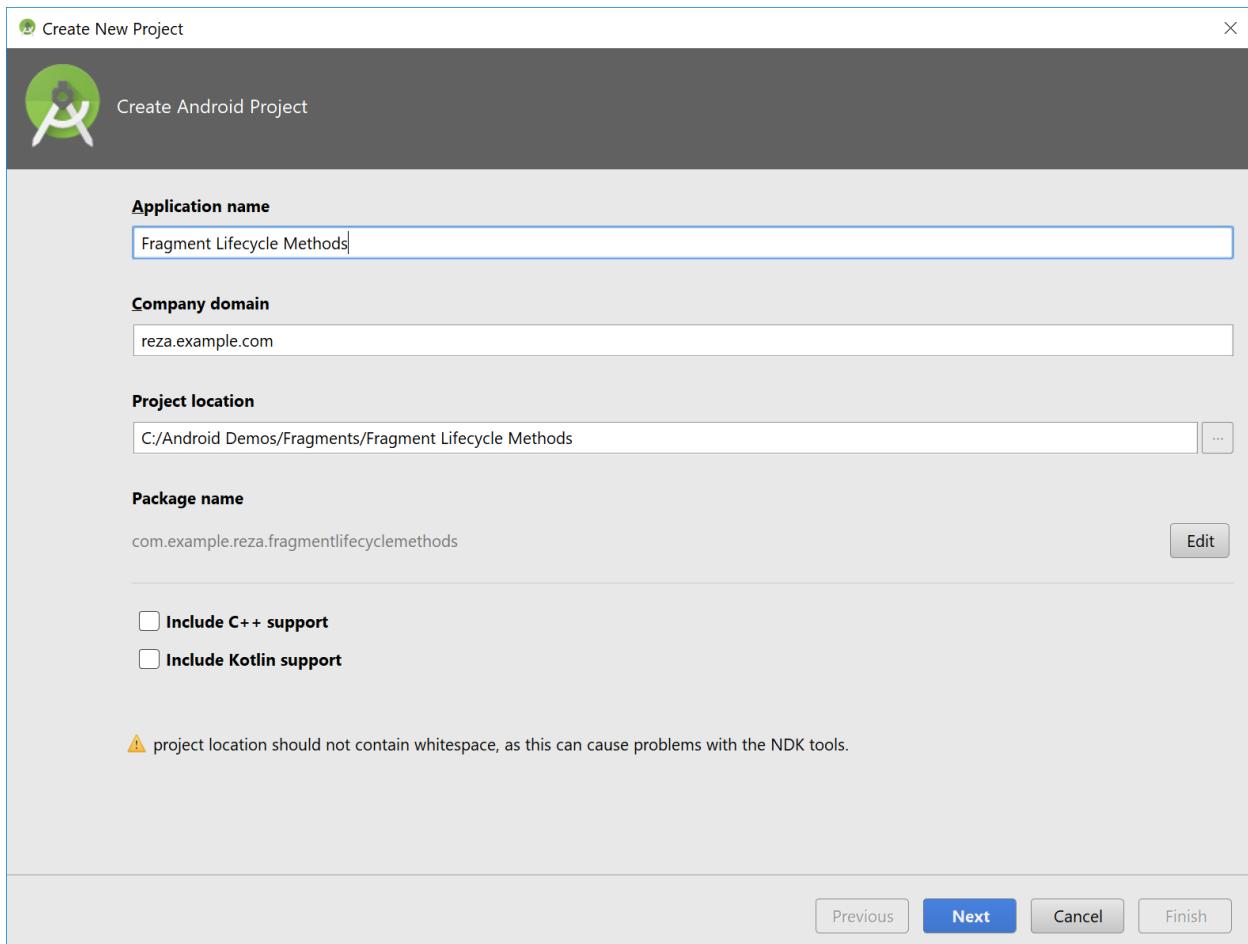


Fragment Lifecycle Methods and How They Are Related to the Host Activity Lifecycle methods

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

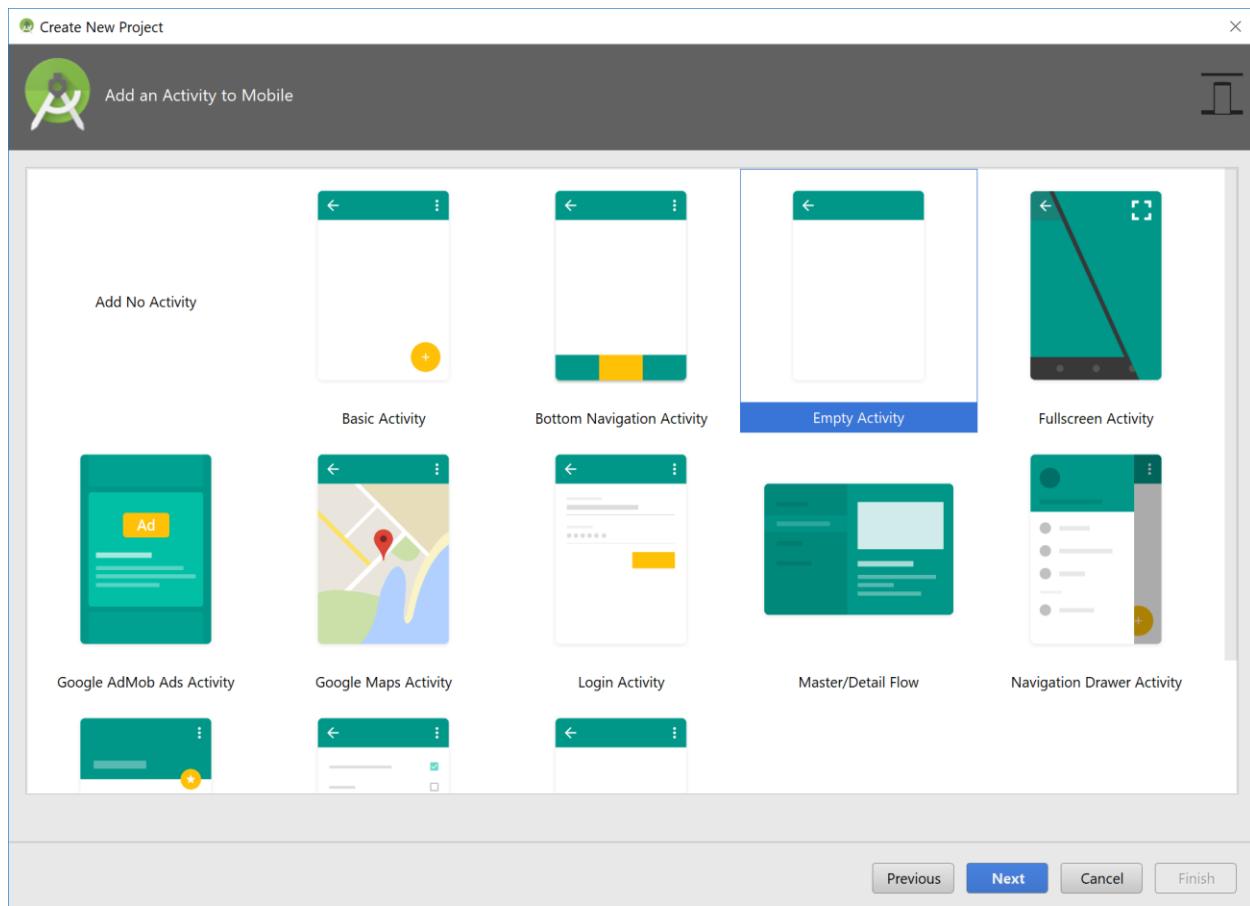
API 21: Android 5.0 (Lollipop) ▼

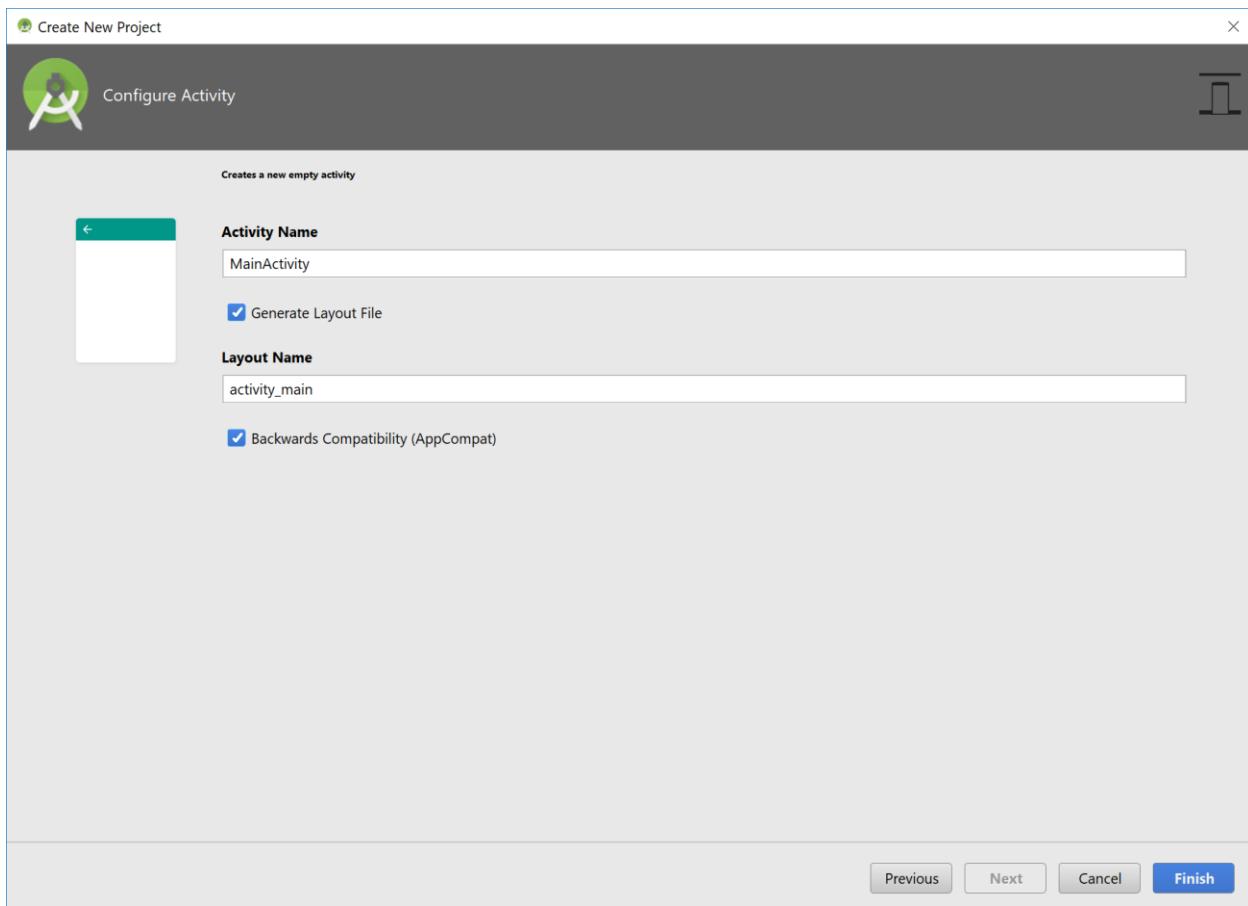
Android Auto

Android Things

API 24: Android 7.0 (Nougat) ▼

Previous Next Cancel Finish





Fragment Lifecycle Methods - [C:\Android Demos\Fragments\Fragment Lifecycle Methods] - ...\\app\\src\\main\\java\\com\\example\\reza\\fragmentlifecyclemethods\\MainActivity.java - Android Studio 3.0.1

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Fragment Lifecycle Methods > app > src > main > java > com > example > reza > fragmentlifecyclemethods > MainActivity >

activity_main.xml > MainActivity.java

Project

Build Variants

captures

Favorites

Gradle Scripts

Device File Explorer

Terminal Logcat Messages TODO

Event Log Gradle Console

3:8 CRLF: UTF-8 Context: <no context>

```
1 package com.example.reza.fragmentlifecyclemethods;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
```

Fragment Lifecycle Methods - [C:\Android Demos\Fragments\Fragment Lifecycle Methods] - [app] - ...\\app\\src\\main\\res\\layout\\activity_main.xml - Android Studio 3.0.1

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Fragment Lifecycle Methods > app > src > main > res > layout > activity_main.xml > MainActivity.java

activity_main.xml > MainActivity.java

Project

Build Variants

Captures

Favorites

Gradle Scripts

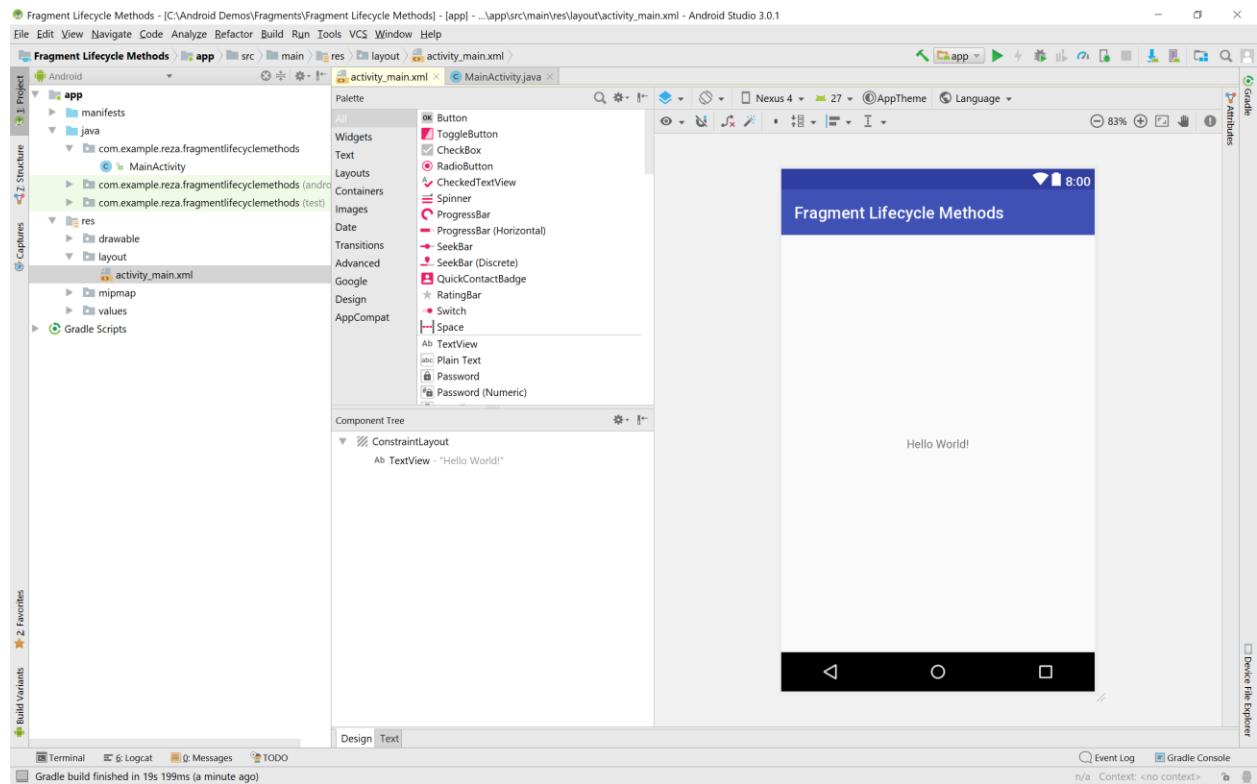
Device File Explorer

Terminal Logcat Messages TODO

Event Log Gradle Console

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

```
<?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.fragmentlifecyclemethods.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>
```



Change the layout file (activity_main.xml) as shown:

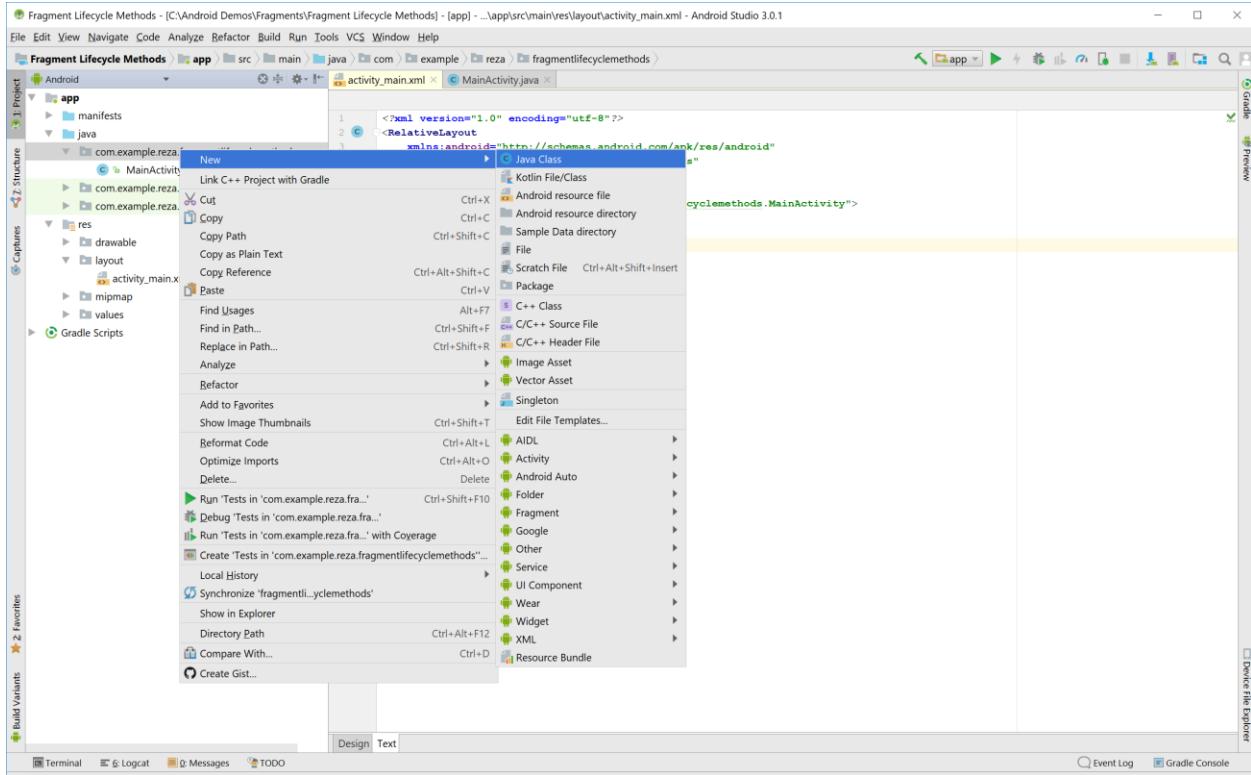
The screenshot shows the Android Studio interface with the following details:

