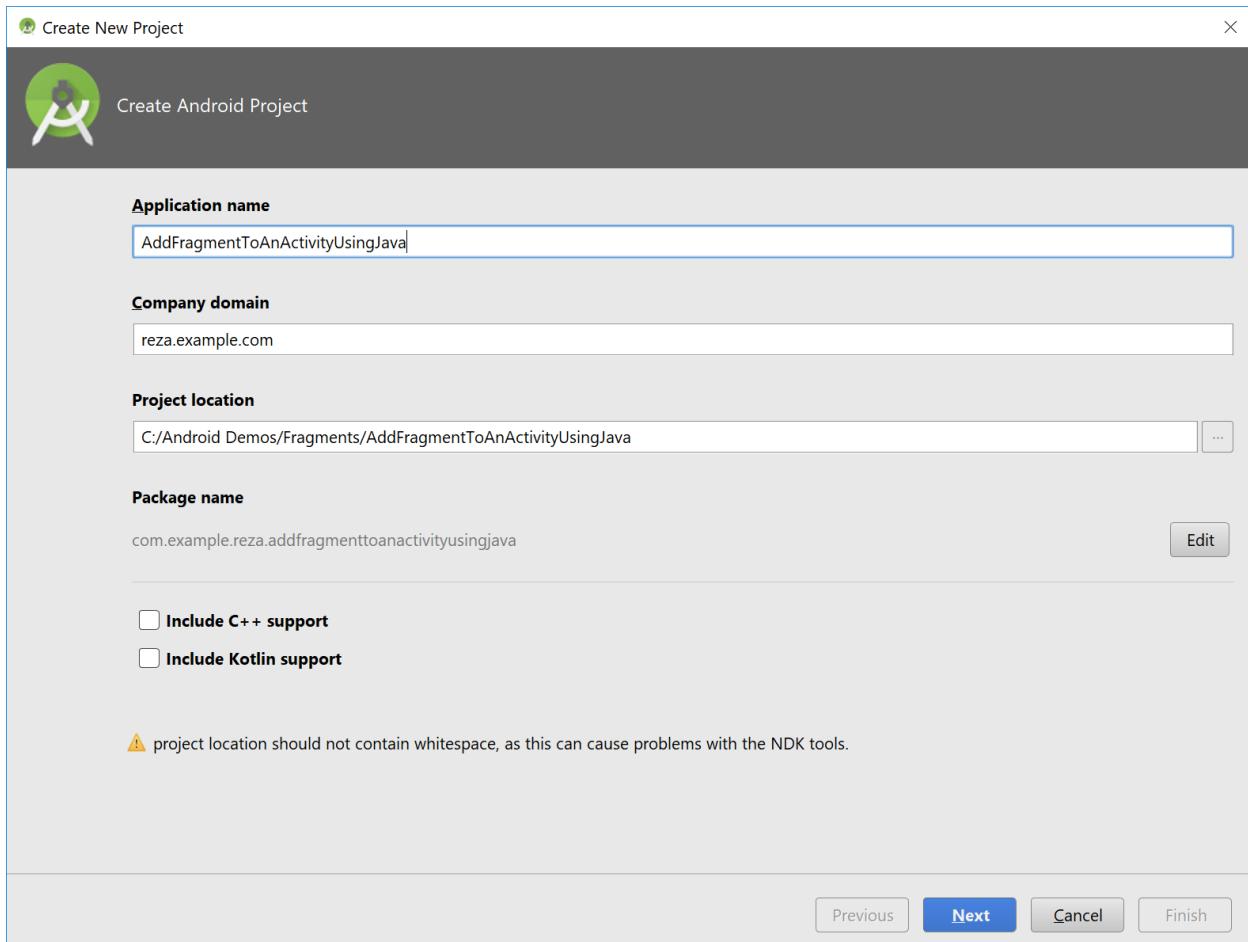


## Adding a Fragment to an Activity using Java (programmatically)

Create a new project as shown:



Create New Project X

## Target Android Devices

Select the form factors and minimum SDK

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

**Phone and Tablet**

API 15: Android 4.0.3 (IceCreamSandwich) ▼

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

Include Android Instant App support

**Wear**

API 21: Android 5.0 (Lollipop) ▼

**TV**

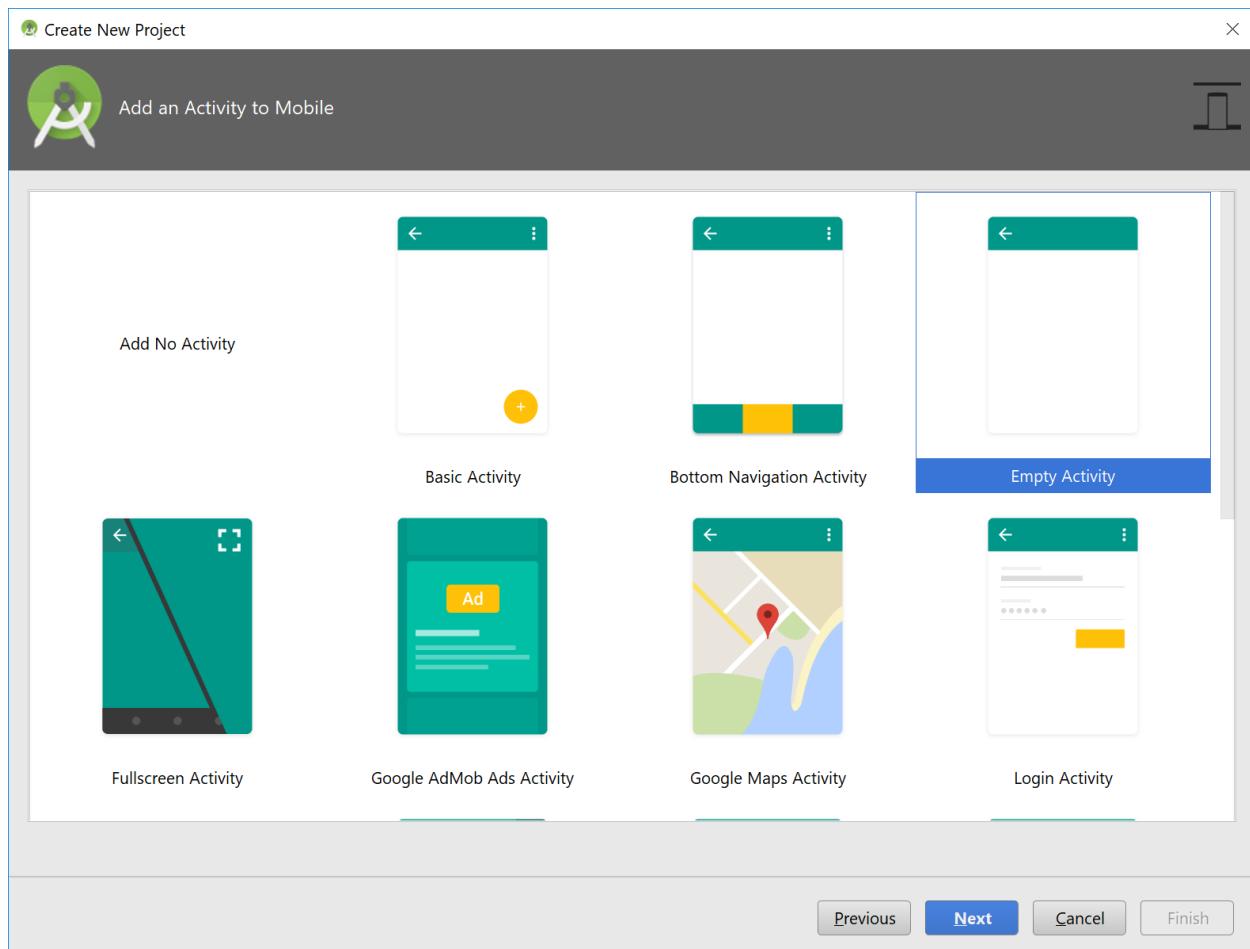
API 21: Android 5.0 (Lollipop) ▼

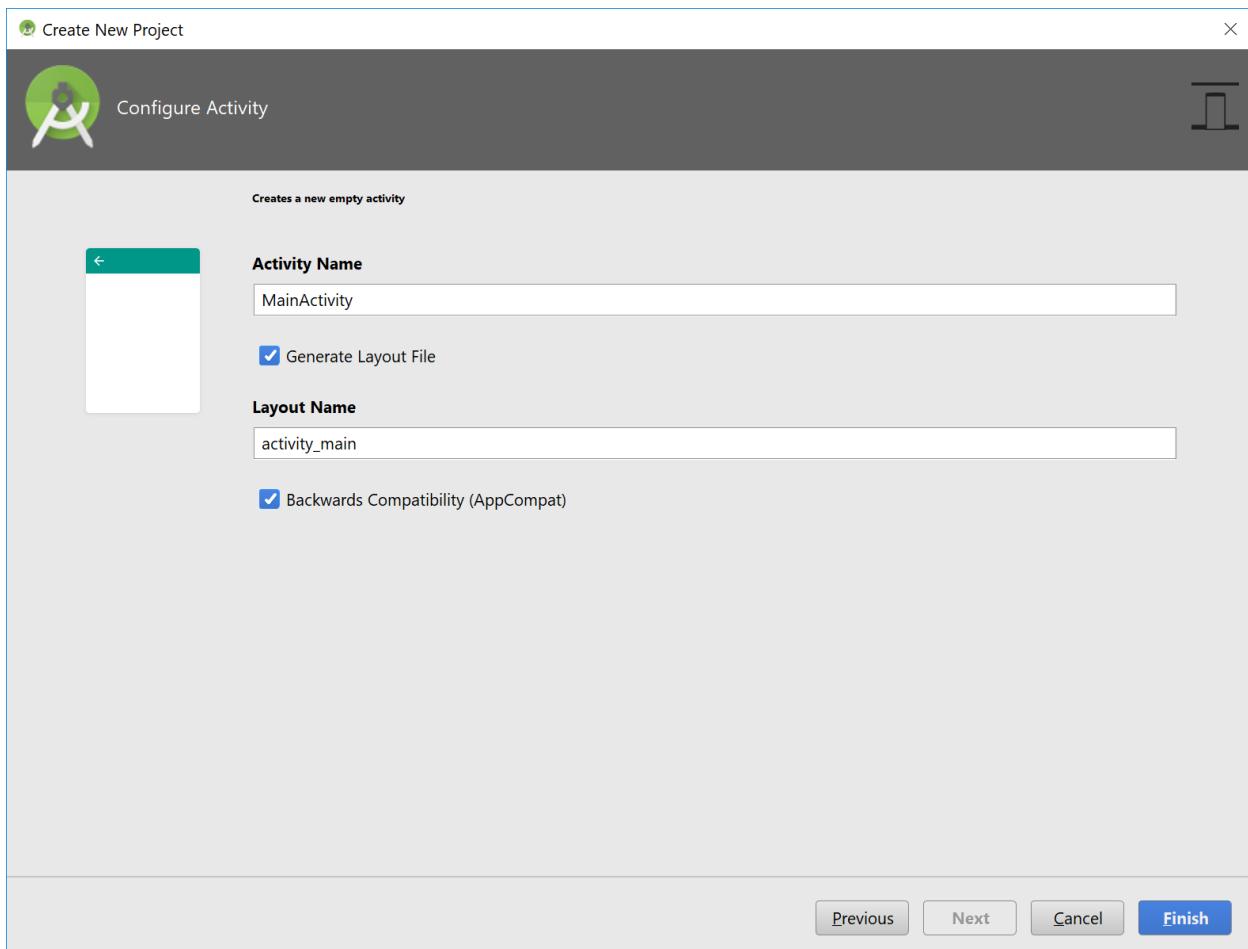
**Android Auto**

**Android Things**

API 24: Android 7.0 (Nougat) ▼

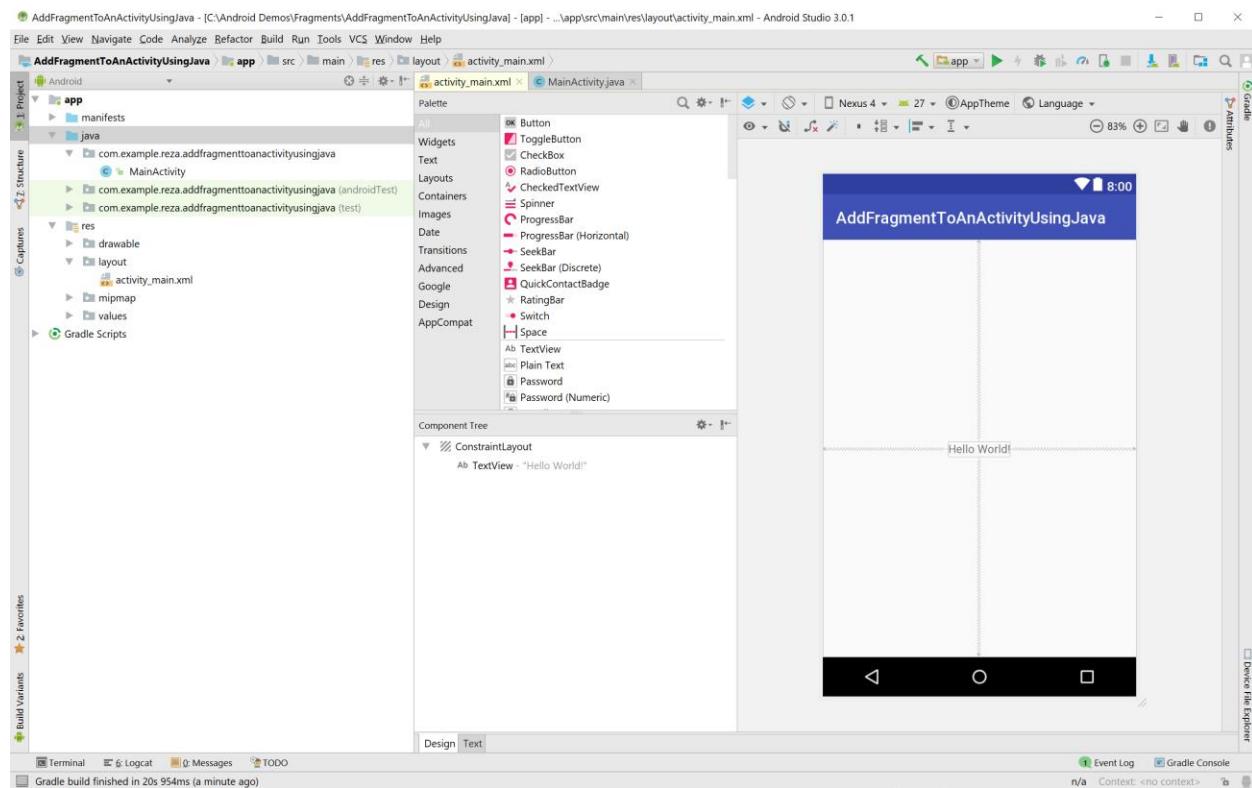
[Previous](#) **Next** [Cancel](#) [Finish](#)





