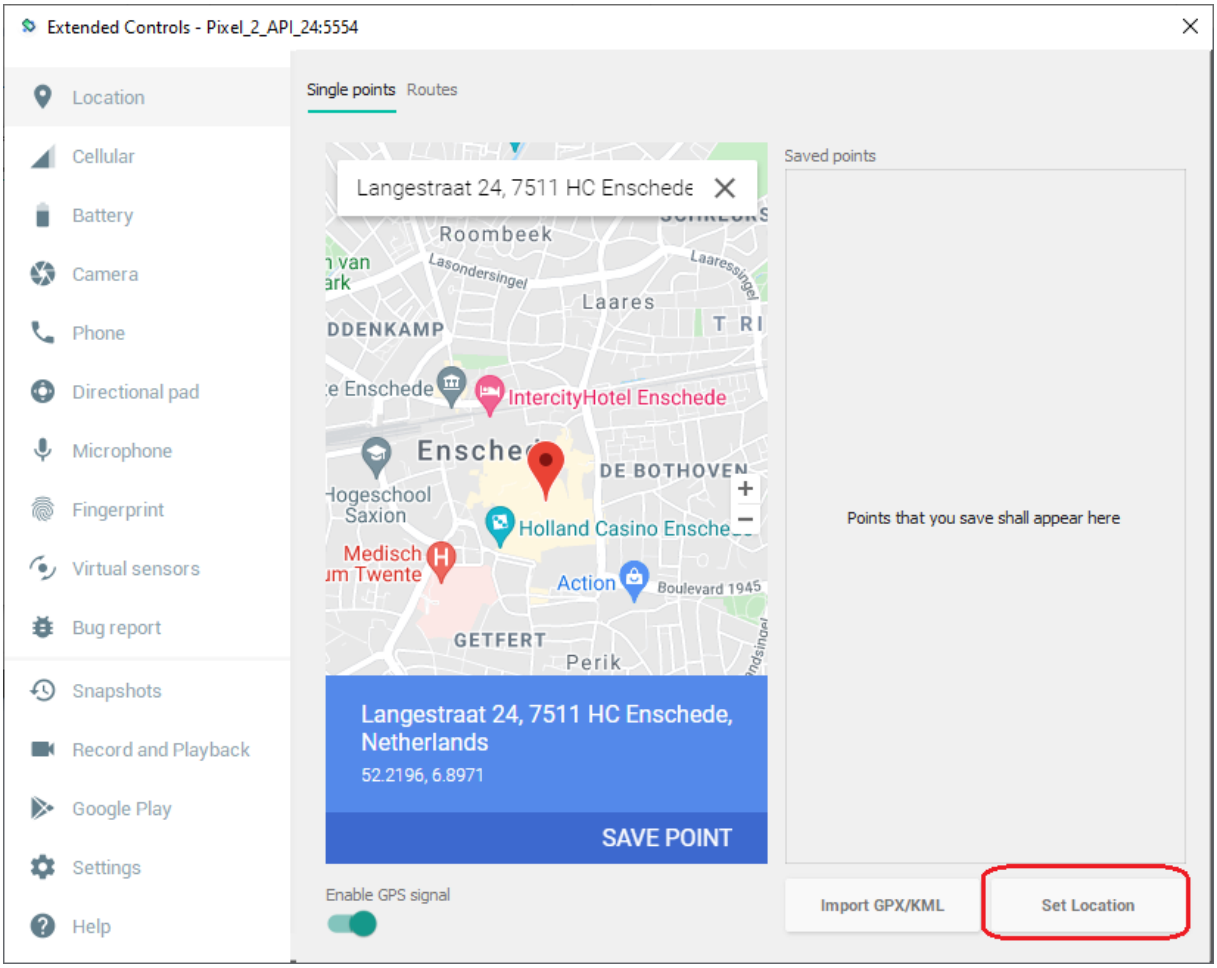


This will open the option to set the device's location.

