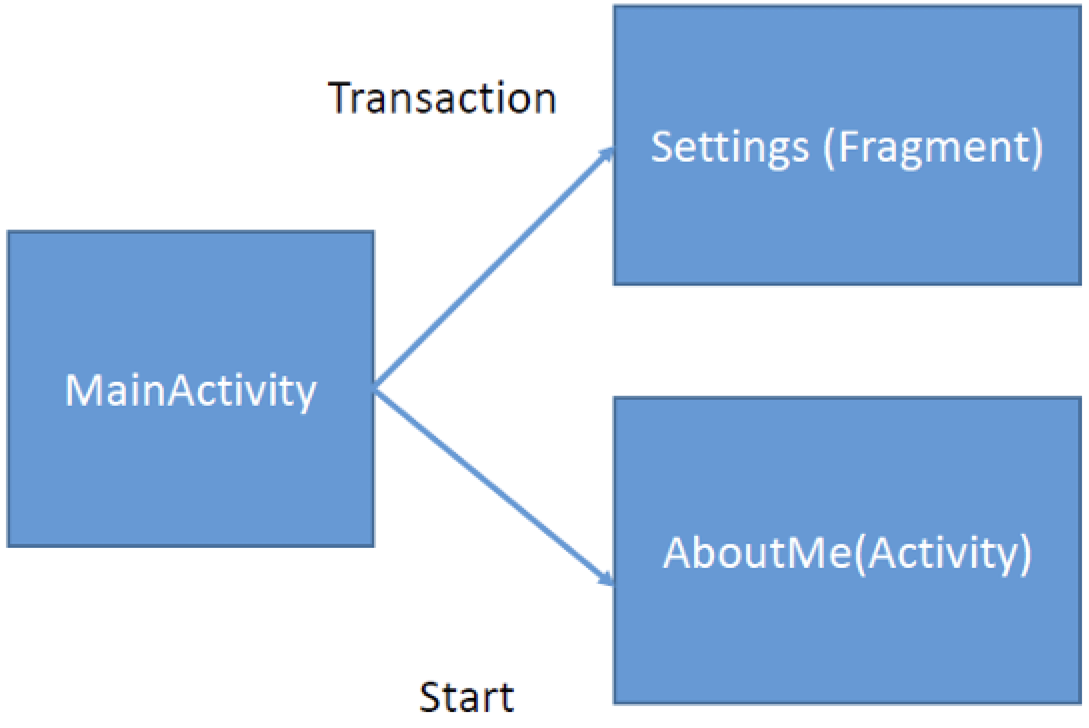
Android Programing 101

Dr. Charles Yu

Outlines

- [Demo2] Activity and Fragment



[Demo2] Activity and Fragment

• Open the project in Android Studio

"ActivityandFragment"

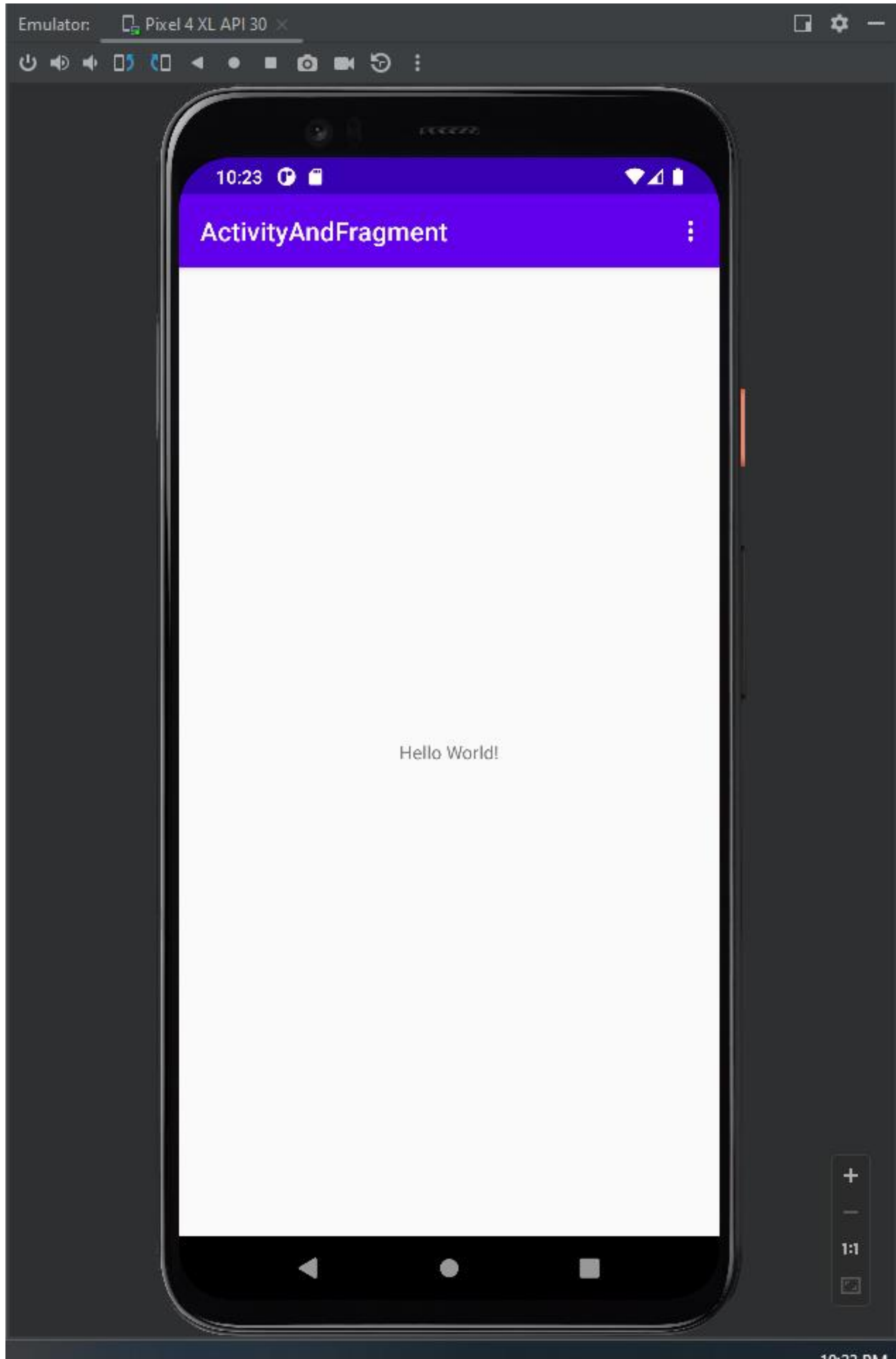
• Turn on the Android virtual OS in

Device Manager

• Toolbar → Run → Run 'app',

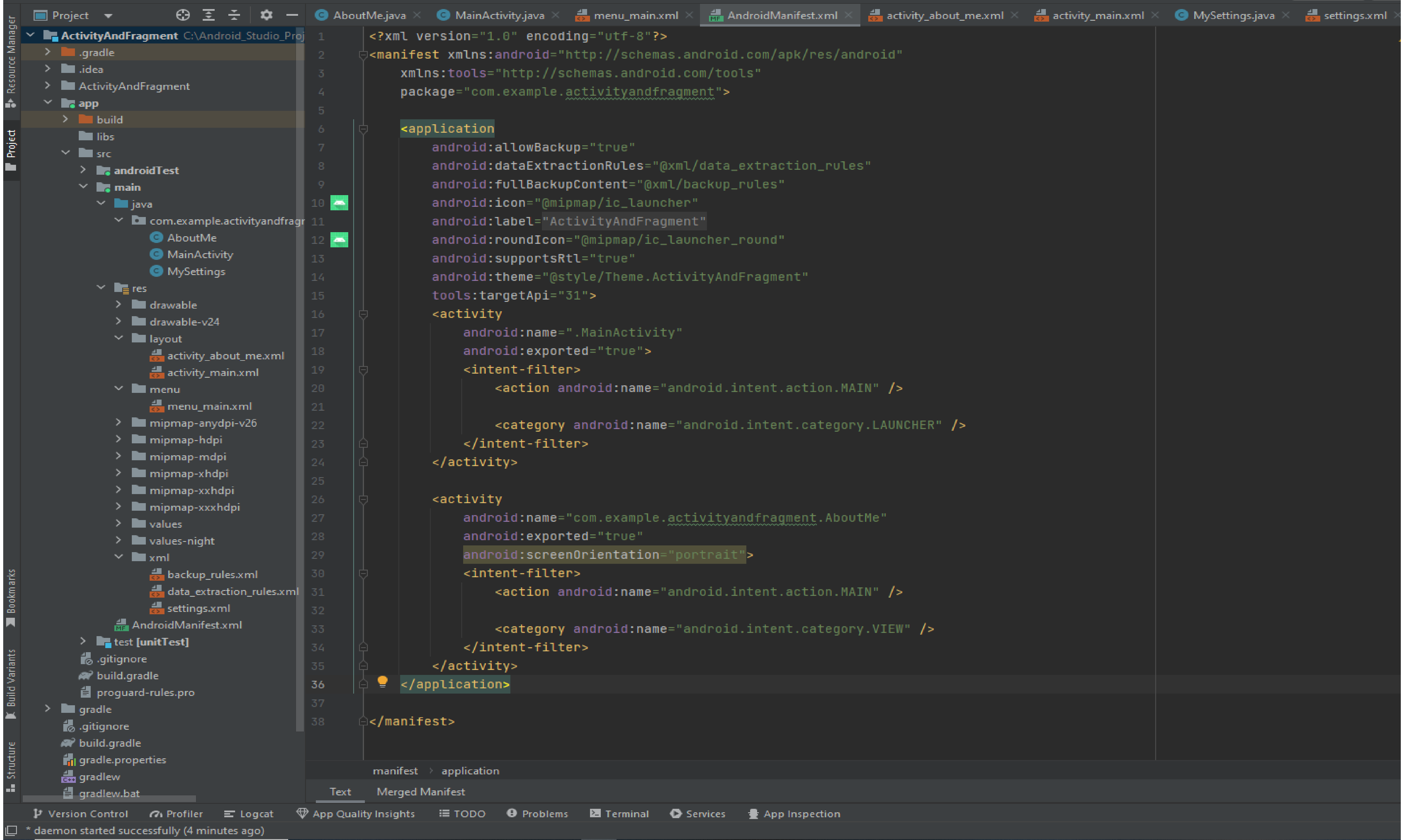
to run this app in the Android virtual OS

• Now, we just stay here temporarily



AndroidManifest.xml

- If I were you, the 1st thing for me to trace someone’s Android code is to see the AndroidManifest.xml
- Check the next page
- When we see the “Launcher” in some “activity”, we realized that “activity” is the 1st launching activity in the App. (when you click the icon)
- The “**android:name**” attribute, it can be in 2 different style
  - Class name: i.e. “.MainActivity”
  - Full name: i.e. “com.example.activityandfragment.AboutMe”
- The “screenOrientation” attribute, it can be “portrait” or “landscape”
  - The later is very good for some special purpose. i.e. Gaming, or showing statistical charts
- OK. I know there are two activities are in this app --- “MainActivity” and “AboutMe”