The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it, the main workspace displays the `MainActivity.java` file under the `app` module. The code defines a simple `MainActivity` that extends `AppCompatActivity` and overrides the `onCreate` method to set the content view to `R.layout.activity_main`.

```
package com.example.reza.addfragmenttoanactivityusingjava;
import ...
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```



FragmentToAnActivityUsingJava - [C:\Android Demos\Fragments\AddFragmentToAnActivityUsingJava] - [app] - ...\\app\\src\\main\\res\\layout\\activity\_main.xml - Android Studio 3.0.1

View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

FragmentToAnActivityUsingJava app src main res layout activity\_main.xml

activity\_main.xml MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="com.example.reza.addfragmenttoanactivityusingjava.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" />

</android.support.constraint.ConstraintLayout>
```

Design Text

Terminal Logcat Messages TODO

Event Log Gradle Console

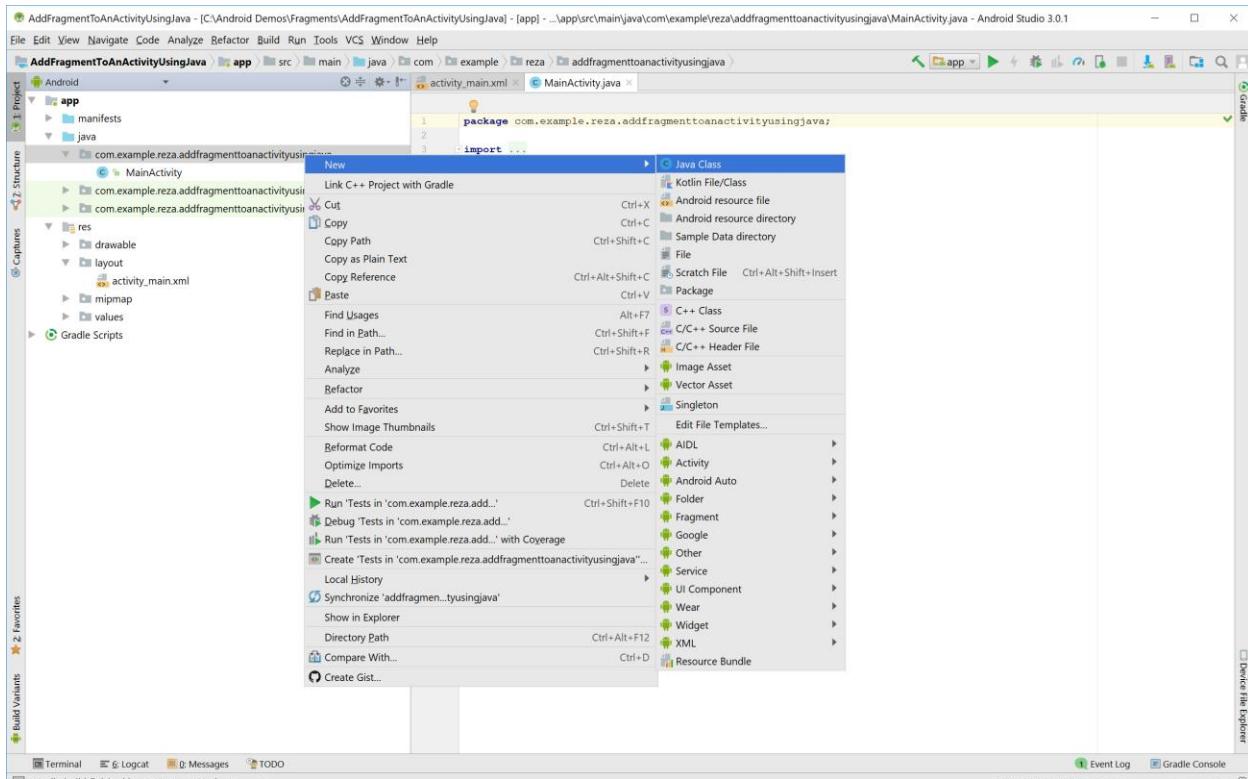
File build finished in 20s 954ms (9 minutes ago)

Change the layout file (activity\_main.xml) as shown:

The screenshot shows the Android Studio interface with the project 'FragmentToAnActivityUsingJava' open. The layout editor is displayed, showing the XML code for 'activity\_main.xml'. The code defines a single `RelativeLayout` with attributes `layout_width="match_parent"` and `layout_height="match_parent"`. The tools context is set to `com.example.reza.addfragmenttoanactivityusingjava.MainActivity`.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.reza.addfragmenttoanactivityusingjava.MainActivity">
</RelativeLayout>
```

Now you need to create your fragment which is a subclass of Fragment class. So create a new Java class:



 Create New Class X

Name:

Kind:  Class ▼

Superclass:

Interface(s):

Package: com.example.reza.addfragmenttoanactivityusingjava

Visibility:  Public  Package Private

Modifiers:  None  Abstract  Final

Show Select Overrides Dialog

OK Cancel Help

 Create New Class X

Name:

Kind:  Class ▼

Superclass:

Interface(s):

Package: com.example.reza.addfragmenttoanactivityusingjava

Visibility:  Public  Package Private

Modifiers:  None  Abstract  Final

Show Select Overrides Dialog

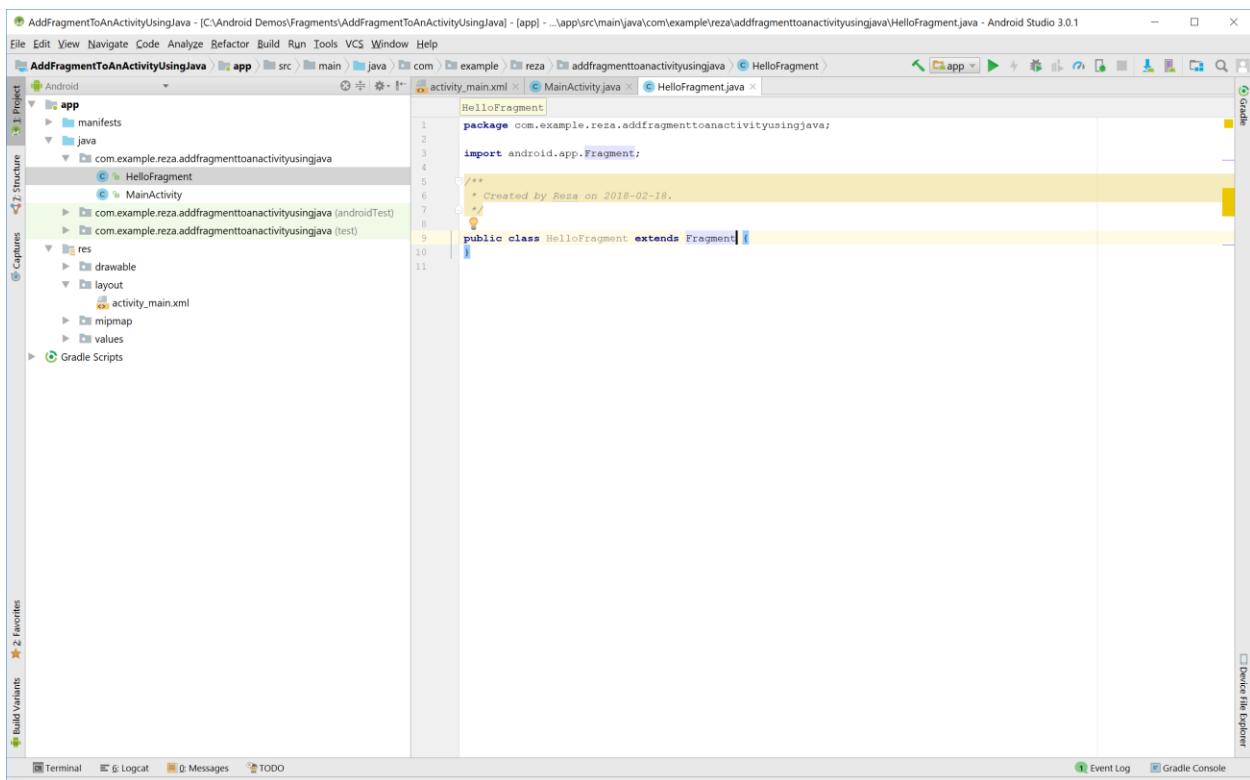
OK Cancel Help

The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The left sidebar displays the project structure under the 'app' module. The main editor window shows the 'HelloFragment.java' file:

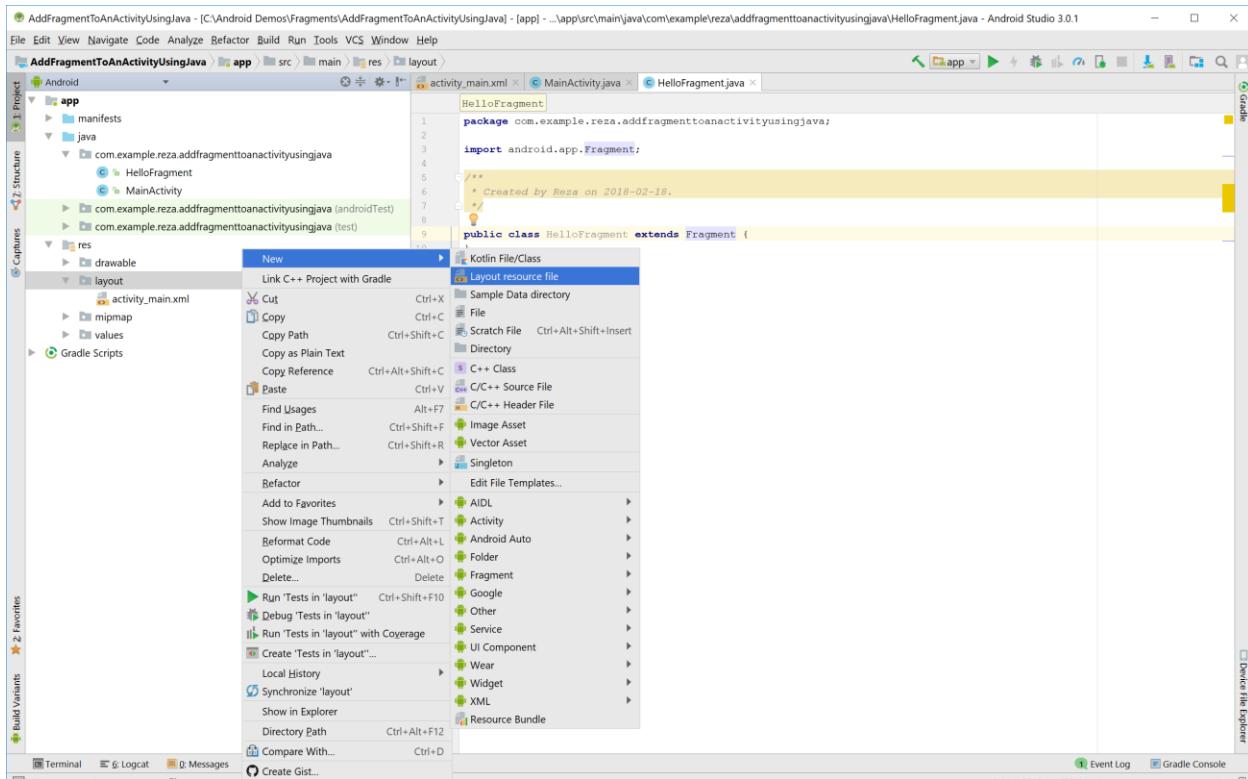
```
1 package com.example.reza.addfragmenttoanactivityusingjava;
2
3 /**
4 * Created by Reza on 2018-02-18.
5 */
6
7 public class HelloFragment {
```

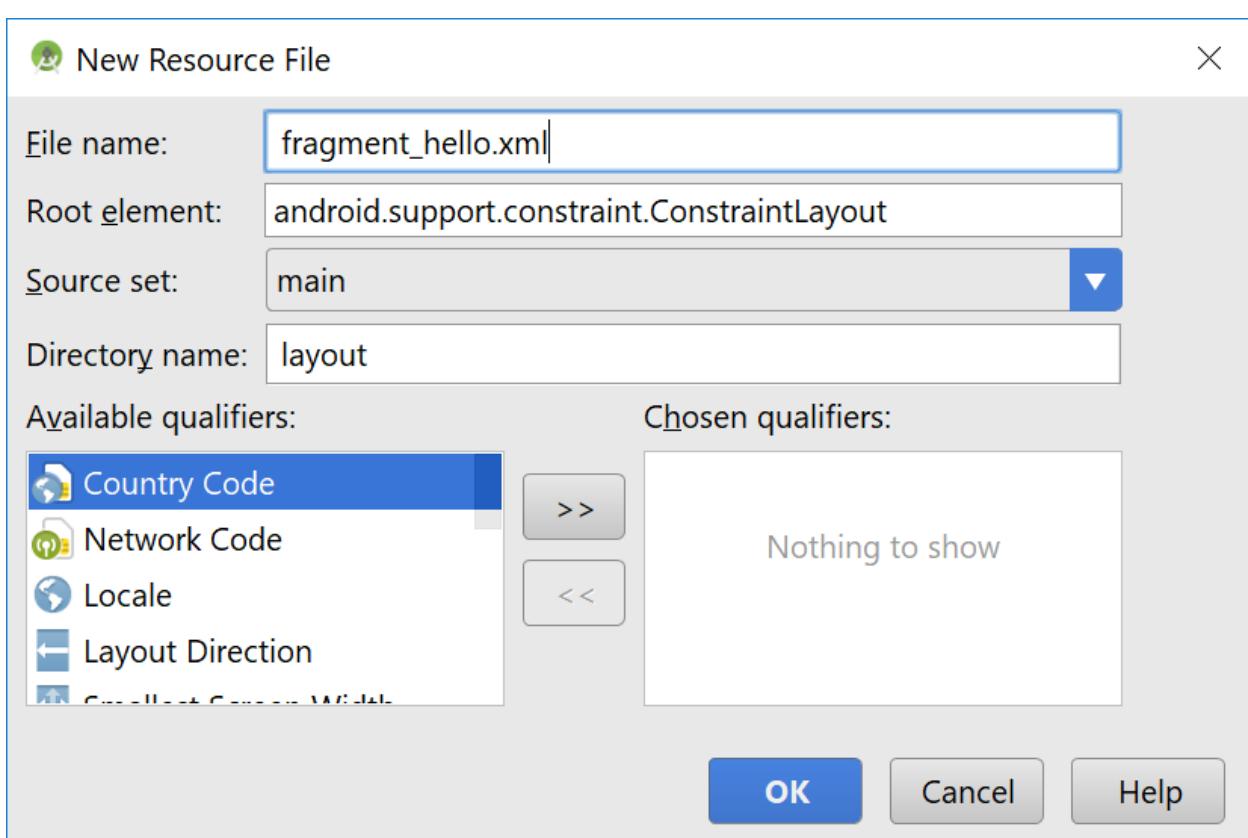
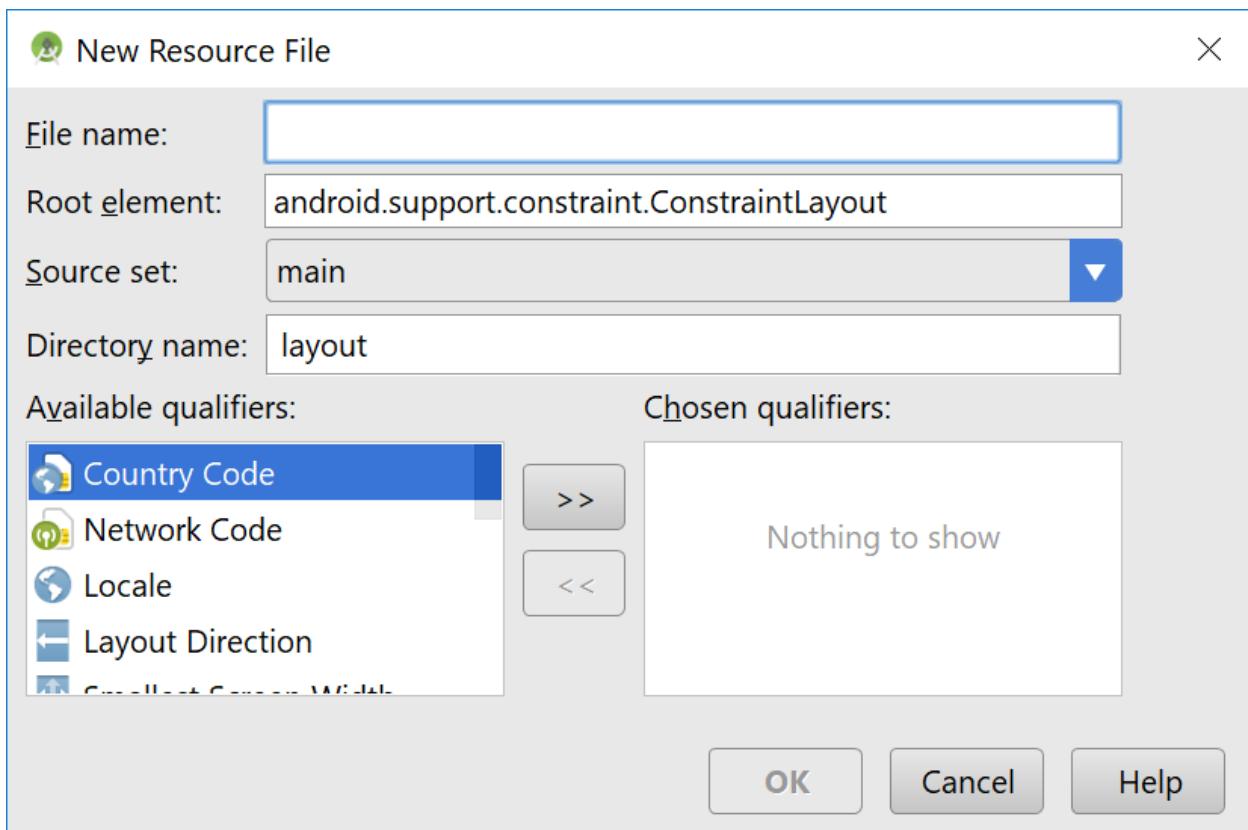
The screenshot shows the same Android Studio interface, but the cursor is hovering over the 'extends Fragment' part of the code. A dropdown menu is open, listing various fragment-related classes from the 'android.app' package:

- Fragment (android.app)
- Fragment (android.support.v4.app)
- FragmentContainer (android.app)
- FragmentController (android.app)
- FragmentHostCallback<E> (android.app)
- FragmentLifecycleCallbacks (android.app.FragmentManager)
- FragmentManager (android.app)
- FragmentManagerNonConfig (android.app)
- FragmentManager (android.support.v4.app)