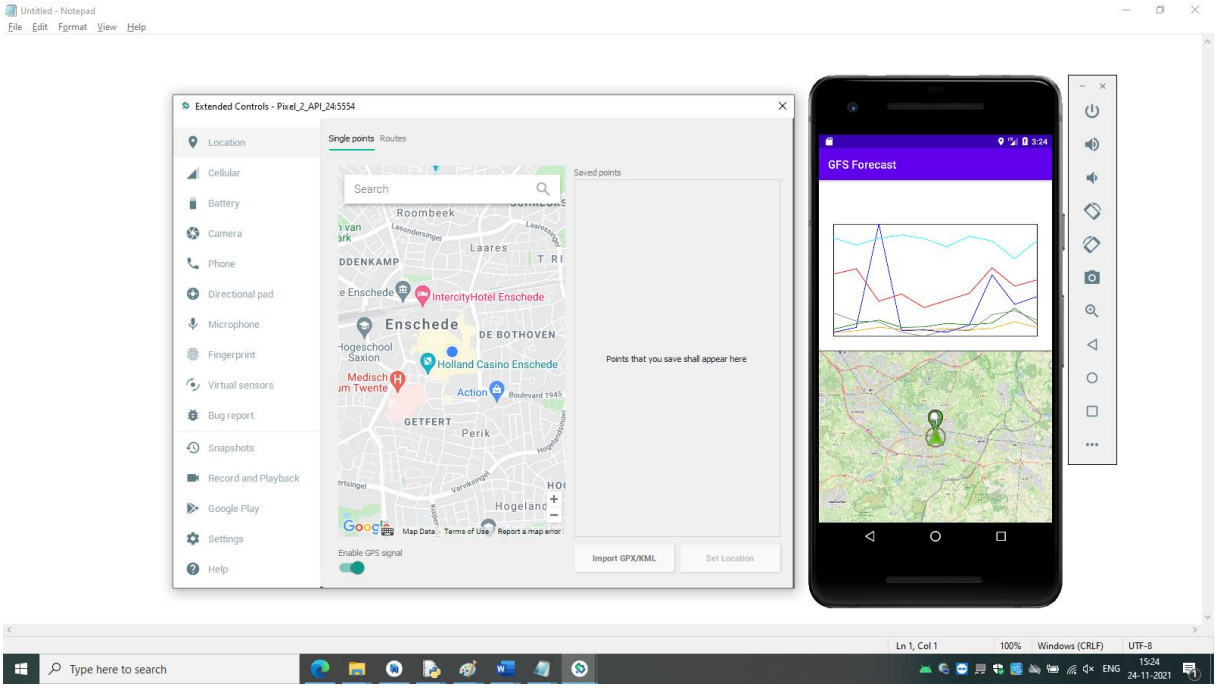


Use the map to find another location.

Go e.g. to Enschede. Then click on the map to place a marker, and click Set Location.



In the app, tap the “find me” () button. The map will now re-center to the new location.



You can move the map around, to see how the graph is re-plotted after every movement. Note that the pixels of this worldwide GFS dataset are quite coarse, namely 0.25 degree, which is roughly 28km, so if you stay in the same pixel, the graph will be the same.

Adding the Graph’s grid, legends and axis labels

To finalize the app, open the file java / com.example.gfsforecast / GraphView.java for editing, and add the functions for the grid, the legends and the axis labels.