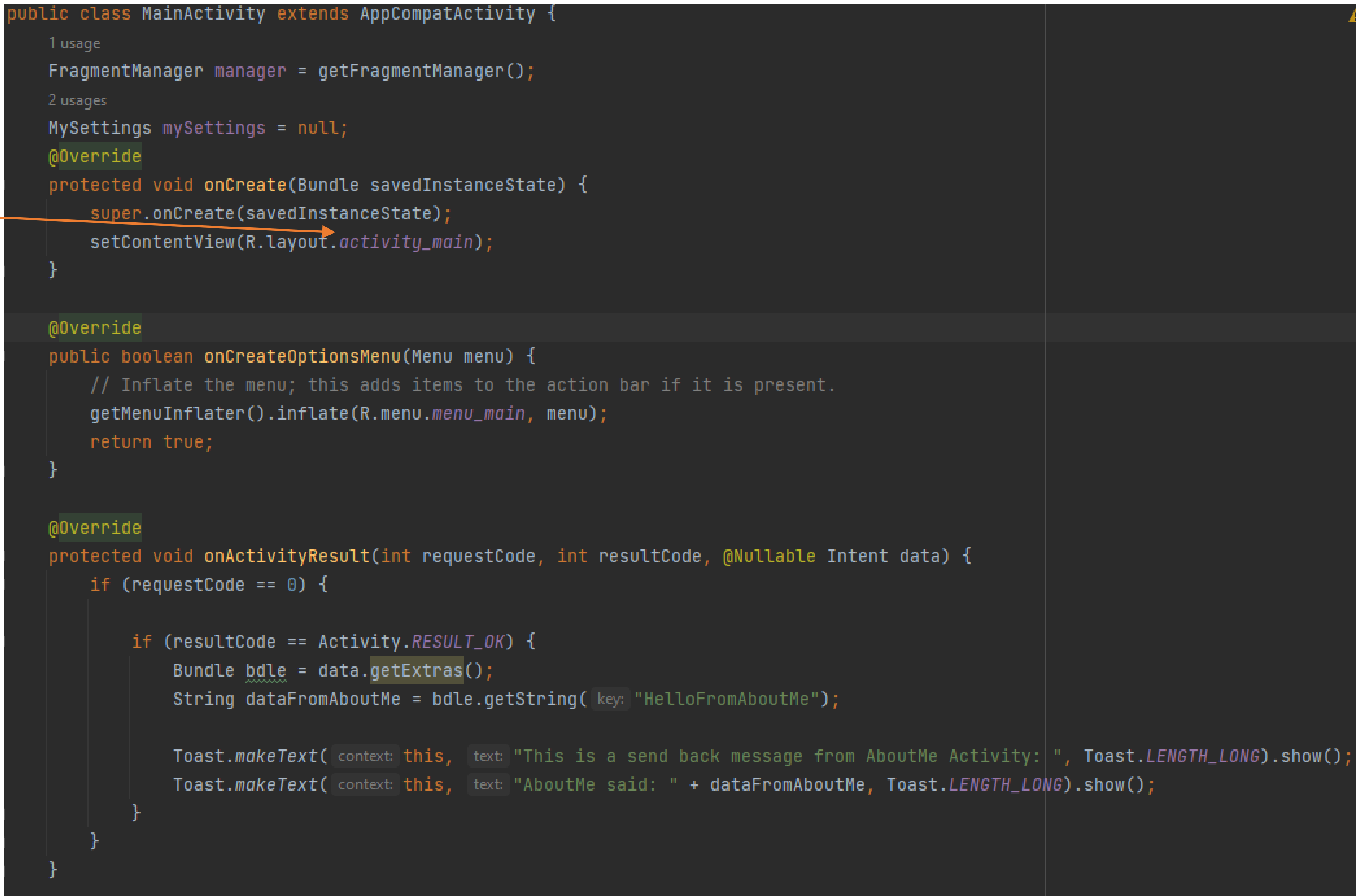


```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.activityandfragment">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="ActivityAndFragment"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ActivityAndFragment"
        tools:targetApi="31">
        <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>

        <activity
            android:name="com.example.activityandfragment.AboutMe"
            android:exported="true"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

* daemon started successfully (4 minutes ago)

MainActivity.java

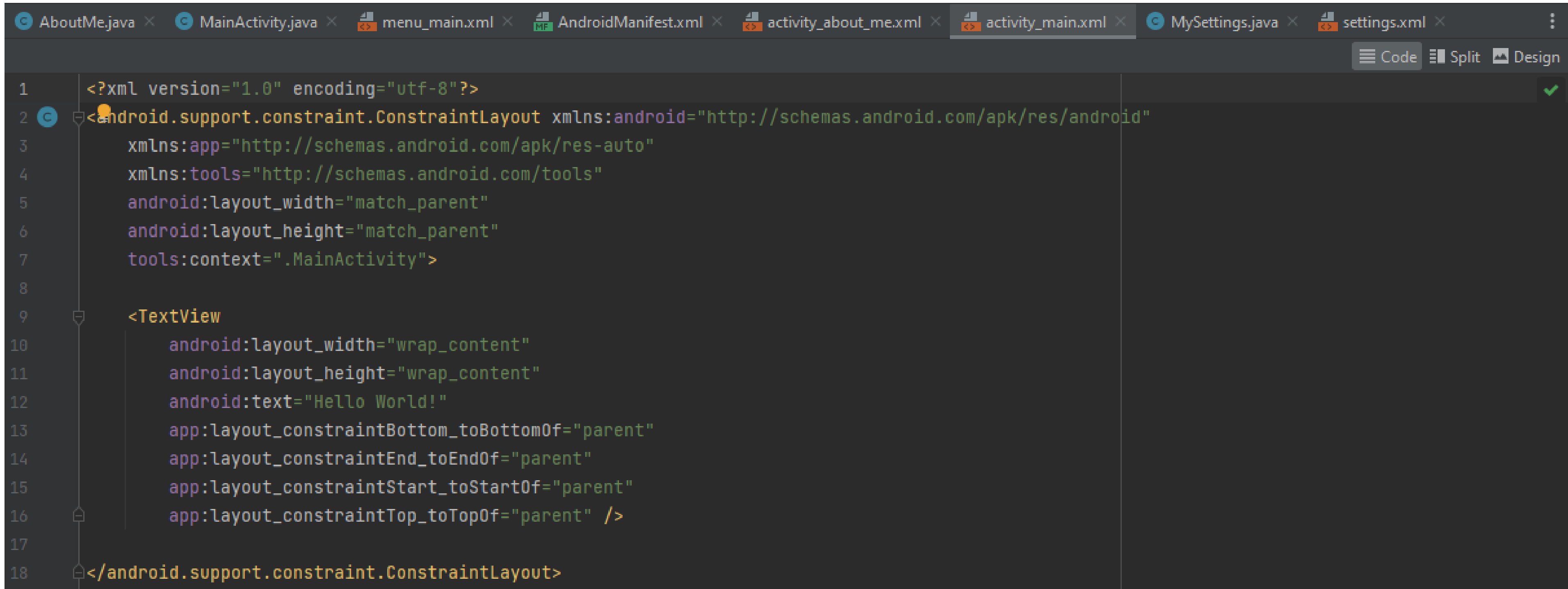
- The 2nd job, I will try to trace this guy --- activity_main.xml



```
public class MainActivity extends AppCompatActivity {
    1 usage
    FragmentManager manager = getFragmentManager();
    2 usages
    MySettings mySettings = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        if (requestCode == 0) {

            if (resultCode == Activity.RESULT_OK) {
                Bundle bdle = data.getExtras();
                String dataFromAboutMe = bdle.getString( key: "HelloFromAboutMe");

                Toast.makeText( context: this,  text: "This is a send back message from AboutMe Activity: ", Toast.LENGTH_LONG).show();
                Toast.makeText( context: this,  text: "AboutMe said: " + dataFromAboutMe, Toast.LENGTH_LONG).show();
            }
        }
    }
```

activity_main.xml

- There is nothing special here. A “TextView” with a string of “Hello World”
- This is in line with our expectation. Check the slide #3.



```xml
<?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=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

MainActivity.java

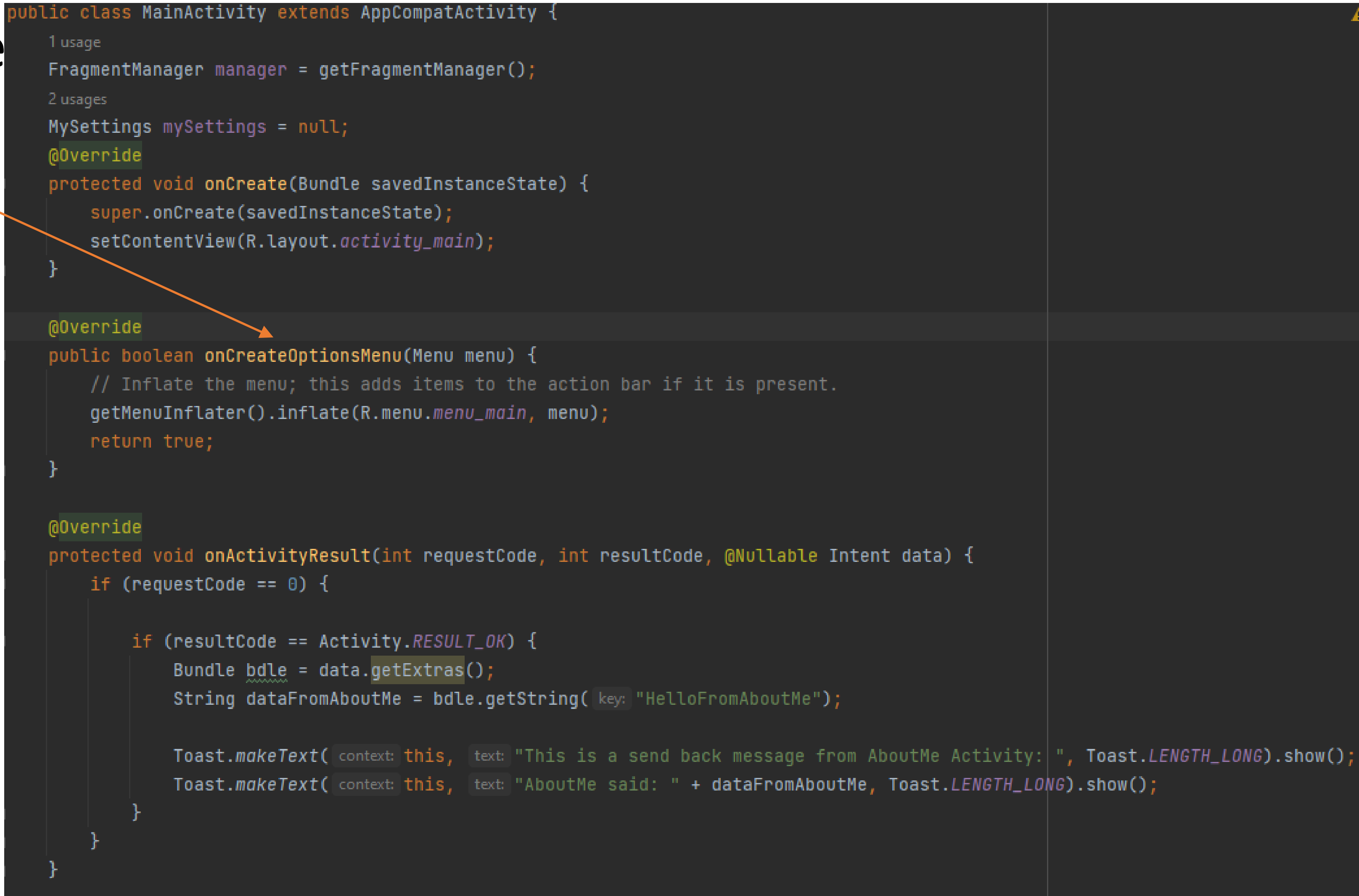
- One thing catch my eye

--- onCreateOptionsMenu()

- Remember the
  - Dot-dot-dot?
- I'm interested to see how

this xml file looks like

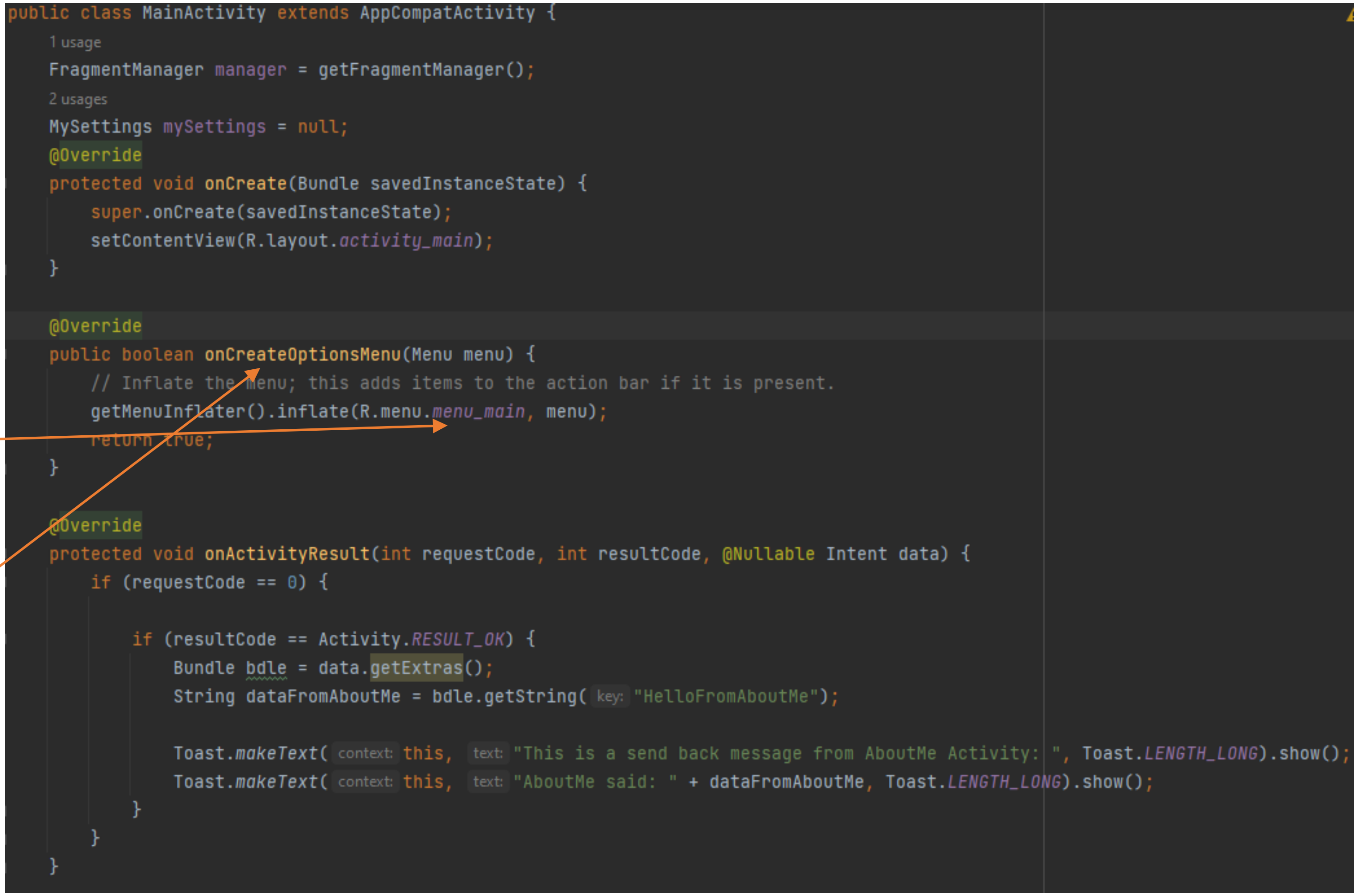
--- menu_main.xml



```java
public class MainActivity extends AppCompatActivity {
    1 usage
    FragmentManager manager = getFragmentManager();
    2 usages
    MySettings mySettings = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        if (requestCode == 0) {

            if (resultCode == Activity.RESULT_OK) {
                Bundle bdle = data.getExtras();
                String dataFromAboutMe = bdle.getString( key: "HelloFromAboutMe");

                Toast.makeText( context: this,  text: "This is a send back message from AboutMe Activity: ", Toast.LENGTH_LONG).show();
                Toast.makeText( context: this,  text: "AboutMe said: " + dataFromAboutMe, Toast.LENGTH_LONG).show();
            }
        }
    }
```