- FragmentTransaction (android.app)

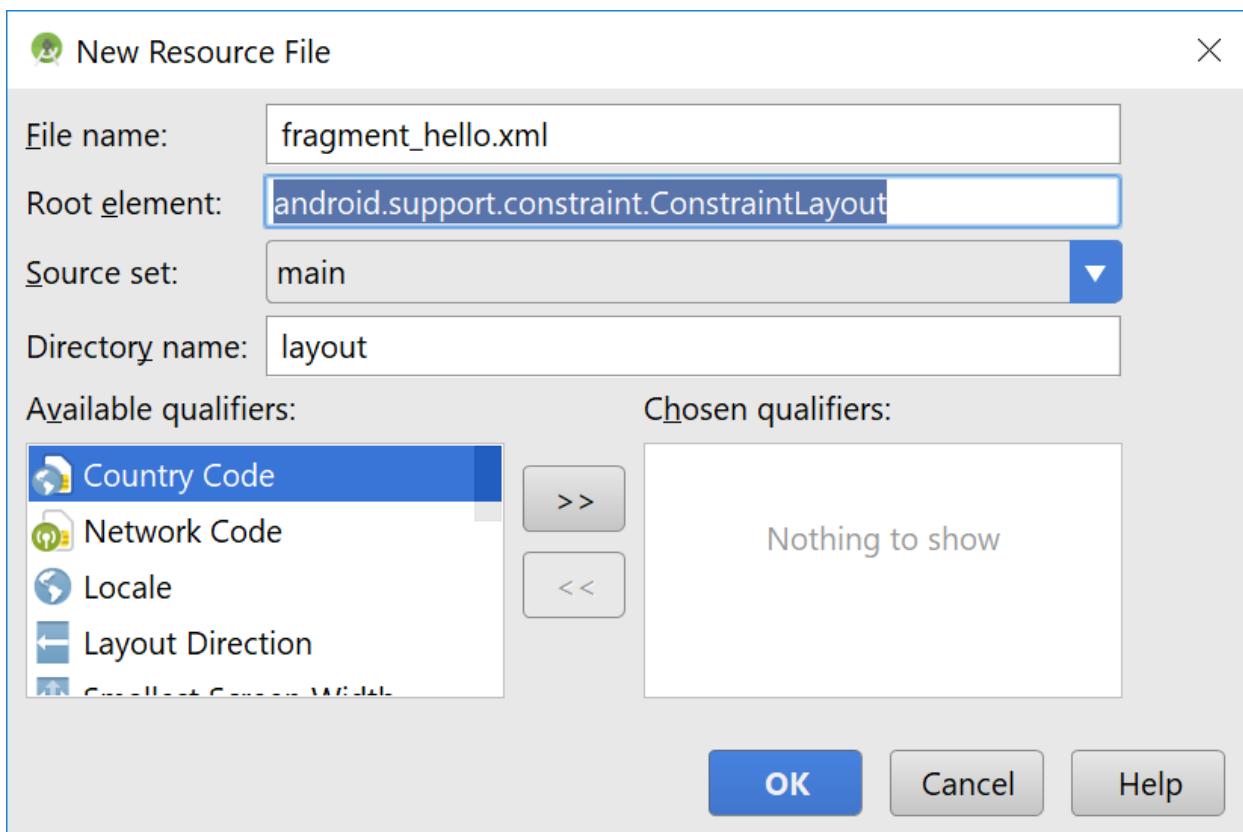


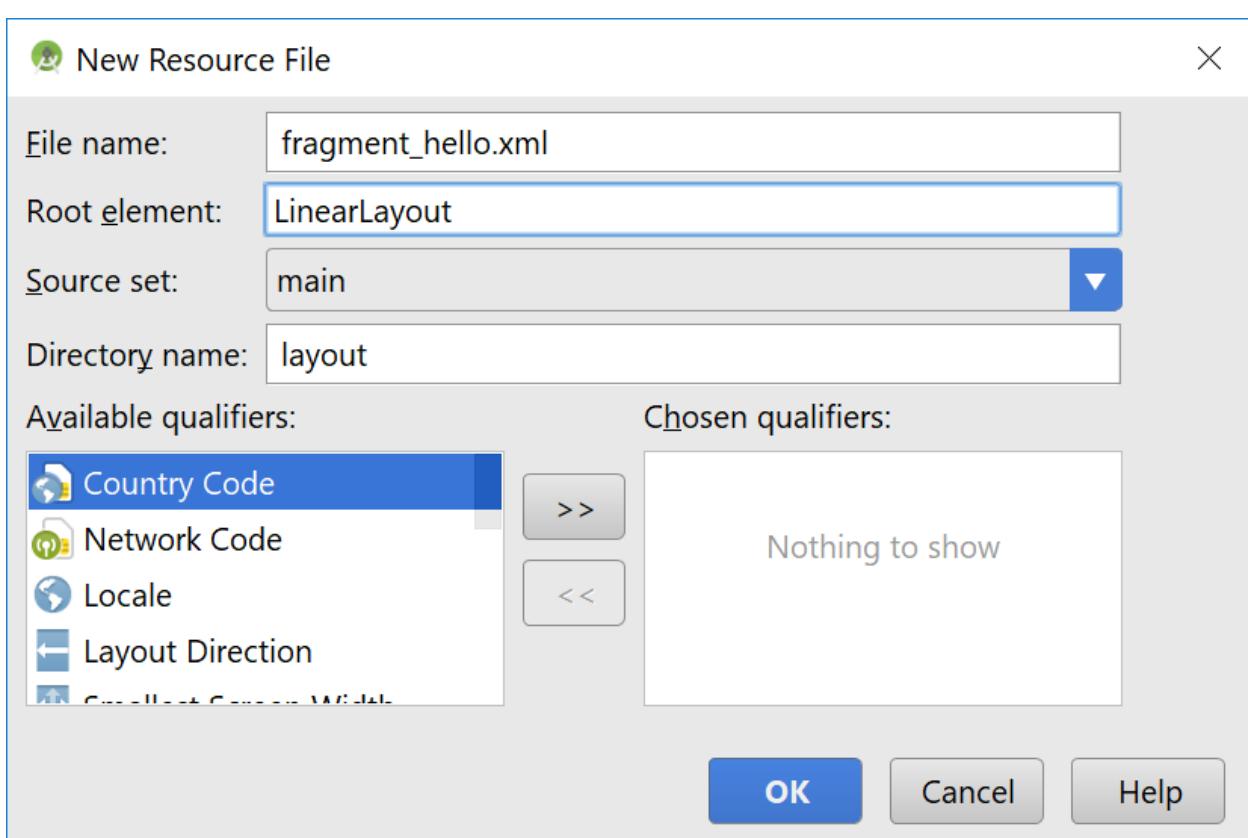
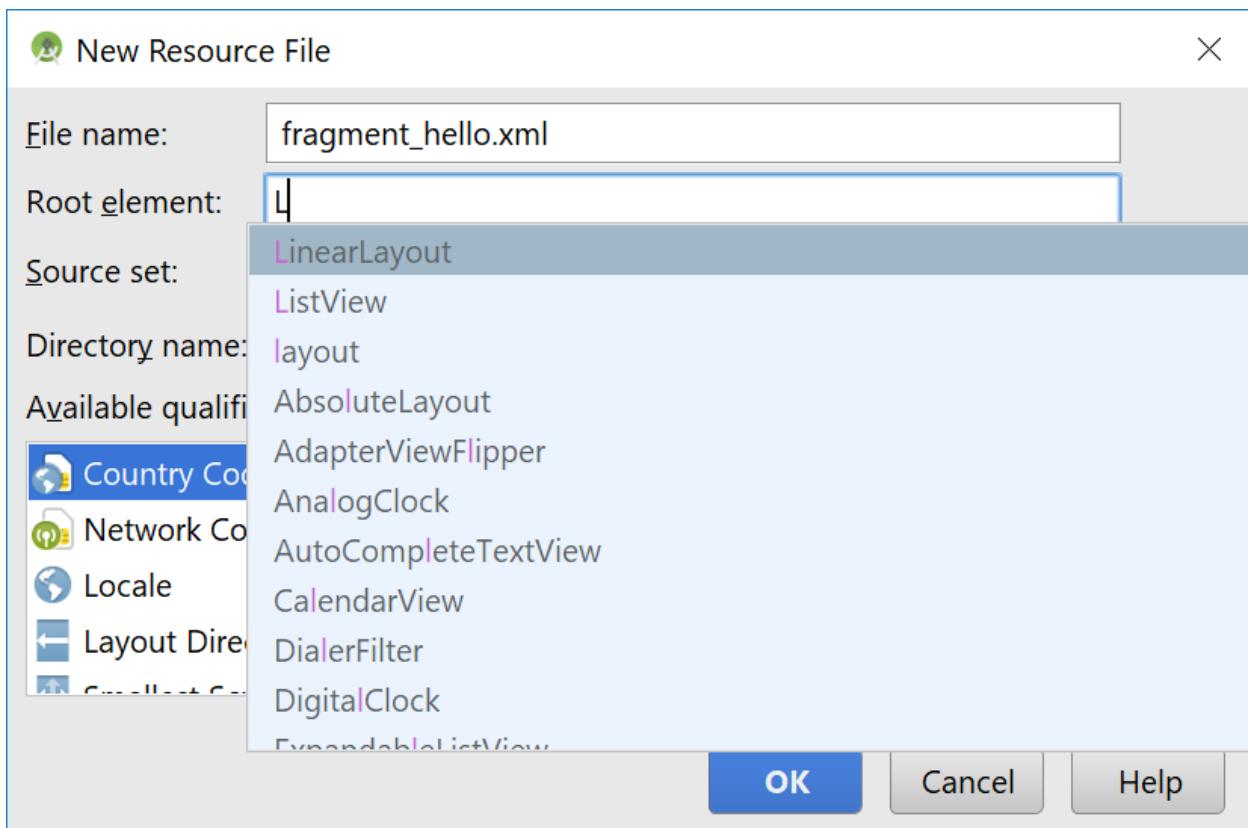
Next create a layout for your fragment class as shown:

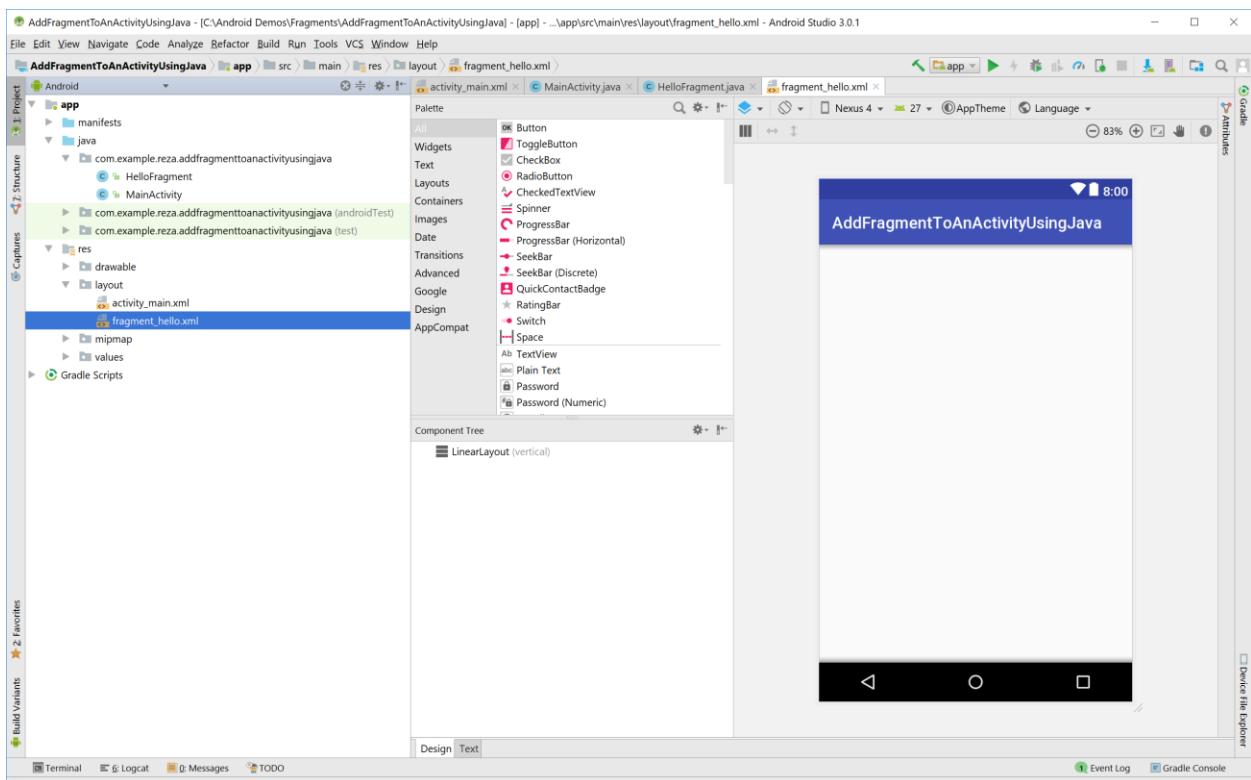




We are going to change the root layout to a LinearLayout as shown:







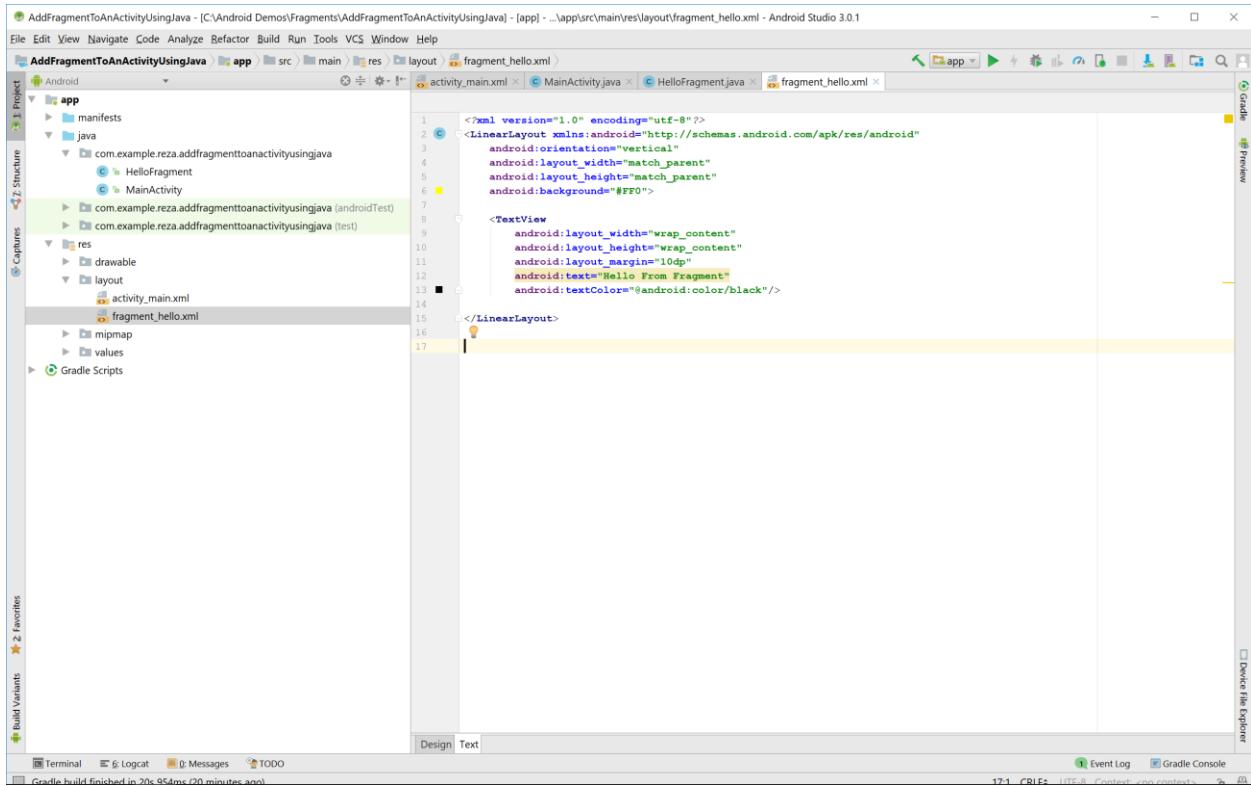
## Here is the layout:

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows the project name "AddFragmentToAnActivityUsingJava" and the file path "...app/src/main/res/layout/fragment\_hello.xml".
- File Menu:** Includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Toolbar:** Standard Android Studio toolbar with icons for Open, Save, Run, Stop, etc.
- Project Structure:** Shows the project tree under "app":
  - manifests
  - java
    - com.example.reza.addfragmenttoanactivityusingjava
      - HelloFragment
      - MainActivity
    - com.example.reza.addfragmenttoanactivityusingjava (androidTest)
    - com.example.reza.addfragmenttoanactivityusingjava (test)
  - res
    - drawable
    - layout
      - activity\_main.xml
      - fragment\_hello.xml
    - mipmap
    - values
  - Gradle Scripts
- Code Editor:** Displays the XML code for "fragment\_hello.xml".

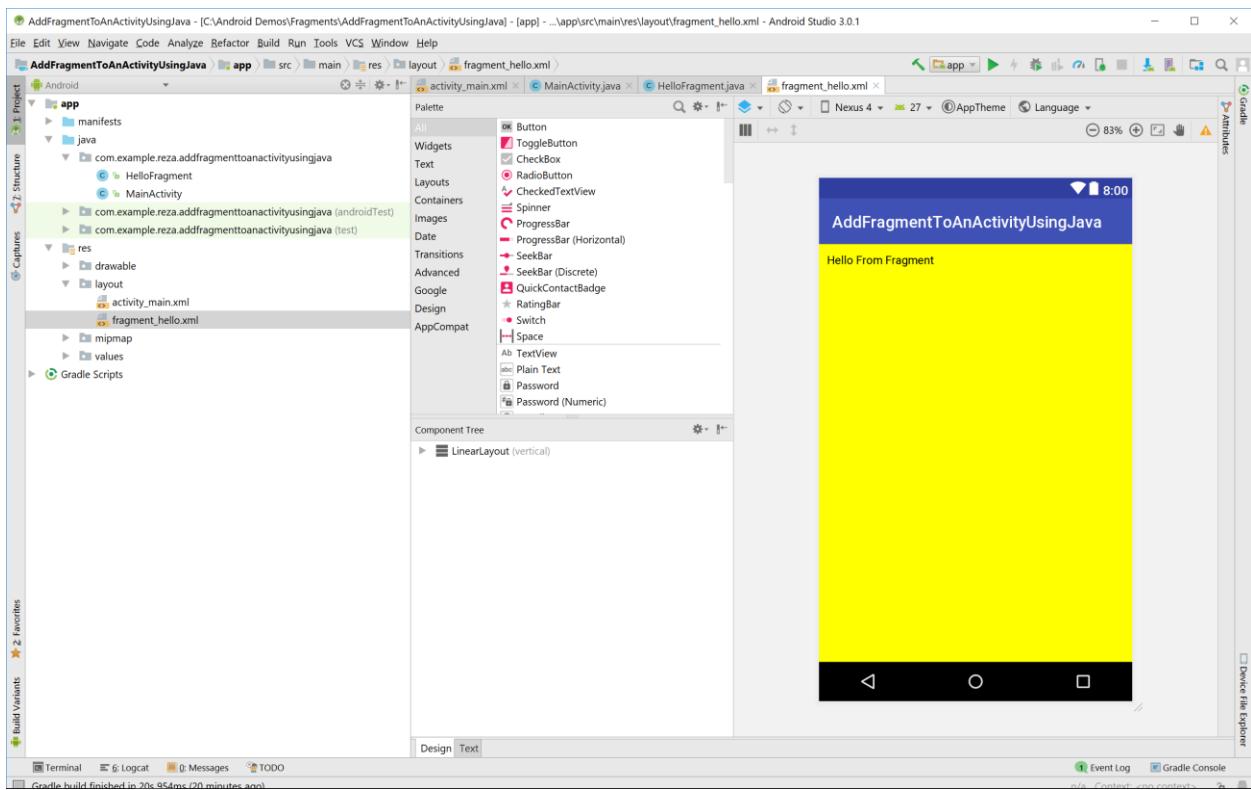
```
<?xml version="1.0" encoding="utf-8"?
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
</LinearLayout>
```
- Toolbars:** Design and Text tabs at the bottom of the code editor.
- Bottom Navigation:** Terminal, Logcat, Messages, TODO, Event Log, Gradle Console.

Add an element and set the background color so it is easy to see the fragment when displayed in the activity:

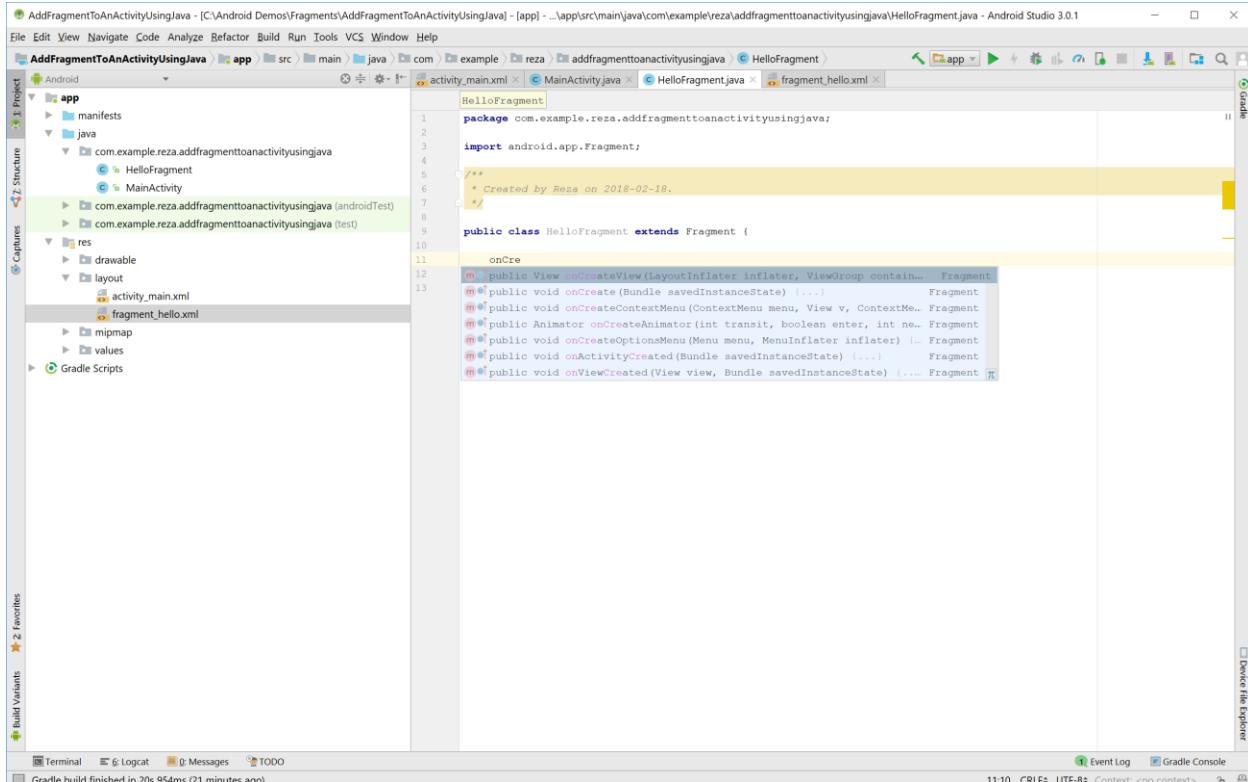


The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The code editor displays the XML layout for a fragment named 'fragment\_hello.xml'. The XML code includes a LinearLayout with a white background and a TextView with black text and a yellow background. The code is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FF0000">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Hello From Fragment"
        android:textColor="@android:color/black"/>
</LinearLayout>
```