- Title Bar:** Fragment Lifecycle Methods - [C:\Android Demos\Fragments\Fragment Lifecycle Methods] - [app] - ...\\app\\src\\main\\res\\layout\\activity_main.xml - Android Studio 3.0.1
- Toolbar:** File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
- Project Structure:** Shows the project structure with modules: app, manifests, java, com.example.reza.fragmentlifecyclemethods, MainActivity, and resources: res, drawable, layout, activity_main.xml.
- Code Editor:** The activity_main.xml file is open in the code 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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.reza.fragmentlifecyclemethods.MainActivity">
```

The code editor has tabs for Design and Text, and a status bar at the bottom showing "Gradle build finished in 19s 199ms (3 minutes ago)".

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.fragmentlifecyclemethods

Visibility: Public Package Private

Modifiers: None Abstract Final

Show Select Overrides Dialog

OK Cancel Help

Fragment Lifecycle Methods - [C:\Android Demos\Fragments\Fragment Lifecycle Methods] - [app] - ...\\app\\src\\main\\java\\com\\example\\reza\\fragmentlifecyclemethods\\HelloFragment.java - Android Studio 3.0.1

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Fragment Lifecycle Methods > app > src > main > java > com > example > reza > fragmentlifecyclemethods > HelloFragment >

activity_main.xml HelloFragment.java MainActivity.java

Android Project Captures Build Variants Favorites Device File Explorer

Gradle Terminal Logcat Messages TODO Event Log Gradle Console