MainActivity.java

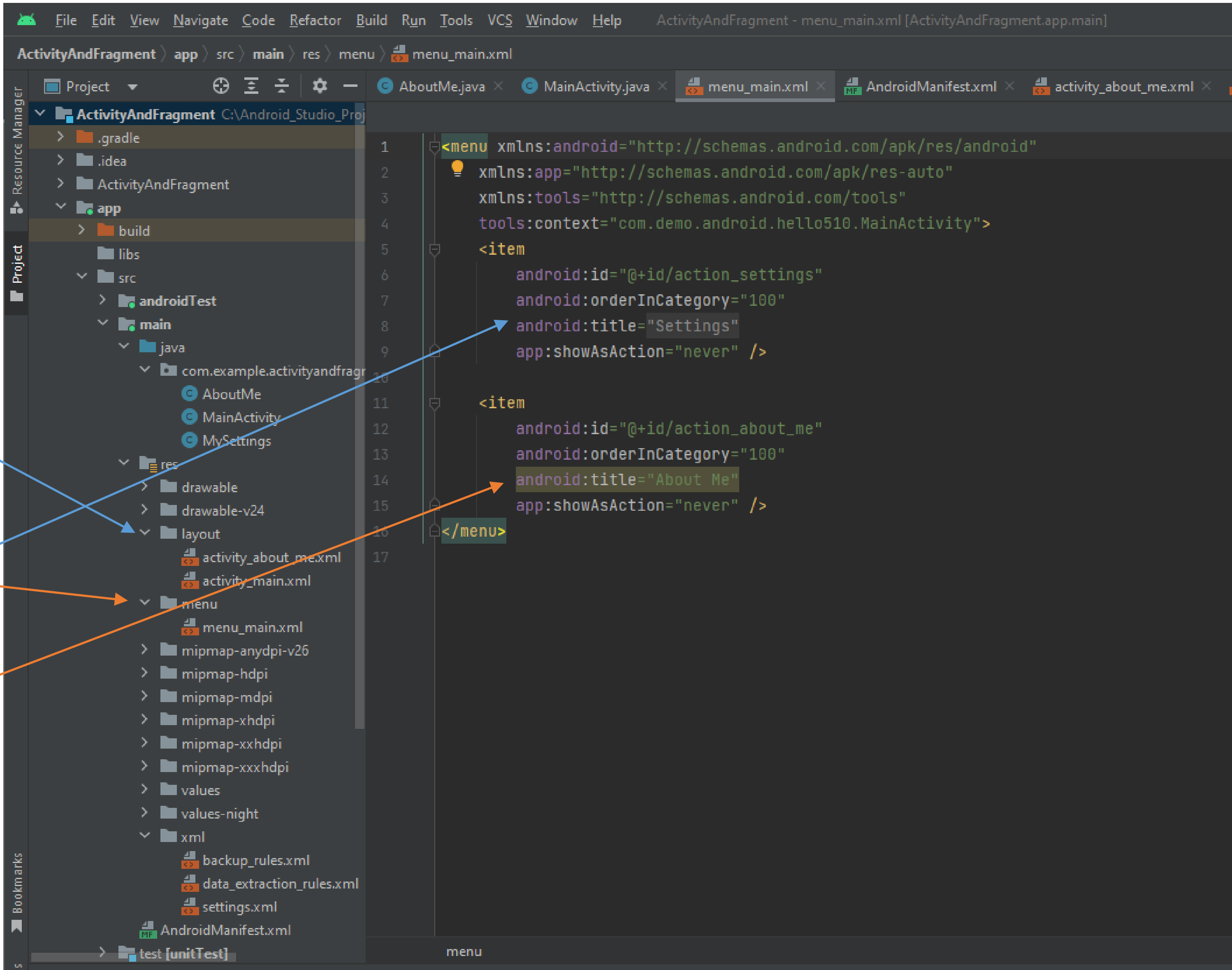
- Before we are moving to menu_main.xml, the getMenuInflater() is a function in AppCompactActivity.
    - What's why we can call that directly
- onCreateOptionsMenu() will be called before onCreate() finishes
- Check this stackoverflow:
    - https://stackoverflow.com/questions/7705927/android-when-is-oncreateoptionsmenu-called-during-activity-lifecycle
- Now, supposedly, the OptionsMenu (dot-dot-dot) is constructed.



```
public class MainActivity extends AppCompatActivity {
    1 usage
    FragmentManager manager = getFragmentManager();
    2 usages
    MySettings mySettings = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        if (requestCode == 0) {

            if (resultCode == Activity.RESULT_OK) {
                Bundle bdle = data.getExtras();
                String dataFromAboutMe = bdle.getString( key: "HelloFromAboutMe");

                Toast.makeText( context: this,  text: "This is a send back message from AboutMe Activity: ", Toast.LENGTH_LONG).show();
                Toast.makeText( context: this,  text: "AboutMe said: " + dataFromAboutMe, Toast.LENGTH_LONG).show();
            }
        }
    }
```

menu_main.xml

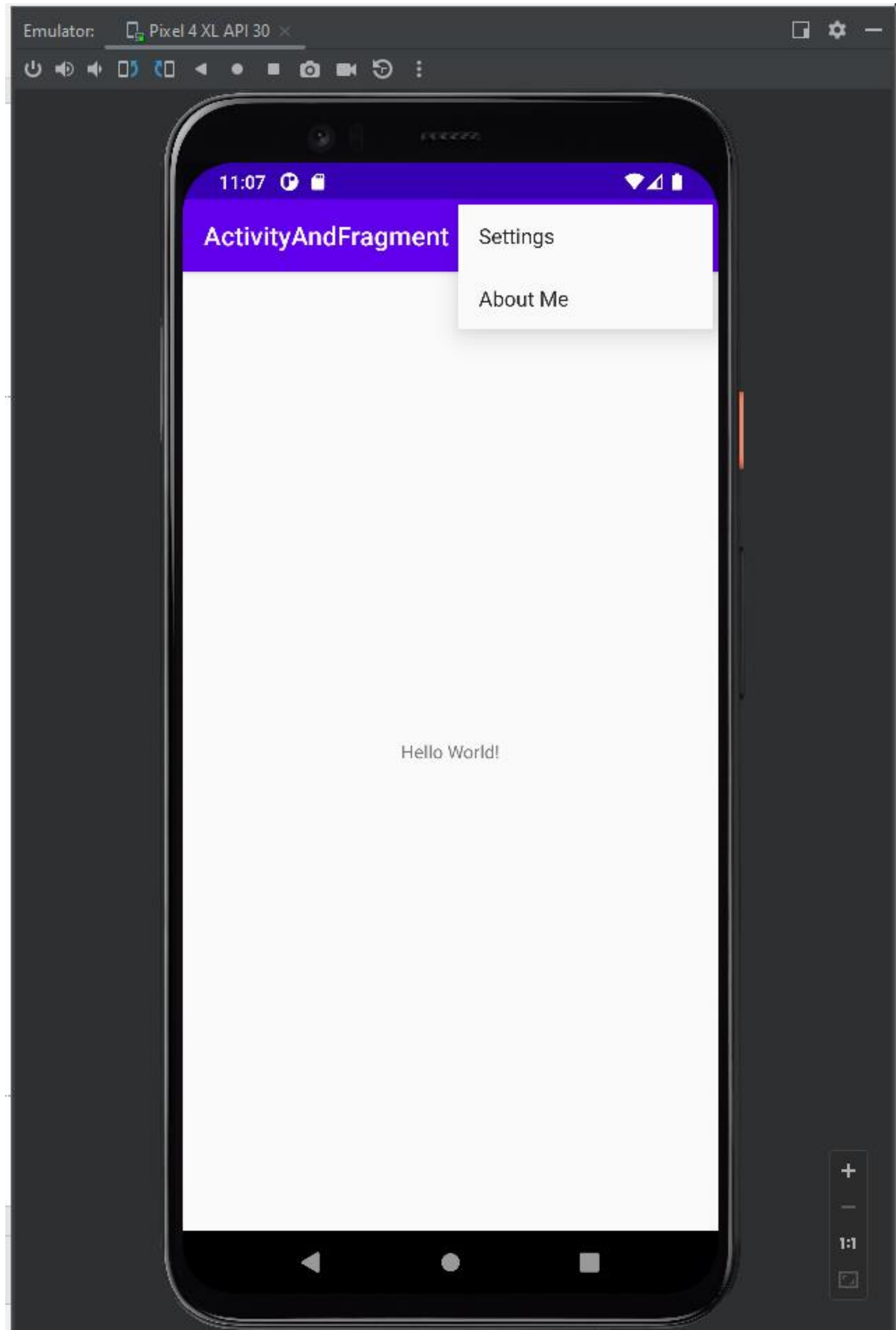
- Like we had seen earlier for the Android layouts, they are located in /res/layout,
- Similarly, our "menu_main.xml" is located in /res/menu
- In this menu, there are 2 items in the menu
  - "Settings" and "About Me"



```xml
<menu 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"
    tools:context="com.demo.android.hello510.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="Settings"
        app:showAsAction="never" />

    <item
        android:id="@+id/action_about_me"
        android:orderInCategory="100"
        android:title="About Me"
        app:showAsAction="never" />
</menu>
```

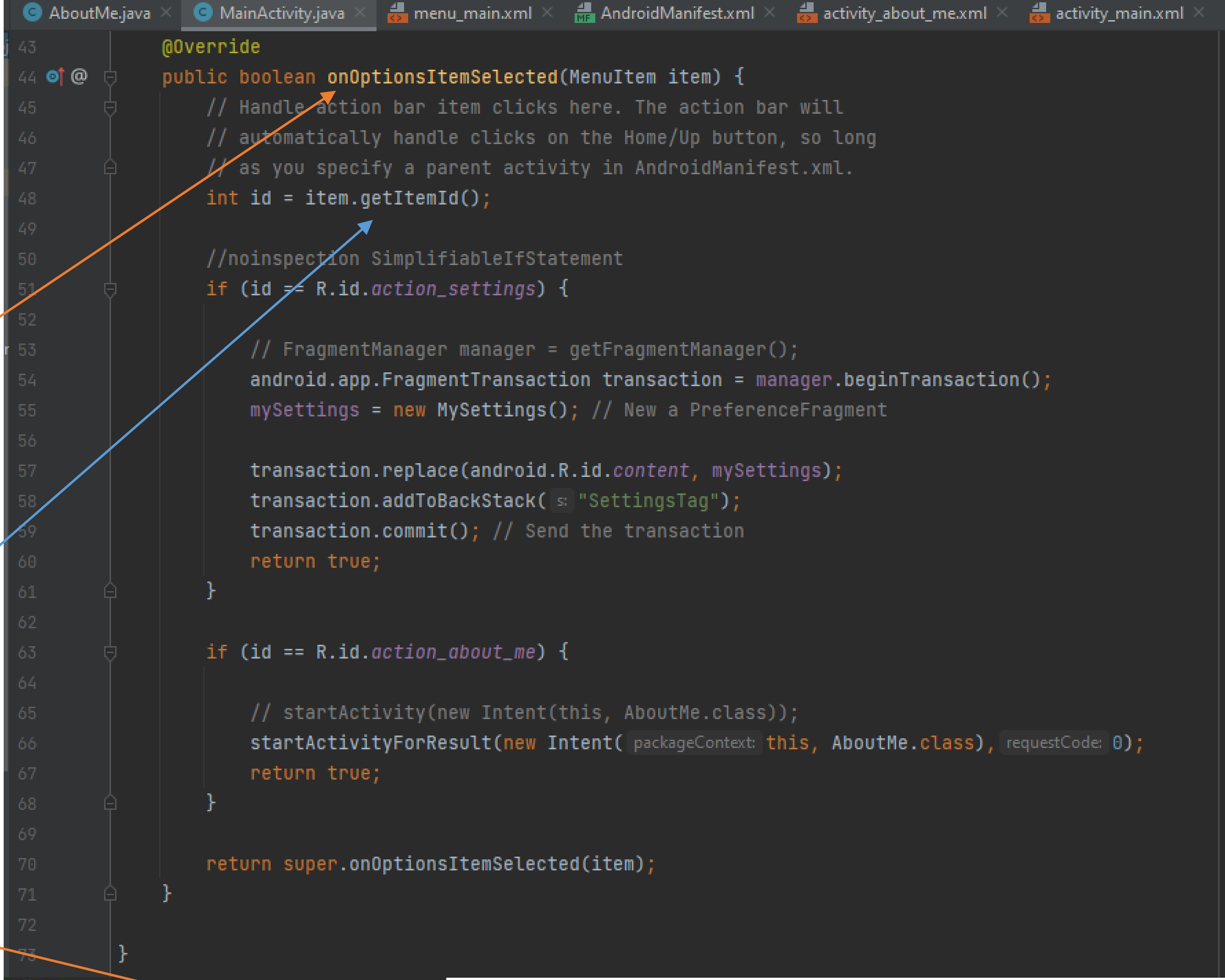
MainActivity.java

- Now we get the menu and let's see onOptionsItemSelected()
- Let's see the corresponding code for these 2 items when they get clicked.