Now go back to HelloFragment.java code and use the onCreate () method to create the connection between the java code and its layout file.



```
1 package com.example.reza.addfragmenttoanactivityusingjava;
2
3 import android.app.Fragment;
4
5 /**
6  * Created by Reza on 2018-02-18.
7 */
8
9 public class HelloFragment extends Fragment {
10
11     @Override
12     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
13         return inflater.inflate(R.layout.activity_main, container, false);
14     }
15 }
```

You need to replace the highlighted code as shown:

```

HelloFragment onCreateView()
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

```

```

public class HelloFragment extends Fragment {
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
        return super.onCreateView(inflater, container, savedInstanceState);
    }
}

```

```

HelloFragment onCreateView()
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

```

```

public class HelloFragment extends Fragment {
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_hello, container, attachToRoot false);
        //Here you can write your own code to initialize any views or widgets you might have in your fragment
        return view;
    }
}

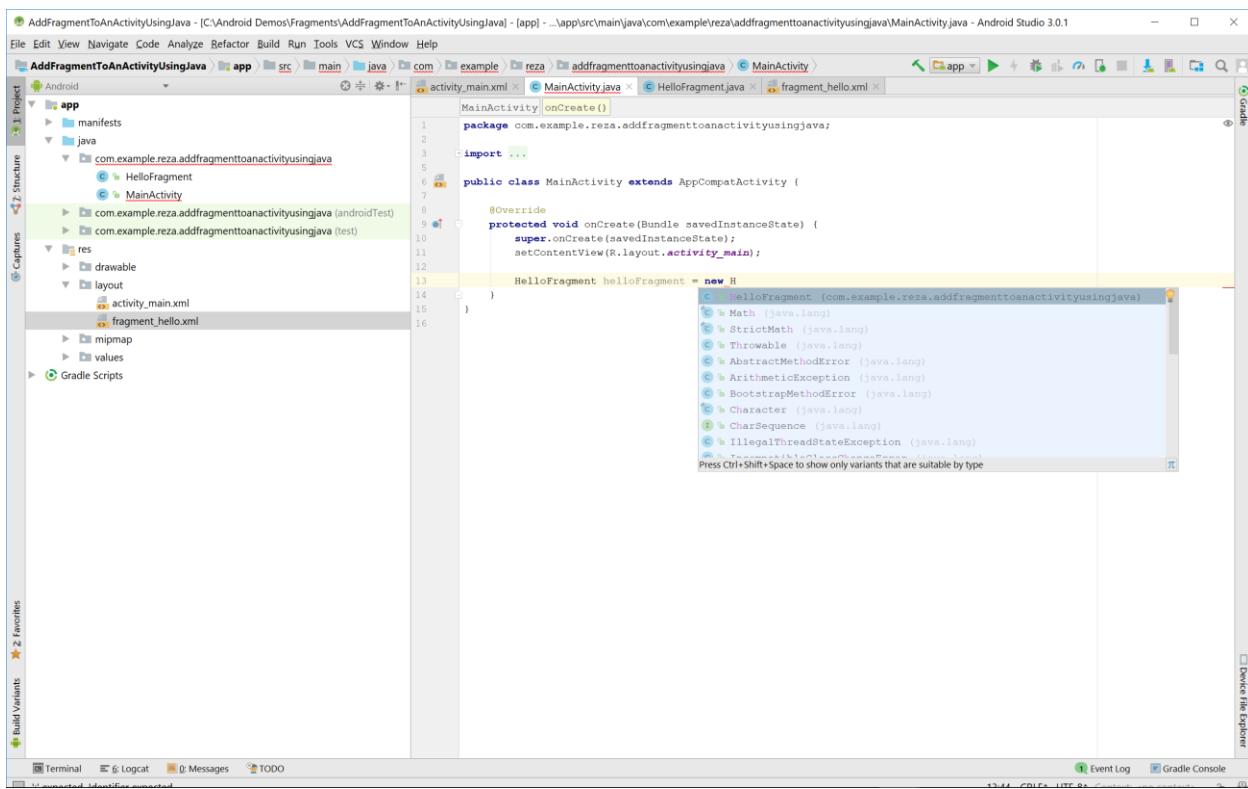
```

In the comment section you can write the code to initialize any other views you might have in your fragment.

To this point everything we did was identical with the steps we took when we created and added a fragment to the activity by dragging it into the user interface of the main activity.

However in this new method we are going to add this fragment to our activity programmatically so the steps from this point forward are different.

Go to MainActivity.java and add the following:



The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The code editor displays the MainActivity.java file, which contains the following Java code:

```
1 package com.example.reza.addfragmenttoanactivityusingjava;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11
12        HelloFragment helloFragment = new H
13            .elloFragment (com.example.reza.addfragmenttoanactivityusingjava)
14        }
15    }
16}
```

A code completion dropdown is visible at the bottom of the code editor, listing various Java classes and methods under the 'HelloFragment' variable. The 'Device File Explorer' tab is visible on the right side of the interface.

The screenshot shows the Android Studio interface with the code editor open. The code being typed is:

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        HelloFragment helloFragment = new HelloFragment();
        FragmentM
```

The cursor is at the end of "FragmentManager", and a dropdown menu is open, listing several options starting with "FragmentManager".

The screenshot shows the Android Studio interface with the code editor open. The code being typed is:

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        HelloFragment helloFragment = new HelloFragment();
        FragmentManager manager = getF
```

The cursor is at the end of "FragmentManager", and a dropdown menu is open, listing several options starting with "FragmentManager".

```

AddFragmentToAnActivityUsingJava - [C:\Android Demos\Fragments]AddFragmentToAnActivityUsingJava - [app] - ...\\app\\src\\main\\java\\com\\example\\reza\\addfragmenttoanactivityusingjava\\MainActivity.java - Android Studio 3.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
AddFragmentToAnActivityUsingJava app src main java com example reza addfragmenttoanactivityusingjava MainActivity
activity_main.xml MainActivity.java HelloFragment.java fragment_hello.xml
MainActivity.onCreate()
1 package com.example.reza.addfragmenttoanactivityusingjava;
2
3 import android.app.FragmentManager;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13
14        HelloFragment helloFragment = new HelloFragment();
15        FragmentManager manager = getFragmentManager();
16        FragmentTransaction transaction = manager.beginTransaction();
17
18        transaction.add(R.id.fragment_hello, helloFragment);
19
}

```

```

AddFragmentToAnActivityUsingJava - [C:\Android Demos\Fragments]AddFragmentToAnActivityUsingJava - [app] - ...\\app\\src\\main\\java\\com\\example\\reza\\addfragmenttoanactivityusingjava\\MainActivity.java - Android Studio 3.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
AddFragmentToAnActivityUsingJava app src main java com example reza addfragmenttoanactivityusingjava MainActivity
activity_main.xml MainActivity.java HelloFragment.java fragment_hello.xml
MainActivity.onCreate()
1 package com.example.reza.addfragmenttoanactivityusingjava;
2
3 import android.app.FragmentManager;
4 import android.app.FragmentTransaction;
5 import android.support.v7.app.AppCompatActivity;
6 import android.os.Bundle;
7
8 public class MainActivity extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_main);
14
15        HelloFragment helloFragment = new HelloFragment();
16        FragmentManager manager = getFragmentManager();
17        FragmentTransaction transaction = manager.beginTransaction();
18        transaction.
19
}

```

The screenshot shows a code completion dropdown for the `transaction.` prefix. The dropdown lists several methods:

- `add(int id, Fragment fragment)`
- `add(Fragment fragment, String s)`
- `add(int id, Fragment fragment, String s)`
- `addSharedElement(View view, String s)`
- `addToBackStack(String s)`
- `attach(Fragment fragment)`
- `commit()`
- `commitAllowingStateLoss()`
- `commitNow()`
- `commitNowAllowingStateLoss()`

Android Studio 3.0.1 interface showing the Java code for `MainActivity`. The code is as follows:

```
package com.example.reza.addfragmenttoanactivityusingjava;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
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);

        HelloFragment helloFragment = new HelloFragment();
        FragmentManager manager = getFragmentManager();
        FragmentTransaction transaction = manager.beginTransaction();
        transaction.add(HelloFragment.class, helloFragment);
        transaction.commit();
    }
}
```

The cursor is at the end of the line `transaction.add(HelloFragment.class, helloFragment);`. A code completion dropdown is open, listing several methods of the `FragmentTransaction` class, including `add(int, Fragment)`, `add(Fragment, String)`, and `add(int, Fragment, String)`.