```
1 package com.example.reza.fragmentlifecyclemethods;
2
3 /**
4 * Created by Reza on 2018-03-02.
5 */
6
7 public class HelloFragment {
```

Fragment Lifecycle Methods - [C:\Android Demos\Fragments\Fragment Lifecycle Methods] - [app] - ...\\app\\src\\main\\java\\com\\example\\reza\\fragmentlifecyclemethods\\HelloFragment.java - Android Studio 3.0.1

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Fragment Lifecycle Methods > app > src > main > java > com > example > reza > fragmentlifecyclemethods > HelloFragment >

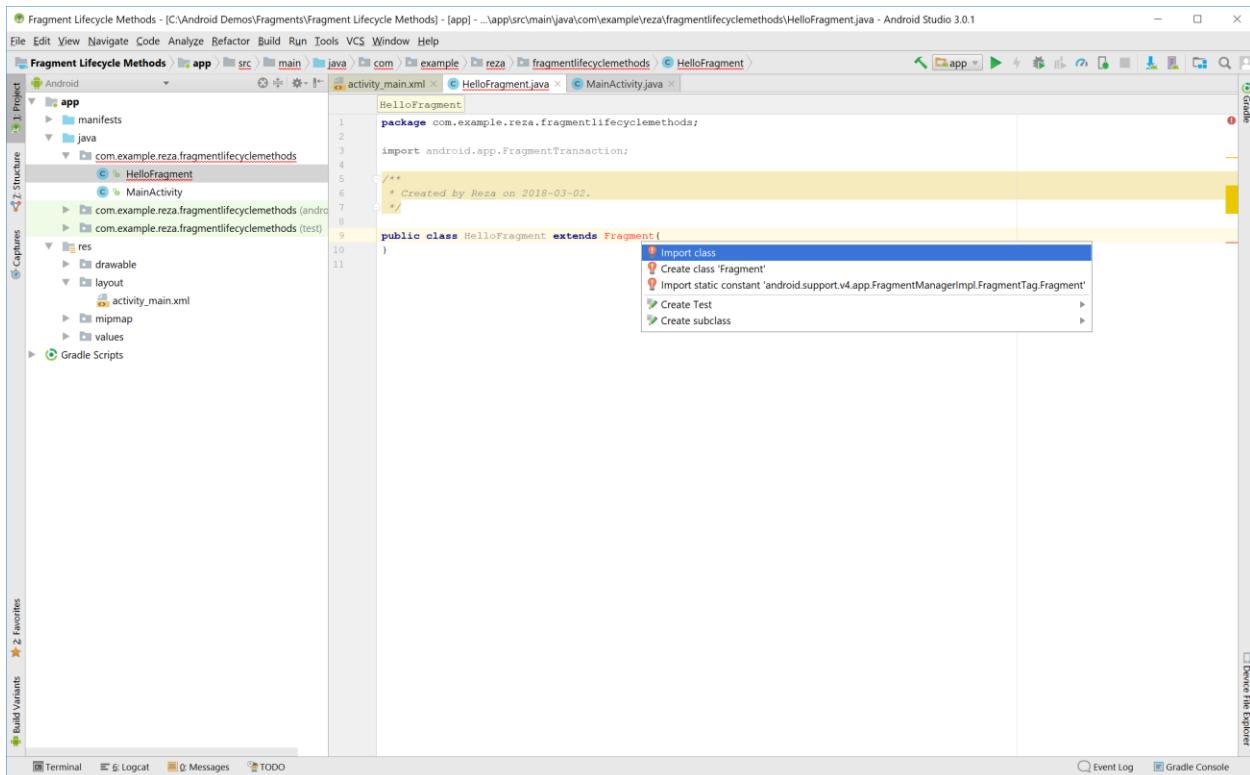
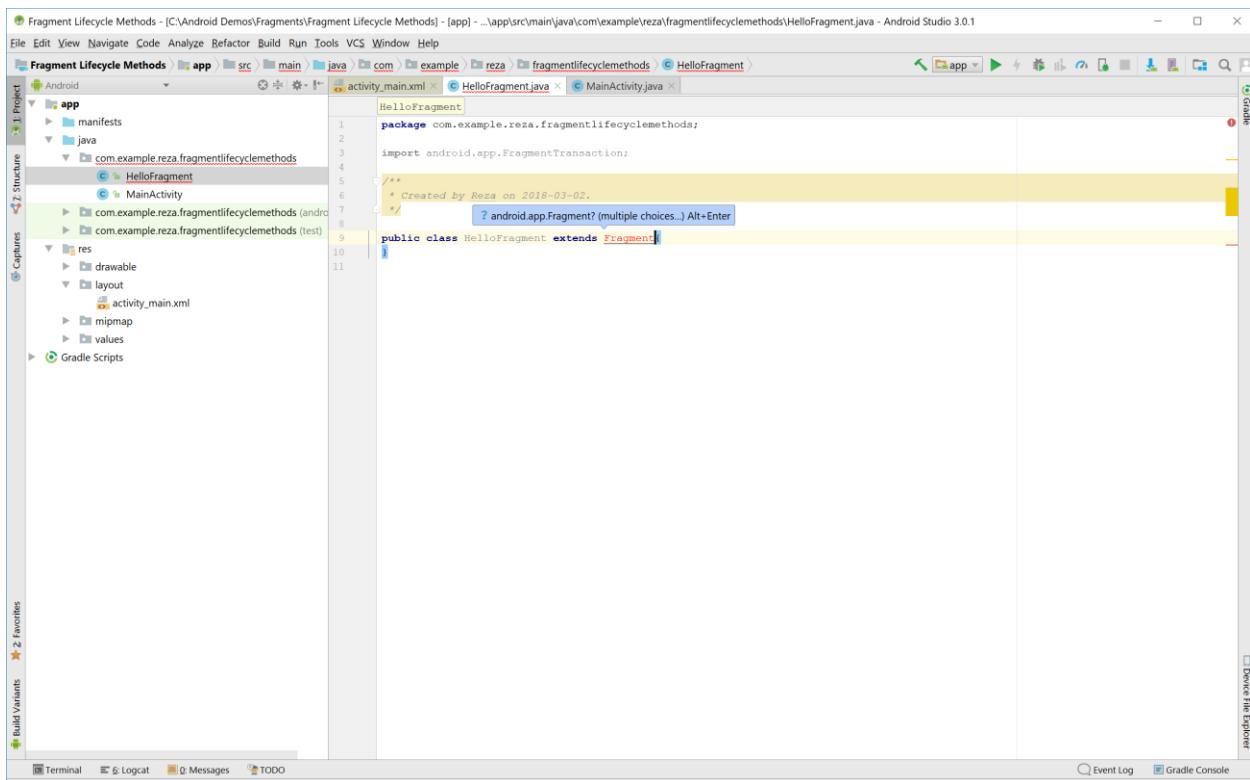
activity_main.xml HelloFragment.java MainActivity.java

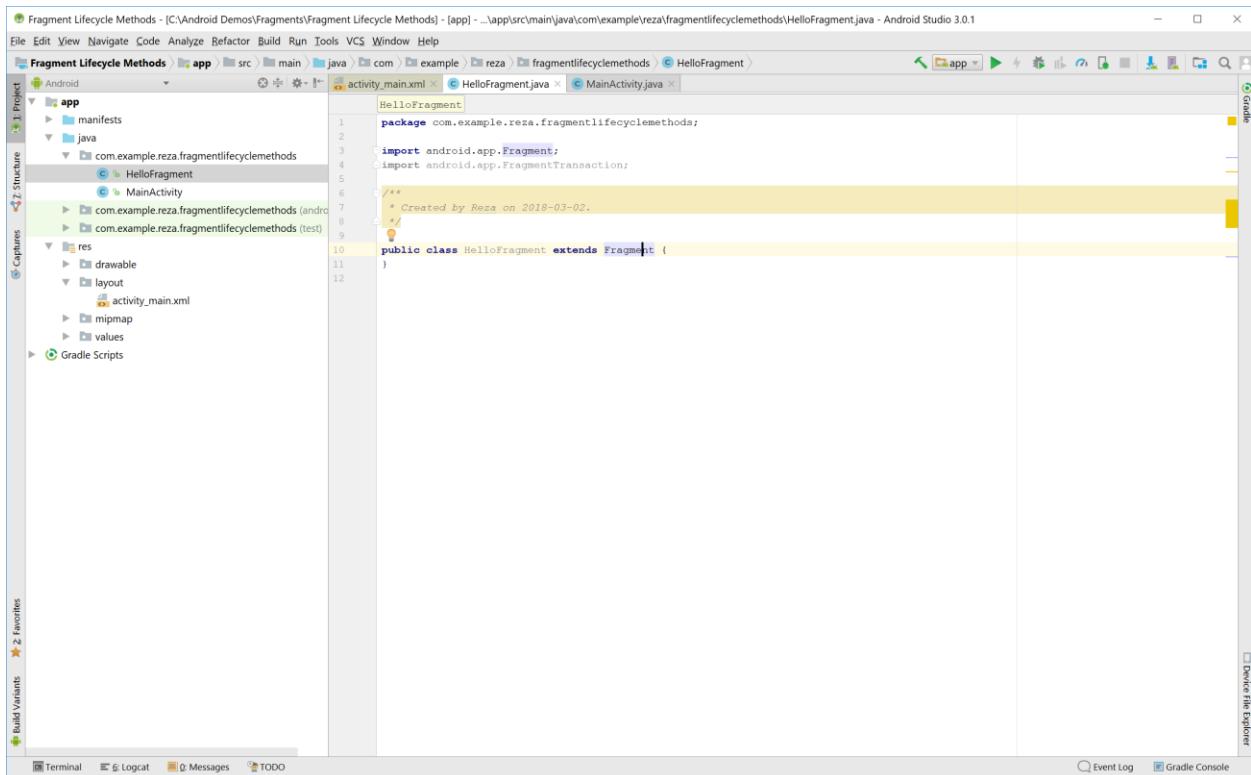
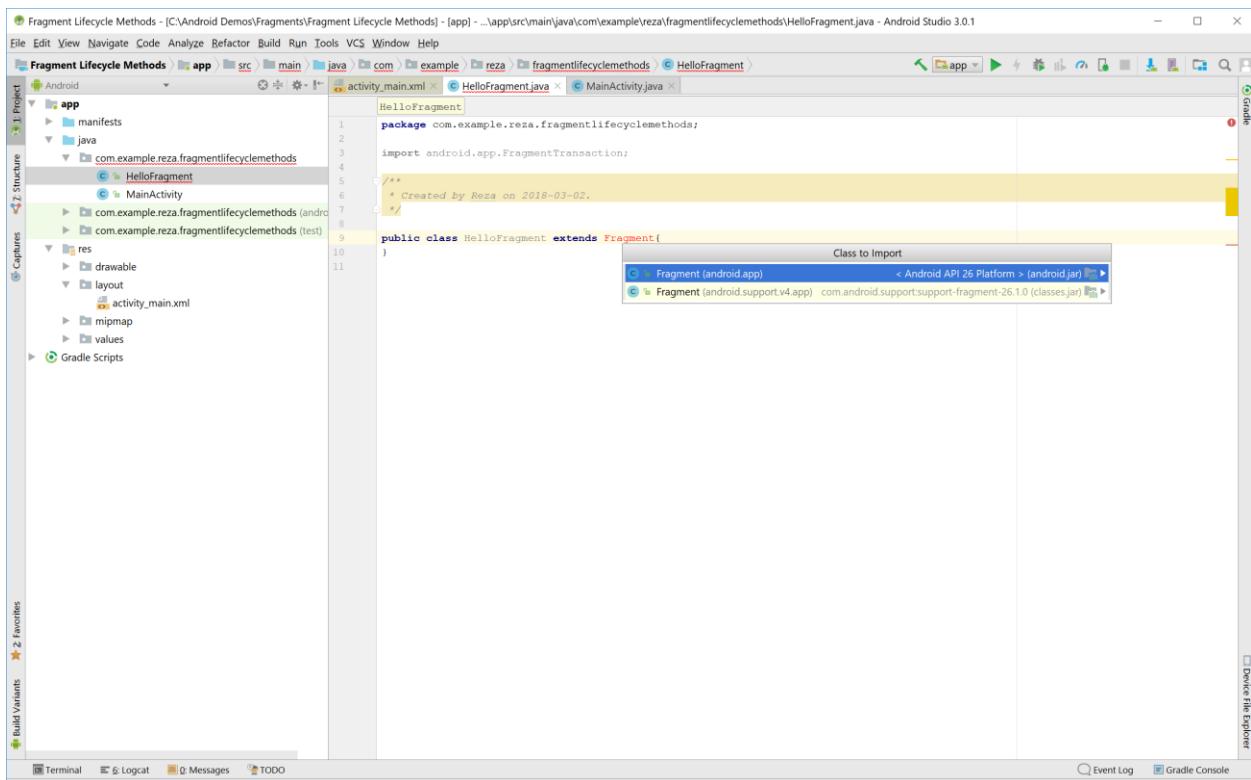
Android Project Captures Build Variants Favorites Device File Explorer

Gradle Terminal Logcat Messages TODO Event Log Gradle Console

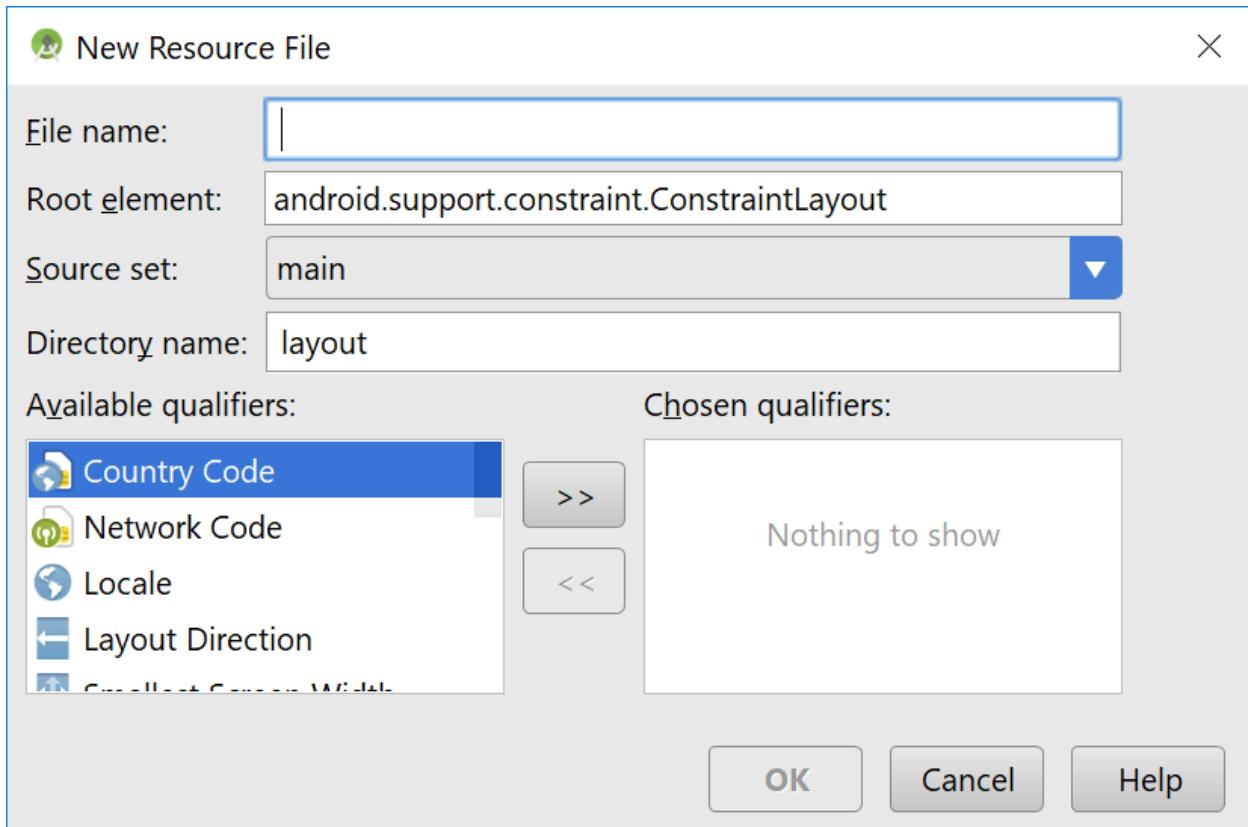
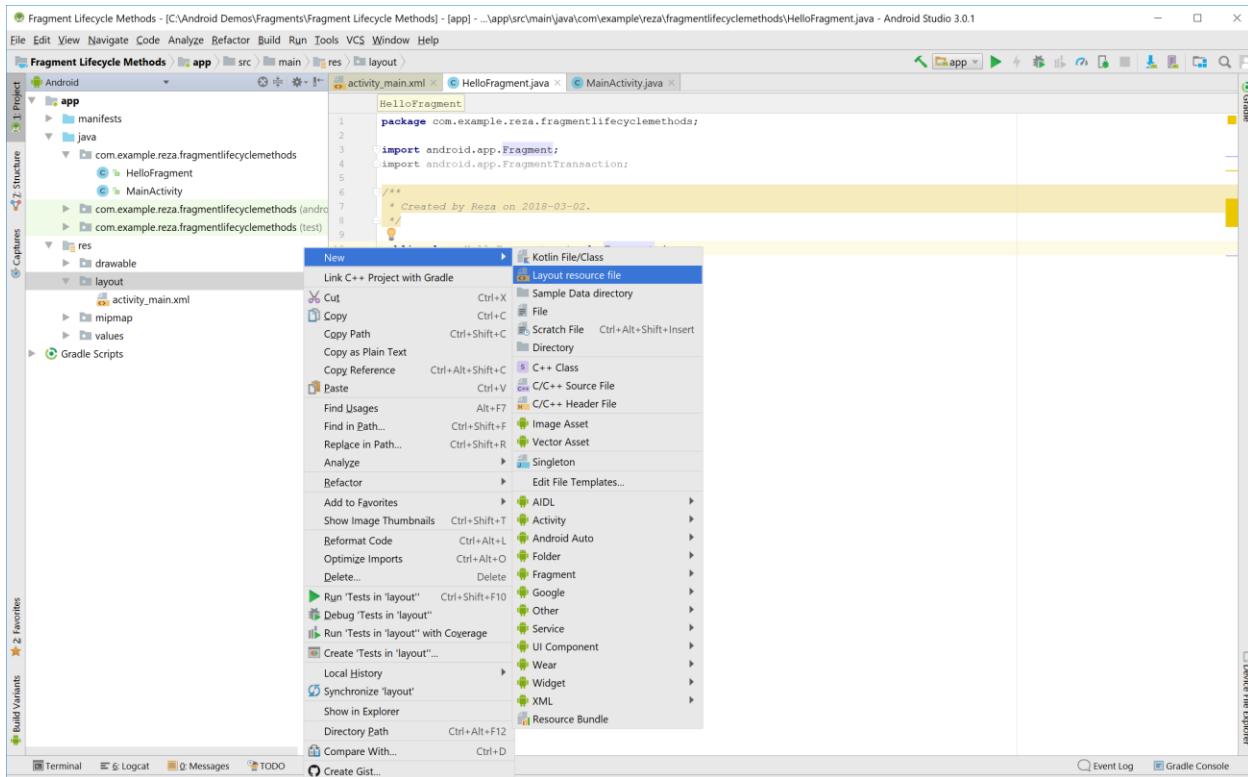
```
1 package com.example.reza.fragmentlifecyclemethods;
2
3 /**
4 * Created by Reza on 2018-03-02.
5 */
6
7 public class HelloFragment extends Fragment {
```

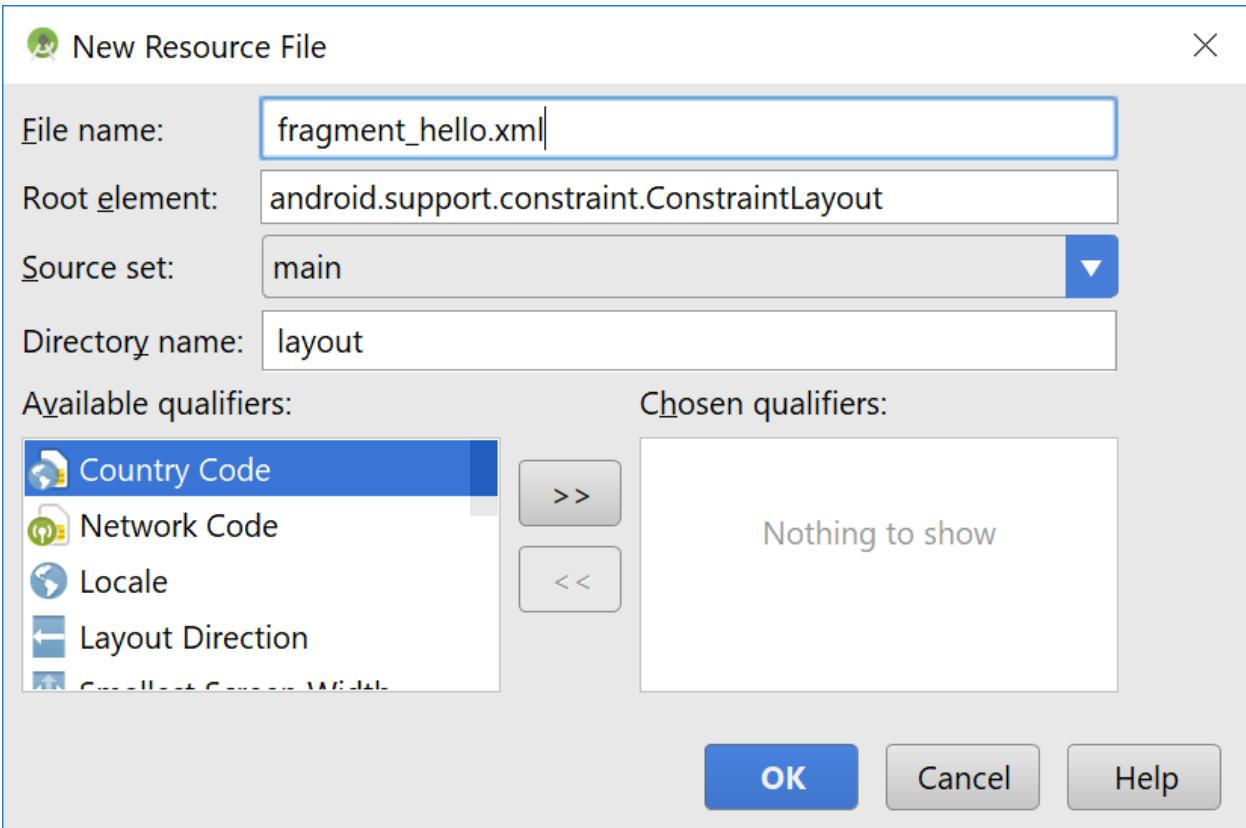
FragmentTransaction (android.app)
↳ Fragment (android.app)
↳ Fragment (android.support.v4.app)
↳ FragmentTransaction (android.support.v4.app)
↳ FragmentManager (android.app)
↳ FragmentManager (android.support.v4.app)
↳ FragmentBuilder (android.text.style.TtsSpan)
↳ FragmentContainer (android.app)
↳ FragmentController (android.app)
↳ FragmentHostCallback (E) (android.app)



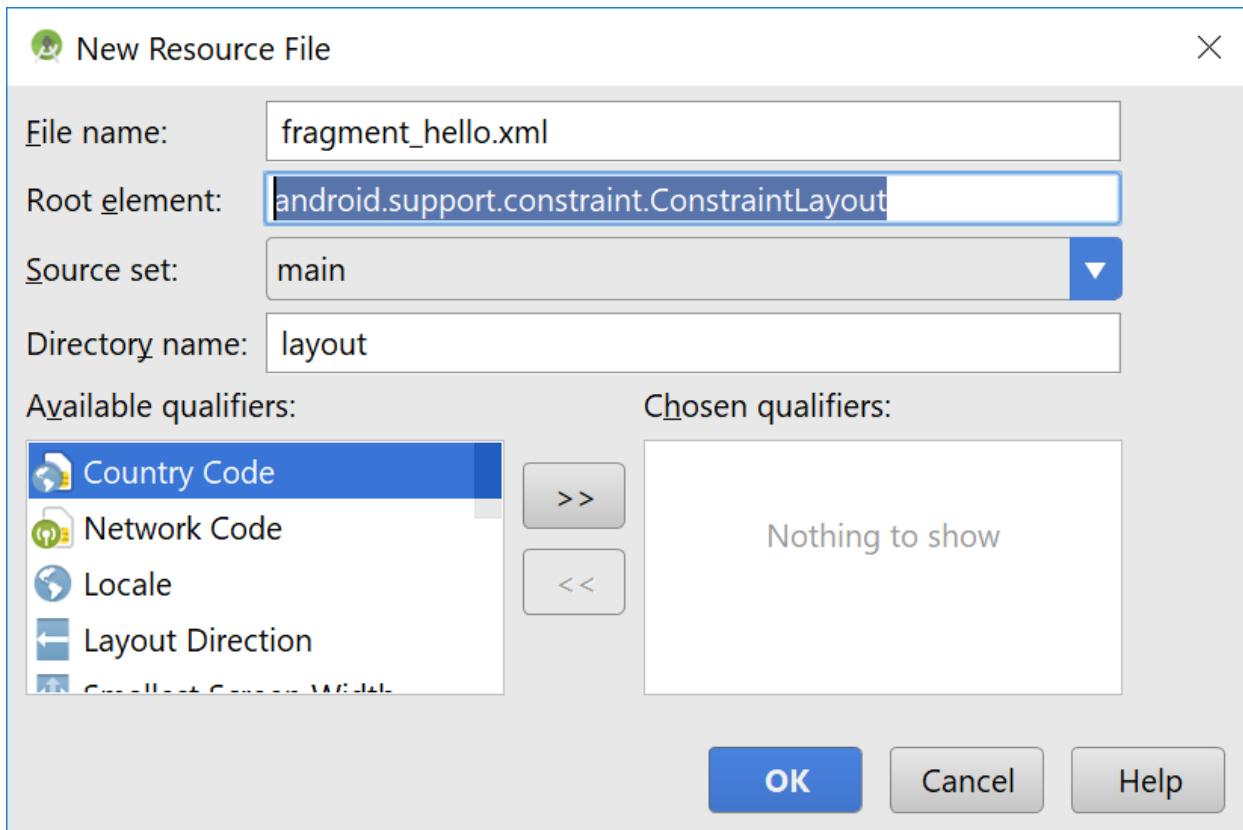


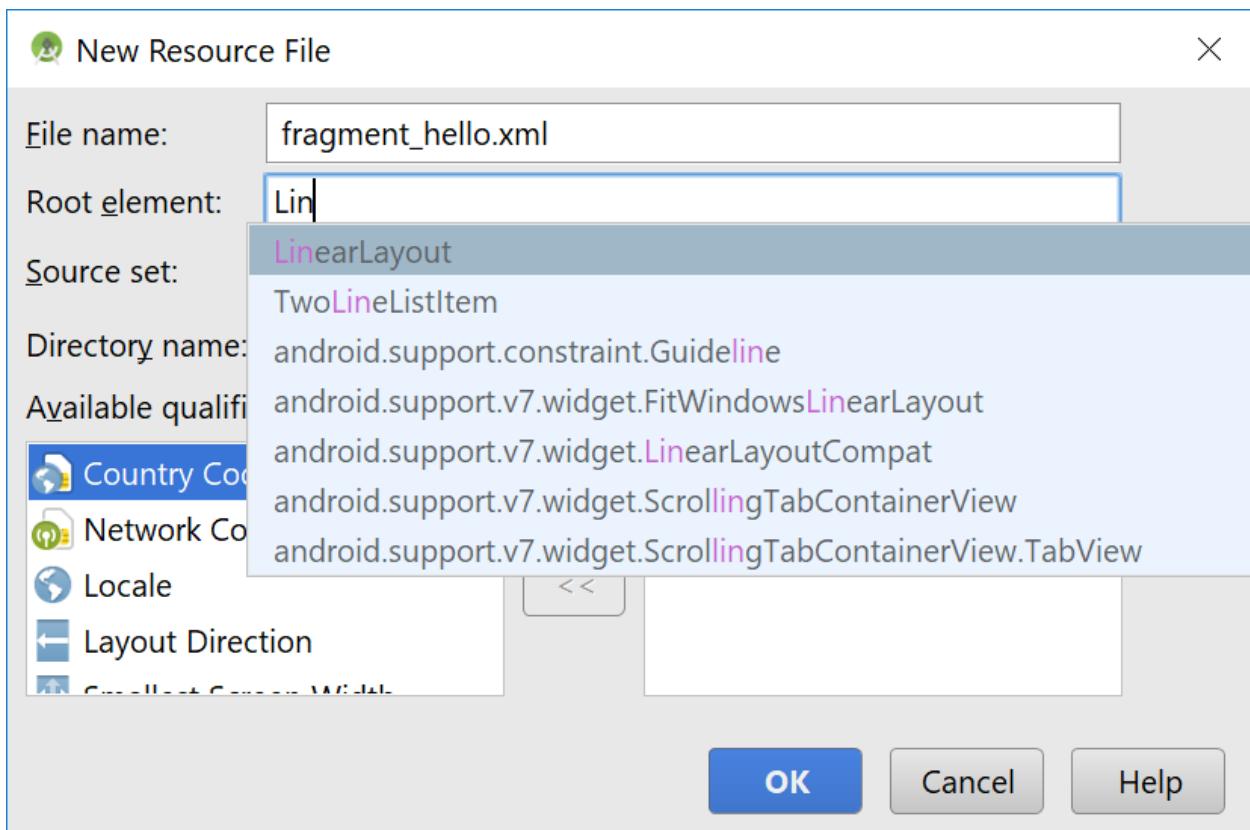
Next create a layout for your fragment class as shown:

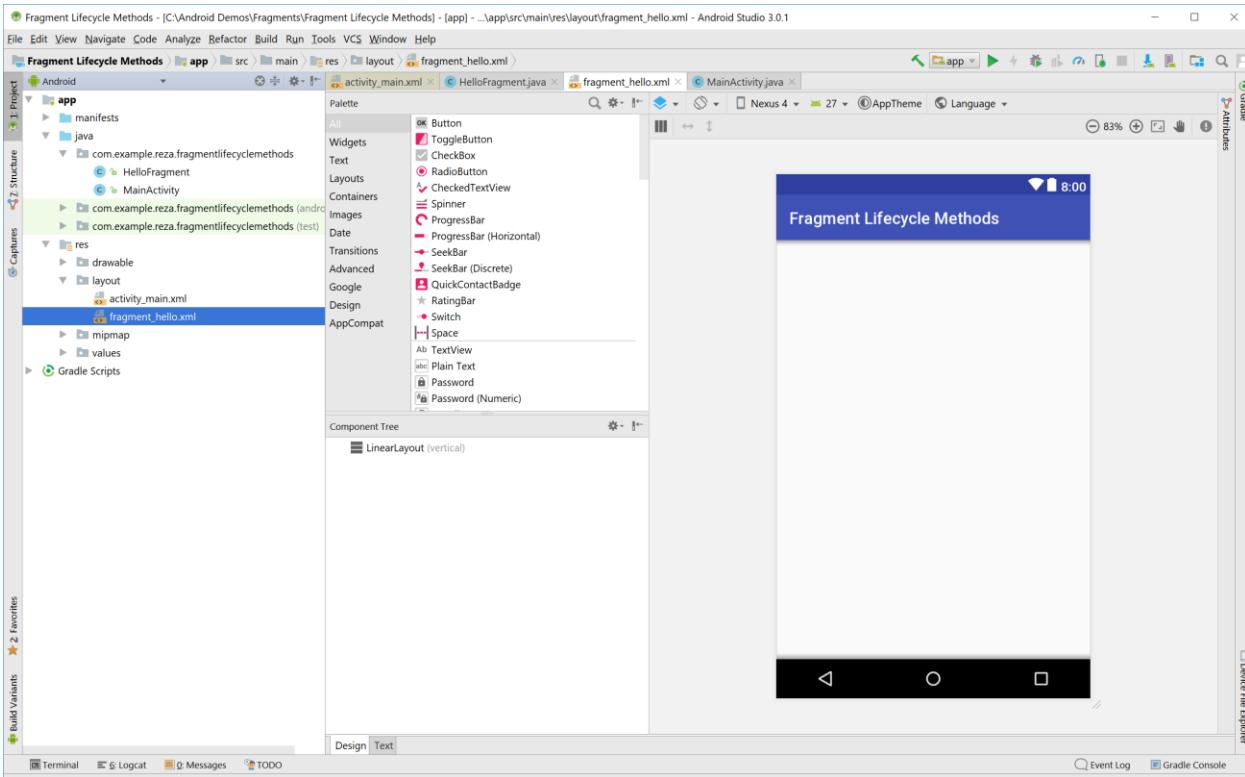
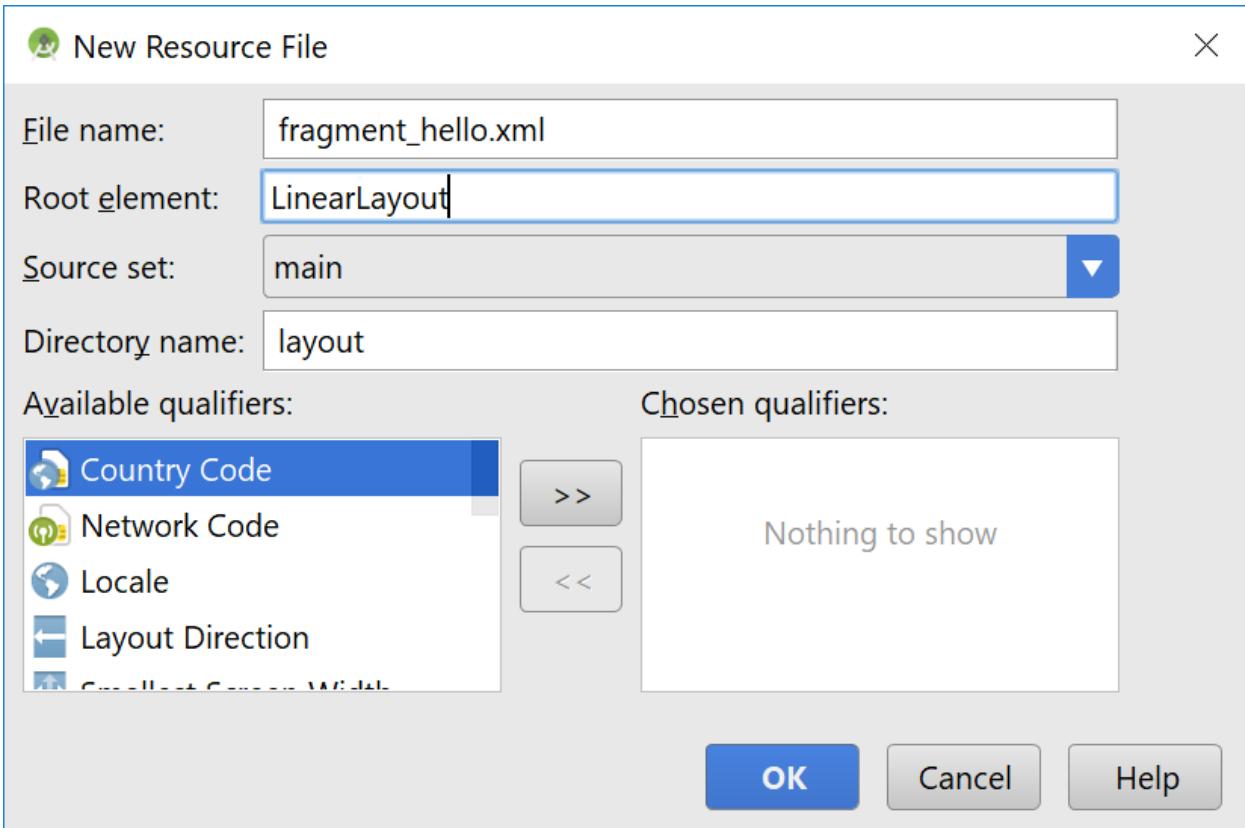




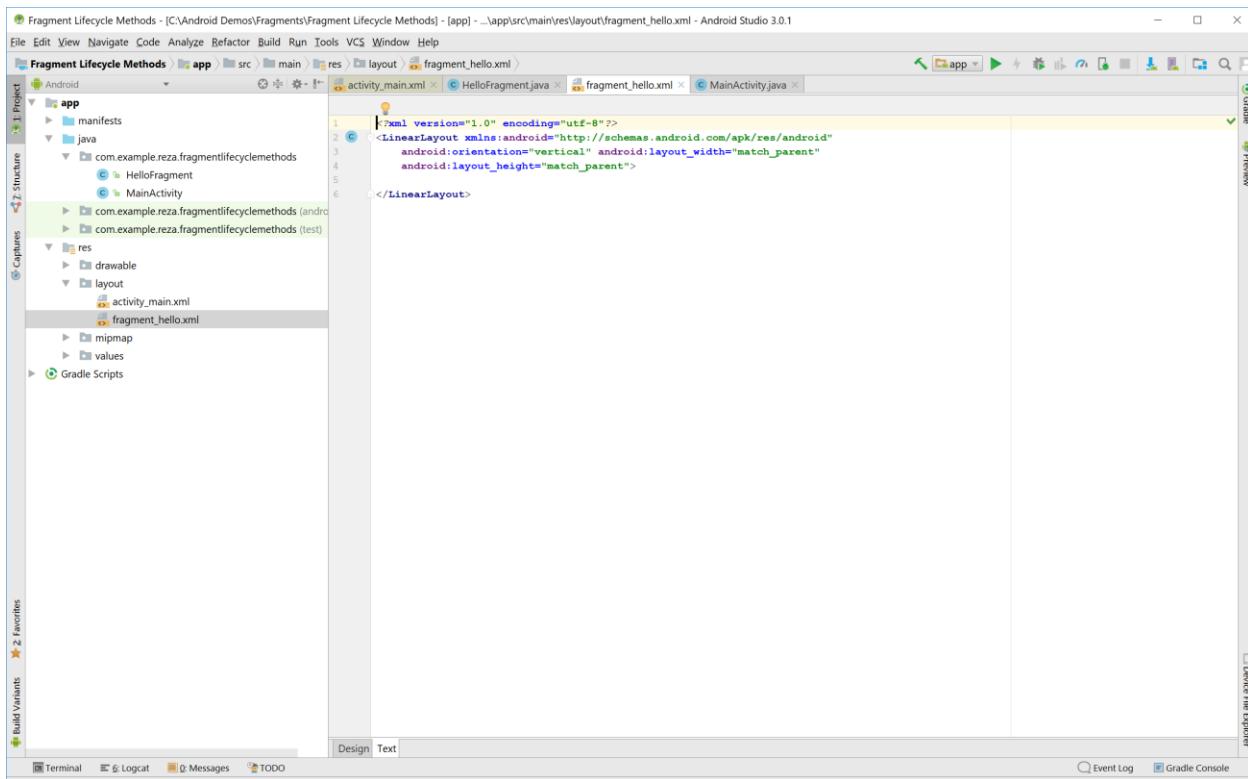
We are going to change the root layout to a LinearLayout as shown:







Here is the layout:



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

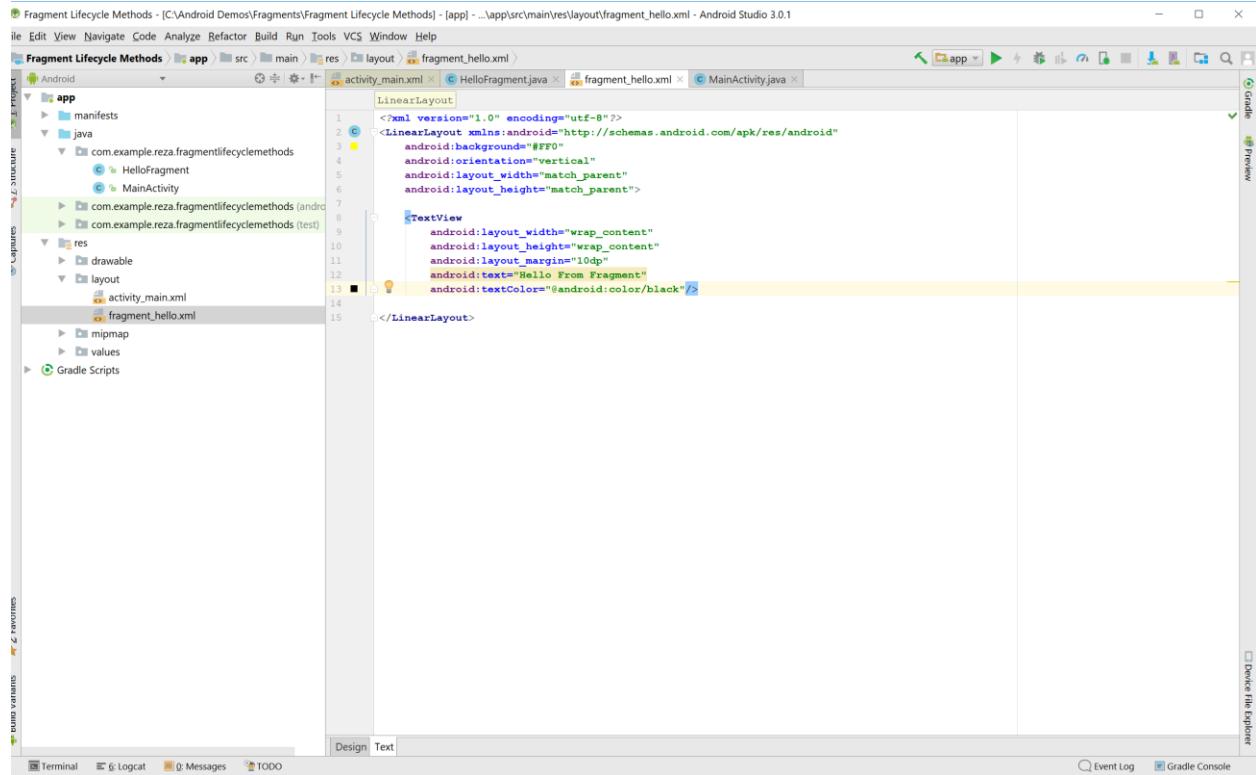
- Project Structure:** The left sidebar shows the project structure under the "app" module. It includes "manifests", "java" (containing "HelloFragment" and "MainActivity"), "res" (containing "drawable" and "layout" folders), and "layout" (containing "activity_main.xml" and "fragment_hello.xml").
- Code Editor:** The main editor window displays the XML code for "fragment_hello.xml". 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">
```