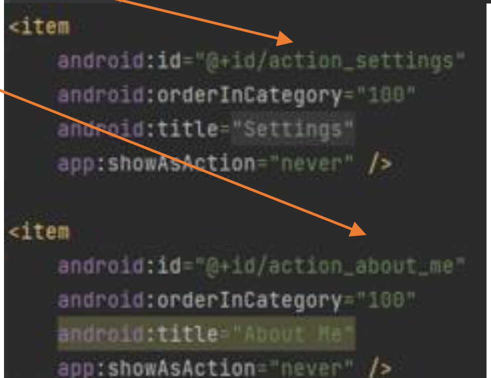
MainActivity.java

- When the menu item is clicked, the call-back function will be executed.
- For each of the item, it has item ID
- We can use the item object to get the item ID



AboutMe.java × | MainActivity.java × | menu_main.xml × | AndroidManifest.xml × | activity_about_me.xml × | activity_main.xml ×

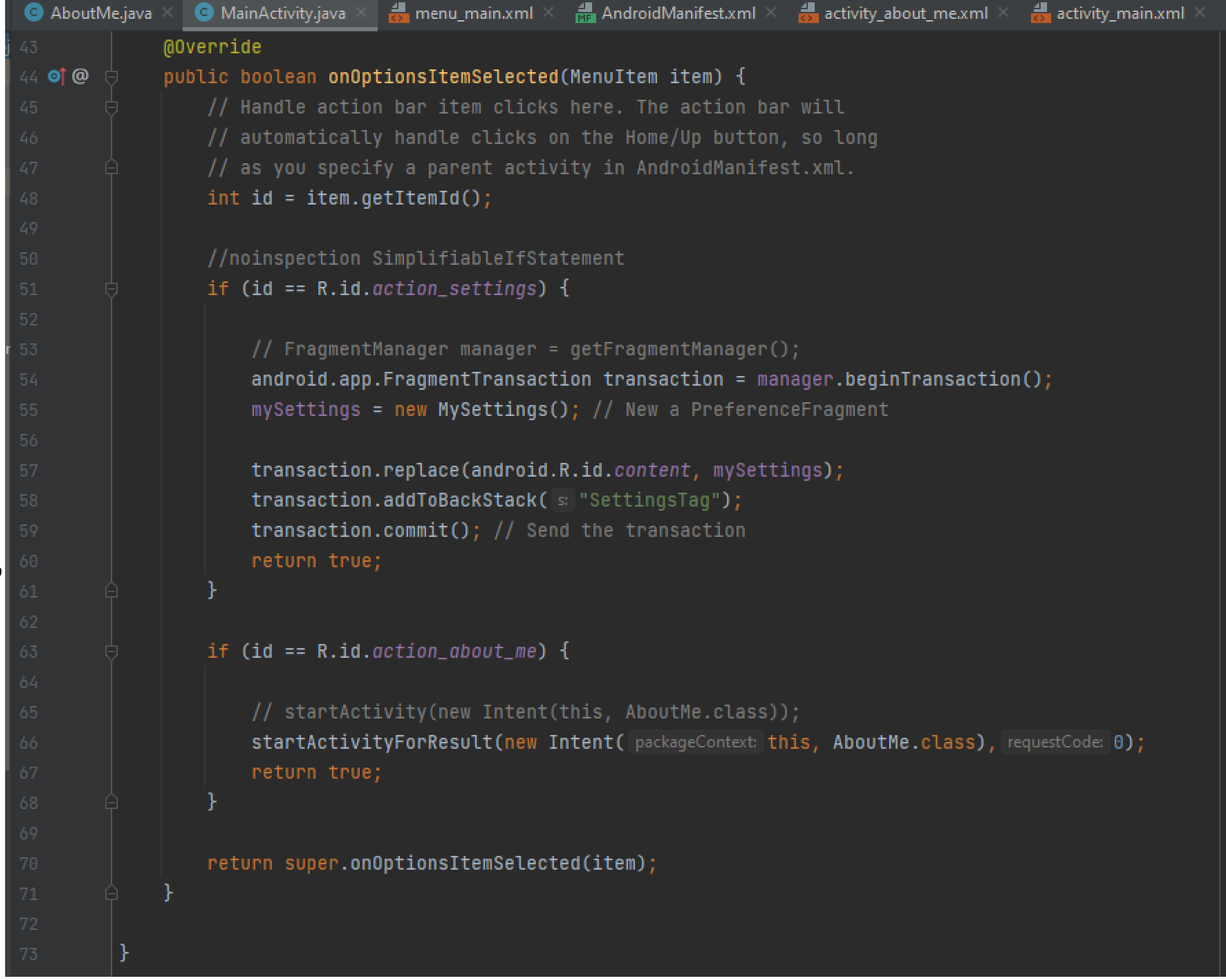
```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {

            // FragmentManager manager = getFragmentManager();
            android.app.FragmentTransaction transaction = manager.beginTransaction();
            mySettings = new MySettings(); // New a PreferenceFragment

            transaction.replace(android.R.id.content, mySettings);
            transaction.addToBackStack( s: "SettingsTag");
            transaction.commit(); // Send the transaction
            return true;
        }

        if (id == R.id.action_about_me) {

            // startActivity(new Intent(this, AboutMe.class));
            startActivityForResult(new Intent( packageContext: this, AboutMe.class), requestCode: 0);
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

}
```



```
<item
    android:id="@+id/action_settings"
    android:orderInCategory="100"
    android:title="Settings"
    app:showAsAction="never" />

<item
    android:id="@+id/action_about_me"
    android:orderInCategory="100"
    android:title="About Me"
    app:showAsAction="never" />
```

MainActivity.java

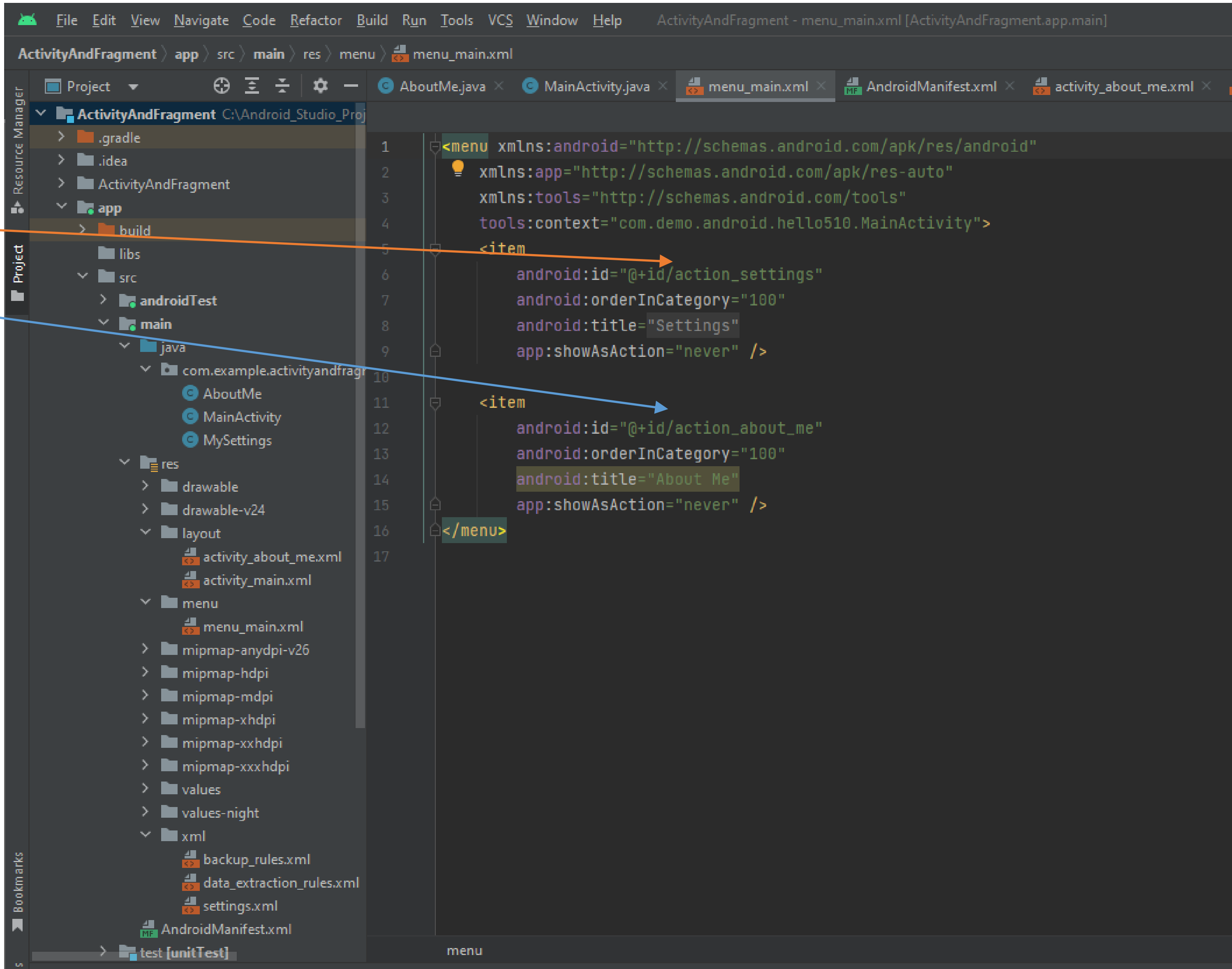
- Depends on the "item ID", if it is action_settings, we are about to do "Fragment Transition" to MySettings.java.
- Otherwise, if it is action_about_me, we make a jump and have our main activity to jump to AboutMe.java (another activity)



```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {

            // FragmentManager manager = getFragmentManager();
            android.app.FragmentTransaction transaction = manager.beginTransaction();
            mySettings = new MySettings(); // New a PreferenceFragment

            transaction.replace(android.R.id.content, mySettings);
            transaction.addToBackStack( s: "SettingsTag");
            transaction.commit(); // Send the transaction
            return true;
        }

        if (id == R.id.action_about_me) {

            // startActivity(new Intent(this, AboutMe.class));
            startActivityForResult(new Intent( packageContext: this, AboutMe.class), requestCode: 0);
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

}
```

menu_main.xml

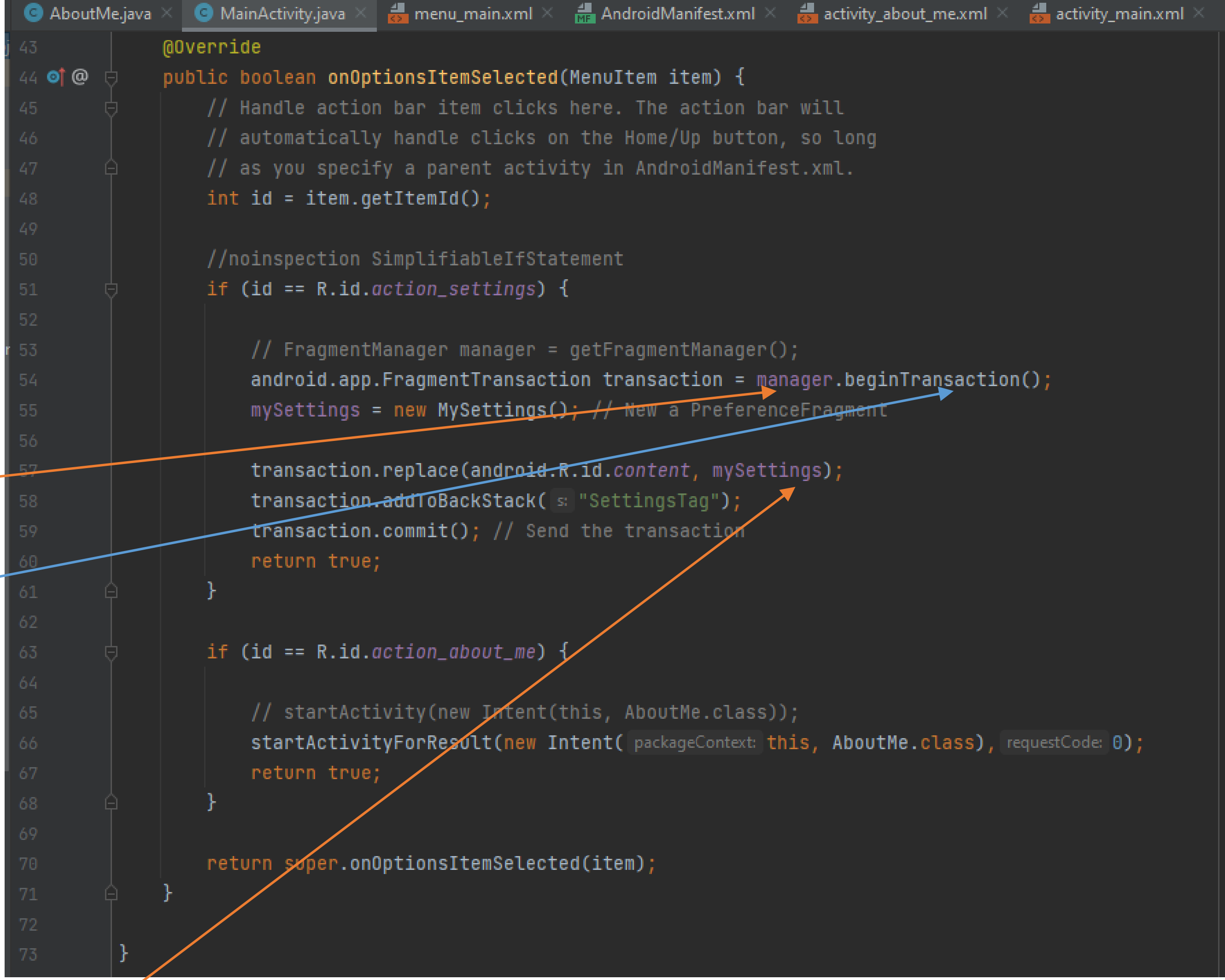
- Wait! What is the?
  - action_settings
  - action_about_me
- Most of the time, in Android, we access the UI components by its “id”



```xml
<menu 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"
    tools:context="com.demo.android.hello510.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="Settings"
        app:showAsAction="never" />

    <item
        android:id="@+id/action_about_me"
        android:orderInCategory="100"
        android:title="About Me"
        app:showAsAction="never" />
</menu>
```