Android Studio 3.0.1 interface showing the Java code for `MainActivity`. The code is as follows:

```
package com.example.reza.addfragmenttoanactivityusingjava;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

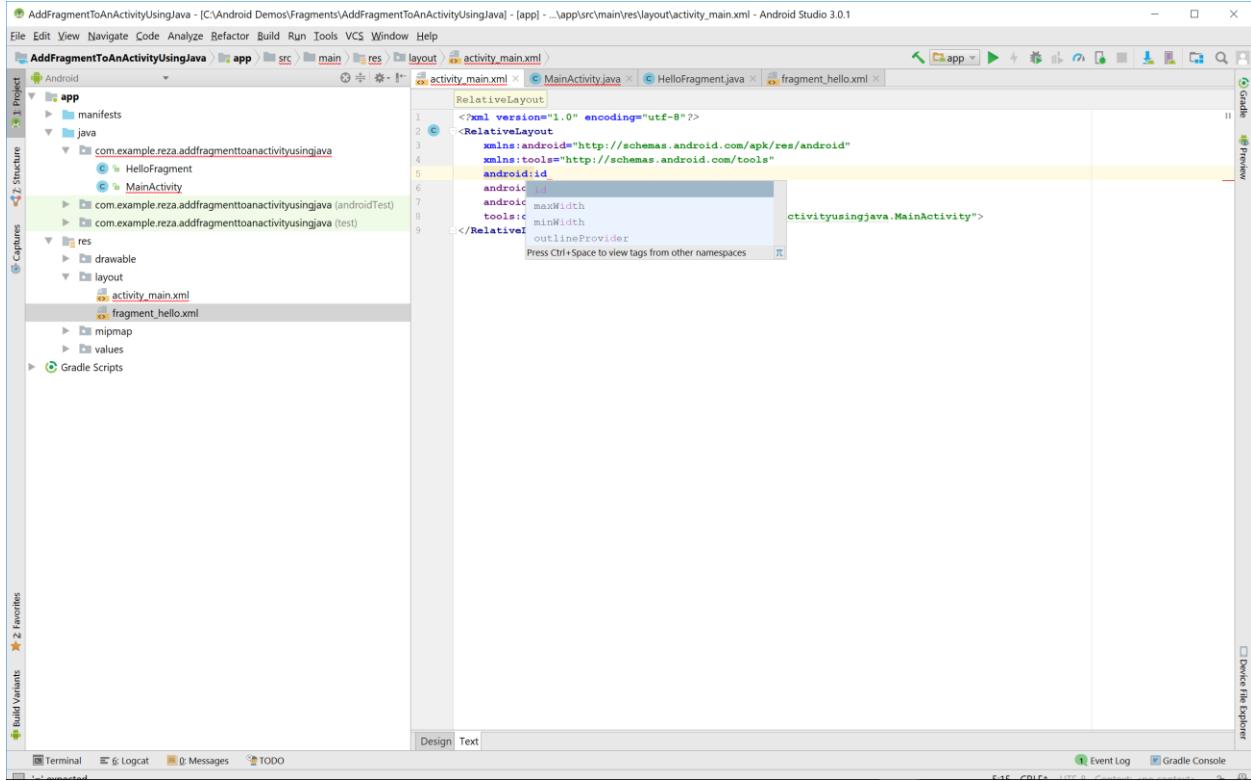
public class MainActivity extends AppCompatActivity {

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

        HelloFragment helloFragment = new HelloFragment();
        FragmentManager manager = getFragmentManager();
        FragmentTransaction transaction = manager.beginTransaction();
        transaction.add();
    }
}
```

The cursor is at the end of the line `transaction.add();`. A code completion dropdown is open, listing parameters for the `add` method, specifically `Fragment fragment, String s`, `@IdRes int i, Fragment fragment`, and `@IdRes int i, Fragment fragment, String s`.

We need to give an id to our container (which is our RelativeLayout here) since we need to provide that as an argument to method add so go back to activity\_main.xml and give your RelativeLayout an id as shown:



The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The 'activity\_main.xml' file is selected in the layout editor. The XML code is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.reza.addfragmenttoanactivityusingjava.MainActivity">
```

The line 'android:id="@+id/activity\_main"' is highlighted in yellow, indicating it has been selected or is being edited. The code editor has tabs for 'Design' and 'Text'. At the bottom, there are tabs for 'Terminal', 'Logcat', 'Messages', 'TODO', 'Event Log', and 'Gradle Console'.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id = "@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.reza.addfragmenttoanactivityusingjava.MainActivity">

```

Now go back to your MainActivity.java and complete the call to the add method:

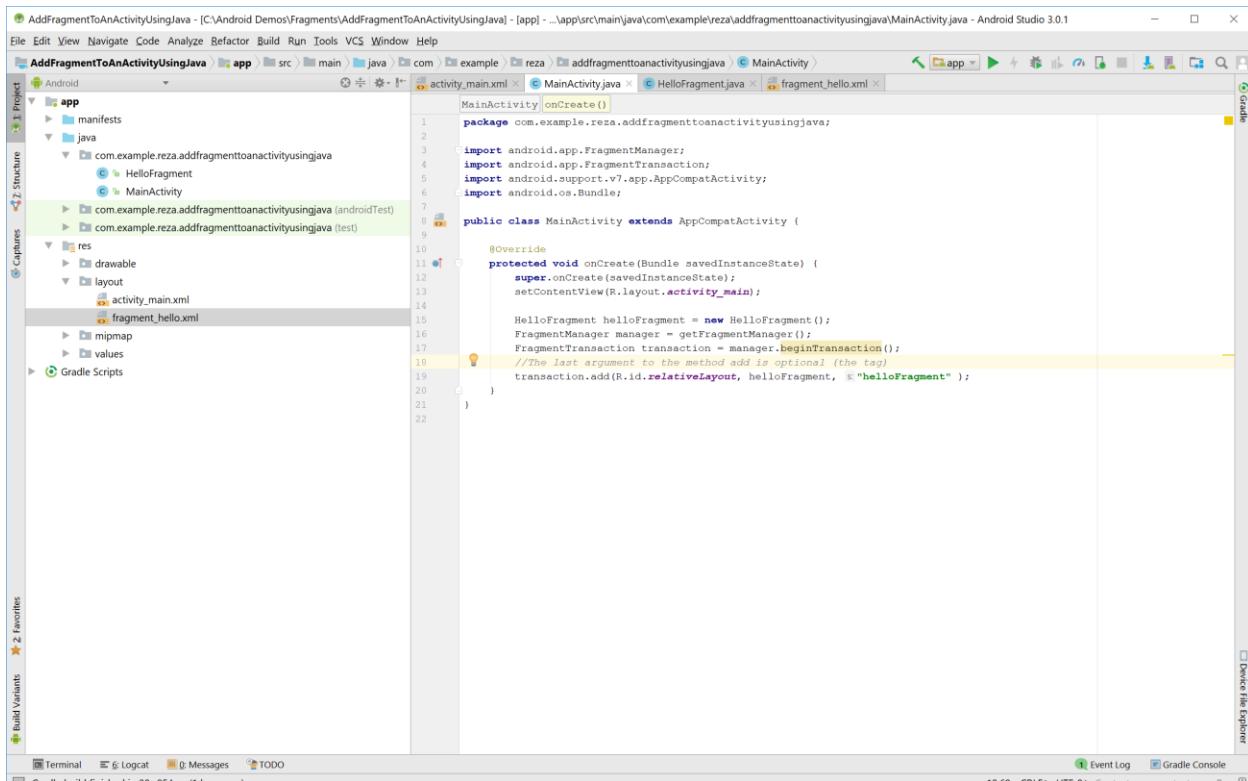
```

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        HelloFragment helloFragment = new HelloFragment();
        FragmentManager manager = getSupportFragmentManager();
        FragmentTransaction transaction = manager.beginTransaction();
        transaction.add(R.id.fragment_hello, helloFragment);
    }
}

```

Here is the completed call to the method add with the last optional argument. We provided the arbitrary string of “helloFragment” here. So with calling the add method, we are asking that fragment “helloFragment” to be added to the container with the id “relativeLayout” which is the layout of our main activity so we are adding our fragment to the main activity.



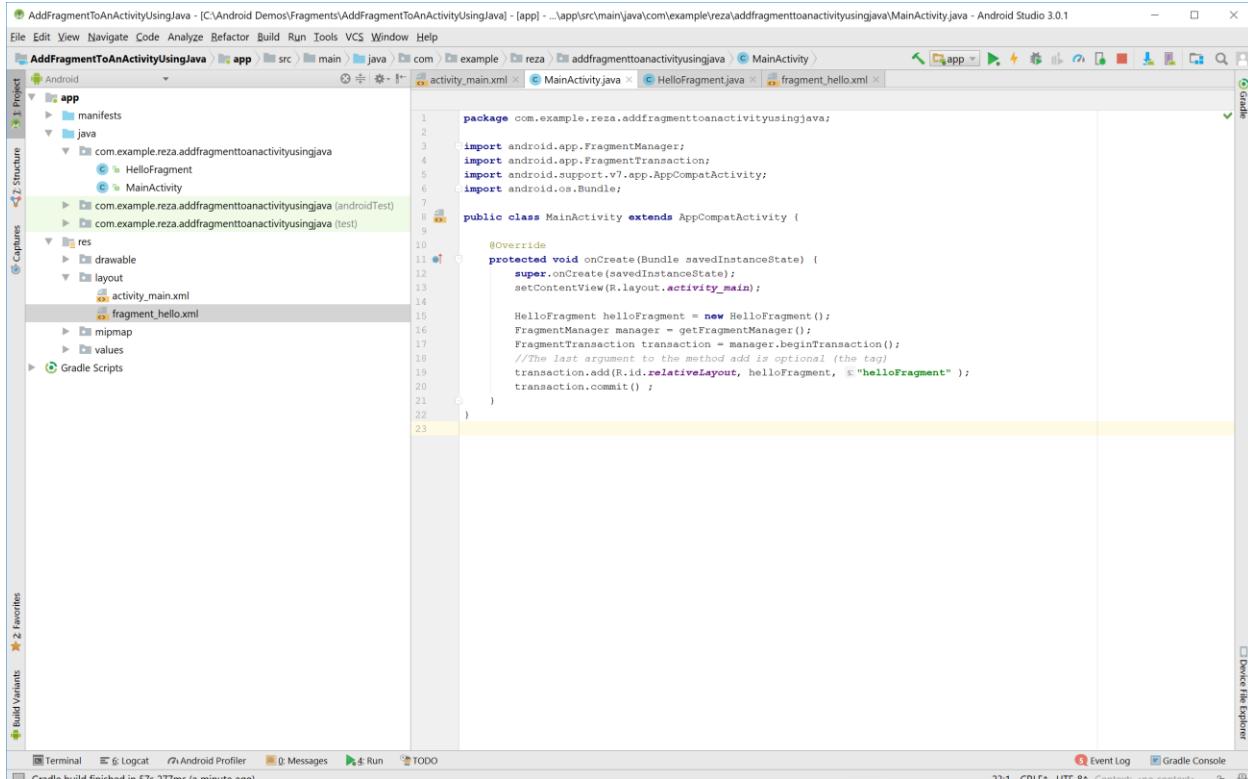
The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The code editor displays the MainActivity.java file, which contains the following Java code:

```
activity_main.xml>MainActivity.java>HelloFragment.java>fragment_hello.xml
```

```
1 package com.example.reza.addfragmenttoanactivityusingjava;
2
3 import android.app.FragmentManager;
4 import android.app.FragmentTransaction;
5 import android.support.v7.app.AppCompatActivity;
6 import android.os.Bundle;
7
8 public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         HelloFragment helloFragment = new HelloFragment();
16         FragmentManager manager = getFragmentManager();
17         FragmentTransaction transaction = manager.beginTransaction();
18         //The last argument to the method add is optional (the tag)
19         transaction.add(R.id.relativeLayout, helloFragment, "helloFragment");
20     }
21 }
```

The code editor highlights the line `transaction.add(R.id.relativeLayout, helloFragment, "helloFragment");` with a yellow background. A tooltip above the line reads: "The last argument to the method add is optional (the tag)". The bottom status bar of the IDE shows the message "Build finished in 20.02s (1 hour ago)".

Then call the commit() method. Without this call you won't be able to perform the add operation that you requested in the previous line.