The code editor has tabs for "Design" and "Text".

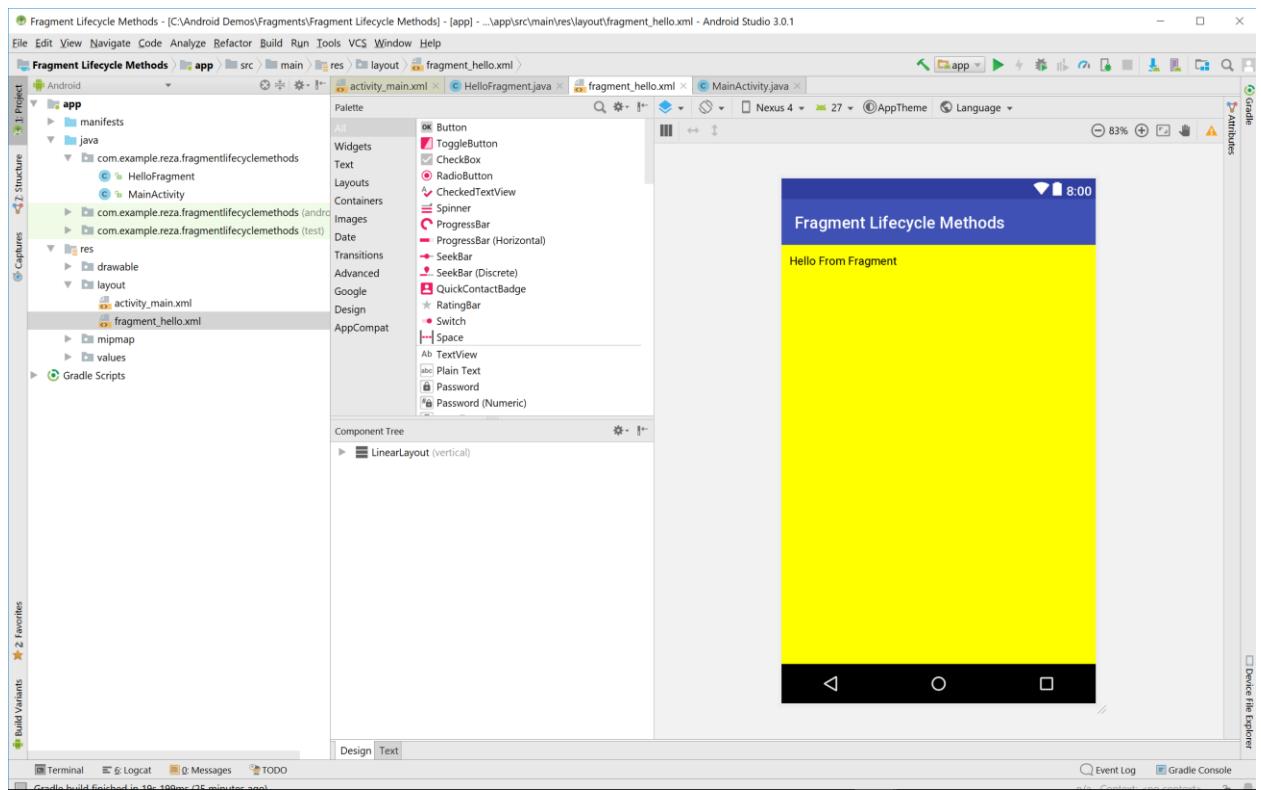
Bottom Navigation: The bottom navigation bar includes "Terminal", "& Logcat", "Messages", "TODO", "Event Log", and "Gradle Console".