MainActivity.java

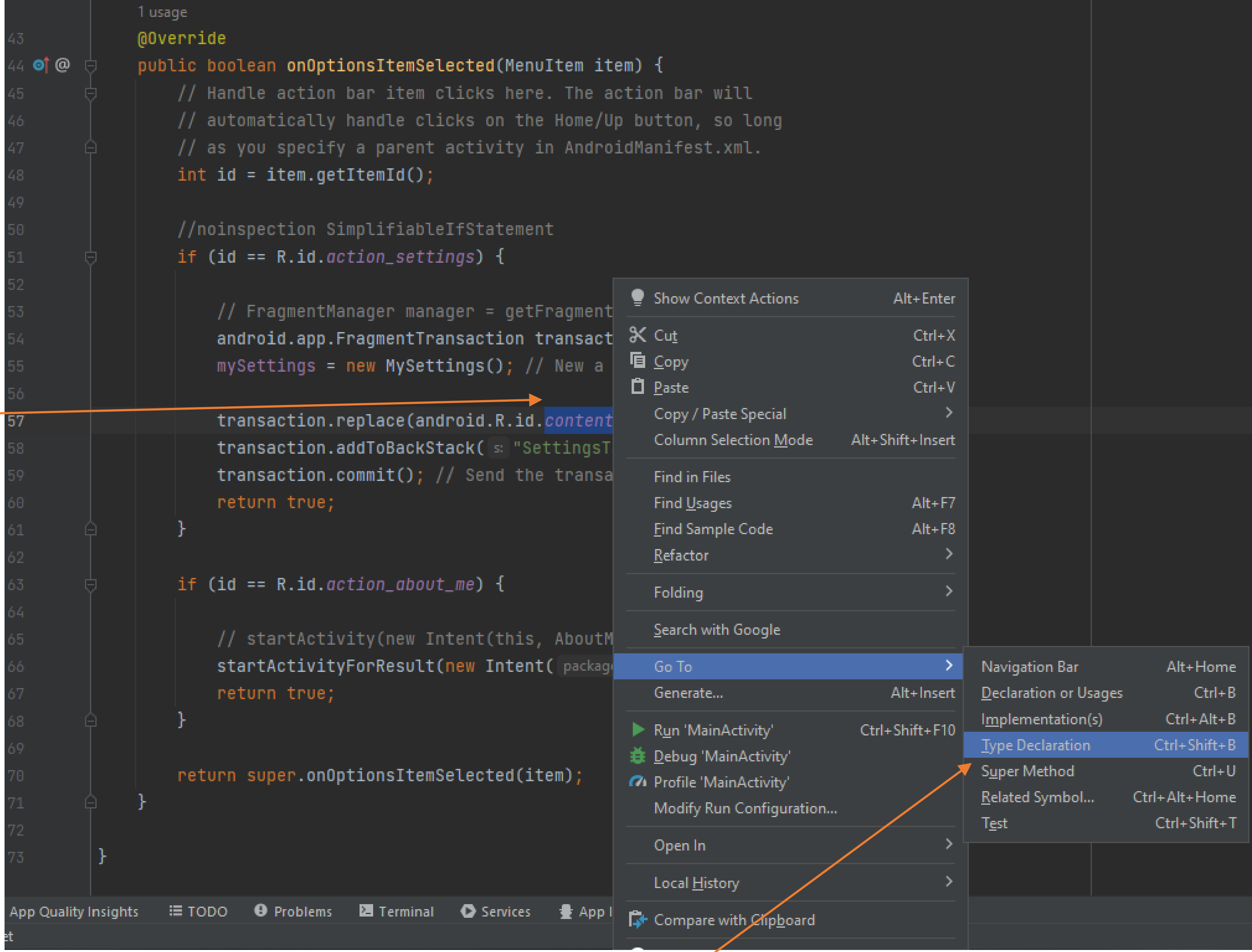
- Check the slide #9 first
  - In slide #9, we get a fragment manager Instance called “**manager**”
  - We now use this **manager** instance to setup our one-time transaction by calling beginTransaction()
- Now, we are going to replace whatever we have in the “fragment container” with this “new” fragment
- mySettings --- Check the slide #9, it was declared earlier (named **mySettings**) and now it is instantiated. It is used in the transaction



```java
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {

            // FragmentManager manager = getFragmentManager();
            android.app.FragmentTransaction transaction = manager.beginTransaction();
            mySettings = new MySettings(); // New a PreferenceFragment

            transaction.replace(android.R.id.content, mySettings);
            transaction.addToBackStack( s: "SettingsTag");
            transaction.commit(); // Send the transaction
            return true;
        }

        if (id == R.id.action_about_me) {

            // startActivity(new Intent(this, AboutMe.class));
            startActivityForResult(new Intent( packageContext: this, AboutMe.class), requestCode: 0);
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

}
```

MainActivity.java

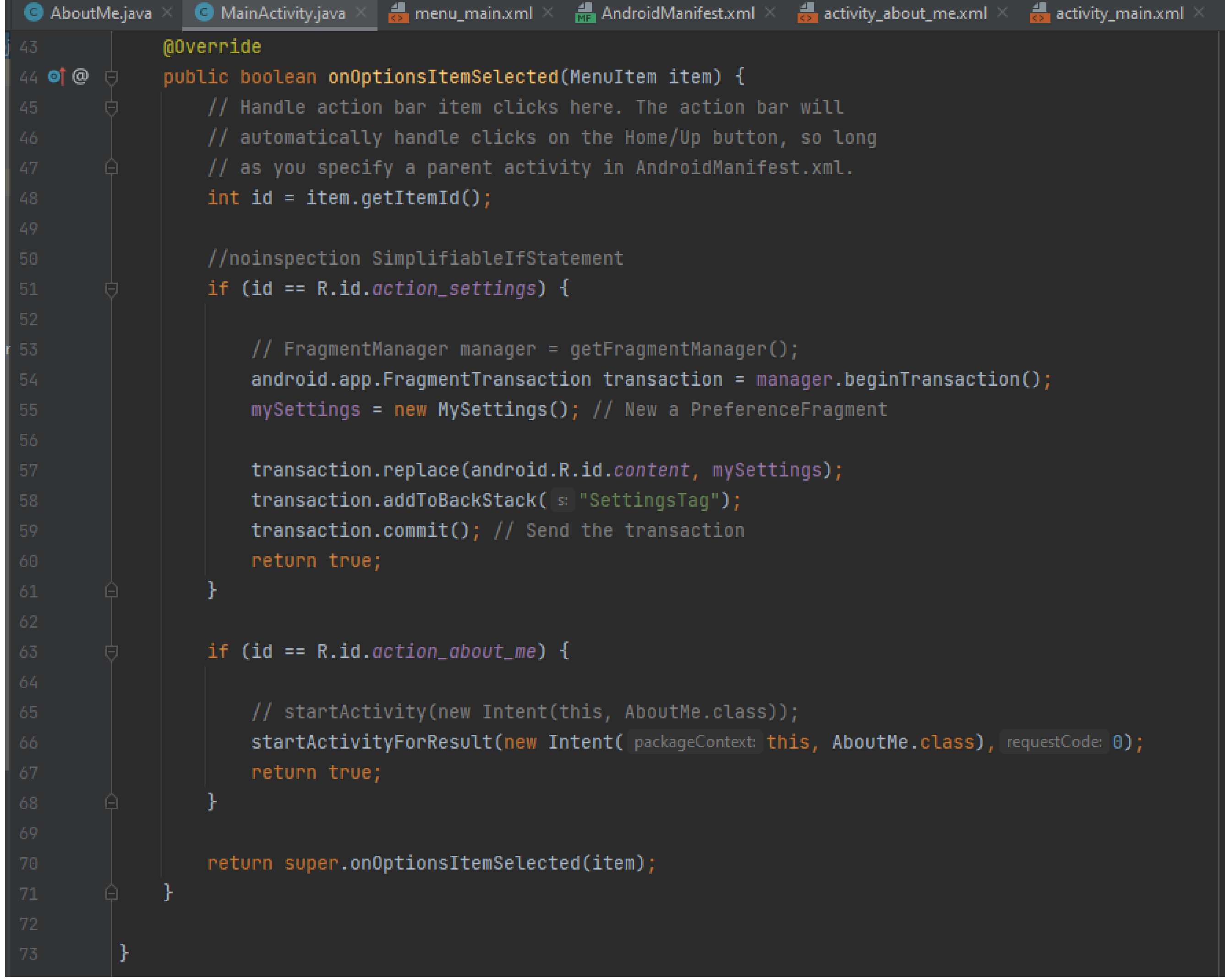
- One more thing, the R.id.content means our current MainActivity, it is actually the “**container**” for other “views” or “fragment”
- R.id.content is a system built-in resource, you can check it by mark it and see its “Type Declaration”



```
1 usage
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {

        // FragmentManager manager = getFragment
        android.app.FragmentTransaction transact
        mySettings = new MySettings(); // New a

        transaction.replace(android.R.id.content
        transaction.addToBackStack( s: "SettingsT
        transaction.commit(); // Send the transa
        return true;
    }

    if (id == R.id.action_about_me) {

        // startActivity(new Intent(this, AboutM
        startActivityForResult(new Intent( packag
        return true;
    }

    return super.onOptionsItemSelected(item);
}

}
```

MainActivity.java

- So far, before the transaction happens, the “container” is empty
- This line, line #57, the replace() means:
  I’m going to grab mySettings, a fragment, and put that on top of the container.

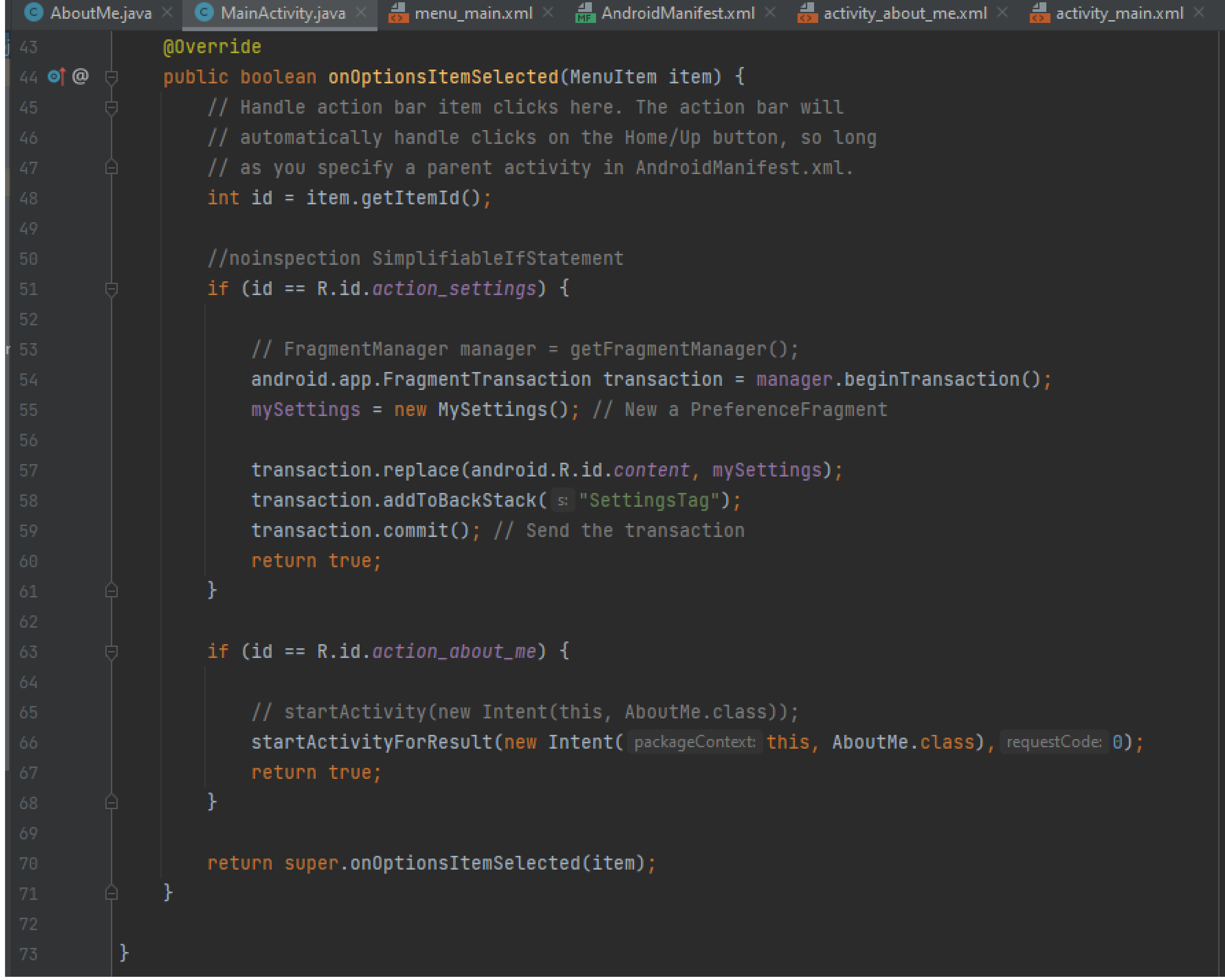


```
43      @Override
44      public boolean onOptionsItemSelected(MenuItem item) {
45          // Handle action bar item clicks here. The action bar will
46          // automatically handle clicks on the Home/Up button, so long
47          // as you specify a parent activity in AndroidManifest.xml.
48          int id = item.getItemId();
49
50          //noinspection SimplifiableIfStatement
51          if (id == R.id.action_settings) {
52
53              // FragmentManager manager = getFragmentManager();
54              android.app.FragmentTransaction transaction = manager.beginTransaction();
55              mySettings = new MySettings(); // New a PreferenceFragment
56
57              transaction.replace(android.R.id.content, mySettings);
58              transaction.addToBackStack( s: "SettingsTag");
59              transaction.commit(); // Send the transaction
60              return true;
61          }
62
63          if (id == R.id.action_about_me) {
64
65              // startActivity(new Intent(this, AboutMe.class));
66              startActivityForResult(new Intent( packageContext: this, AboutMe.class), requestCode: 0);
67              return true;
68          }
69
70          return super.onOptionsItemSelected(item);
71      }
72
73  }
```

- Transaction means one time of the UI change, usually means a fragment is flying into our main display area and hovering on top of that.