Add the following code after the last function which is getFontHeight():

```
private void drawGrid(Canvas canvas) {
    if (values.size() > 0) {
        HashMap<String, Object> parameter = values.get(0);
        Vector<Pair<Double, Date>> series = (Vector<Pair<Double, Date>>)
parameter.get("series");
        if ((series.size() > 1) && (right > left) && (Math.abs(topHalf -
bottom) > 0)) {
            paint.setARGB(255, 231, 231, 231);
            paint.setStrokeWidth(3);
            for (int i = 1; i < series.size() - 1; ++i) {
                float x = series.get(i).second.getTime();
                x -=minDate.getTime();
                x *= (float) (right - left - marginLeft - marginRight) /
(float) (maxDate.getTime() - minDate.getTime());
                x += marginLeft;
                canvas.drawLine(x, topHalf + marginTop, x, bottom -
marginBottom, paint);
            }
            int parts = 5;
            for (float y = (float)minVal + (float)(maxVal - minVal) /
parts; y < maxVal - (float)(maxVal - minVal) / (parts * 2); y = y +
(float)(maxVal - minVal) / parts) {
```

```
                float yy = y -(float) minVal;
                yy *= (float) ((topHalf + marginTop + marginBottom -bottom)
/ (maxVal - minVal));
                yy += bottom - marginBottom;
                canvas.drawLine(marginLeft, yy, right - marginRight, yy,
paint);
            }
        }
    }
}
private float getBaseline(Paint p) {
    Paint.FontMetrics fontMetrics = p.getFontMetrics();
    return (fontMetrics.descent - fontMetrics.ascent) / 2 -
fontMetrics.descent;
}

private void drawLabels(Canvas canvas) {
    if (values.size() > 0) {
        HashMap<String, Object> parameter = values.get(0);
        Vector<Pair<Double, Date>> series = (Vector<Pair<Double, Date>>)
parameter.get("series");
        if ((series.size() > 1) && (right > left) && (Math.abs(topHalf -
bottom) > 0)) {
            paint.setARGB(255, 0, 0, 0);
            paint.setStrokeWidth(1);
            paint.setTextAlign(Paint.Align.CENTER);
            float yoffset = getFontHeight(paint);
            for (int i = 0; i < series.size(); ++i) {
                Date date = series.get(i).second;
                int day = date.getDate();
                float x = date.getTime();
                x -=minDate.getTime();
                x *= (float) (right - left - marginLeft - marginRight) /
(float) (maxDate.getTime() - minDate.getTime());
                x += marginLeft;
                canvas.drawText(String.valueOf(day), x, bottom -
marginBottom + yoffset, paint);
            }
            paint.setTextAlign(Paint.Align.RIGHT);
            float xoffset = yoffset / 4;
            yoffset = getBaseline(paint);
            int parts = 5;
            for (float y = (float)minVal; y < maxVal; y = y +
(float)(maxVal - minVal) / parts) {
                float yy = y -(float) minVal;
                yy *= (float) ((topHalf + marginTop + marginBottom -bottom)
/ (maxVal - minVal));
                yy += bottom - marginBottom;
                canvas.drawText(String.format("%.1f", y), marginLeft -
xoffset, yy + yoffset, paint);
            }
            paint.setTextAlign(Paint.Align.LEFT);
            for (float y = 0; y <= 100; y = y + (float)100 / parts) {
                float yy = y;
                yy *= (float) ((topHalf + marginTop + marginBottom -bottom)
/ 100);
                yy += bottom - marginBottom;
                canvas.drawText(String.format("%.0f", y), right -
marginRight + xoffset, yy + yoffset, paint);
            }
            paint.setTextAlign(Paint.Align.CENTER);
```

```
            yoffset = getFontHeight(paint);
            String labelRight = "";
            String labelLeft = "";
            for (int i = 0; i < values.size(); ++i) {
                parameter = values.get(i);
                String name = (String)parameter.get("parametername");
                String unit = (String)parameter.get("unit");
                boolean left = parameter.get("axisLeft").equals("true");
                if (left) {
                    if (labelLeft.length() > 0)
                        labelLeft += ',';
                    labelLeft += name + '(' + unit + ')';
                } else {
                    if (labelRight.length() > 0)
                        labelRight += ',';
                    labelRight +=name + '(' + unit + ')';
                }
            }
            canvas.rotate(90);
            canvas.drawText(labelRight, bottom - marginBottom - (bottom -
topHalf - marginBottom - marginTop) / 2, marginRight - right - yoffset -
yoffset, paint);
            canvas.rotate(-180);
            canvas.drawText(labelLeft, -bottom + marginBottom + (bottom -
topHalf - marginBottom - marginTop) / 2, marginLeft - yoffset - yoffset,
paint);
            canvas.rotate(90);
        }
    }
}

private void drawLegend(Canvas canvas) {
    float itemWidth = (right - left - marginRight - marginLeft) / 10;
    paint.setStrokeWidth(1);
    paint.setStyle(Paint.Style.FILL_AND_STROKE);
    for (int i = 0; i < values.size(); ++i) {
        HashMap<String, Object> parameter = values.get(i);
        String color = (String) parameter.get("color");
        setARGB(color);
        float posX = (float) (marginLeft + Math.floor(i / 2) * (right -
left - marginRight - marginLeft) / 3);
        float posY = topFull + marginTop + (i % 2) * (topHalf - topFull -
marginTop);
        canvas.drawRect(posX, posY, posX + itemWidth, posY + 8, paint);
    }
    paint.setStyle(Paint.Style.STROKE);
    paint.setARGB(255, 0, 0, 0);
    paint.setTextAlign(Paint.Align.LEFT);
    float yoffset = getBaseline(paint);
    for (int i = 0; i < values.size(); ++i) {
        HashMap<String, Object> parameter = values.get(i);
        String name = (String) parameter.get("parametername");
        String unit = (String) parameter.get("unit");
        String label = name + " (" + unit + ")";
        float posX = (float) (marginLeft + Math.floor(i / 2) * (right -
left - marginRight - marginLeft) / 3);
        float posY = topFull + marginTop + (i % 2) * (topHalf - topFull -
marginTop);
        canvas.drawText(label, posX + itemWidth * 1.1f, posY + yoffset,
paint);
```

```
    }
}
```

Then go to function onDraw() and change it as follows, in order to include drawing the new elements as well:

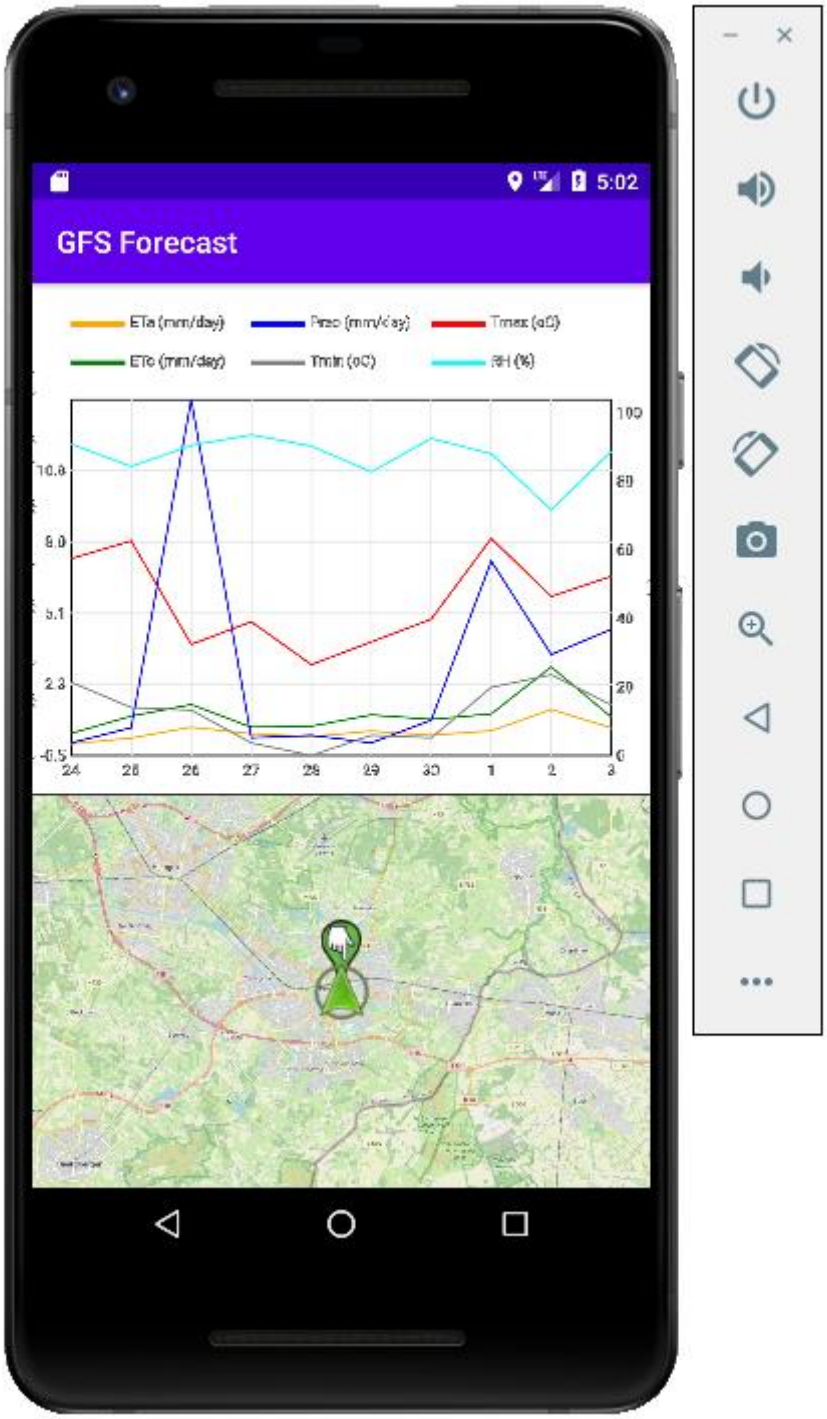
```
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    paint.setAntiAlias(true); // crisp graphs and lines
    paint.setStyle(Paint.Style.FILL);
    paint.setARGB(255, 255, 255, 255);
    canvas.drawRect(left, topFull, right, bottom, paint); // clear the
background
    paint.setStyle(Paint.Style.STROKE);
    if (bottom - marginBottom > topHalf + marginTop) {
        drawGraphRect(canvas);
        drawGrid(canvas);
        drawLabels(canvas);
        drawGraphs(canvas);
    }
    drawLegend(canvas);
}
```

Also add the following line to function onLayout(), as its last line:

```
redrawPaths();
```

This will ensure that the paths are recomputed (re-scaled) and redrawn when the user manipulates the divider.

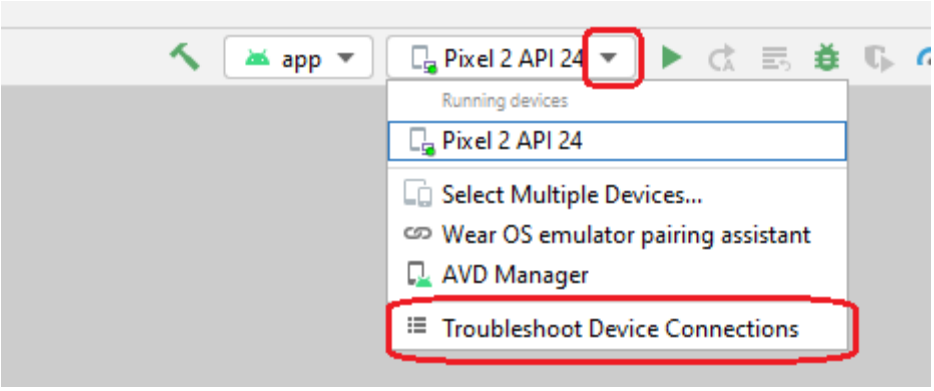
Run the app (Run -> Run ‘app’):