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

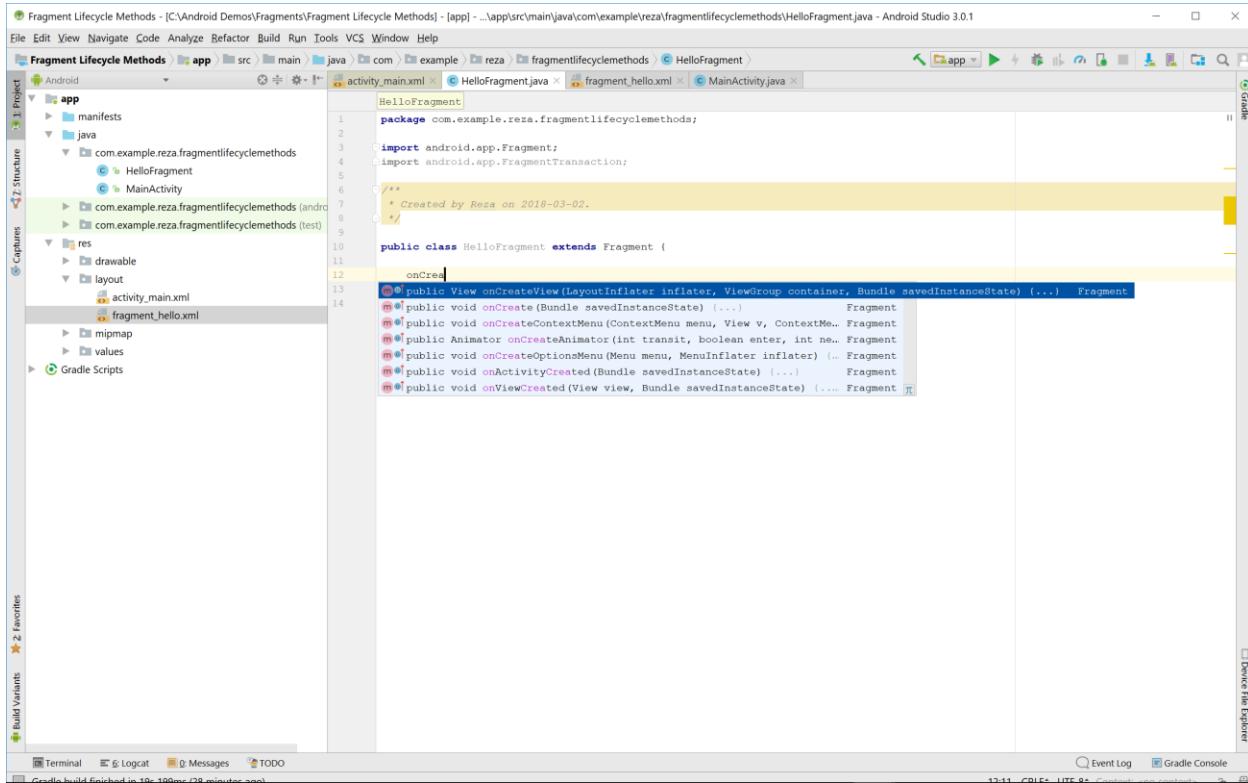


The screenshot shows the Android Studio interface with the project 'Fragment Lifecycle Methods' open. The 'fragment_hello.xml' file is selected in the layout editor. The XML code defines a linear layout with a yellow background and a text view with black text. A yellow lightbulb icon is visible next to the text view, indicating a potential issue or suggestion.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#FF0000"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <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.fragmentlifecyclemethods;
2
3 import android.app.Fragment;
4 import android.app.FragmentManager;
5
6 /**
7  * Created by Reza on 2018-03-02.
8 */
9
10 public class HelloFragment extends Fragment {
11
12     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
13         return null;
14     }
15 }
```

You need to replace the highlighted code as shown:

The screenshot shows the Android Studio interface with the project 'Fragment Lifecycle Methods' open. The code editor displays the 'HelloFragment.java' file under the 'src/main/java/com/example/reza/fragmentlifecyclemethods' package. The code implements the onCreateView() method, which returns the inflated fragment layout. The line 'return super.onCreateView(inflater, container, savedInstanceState);' is highlighted in yellow.

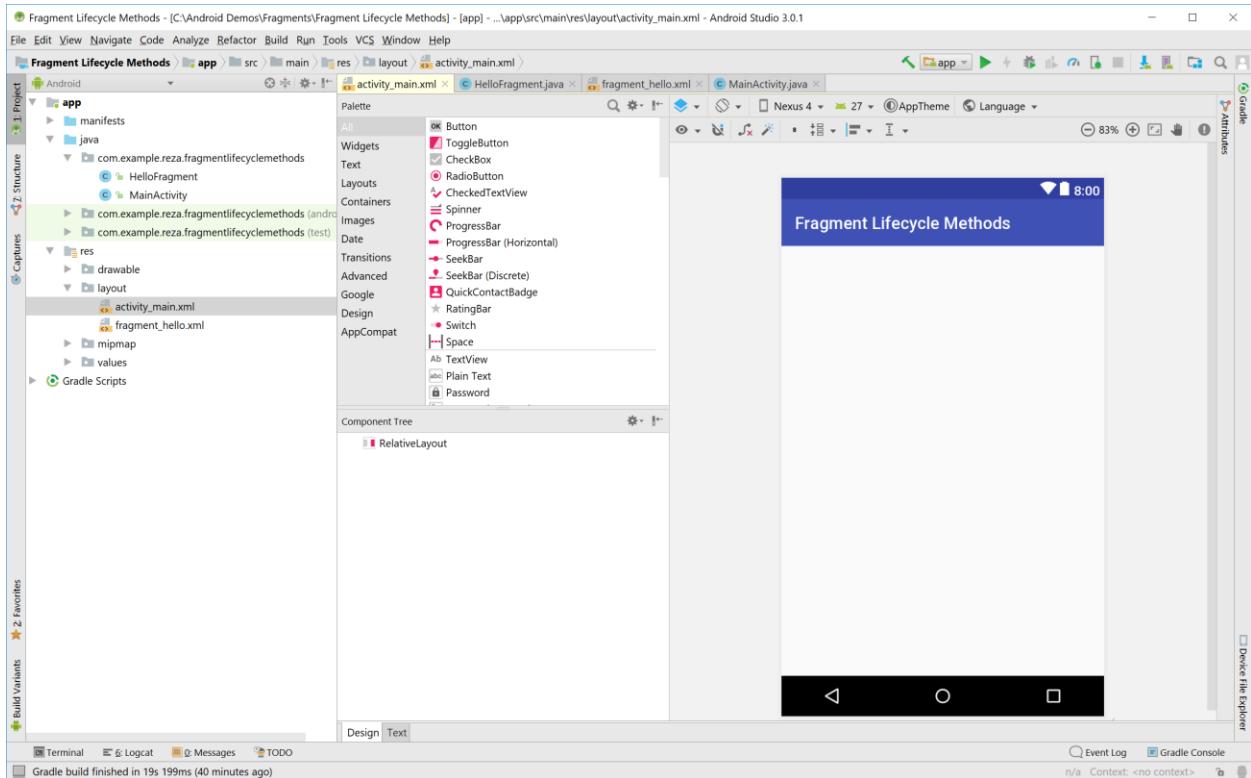
```
1 package com.example.reza.fragmentlifecyclemethods;
2
3 import android.app.Fragment;
4 import android.app.FragmentManager;
5 import android.os.Bundle;
6 import android.support.annotation.Nullable;
7 import android.view.LayoutInflater;
8 import android.view.View;
9 import android.view.ViewGroup;
10
11 /**
12 * Created by Reza on 2018-03-02.
13 */
14
15 public class HelloFragment extends Fragment {
16
17     @Nullable
18     @Override
19     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
20         return super.onCreateView(inflater, container, savedInstanceState);
21     }
22 }
```

The screenshot shows the same Android Studio interface after the modification. The line 'return super.onCreateView(inflater, container, savedInstanceState);' has been replaced with 'return view;'. The code now inflates the fragment layout and returns it directly.

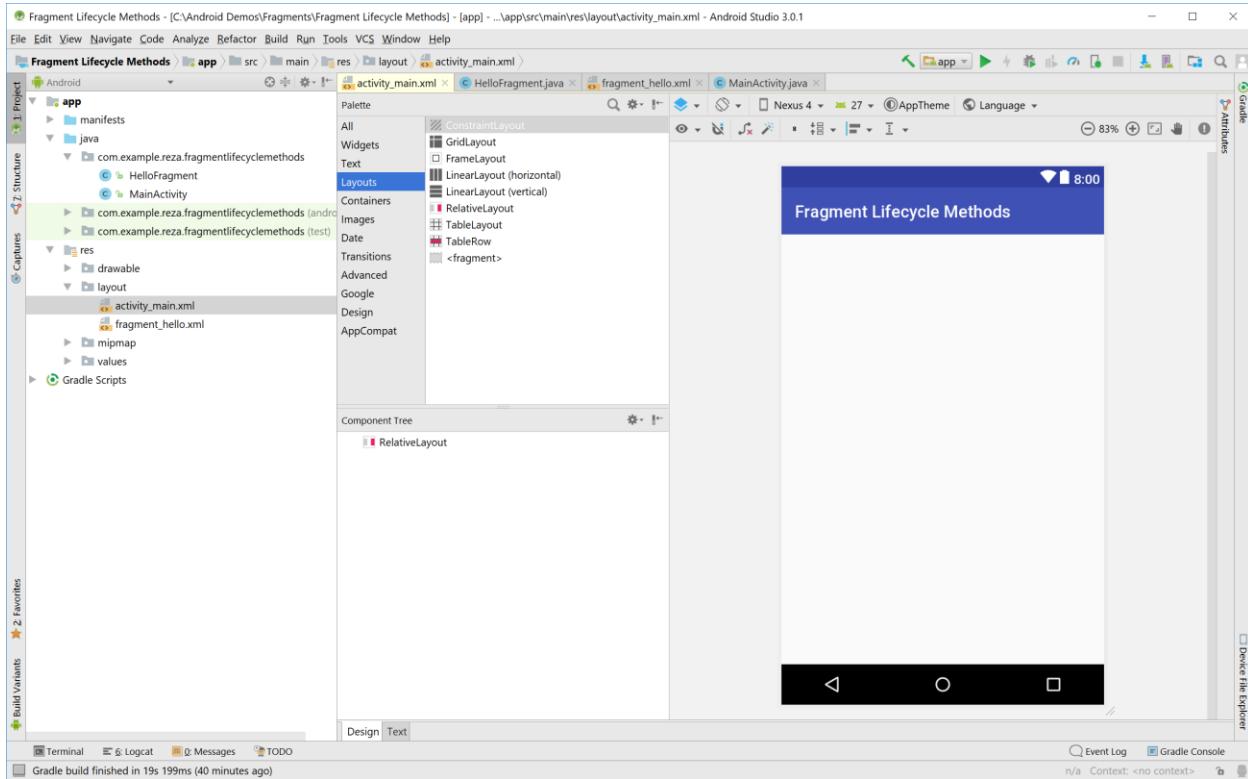
```
1 package com.example.reza.fragmentlifecyclemethods;
2
3 import android.app.Fragment;
4 import android.app.FragmentManager;
5 import android.os.Bundle;
6 import android.support.annotation.Nullable;
7 import android.view.LayoutInflater;
8 import android.view.View;
9 import android.view.ViewGroup;
10
11 /**
12 * Created by Reza on 2018-03-02.
13 */
14
15 public class HelloFragment extends Fragment {
16
17     @Nullable
18     @Override
19     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
20         View view = inflater.inflate(R.layout.fragment_hello, container, attachToRoot: false);
21         return view;
22     }
23 }
```

Next you need to add this fragment to the layout for your main activity. The process is the same as adding a TextView or EditText to the main activity layout.

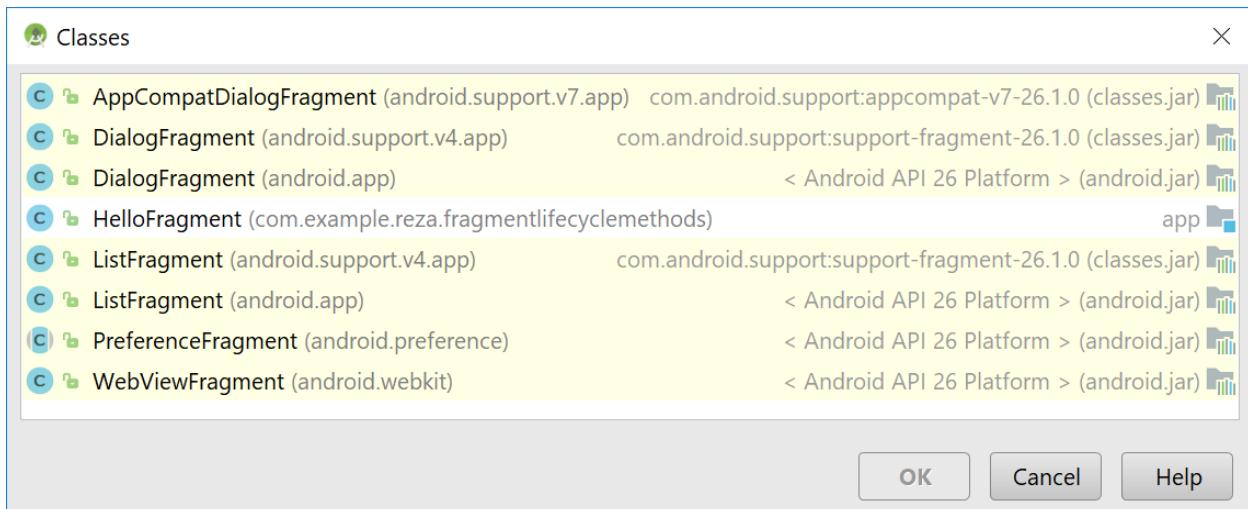
So select activity_main.xml in the design view and under Palette select Layouts and as shown:

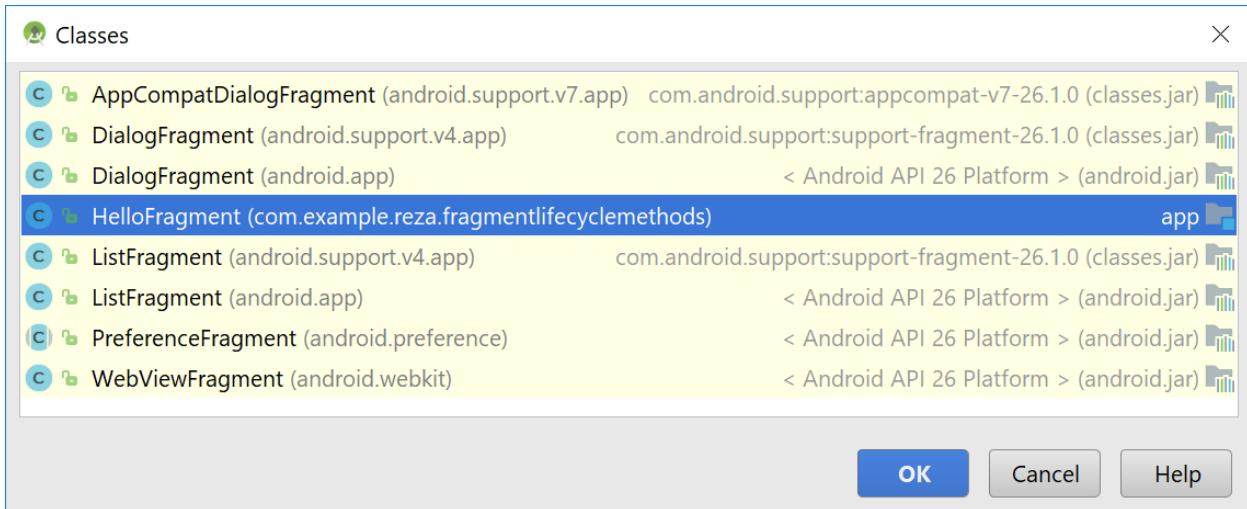


Now select <fragment> and drag into the activity_main.xml file and you will get this pop up that shows you several different fragments as shown:

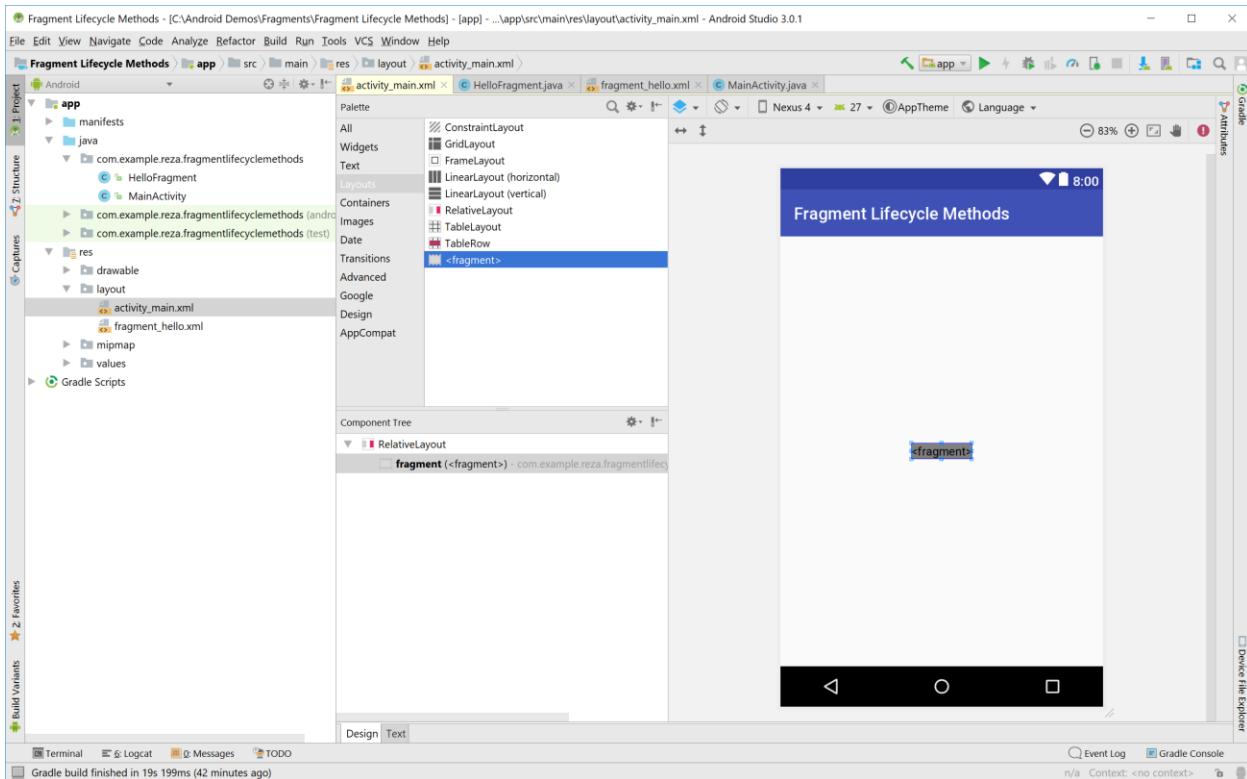


You see you can add many different predefined fragments but what you are interested is the HelloFragment fragment so select HelloFragment and click Ok as shown:

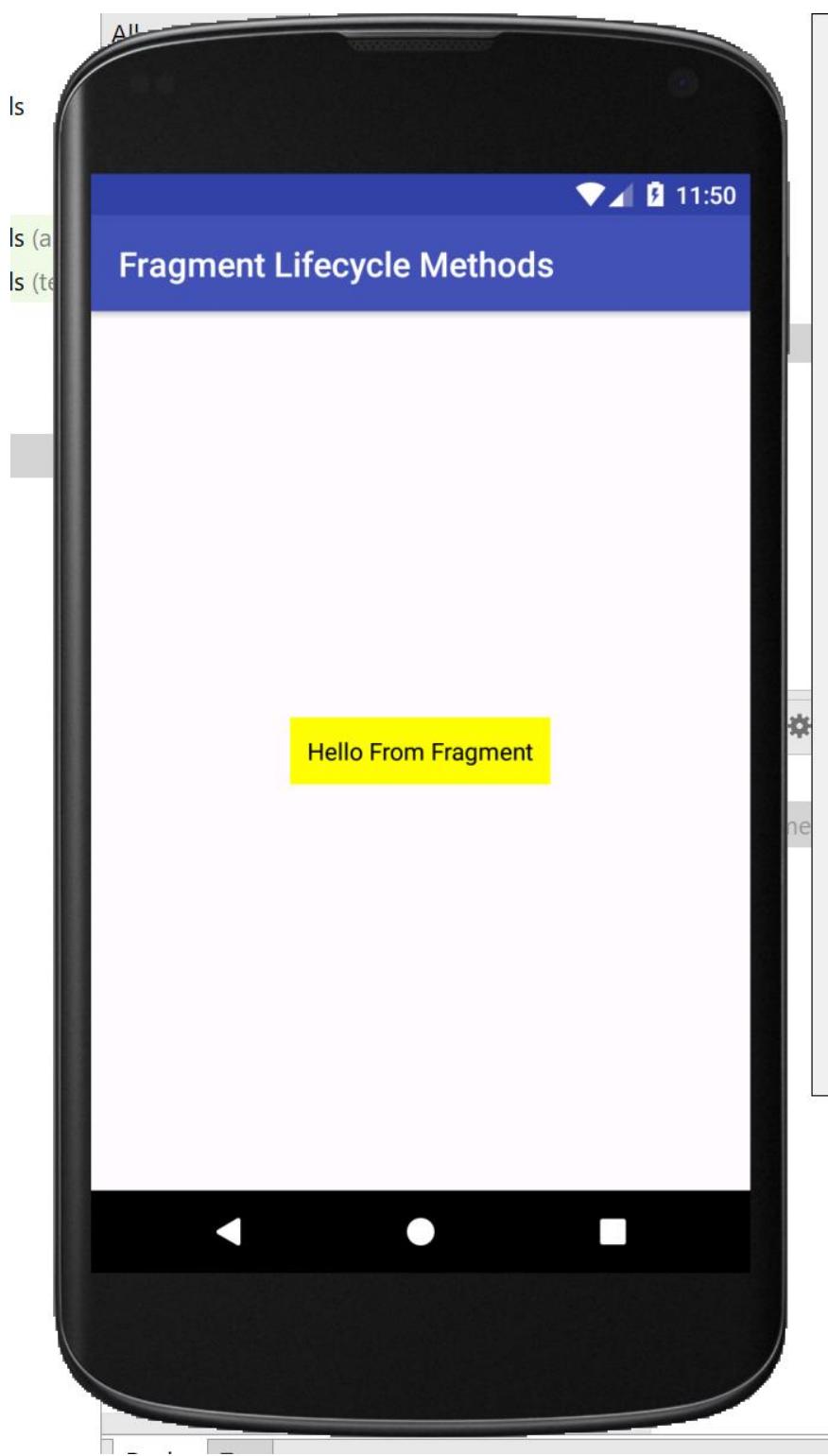




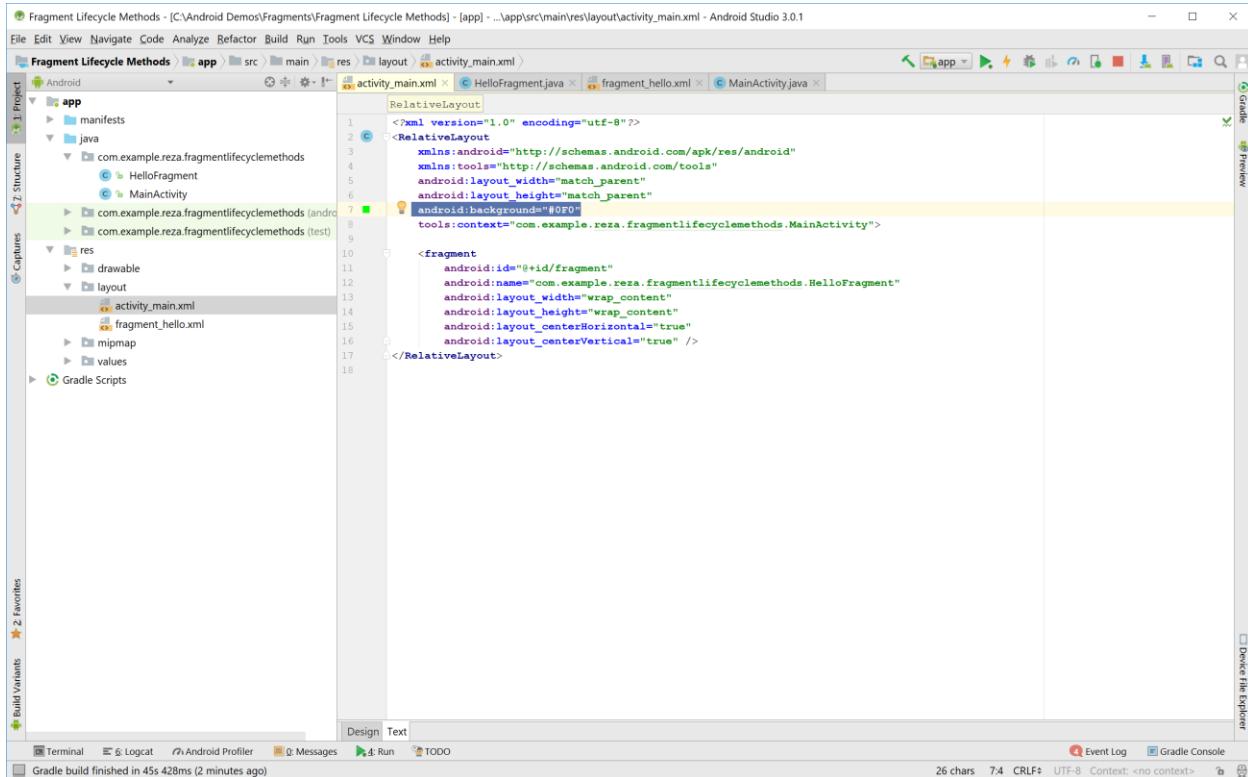
The fragment has been added to the layout file of the main activity. Again you can think of a fragment as another view that you drag to your design of your main activity:



Now if you run the app, you will see your fragment in the activity as shown:



We can change the background color of layout for the main activity to a different color. So add the following to the RelativeLayout tag to assign a background color to the main activity so we are giving a green background to the main activity and our fragment has the yellow background:



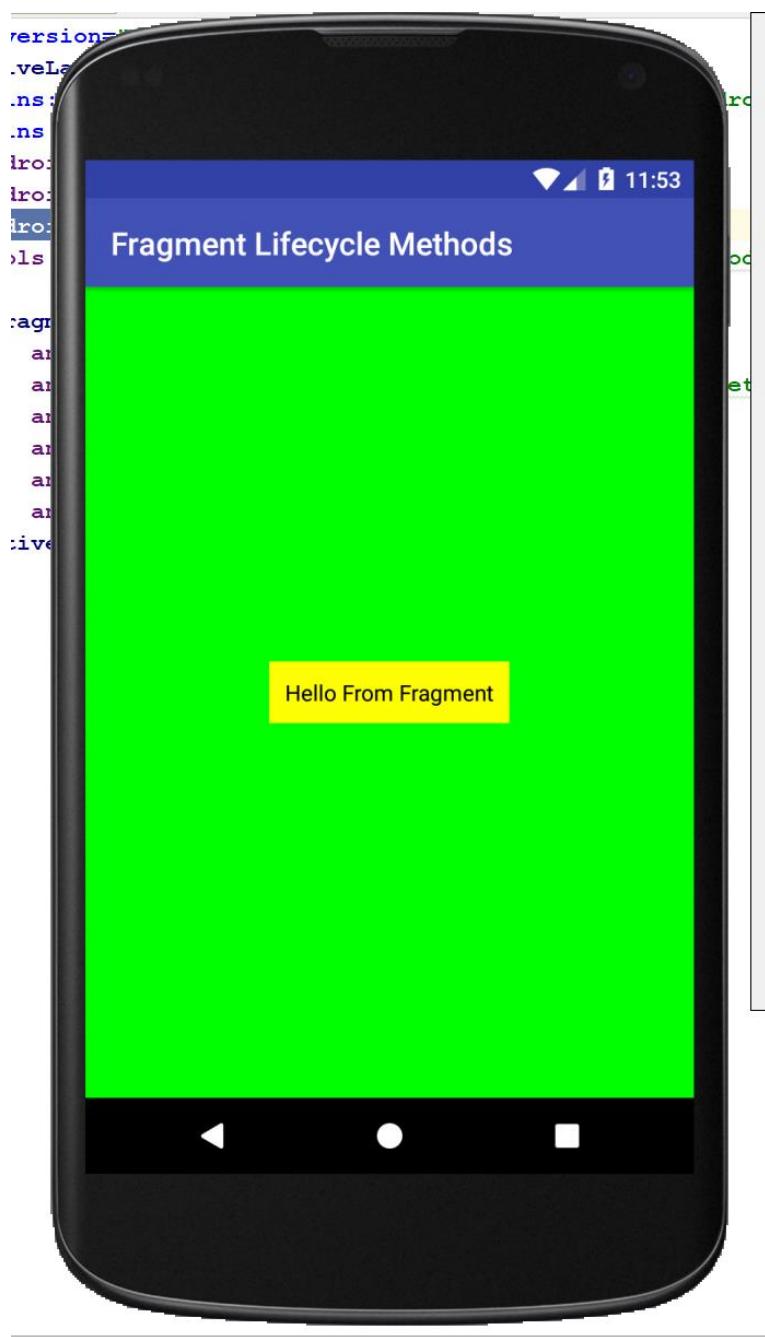
The screenshot shows the Android Studio interface with the project 'Fragment Lifecycle Methods' open. The 'activity_main.xml' file is selected in the 'layout' directory of the 'res' folder. The code editor displays the XML configuration for the main activity's layout:

```
<?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"
    android:background="#00FF00" <-- Line 7, highlighted in yellow
    tools:context="com.example.reza.fragmentlifecyclemethods.MainActivity">

    <fragment
        android:id="@+id/fragment"
        android:name="com.example.reza.fragmentlifecyclemethods.HelloFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true" />
</RelativeLayout>
```

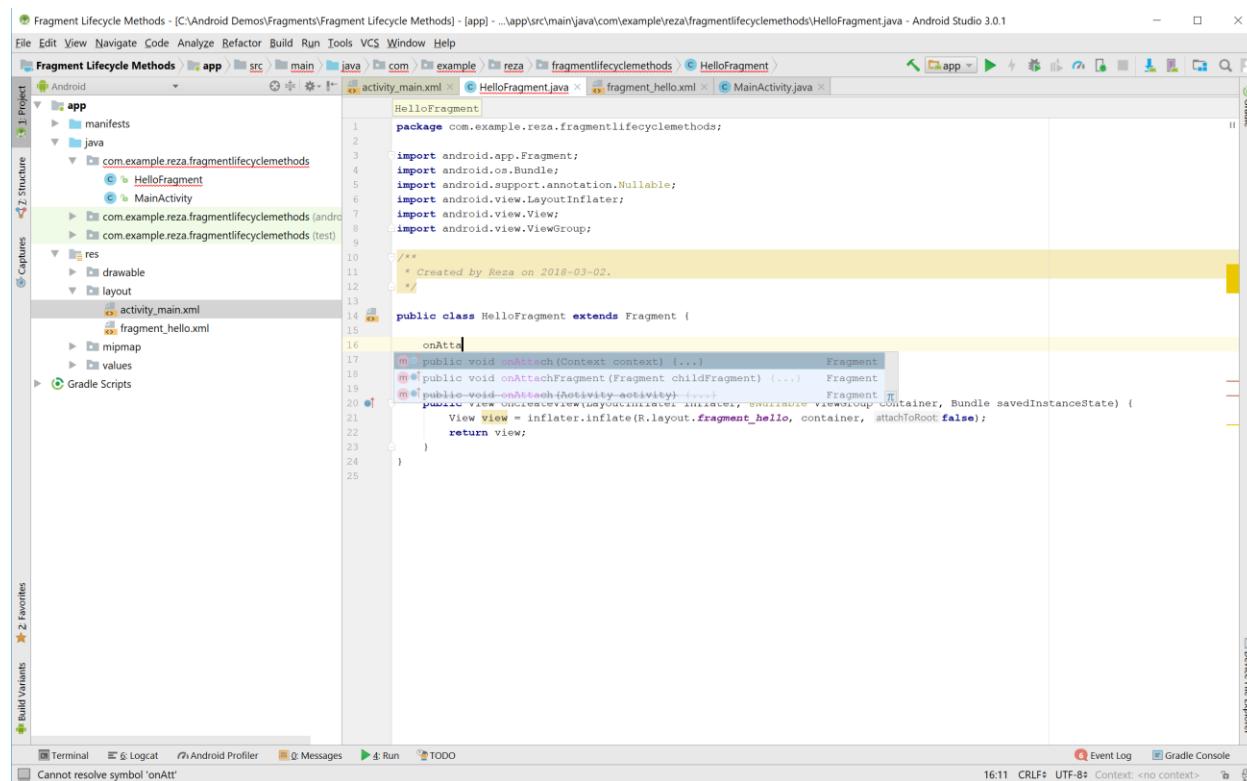
The line `android:background="#00FF00"` is highlighted in yellow, indicating it is the current selection or being edited.

Now you run the app again and you will see your fragment displayed in your main activity. Notice that our fragment with its yellow background contains only one text view which has the option wrap-content for both of its height and width, so it appears the way it does but it could also contain any other view elements.



Now we need to override our methods so go to HelloFragment.java and add the lifecycle methods as shown. Don't forget to add the log statement to each method.

The following screen capture shows the onAttach() method. Note that onAttach() method with the activity argument has been deprecated and you should use the onAttach() method with the context argument. This new method was introduced in API 23, the marshmallow:



```
package com.example.reza.fragmentlifecyclemethods;

import android.app.Fragment;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * Created by Reza on 2018-03-02.
 */

public class HelloFragment extends Fragment {

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
    }

    @Override
    public void onAttachFragment(Fragment childFragment) {
        super.onAttachFragment(childFragment);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_hello, container, false);
        return view;
    }
}
```

The screenshot shows the Android Studio interface with the project 'Fragment Lifecycle Methods' open. The code editor displays the 'HelloFragment.java' file, which contains the following Java code:

```
1 package com.example.reza.fragmentlifecyclemethods;
2
3 import android.app.Fragment;
4 import android.content.Context;
5 import android.os.Bundle;
6 import android.support.annotation.Nullable;
7 import android.view.LayoutInflater;
8 import android.view.View;
9 import android.view.ViewGroup;
10
11 /**
12 * Created by Reza on 2018-03-02.
13 */
14
15 public class HelloFragment extends Fragment {
16
17     @Override
18     public void onAttach(Context context) {
19         super.onAttach(context);
20     }
21
22     @Nullable
23     @Override
24     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
25         View view = inflater.inflate(R.layout.fragment_hello, container, attachToRoot: false);
26         return view;
27     }
28 }
29
30
```

The screenshot shows the same Android Studio interface, but now with code completion suggestions displayed. The cursor is at the end of the 'onCreate' method, and a tooltip shows several methods available for the 'Fragment' type:

- public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.OnCreateContextMenuListener menuInfo)
- public void onCreate(Bundle savedInstanceState) ...
- public Animator onCreateAnimator(int transit, boolean enter, int nextAnim)
- public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) ...
- public void onActivityCreated(Bundle savedInstanceState) ...
- public void onViewCreated(View view, Bundle savedInstanceState) ...

```

1 package com.example.reza.fragmentlifecyclemethods;
2
3 import android.app.Fragment;
4 import android.content.Context;
5 import android.os.Bundle;
6 import android.support.annotation.Nullable;
7 import android.view.LayoutInflater;
8 import android.view.View;
9 import android.view.ViewGroup;
10
11 /**
12 * Created by Reza on 2018-03-02.
13 */
14
15 public class HelloFragment extends Fragment {
16
17     @Override
18     public void onAttach(Context context) {
19         super.onAttach(context);
20     }
21
22     @Override
23     public void onCreate(@Nullable Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25     }
26
27     @Nullable
28     @Override
29     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
30         View view = inflater.inflate(R.layout.fragment_hello, container, attachToRoot=false);
31         return view;
32     }
33 }
34

```

The following screen capture show the completed version of HelloFragment.java:

```

1 package com.example.reza.fragmentlifecyclemethods;
2
3 import android.app.Fragment;
4 import android.content.Context;
5 import android.os.Bundle;
6 import android.support.annotation.Nullable;
7 import android.util.Log;
8 import android.view.LayoutInflater;
9 import android.view.View;
10 import android.view.ViewGroup;
11
12 /**
13 * Created by Reza on 2018-03-02.
14 */
15
16 public class HelloFragment extends Fragment {
17
18     private static final String TAG = HelloFragment.class.getSimpleName();
19
20     @Override
21     public void onAttach(Context context) {
22         super.onAttach(context);
23         Log.i(TAG, msg:"onAttach");
24     }
25
26     @Override
27     public void onCreate(Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29         Log.i(TAG, msg:"onCreate");
30     }
31
32     @Nullable
33     @Override
34     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
35         Log.i(TAG, msg:"onCreateView");
36         View view = inflater.inflate(R.layout.fragment_hello, container, attachToRoot=false);
37         return view;
38     }
39 }
40

```

Fragment Lifecycle Methods - [C:\Android Demos\Fragments\Fragment Lifecycle Methods] - [app] - ...\\app\\src\\main\\java\\com\\example\\reza\\fragmentlifecyclemethods\\HelloFragment.java - Android Studio 3.0.1

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Fragment Lifecycle Methods app src main java com example reza fragmentlifecyclemethods HelloFragment
activity_main.xml HelloFragment.java fragment_hello.xml MainActivity.java
Gradle
Project Structure Captures Favorites Build Variants Device File Explorer
Build Scripts
Terminal Logcat Android Profiler Messages Run TODO Event Log Gradle Console
```

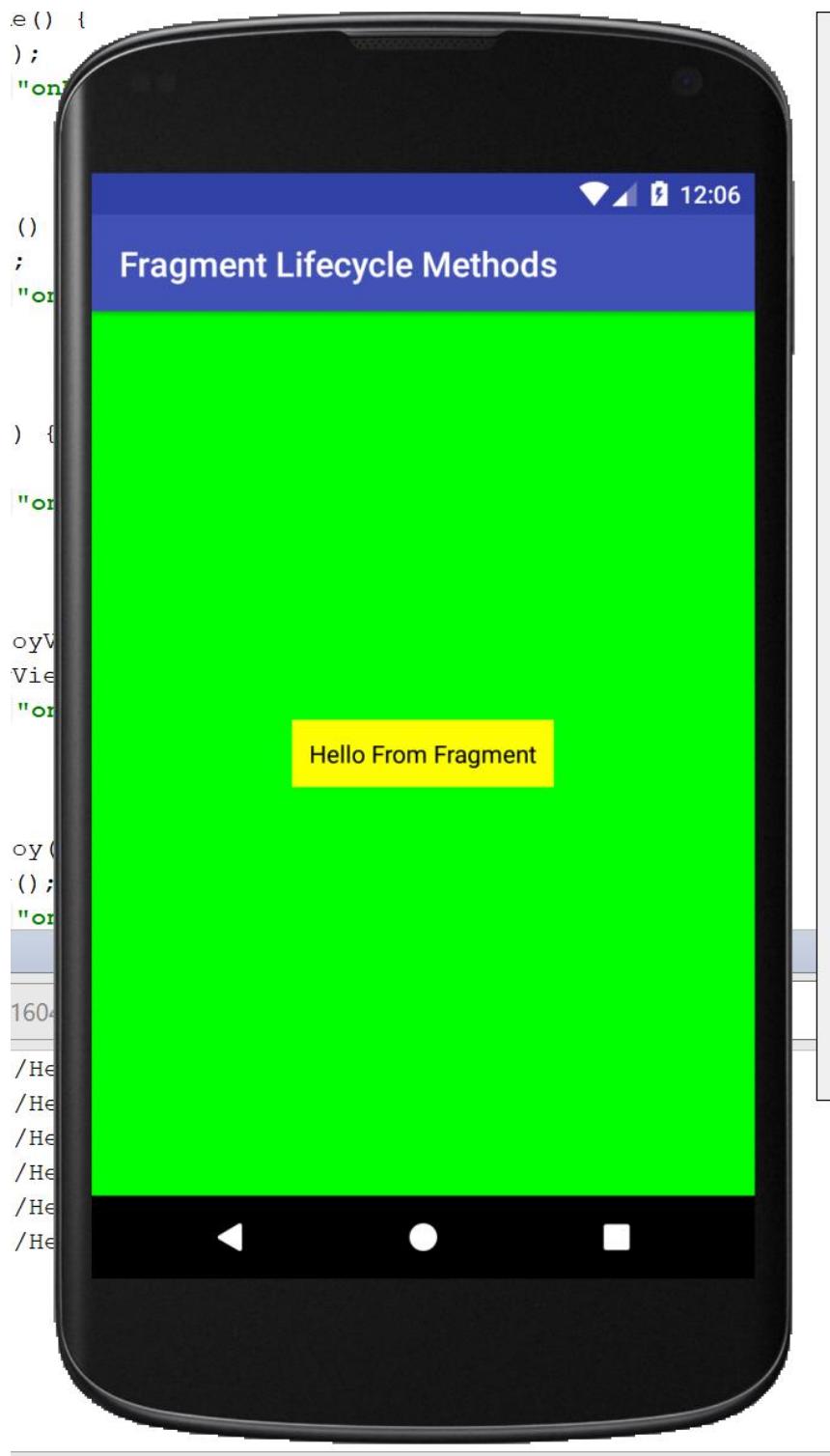
```
31
32     @Nullable
33     @Override
34     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
35         Log.i(TAG, msg: "onCreateView");
36         View view = inflater.inflate(R.layout.fragment_hello, container, attachToRoot: false);
37         return view;
38     }
39     @Override
40     public void onActivityCreated(Bundle savedInstanceState) {
41         super.onActivityCreated(savedInstanceState);
42         Log.i(TAG, msg: "onActivityCreated");
43     }
44     @Override
45     public void onStart() {
46         super.onStart();
47         Log.i(TAG, msg: "onStart");
48     }
49     @Override
50     public void onResume() {
51         super.onResume();
52         Log.i(TAG, msg: "onResume");
53     }
54     @Override
55     public void onPause() {
56         super.onPause();
57         Log.i(TAG, msg: "onPause");
58     }
59     @Override
60     public void onStop() {
61         super.onStop();
62         Log.i(TAG, msg: "onStop");
63     }
64     @Override
65     public void onDestroyView() {
66         super.onDestroyView();
67         Log.i(TAG, msg: "onDestroyView");
68     }
69
70     @Override
71     public void onDestroy() {
72         super.onDestroy();
73         Log.i(TAG, msg: "onDestroy");
74     }
75
76     @Override
77     public void onDetach() {
78         super.onDetach();
79         Log.i(TAG, msg: "onDetach");
80     }
81
82     @Override
83     public void onAttach() {
84         super.onAttach();
85         Log.i(TAG, msg: "onAttach");
86     }
87 }
```

Fragment Lifecycle Methods - [C:\Android Demos\Fragments\Fragment Lifecycle Methods] - [app] - ...\\app\\src\\main\\java\\com\\example\\reza\\fragmentlifecyclemethods\\HelloFragment.java - Android Studio 3.0.1

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Fragment Lifecycle Methods app src main java com example reza fragmentlifecyclemethods HelloFragment
activity_main.xml HelloFragment.java fragment_hello.xml MainActivity.java
Gradle
Project Structure Captures Favorites Build Variants Device File Explorer
Build Scripts
Terminal Logcat Android Profiler Messages Run TODO Event Log Gradle Console
```

```
51
52     @Override
53     public void onResume() {
54         super.onResume();
55         Log.i(TAG, msg: "onResume");
56     }
57
58     @Override
59     public void onPause() {
60         super.onPause();
61         Log.i(TAG, msg: "onPause");
62     }
63
64     @Override
65     public void onStop() {
66         super.onStop();
67         Log.i(TAG, msg: "onStop");
68     }
69
70     @Override
71     public void onDestroyView() {
72         super.onDestroyView();
73         Log.i(TAG, msg: "onDestroyView");
74     }
75
76     @Override
77     public void onDestroy() {
78         super.onDestroy();
79         Log.i(TAG, msg: "onDestroy");
80     }
81
82     @Override
83     public void onDetach() {
84         super.onDetach();
85         Log.i(TAG, msg: "onDetach");
86     }
87 }
```

Now run the app:



And examine the logcat window, use the filter “Hello” as shown. Notice the order that the fragment lifecycle methods are called:

```

@Override
public void onResume() {
    super.onResume();
    Log.i(TAG, msg: "onResume");
}

@Override
public void onPause() {
    super.onPause();
    Log.i(TAG, msg: "onPause");
}

@Override
public void onStop() {
    super.onStop();
    Log.i(TAG, msg: "onStop");
}

@Override
public void onDestroyView() {
    super.onDestroyView();
    Log.i(TAG, msg: "onDestroyView");
}

@Override
public void onDestroy() {
    super.onDestroy();
    Log.i(TAG, msg: "onDestroy");
}

```

Logcat output (Emulator Nexus_4_API_25 Android 7.1.1, API 25) for com.example.reza.fragmentlifecyclemethods (11604):

```

03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onAttach
03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onCreate
03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onCreateView
03-02 11:54:14.818 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onActivityCreated
03-02 11:54:14.819 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onStart
03-02 11:54:14.819 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onResume

```

```

@Override
public void onResume() {
    super.onResume();
    Log.i(TAG, msg: "onResume");
}

@Override
public void onPause() {
    super.onPause();
    Log.i(TAG, msg: "onPause");
}

@Override
public void onStop() {
    super.onStop();
    Log.i(TAG, msg: "onStop");
}

@Override
public void onDestroyView() {
    super.onDestroyView();
    Log.i(TAG, msg: "onDestroyView");
}

@Override
public void onDestroy() {
    super.onDestroy();
    Log.i(TAG, msg: "onDestroy");
}

```

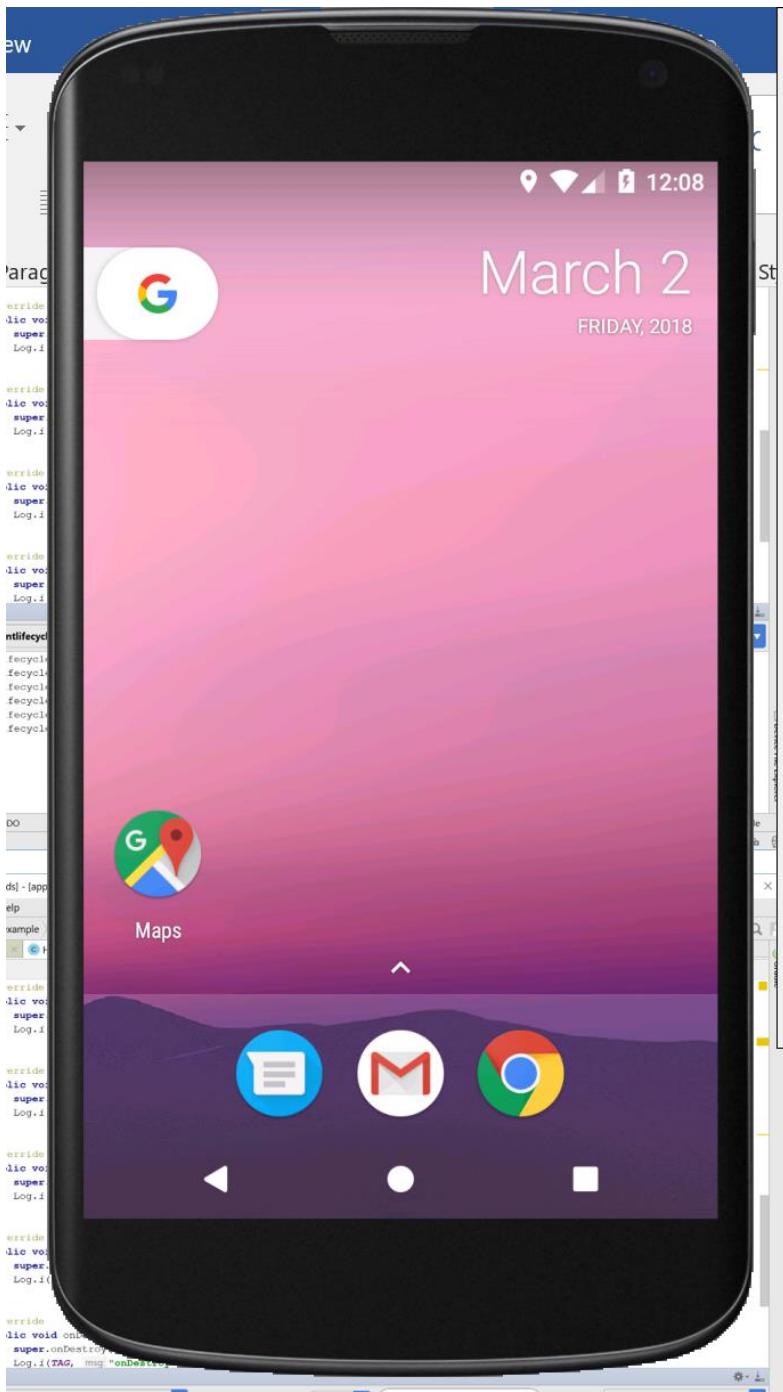
Logcat output (Emulator Nexus_4_API_25 Android 7.1.1, API 25) for com.example.reza.fragmentlifecyclemethods (11604):

```

03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onAttach
03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onCreate
03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onCreateView
03-02 11:54:14.818 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onActivityCreated
03-02 11:54:14.819 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onStart
03-02 11:54:14.819 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onResume

```

Now press the back button on your device (the simulator) as shown:



Examine the lifecycle methods that were called. I have highlighted them for easy reference:

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

- Project Structure:** Shows the project tree with modules: app, com.example.reza.fragmentlifecyclemethods, and com.example.reza.fragmentlifecyclemethods (test).
- Code Editor:** Displays `HelloFragment.java` containing lifecycle methods:

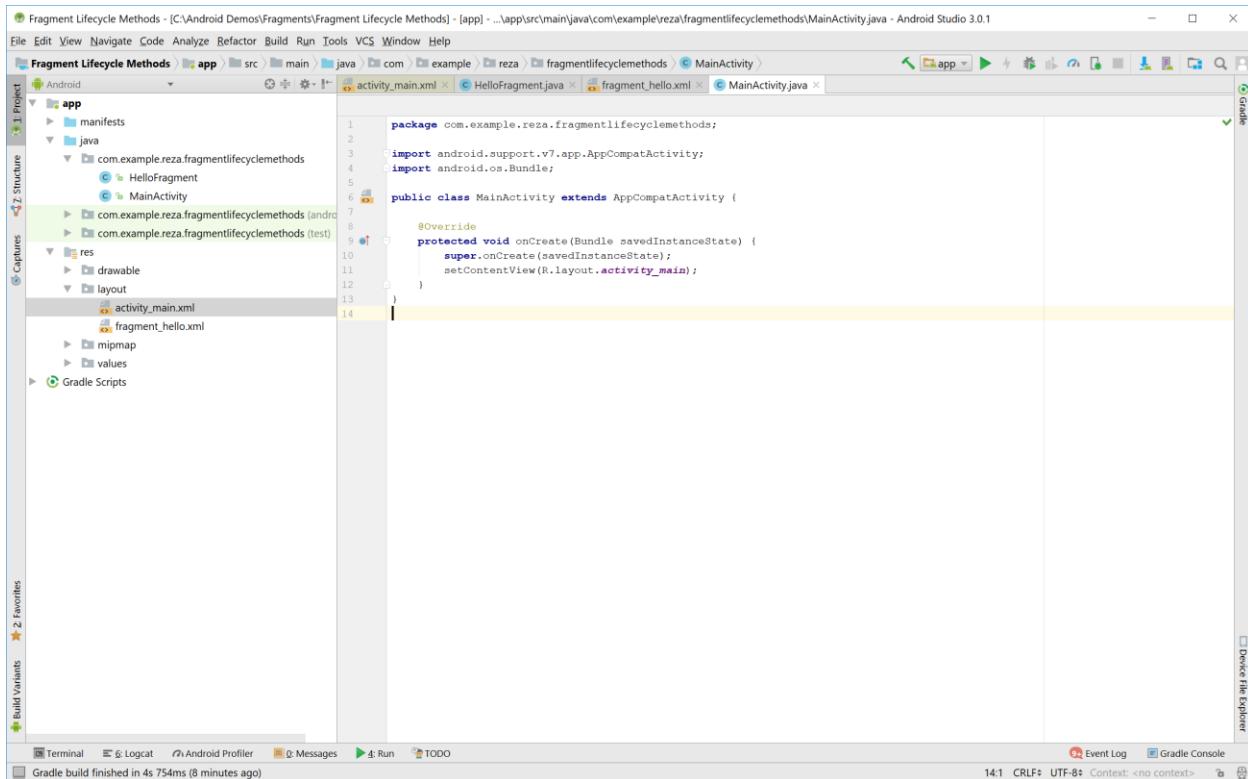
```
51     @Override
52     public void onResume() {
53         super.onResume();
54         Log.i(TAG, msg: "onResume");
55     }
56
57     @Override
58     public void onPause() {
59         super.onPause();
60         Log.i(TAG, msg: "onPause");
61     }
```
- Logcat:** Shows log entries for the Emulator Nexus_4_API_25 device. The log highlights several lifecycle events:

```
03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onAttach
03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onCreate
03-02 11:54:14.817 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onCreateView
03-02 11:54:14.818 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onActivityCreated
03-02 11:54:14.819 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onStart
03-02 11:54:14.819 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onResume
03-02 12:08:23.092 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onPause
03-02 12:08:24.136 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onStop
03-02 12:08:24.147 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onDestoryView
03-02 12:08:24.147 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onDestory
03-02 12:08:24.147 11604-11604/com.example.reza.fragmentlifecyclemethods I/HelloFragment: onDetach
```
- Bottom Status Bar:** Shows "Gradle build finished in 4s 754ms (3 minutes ago)".

Examining the relationship between the fragment lifecycle methods and its host activity lifecycle methods

Now to examine the relationship between the Fragment lifecycle methods and the Activity lifecycle methods. To do that we are going to modify our MainActivity.java with adding its lifecycle methods as shown:

This is how currently our MainActivity.java looks like:



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

```
package com.example.reza.fragmentlifecyclemethods;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

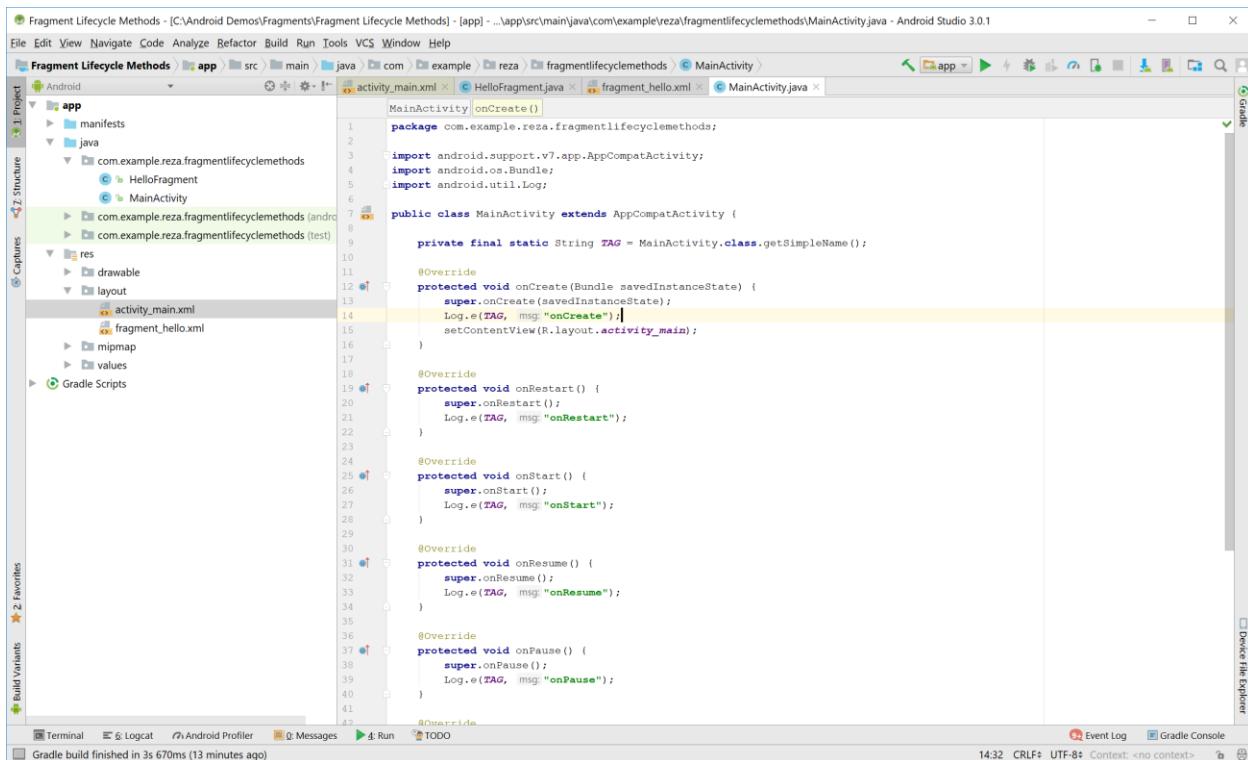
public class MainActivity extends AppCompatActivity {

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

The Project tool window on the left shows the app module structure with Java files like HelloFragment and MainActivity, and XML files like activity_main.xml and fragment_hello.xml. The bottom navigation bar includes Terminal, Logcat, Android Profiler, Messages, Run, and TODO.

The following screen capture shows the completed MainActivity.java file:

Notice that we are using Log.e (e stands for error) for the Activity lifecycle methods versus Log.i (i stands for info) which we used for Fragment lifecycle methods. It is just so we can easily differentiate which one is which.



A screenshot of the Android Studio interface showing the code editor for MainActivity.java. The code implements an AppCompatActivity and overrides several lifecycle methods. The 'onCreate' method uses Log.e for its message. The other methods ('onRestart', 'onStart', 'onResume', 'onPause') also use Log.e. The code is as follows:

```
package com.example.reza.fragmentlifecyclemethods;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    private final static String TAG = MainActivity.class.getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.e(TAG, msg: "onCreate");
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.e(TAG, msg: "onRestart");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.e(TAG, msg: "onStart");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.e(TAG, msg: "onResume");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.e(TAG, msg: "onPause");
    }
}
```

Fragment Lifecycle Methods - [C:\Android Demos\Fragments\Fragment Lifecycle Methods] - [app] - ...\\app\\src\\main\\java\\com\\example\\reza\\fragmentlifecyclemethods\\MainActivity.java - Android Studio 3.0.1

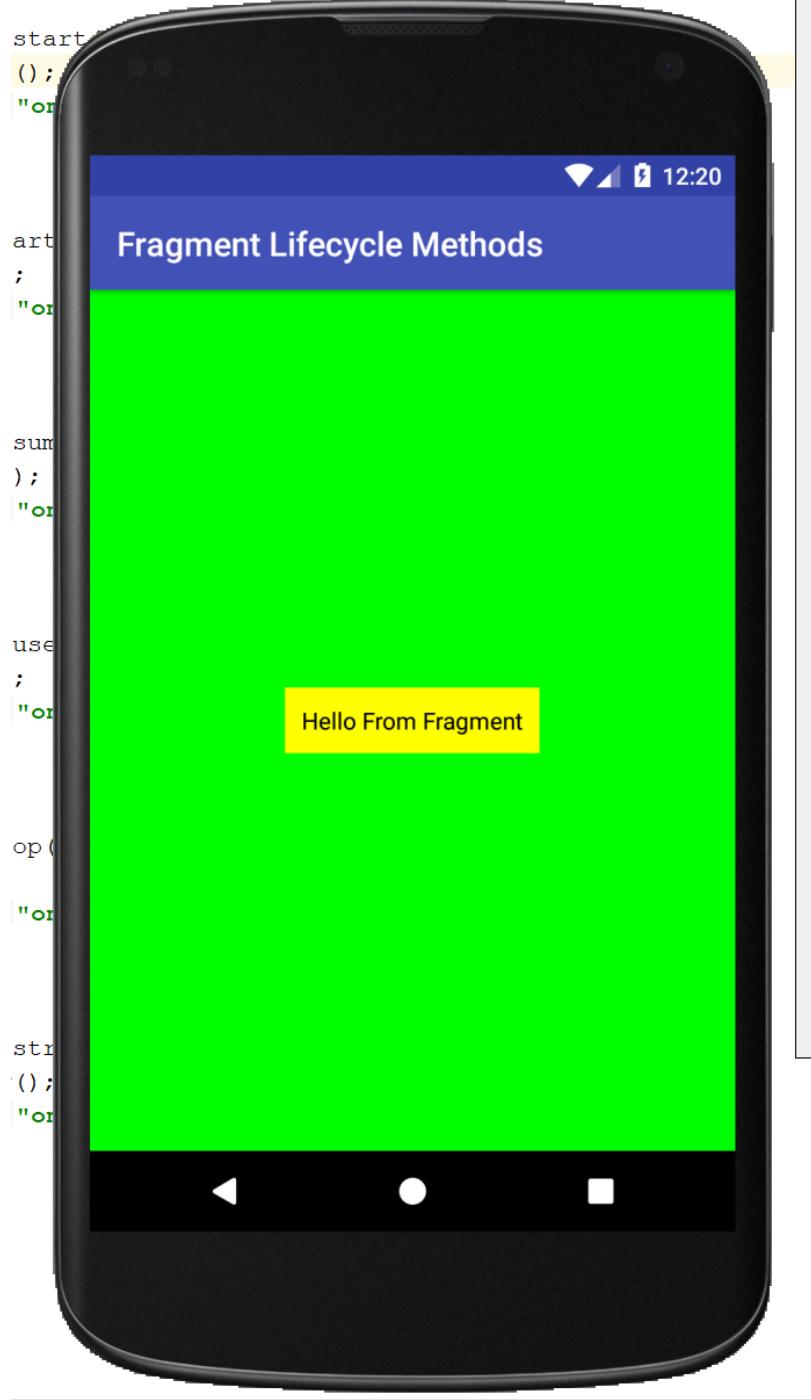
```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Fragment Lifecycle Methods app src main java com example com.example.reza.fragmentlifecyclemethods MainActivity
activity_main.xml HelloFragment fragment_hello.xml MainActivity.java
18 @Override
19 protected void onCreate() {
20     super.onCreate();
21     Log.e(TAG, msg: "onCreate");
22 }
23
24 @Override
25 protected void onRestart() {
26     super.onRestart();
27     Log.e(TAG, msg: "onRestart");
28 }
29
30 @Override
31 protected void onStart() {
32     super.onStart();
33     Log.e(TAG, msg: "onStart");
34 }
35
36 @Override
37 protected void onResume() {
38     super.onResume();
39     Log.e(TAG, msg: "onResume");
40 }
41
42 @Override
43 protected void onPause() {
44     super.onPause();
45     Log.e(TAG, msg: "onPause");
46 }
47
48 @Override
49 protected void onStop() {
50     super.onStop();
51     Log.e(TAG, msg: "onStop");
52 }
53
54 }
```

1 Project 2 Favorites 3 Build Variants

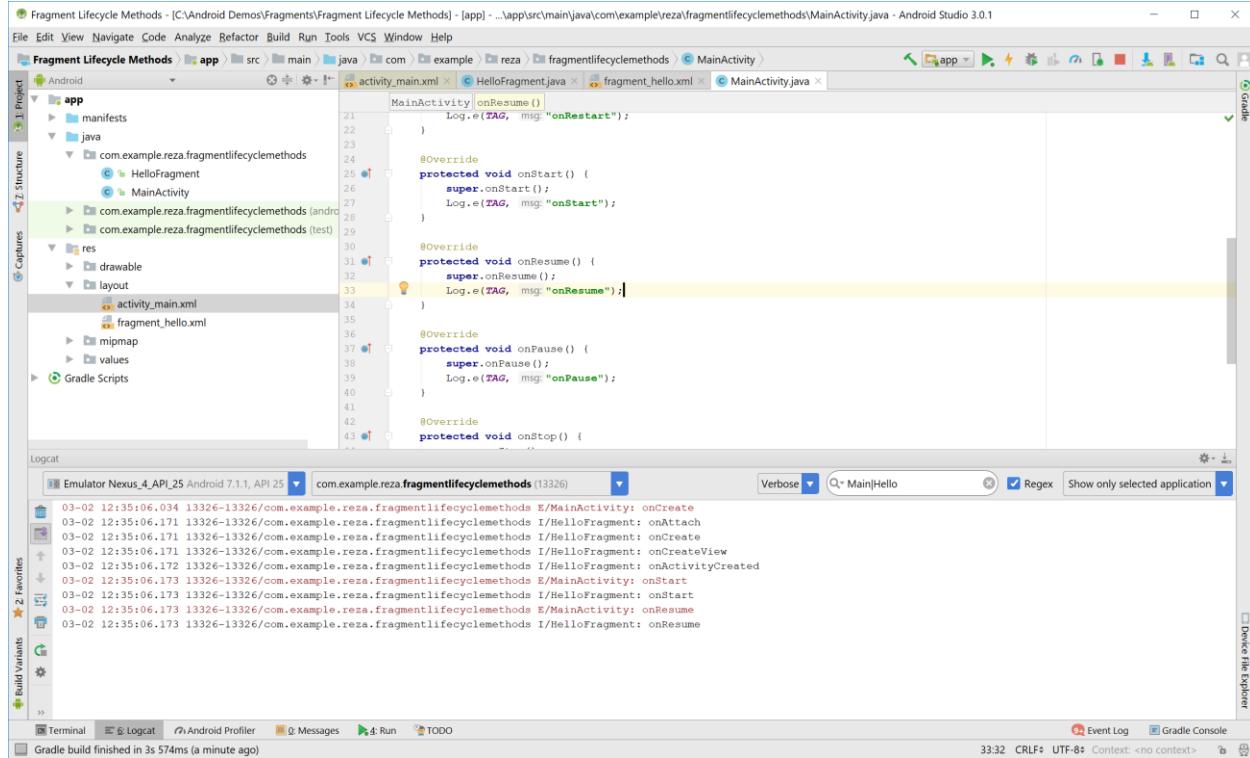
Device File Explorer Event Log Gradle Console

Gradle build finished in 3s 670ms (13 minutes ago)

Now run the app so we can examine the methods that were called:

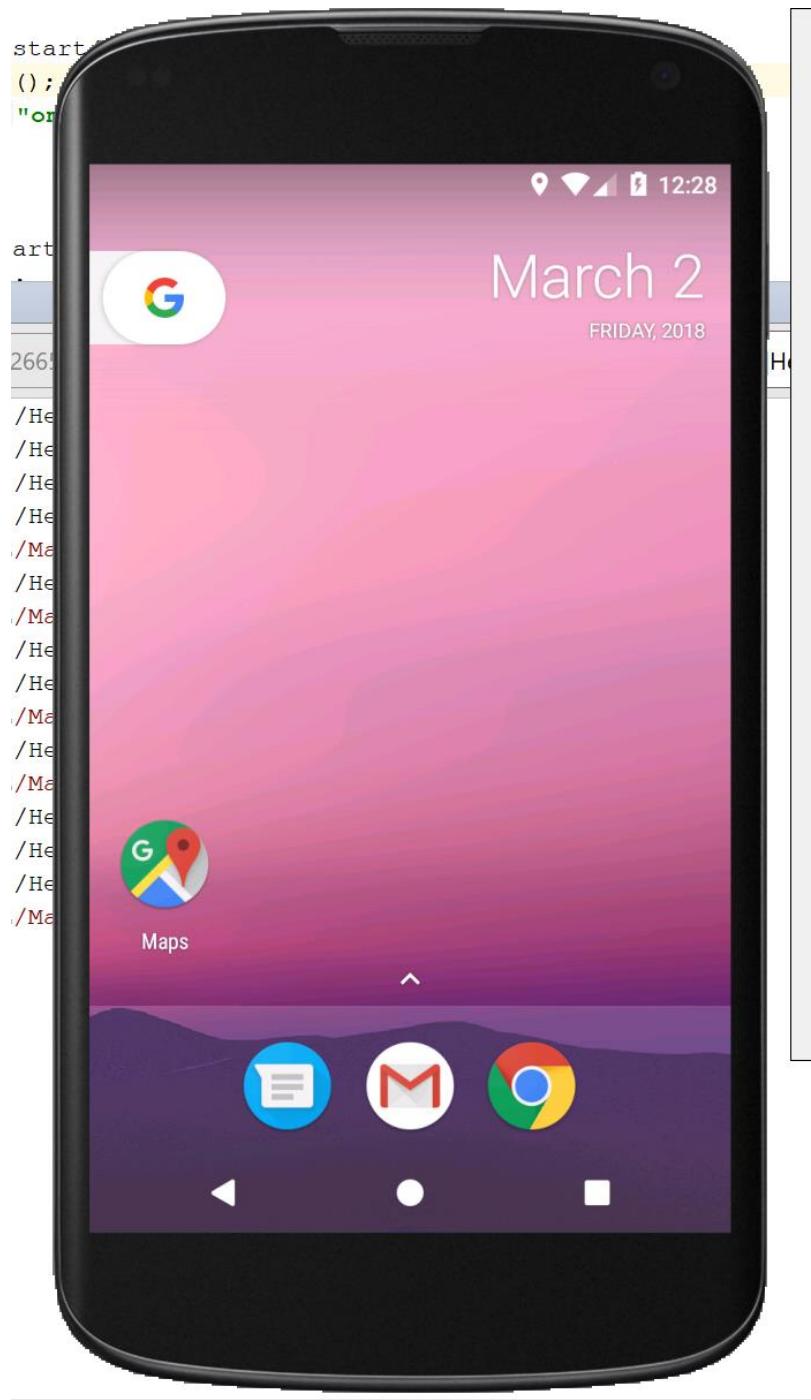


Now examine the Logcat. Notice that the filter we are using is “Main|Hello” (that’s the word Main followed by the pipe symbol followed by the word Hello) where for examining the Fragment lifecycle methods we used “Hello” filter:



Notice that since we used `Log.e` instead of `Log.i` and `Log.e` is for reporting errors, the color used is red and makes it easier to see the methods called belong to the activity or the fragment. It should be easy now to see how the fragment and the activity lifecycle methods are related and in which order they are being called.

Now press the back button on the phone as shown:



```
MainActivity#onCreate()
    Log.e(TAG, msg: "onCreate");

@Override
protected void onStart() {
    super.onStart();
    Log.e(TAG, msg: "onStart");
}

@Override
protected void onResume() {
    super.onResume();
    Log.e(TAG, msg: "onResume");
}
```

Emulator Nexus_4 API_25 Android 7.1.1, API 25 com.example.reza.fragmentlifeclemethods (13326) Verbose Q MainHello Regex Show only selected application

```
03-02 12:35:06.034 13326-13326/com.example.reza.fragmentlifeclemethods E/MainActivity: onCreate
03-02 12:35:06.171 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onAttach
03-02 12:35:06.171 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onCreate
03-02 12:35:06.171 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onCreateView
03-02 12:35:06.172 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onActivityCreated
03-02 12:35:06.173 13326-13326/com.example.reza.fragmentlifeclemethods I>MainActivity: onStart
03-02 12:35:06.173 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onStart
03-02 12:35:06.173 13326-13326/com.example.reza.fragmentlifeclemethods I>MainActivity: onResume
03-02 12:35:06.173 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onResume
03-02 12:35:57.494 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onPause
03-02 12:35:58.581 13326-13326/com.example.reza.fragmentlifeclemethods E>MainActivity: onPause
03-02 12:35:58.581 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onStop
03-02 12:35:58.581 13326-13326/com.example.reza.fragmentlifeclemethods E>MainActivity: onStop
03-02 12:35:58.581 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onDestoryView
03-02 12:35:58.581 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onDestory
03-02 12:35:58.581 13326-13326/com.example.reza.fragmentlifeclemethods I>HelloFragment: onDetach
03-02 12:35:58.587 13326-13326/com.example.reza.fragmentlifeclemethods E>MainActivity: onDestory
```

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

- Project Structure:** The project is named "Fragment Lifecycle Methods". It contains an "app" module with Java, XML, and resources. The Java folder includes "HelloFragment" and "MainActivity".
- MainActivity.java:** The code shows the implementation of the fragment lifecycle methods. It includes annotations like `@Override` and `@Protected`. The log output below shows the execution of these methods.
- Logcat:** The logcat window displays a series of log entries from the emulator. The entries show the sequence of lifecycle events: `onCreate`, `onAttach`, `onCreateView`, `onActivityCreated`, `onStart`, `onResume`, `onPause`, `onStop`, `onDestroyView`, `onStop`, `onDestroy`, and `onDetach`. The log entries are timestamped at 03-02 12:35:06.173.