MainActivity.java

- addToBackStack()
  - We use this function to add the transaction to the stack.
- This means the transaction will be **remembered** after it is committed, and will reverse its operation when later popped off the stack.
- We provide a **string** to memorize this time of the transaction

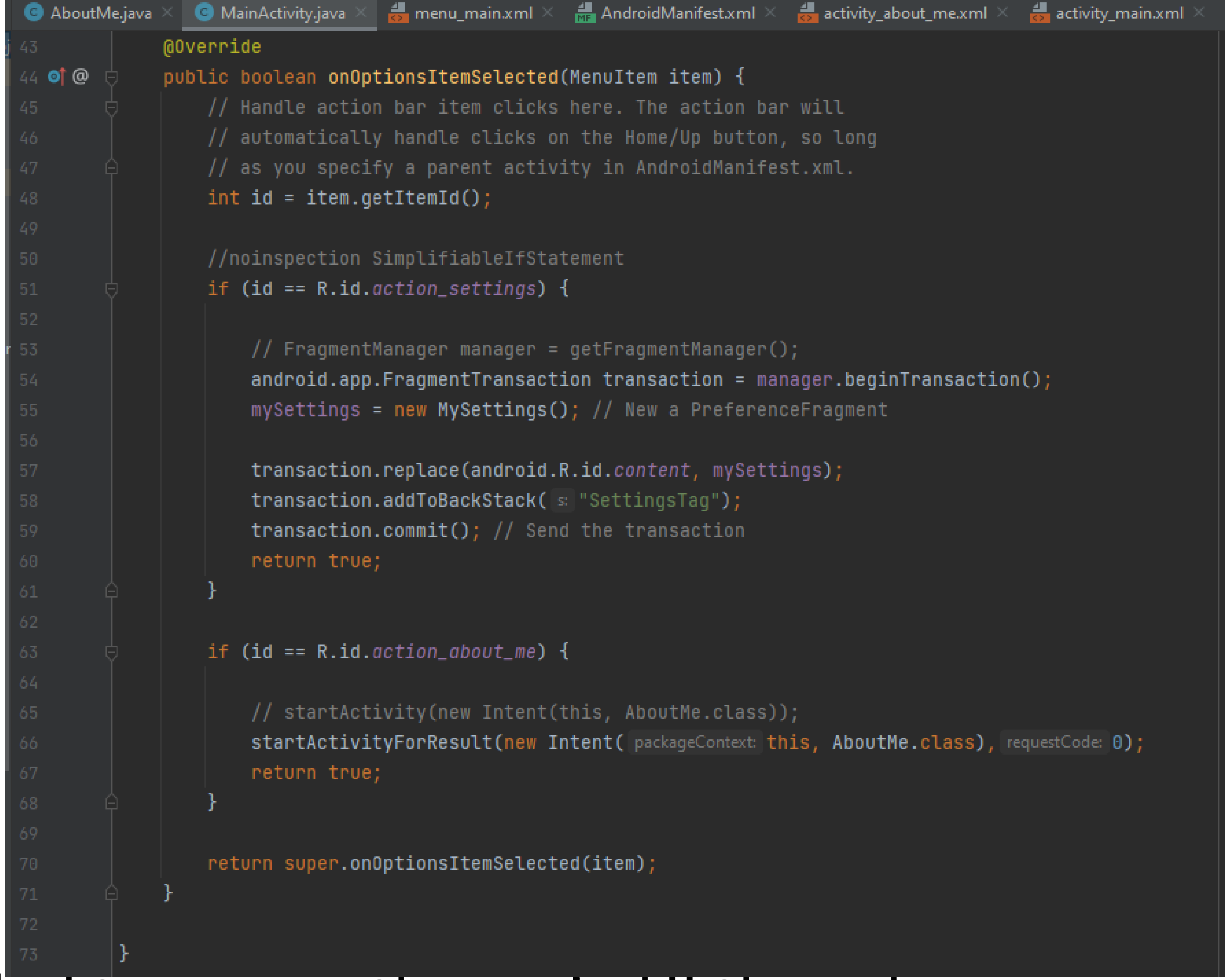


AboutMe.java | MainActivity.java | menu_main.xml | AndroidManifest.xml | activity_about_me.xml | activity_main.xml

```java
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {

            // FragmentManager manager = getFragmentManager();
            android.app.FragmentTransaction transaction = manager.beginTransaction();
            mySettings = new MySettings(); // New a PreferenceFragment

            transaction.replace(android.R.id.content, mySettings);
            transaction.addToBackStack( s: "SettingsTag");
            transaction.commit(); // Send the transaction
            return true;
        }

        if (id == R.id.action_about_me) {

            // startActivity(new Intent(this, AboutMe.class));
            startActivityForResult(new Intent( packageContext: this, AboutMe.class), requestCode: 0);
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

}
```

MainActivity.java

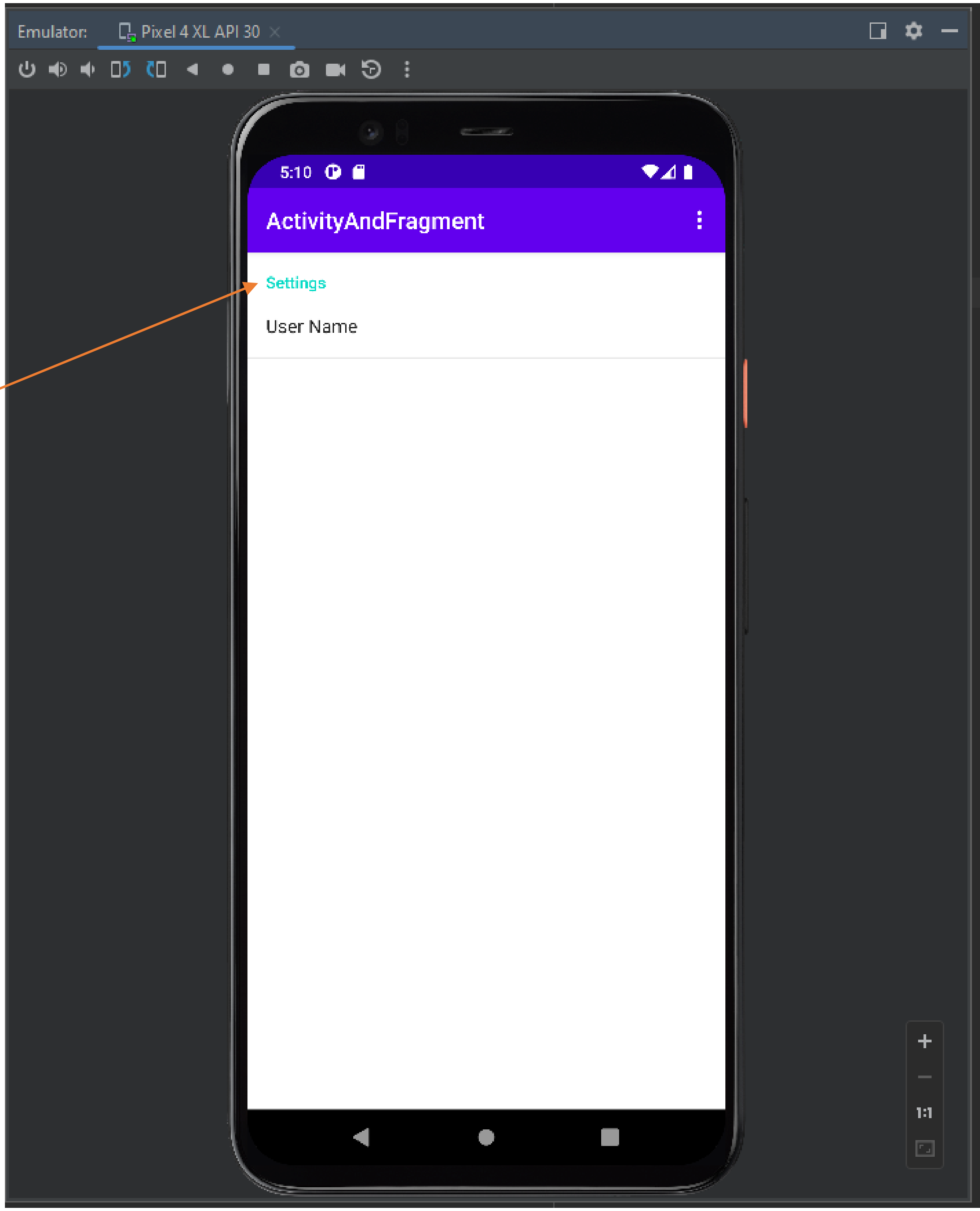
- Finally, according to this Document, the commit() is asynchronous.
- If we call the commit() doesn't perform the transaction immediately.
- Rather, the transaction is scheduled to run on the main UI thread, as soon as it is able to do so



```java
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {

            // FragmentManager manager = getFragmentManager();
            android.app.FragmentTransaction transaction = manager.beginTransaction();
            mySettings = new MySettings(); // New a PreferenceFragment

            transaction.replace(android.R.id.content, mySettings);
            transaction.addToBackStack("SettingsTag");
            transaction.commit(); // Send the transaction
            return true;
        }

        if (id == R.id.action_about_me) {

            // startActivity(new Intent(this, AboutMe.class));
            startActivityForResult(new Intent(this, AboutMe.class), 0);
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

}
```

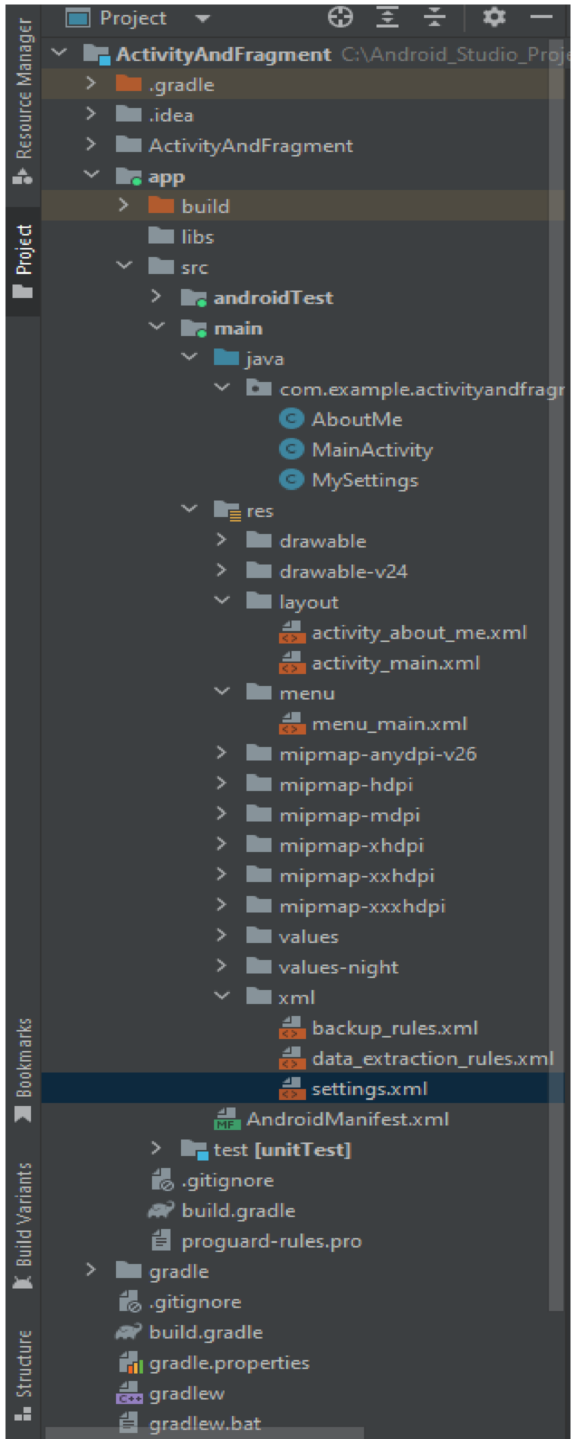
MySettings.java

- See? The new UI (**fragment**) flies in and it **shares** the “dot-dot-dot menu” with MainActivity
- In this MySettings.java, I load a “PreferenceCategory” and its title is “Settings”
- The EditTextPreference is used to store someone’s User Name, for example
- Let’s see how the settings.xml is loaded from MySettings.java ?



Settings.xml

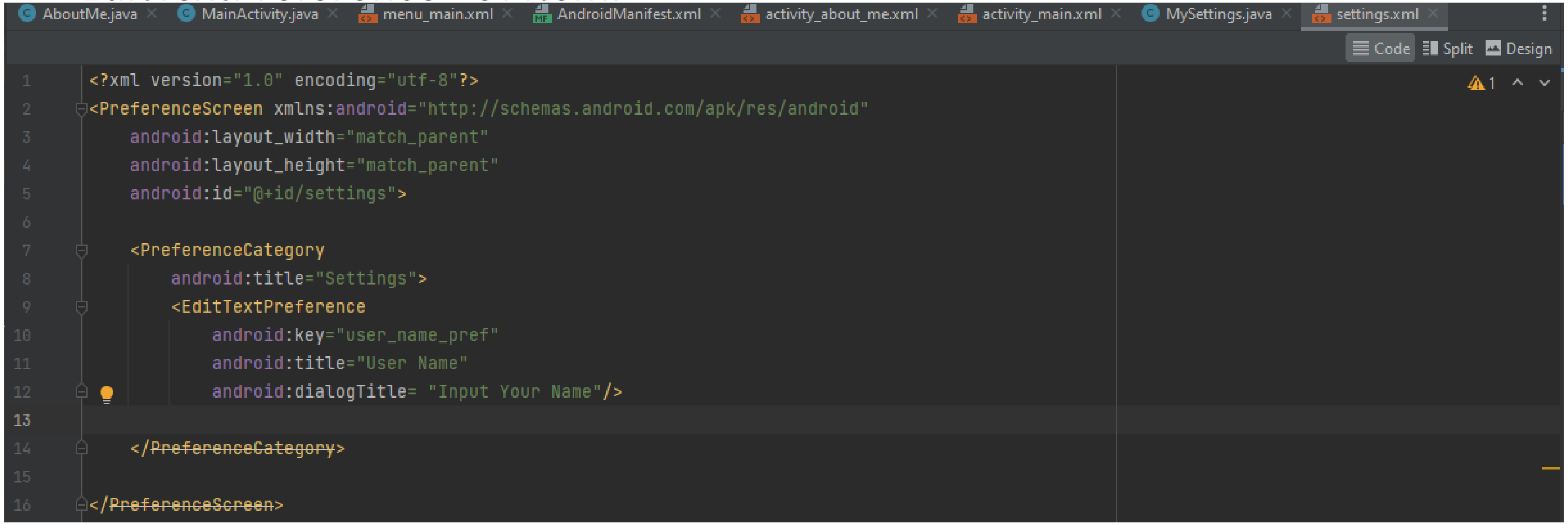
- This is the settings.xml



Settings.xml

We don’t talk about the UI controls into detail. It would be a lot and as you can see, their evolutions are fast. Some of the UI controls are deprecated! If you are interested, please check the Google Android Developer’s website