The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The code editor displays the file 'MainActivity.java' which contains the following Java code:

```
package com.example.reza.addfragmenttoanactivityusingjava;

import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

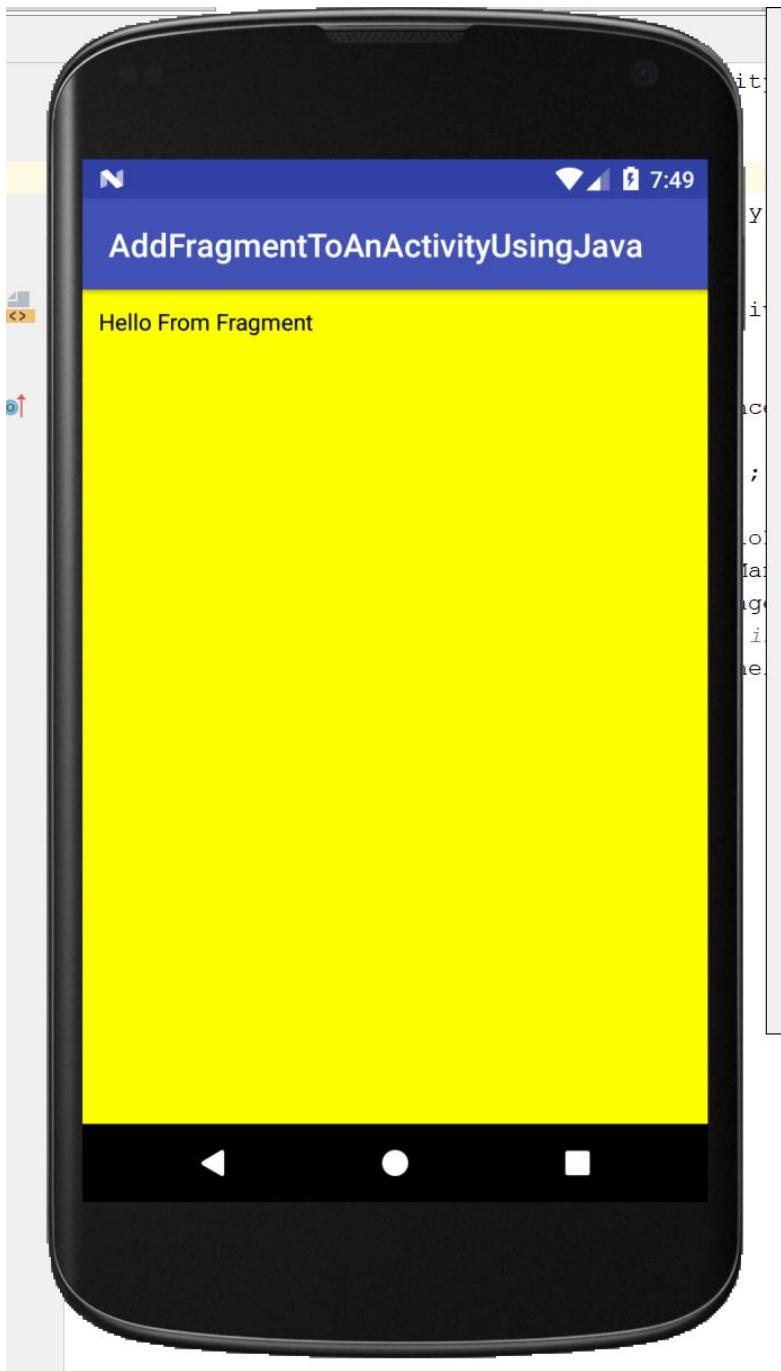
public class MainActivity extends AppCompatActivity {

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

        HelloFragment helloFragment = new HelloFragment();
        FragmentManager manager = getFragmentManager();
        FragmentTransaction transaction = manager.beginTransaction();
        //The last argument to the method add is optional (the tag)
        transaction.add(R.id.relativeLayout, helloFragment, "helloFragment");
        transaction.commit();
    }
}
```

The code editor has syntax highlighting and line numbers. Below the code editor, the bottom navigation bar of Android Studio is visible, showing tabs for Terminal, Logcat, Android Profiler, Messages, Run, TODO, Event Log, and Gradle Console.

Now if you run the app, you'll see your fragment displayed in the Relative layout.



Now this fragment has occupied the whole screen, another word the whole layout of the main activity. The reason is that the layout width and layout height of the fragment is `match_parent` as shown in the following screen capture so we can change it:

The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The 'fragment\_hello.xml' file is selected in the layout editor. The code in the Text tab is:

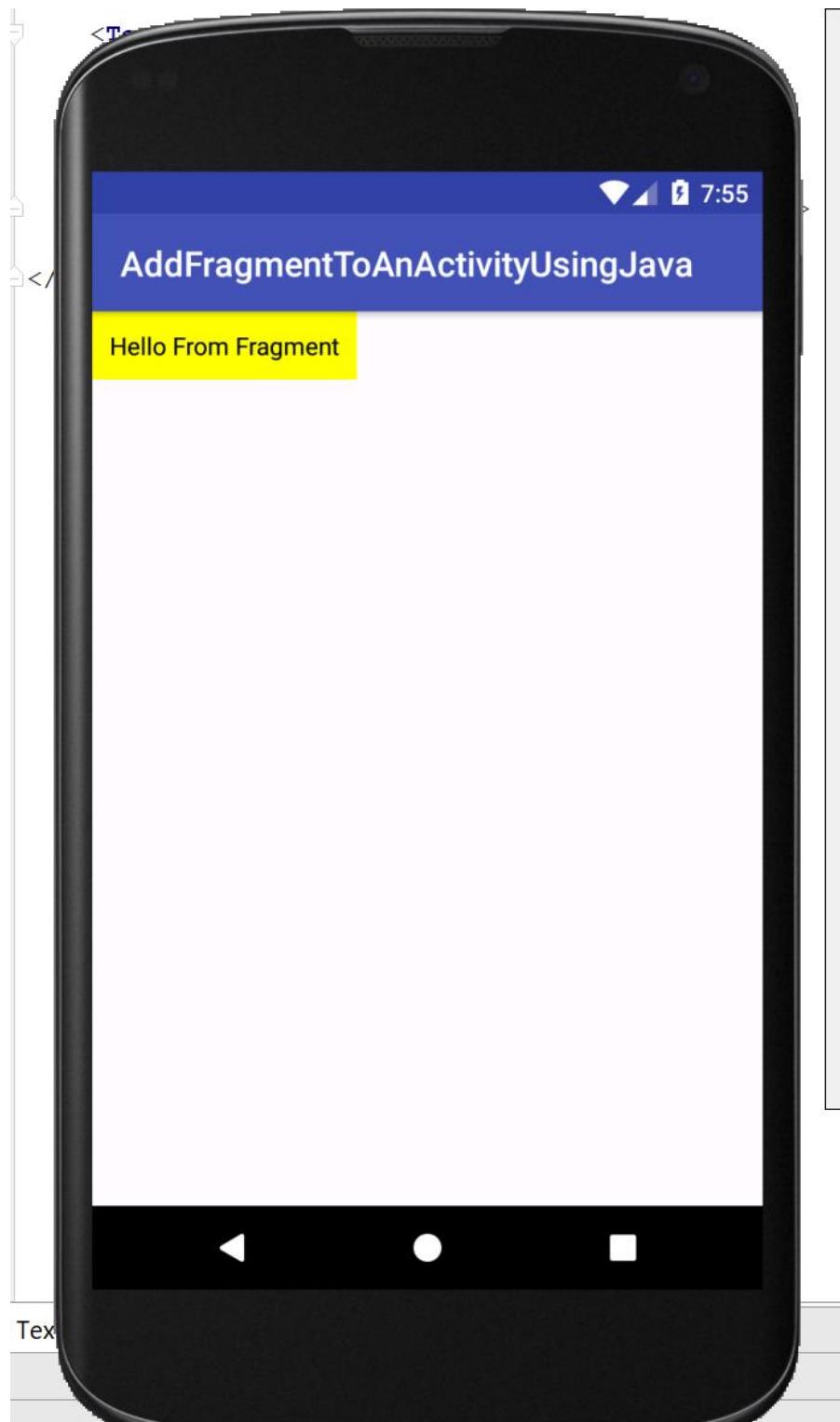
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FF0000">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Hello From Fragment"
        android:textColor="@android:color/black"/>

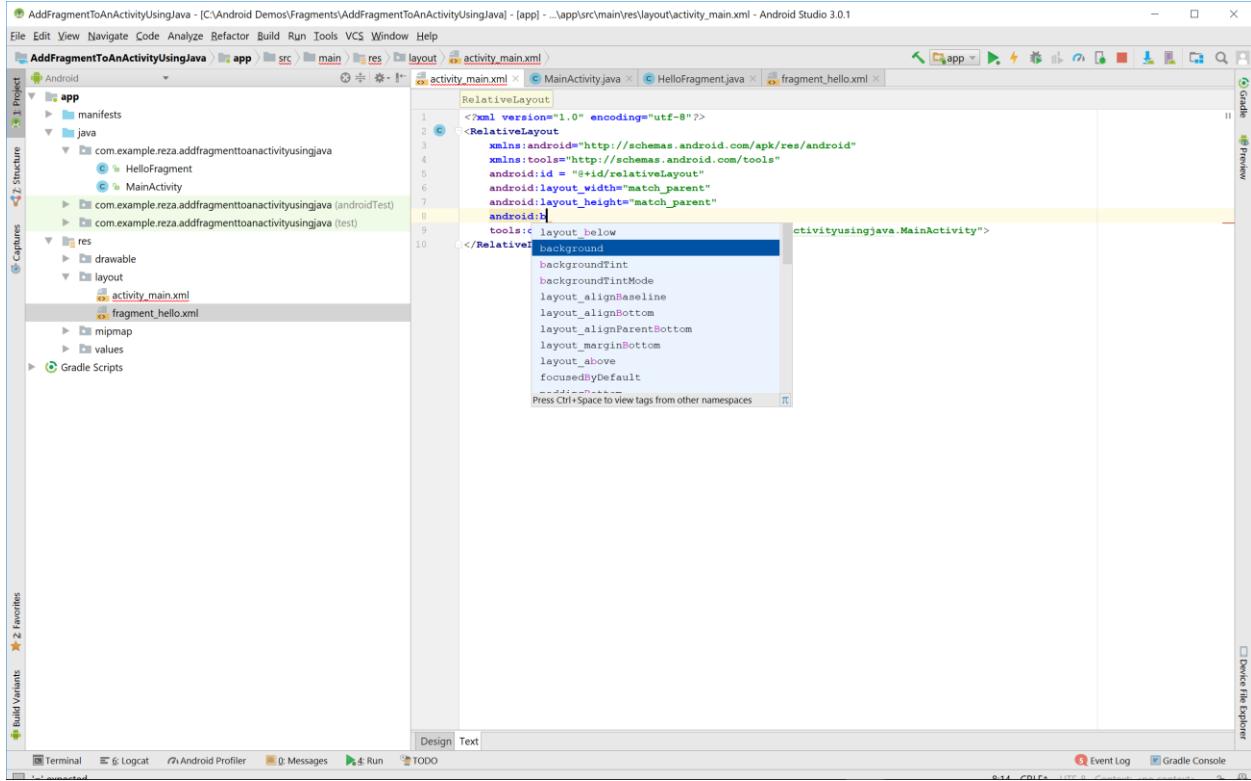
</LinearLayout>
```

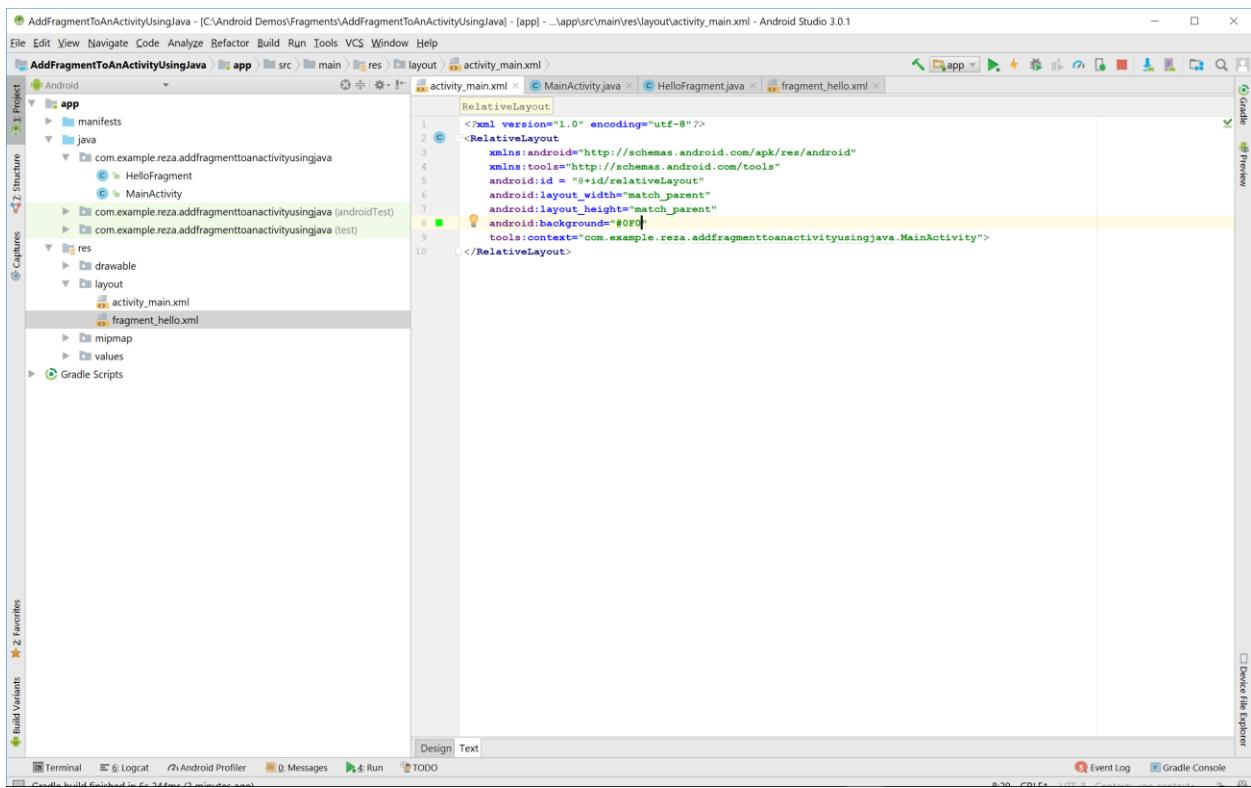
This screenshot is identical to the one above, but the line ' android:background="#FF0000"' is highlighted with a yellow background, indicating it is selected or being edited.

Once you make that change and run your app, this is what you get:

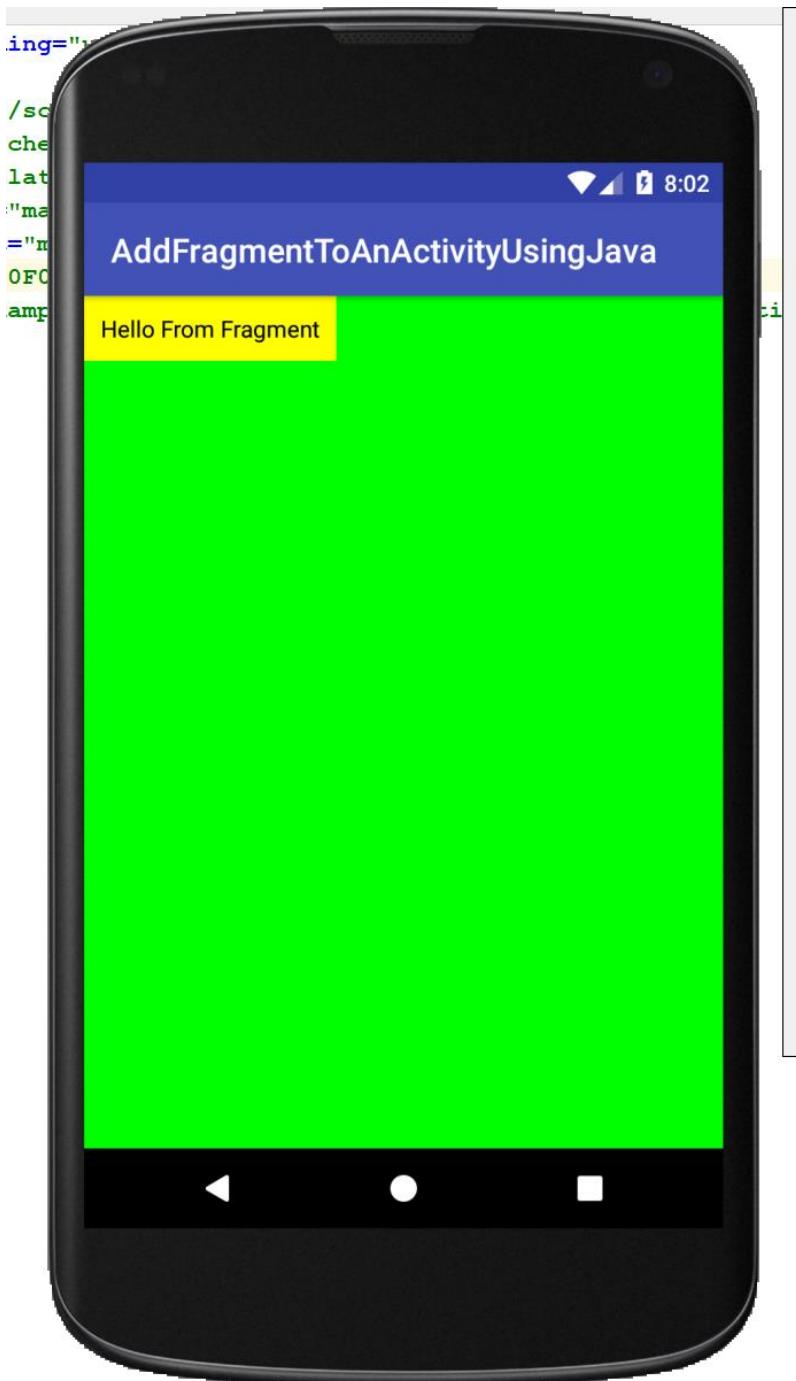


The fragment doesn't appear at the center as it did in the previous example so we can fix it as follows and as long as we are at it we can change the background color of the main activity to green so it matches with the previous example:





Here is the background:



Let's say if we wanted to put our fragment at the center of the screen of the layout for the main activity, we could add a Frame layout to our Relative layout and center this Frame layout in our Relative layout and then add our fragment to this Frame layout as shown below:

The screenshot shows the Android Studio interface with the project navigation bar at the top. Below it is the Project structure, which includes the app module with its manifest, Java files (HelloFragment, MainActivity), and resources (layout, drawable, mipmap, values). The layout folder contains two XML files: activity\_main.xml and fragment\_hello.xml. The activity\_main.xml file is currently selected and displayed in the main editor area. The code shows a RelativeLayout with a FrameLayout as a child. A tooltip is visible over the FrameLayout, providing documentation for the wrap\_content attribute.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00F0"
    tools:context="com.example.reza.addfragmenttoanactivityusingjava.MainActivity">
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/frameLayout"
        android:layout_centerInParent="true"/>