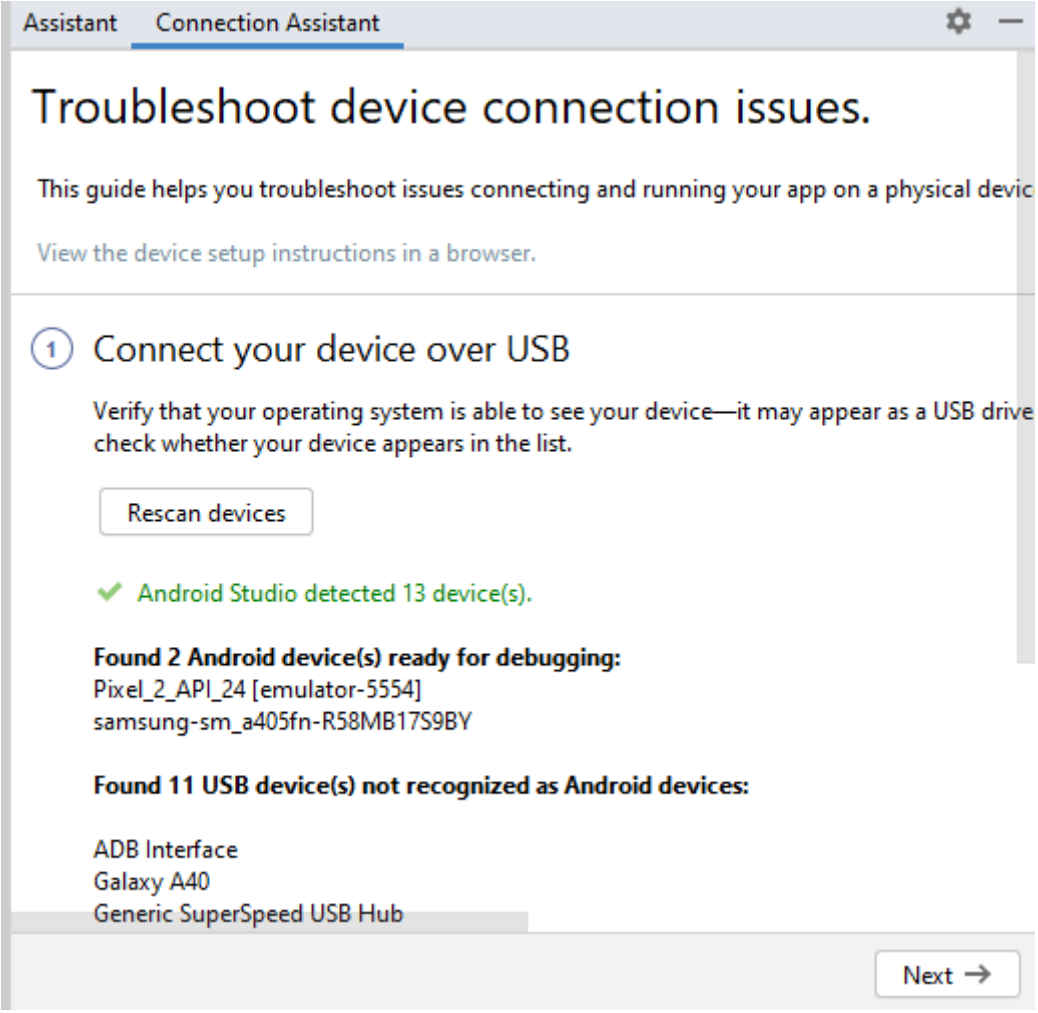
Install the app on your own phone

For this you will need to connect your phone with the laptop, with the USB cable that came with your phone, and you have to enable developer options and USB debugging in the phone settings.

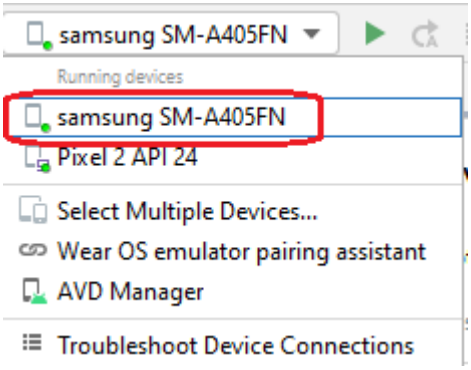
Follow the instructions that come when you click on Troubleshooting Device Connections.



You will see your phone on the list, and the instructions that you need to follow, to allow Android Studio to communicate with your phone.



When succeeding, your phone will be listed as a selectable target device. You can then run the app on the phone. Running will install a copy of the app on the phone, so it will be available even after you disconnect the USB cable.



Questions

What improvements or beautifications can you think of?