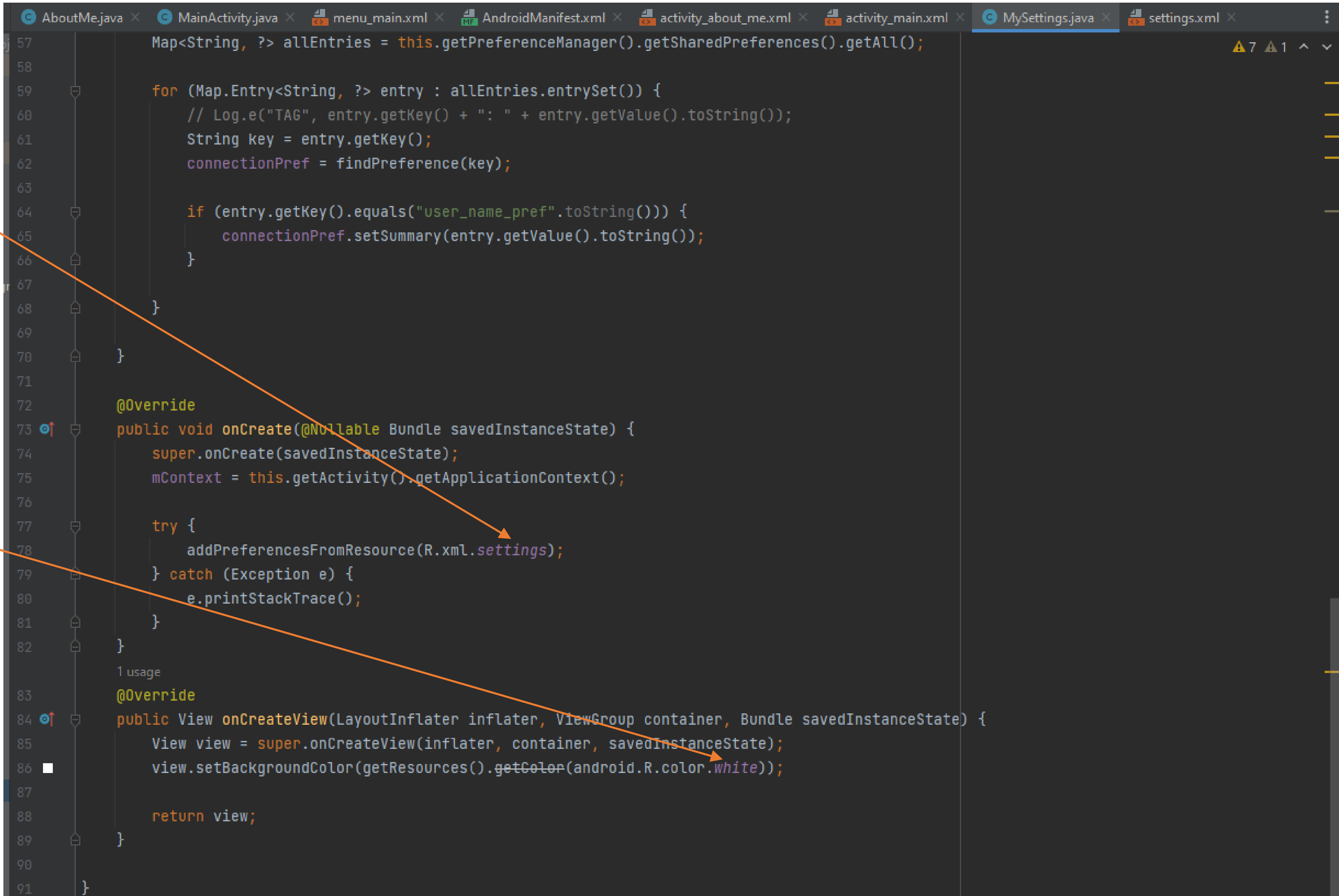
- What is in the settings.xml?
  - A “PreferenceScreen” with
  - a “PreferenceCategory” in it. And this guy has a
  - “EditTextPreference” UI item.



```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/settings">

    <PreferenceCategory
        android:title="Settings">
        <EditTextPreference
            android:key="user_name_pref"
            android:title="User Name"
            android:dialogTitle= "Input Your Name"/>

    </PreferenceCategory>

</PreferenceScreen>
```

MySettings.java

- settings.xml is loaded in the onCreate()
- Check slide #20, its background is set to white color

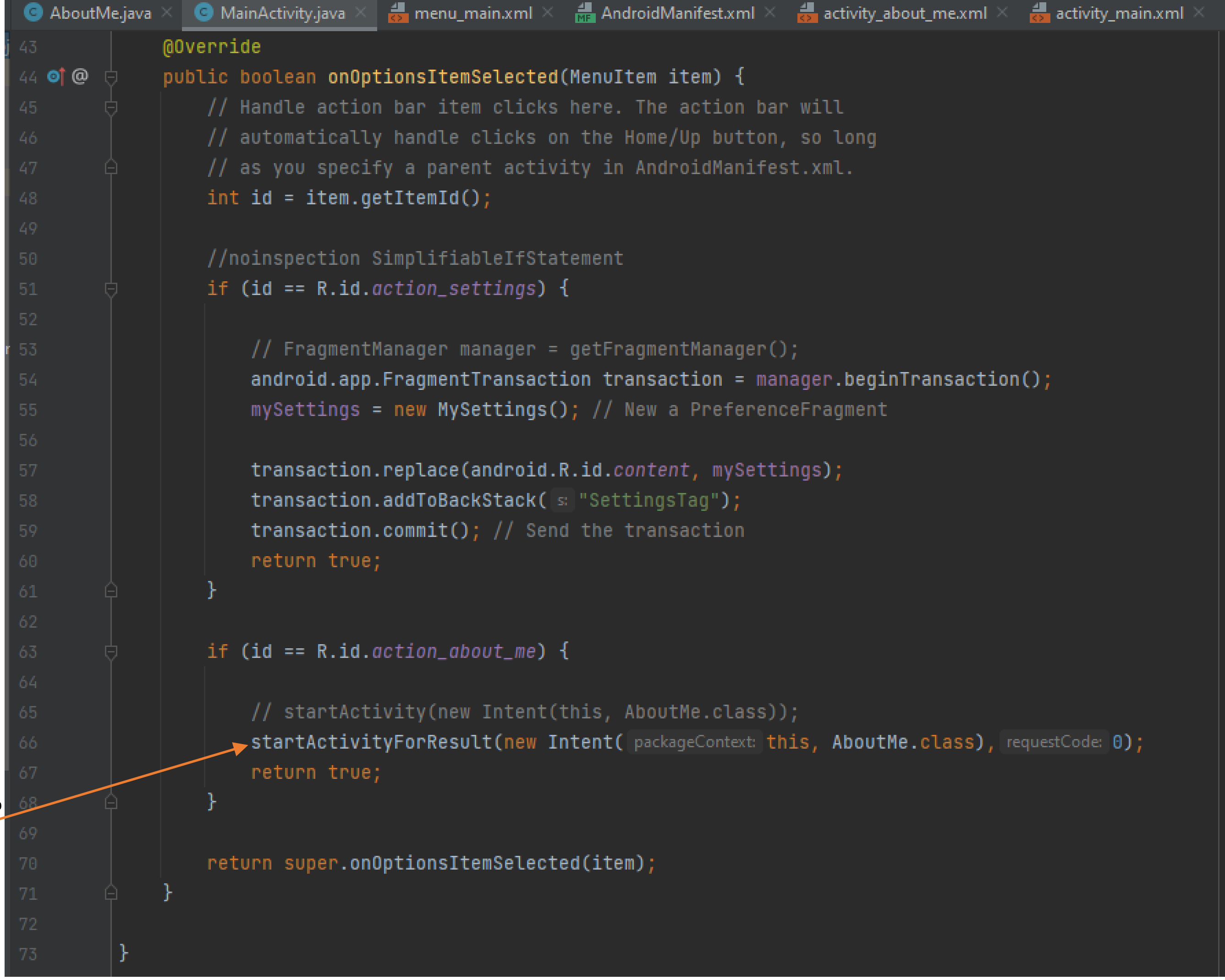


```
        Map<String, ?> allEntries = this.getPreferenceManager().getSharedPreferences().getAll();

        for (Map.Entry<String, ?> entry : allEntries.entrySet()) {
            // Log.e("TAG", entry.getKey() + ": " + entry.getValue().toString());
            String key = entry.getKey();
            connectionPref = findPreference(key);

            if (entry.getKey().equals("user_name_pref".toString())) {
                connectionPref.setSummary(entry.getValue().toString());
            }

        }

    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mContext = this.getActivity().getApplicationContext();

        try {
            addPreferencesFromResource(R.xml.settings);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    1 usage
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View view = super.onCreateView(inflater, container, savedInstanceState);
        view.setBackgroundColor(getResources().~~getColor~~(android.R.color.white));

        return view;
    }

}
```

MainActivity.java

- We setup an activity jump from MainActivity to AboutMe, the second activity.
- We need to setup an intent (anonymous object) as parameter.
- Then, call the startActivityForResult(),

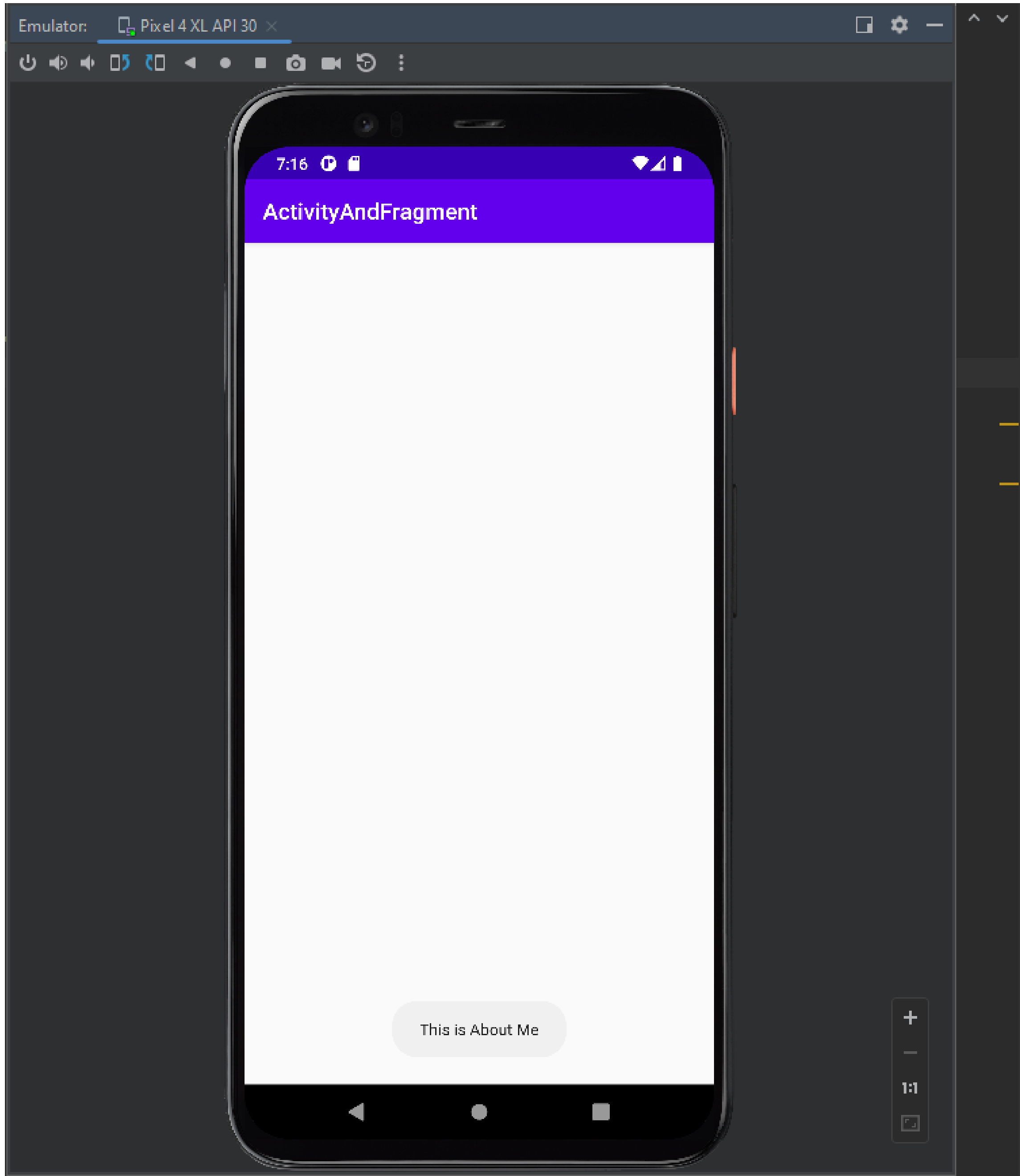
like we have done in the previous demo



```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {

            // FragmentManager manager = getFragmentManager();
            android.app.FragmentTransaction transaction = manager.beginTransaction();
            mySettings = new MySettings(); // New a PreferenceFragment

            transaction.replace(android.R.id.content, mySettings);
            transaction.addToBackStack( s: "SettingsTag");
            transaction.commit(); // Send the transaction
            return true;
        }

        if (id == R.id.action_about_me) {

            // startActivity(new Intent(this, AboutMe.class));
            startActivityForResult(new Intent( packageContext: this, AboutMe.class), requestCode: 0);
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

}
```

AboutMe.java

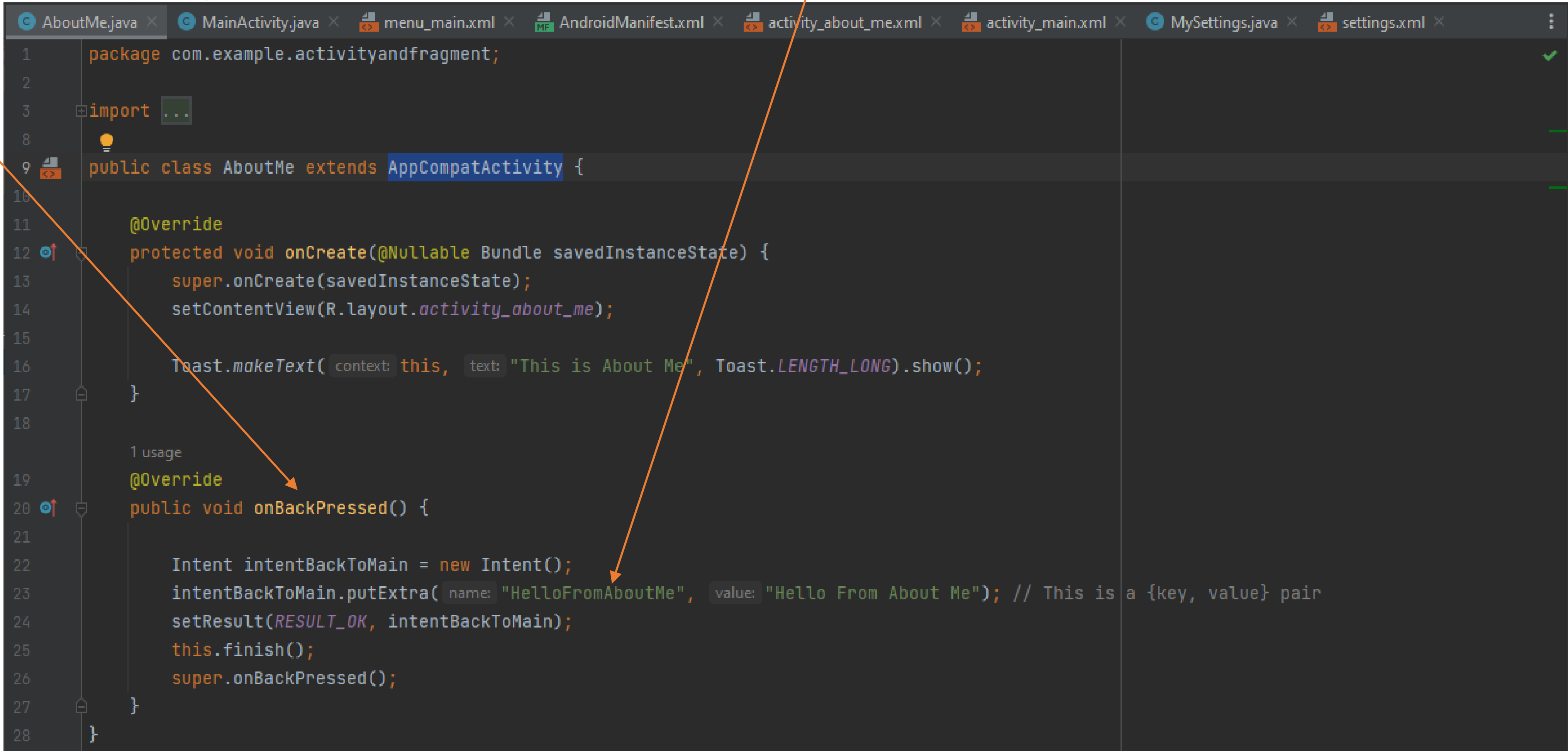
- There is basically nothing in the "AboutMe"



Key: “HelloFromAboutMe”
Value: “Hello From About Me”

AboutMe.java

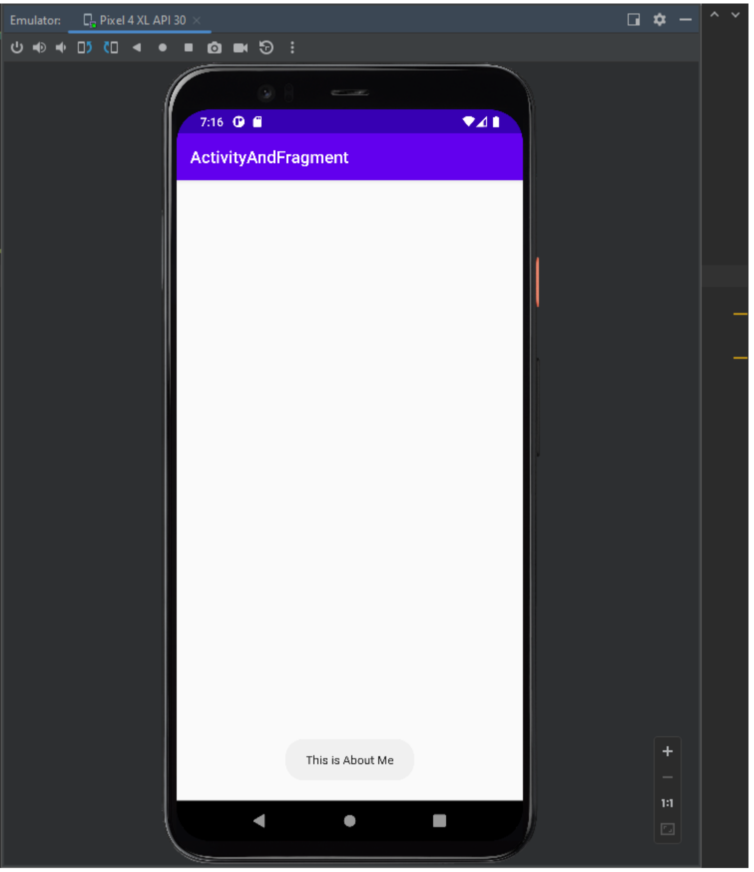
- I put a Toast saying, “This is About Me” **and intercept** the [Back] key pressed



AboutMe.java | MainActivity.java | menu_main.xml | AndroidManifest.xml | activity_about_me.xml | activity_main.xml | MySettings.java | settings.xml

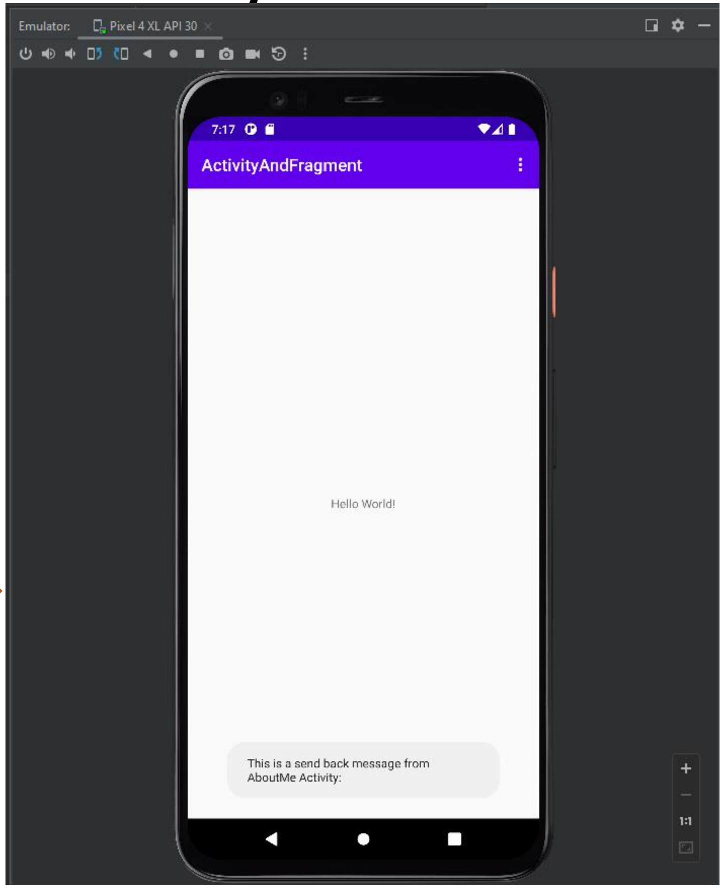
```java
package com.example.activityandfragment;

import ...

public class AboutMe extends AppCompatActivity {

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

        Toast.makeText( context: this,  text: "This is About Me", Toast.LENGTH_LONG).show();
    }

    1 usage
    @Override
    public void onBackPressed() {

        Intent intentBackToMain = new Intent();
        intentBackToMain.putExtra( name: "HelloFromAboutMe",  value: "Hello From About Me"); // This is a {key, value} pair
        setResult(RESULT_OK, intentBackToMain);
        this.finish();
        super.onBackPressed();
    }
}
```

AboutMe → MainActivity



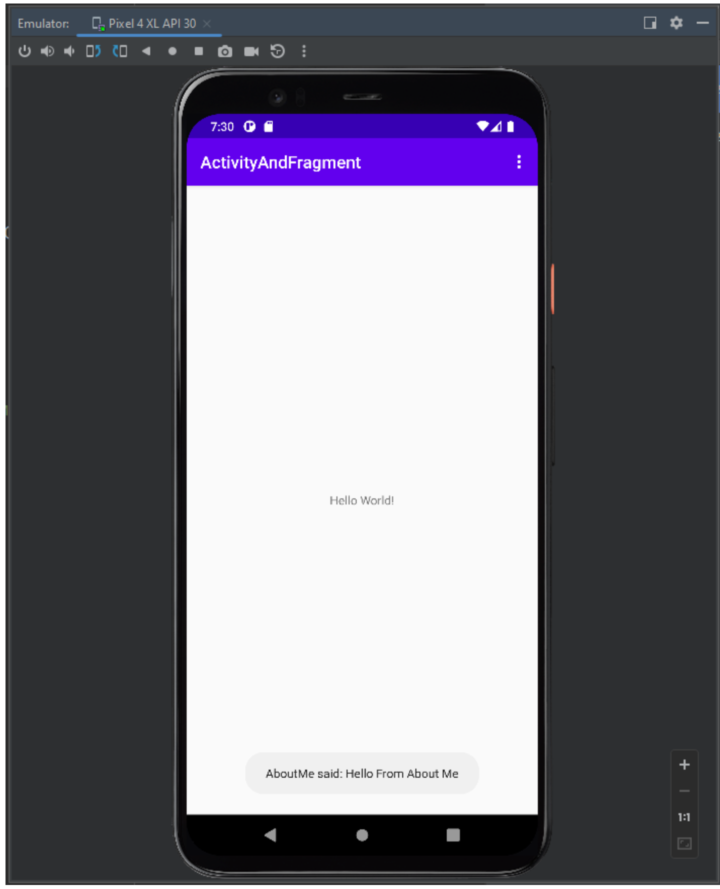
This is About Me

When the [Back] key is pressed



Hello World!

This is a send back message from AboutMe Activity:



Hello World!

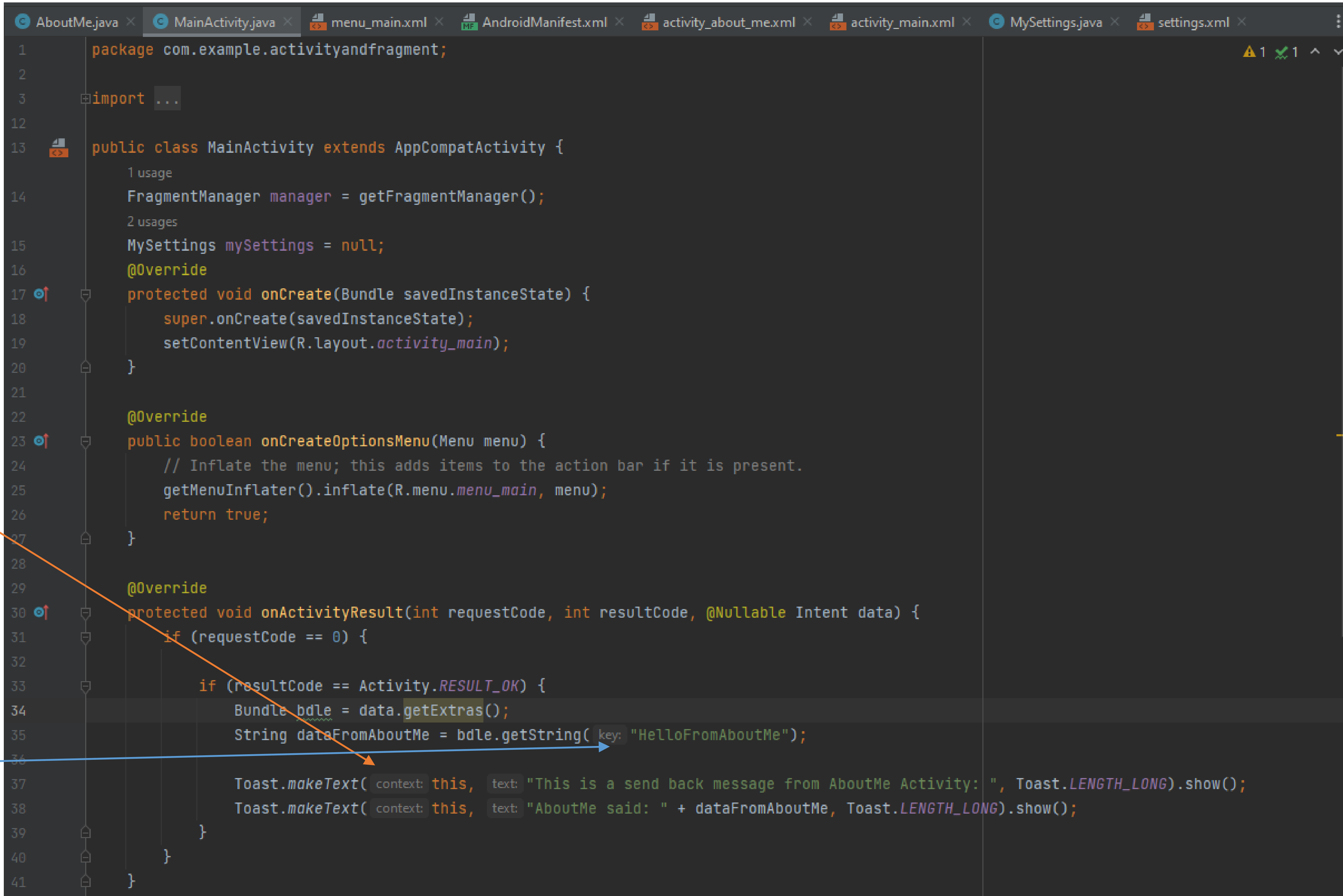
AboutMe said: Hello From About Me

AboutMe

MainActivity

MainActi

- The reason I get 2 Toast(s) in the MainActivity is:
- Like we did before.
- The bundle use the “key” to extract the value



```java
package com.example.activityandfragment;

import ...

public class MainActivity extends AppCompatActivity {
    1 usage
    FragmentManager manager = getFragmentManager();
    2 usages
    MySettings mySettings = null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        if (requestCode == 0) {

            if (resultCode == Activity.RESULT_OK) {
                Bundle bdle = data.getExtras();
                String dataFromAboutMe = bdle.getString( key: "HelloFromAboutMe");

                Toast.makeText( context: this,  text: "This is a send back message from AboutMe Activity: ", Toast.LENGTH_LONG).show();
                Toast.makeText( context: this,  text: "AboutMe said: " + dataFromAboutMe, Toast.LENGTH_LONG).show();
            }
        }
    }
```