```

This screenshot is similar to the previous one, but the tooltip for the wrap\_content attribute is more prominent. It includes a warning message: "The view should be only big enough to enclose its content (plus padding)".

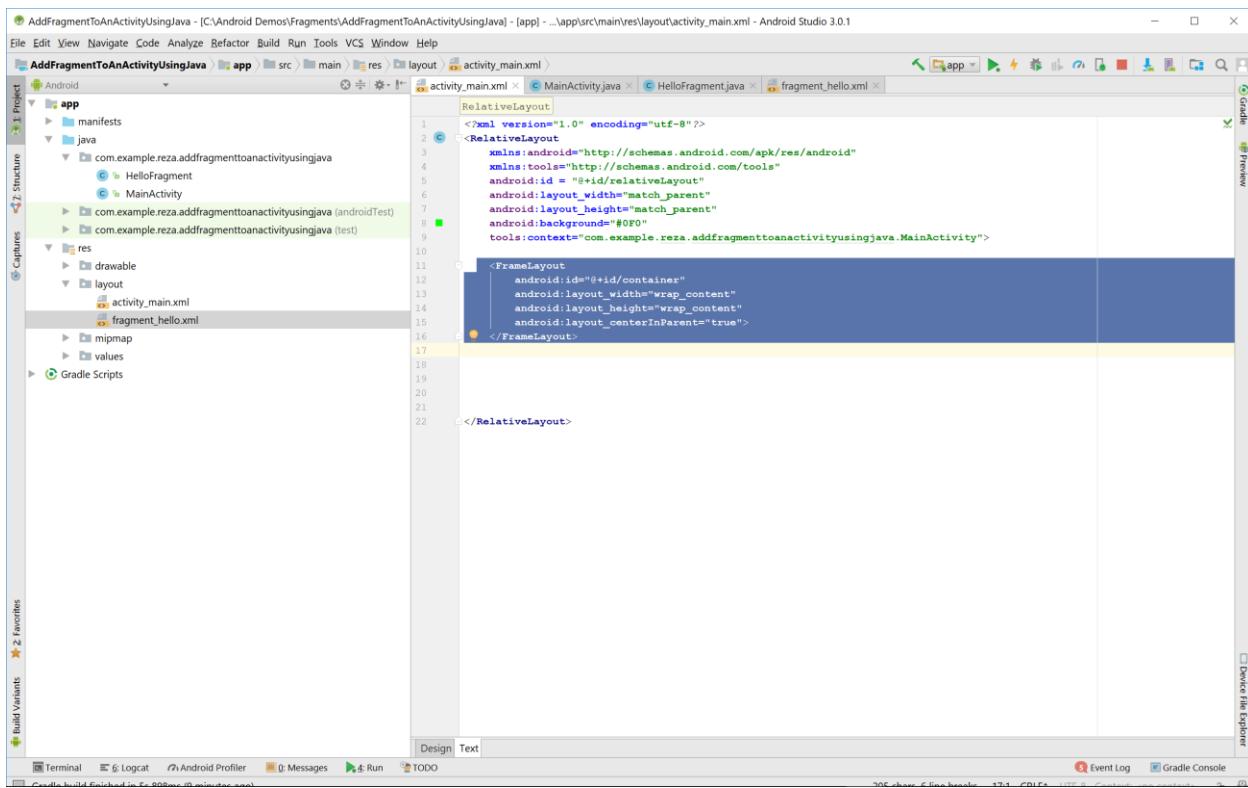
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00F0"
    tools:context="com.example.reza.addfragmenttoanactivityusingjava.MainActivity">
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/frameLayout"
        android:layout_centerInParent="true"/>

```

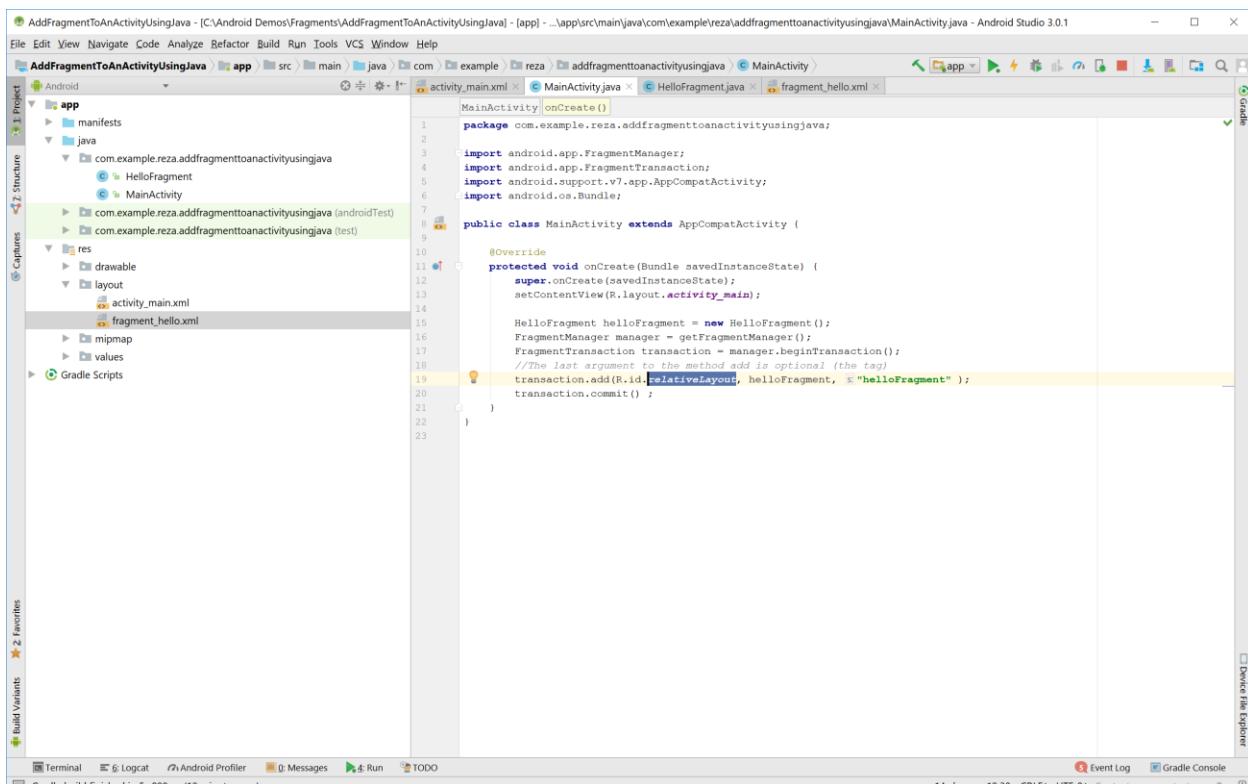
The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The left sidebar displays the project structure under the 'app' module. The main editor window shows the XML code for 'activity\_main.xml'. The cursor is positioned on the line '14 <FrameLayout'. A context menu is open over the code, with the path 'Edit | Paste' highlighted in blue. The code itself is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout>
    <FrameLayout
        android:id="@+id/relativeLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#0F0"
        tools:context="com.example.reza.addfragmenttoanactivityusingjava.MainActivity">
        <FrameLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_centerInParent="true"
            android:clickable="false"
            android:clipChildren="true"
            android:clipToPadding="true"
            android:contentDescription="Hello Fragment"
            android:contextClickable="true"
            android:layout_marginVertical="10dp"
            android:padding="10dp"
            android:background="#FFF"
            android:outlineProvider="view"
            android:stateListAnimator="null"
            android:transitionName="null">
            <HelloFragment />
        </FrameLayout>
    </FrameLayout>
</RelativeLayout>
```

This screenshot shows the same Android Studio session with the XML code for 'activity\_main.xml'. The cursor is now on the line '14 <FrameLayout>'. The context menu is again open, but the path 'Edit | Paste' is not highlighted. Instead, the path 'Edit | Cut' is highlighted in blue. The code remains the same as in the previous screenshot.



Now go back to MainActivity.java and in the add method instead of “R.id.relativeLayout” pass “R.id.container” to the add method as shown:



The screenshot shows the Android Studio interface with the project 'AddFragmentToAnActivityUsingJava' open. The left sidebar displays the project structure, including the app module with its Java, XML, and resources. The main editor window shows the code for `MainActivity.java`, which contains the logic for adding a fragment to the activity's layout.

```
1 package com.example.reza.addfragmenttoanactivityusingjava;
2
3 import android.app.FragmentManager;
4 import android.app.FragmentTransaction;
5 import android.support.v7.app.AppCompatActivity;
6 import android.os.Bundle;
7
8 public class MainActivity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14
15         HelloFragment helloFragment = new HelloFragment();
16         FragmentManager manager = getFragmentManager();
17         FragmentTransaction transaction = manager.beginTransaction();
18         //The last argument to the method add is optional (the tag)
19         transaction.add(R.id.container, helloFragment, "helloFragment");
20         transaction.commit();
21     }
22 }
```

Now run your app and you have the exact result as the previous example when you added the fragment by dragging it to the main activity layout:

