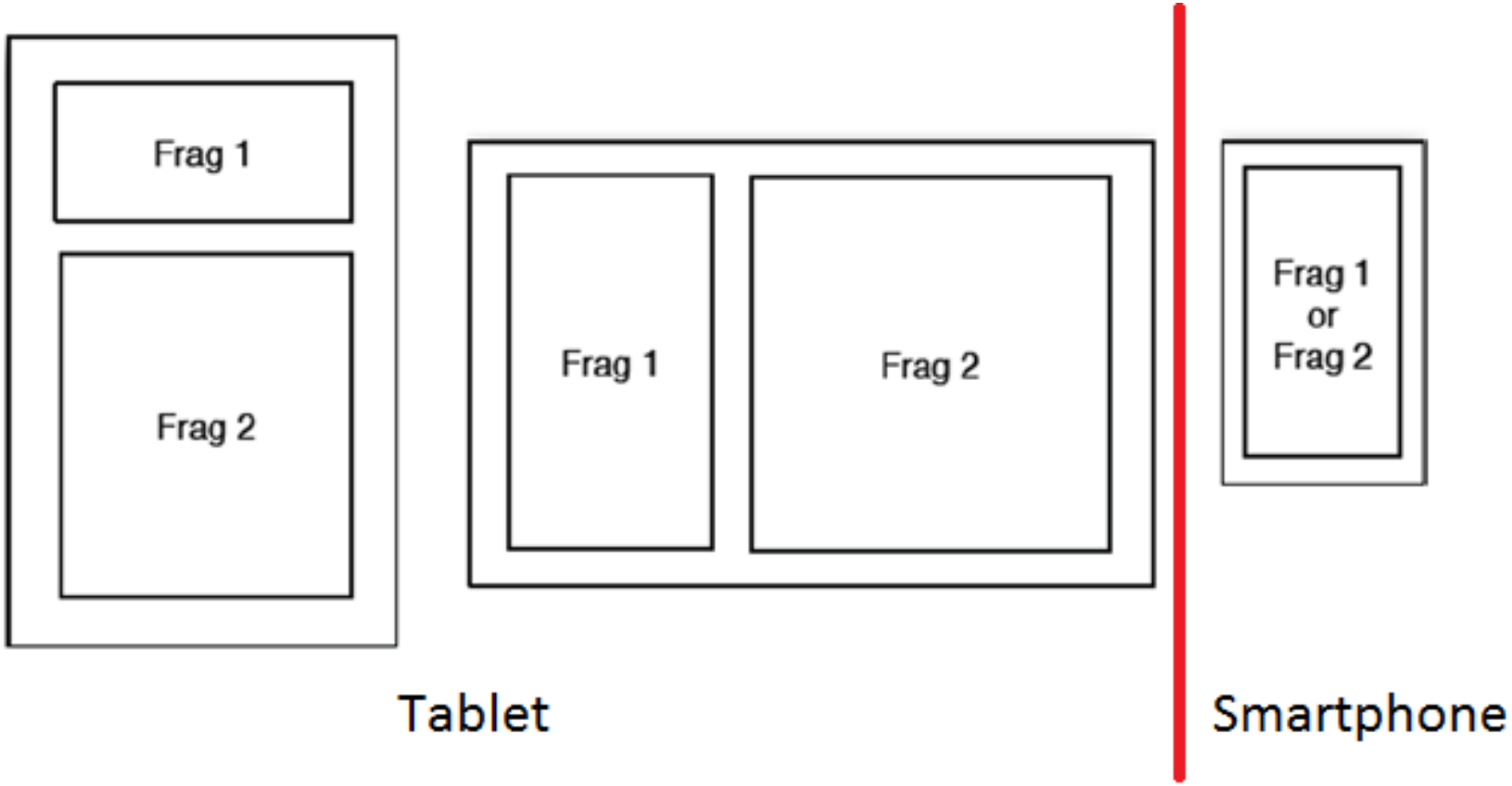
Android Studio

West University of Timisoara, Romania
Computer Science Department
IE3, Fall 2015
Dr. Liviu Octavian Mafteiu-Scai

Working with Fragments

A reason to use fragments



When a screen is large, it becomes difficult to manage all of its functionality in a single activity. *Fragments* are like sub-activities, and an activity can display one or more fragments on the screen at the same time. When a screen is small, an activity is more likely to contain just one fragment, and that fragment can be the same one used within larger screens.

What Is a Fragment?

One way to think of a ***fragment*** is as a *sub-activity*:

- can respond to the Back button like activities do;
- fragments are contained within an activity and can only exist within the context of an activity ie, it can't use a fragment without an activity.

But, fragments are not like activities, because the fragments framework provides several features to make saving and restoring fragments much simpler than on activities.

A fragment is not an extension of Activity class. It extends Fragment class or one of its subclasses.

An activity can have multiple fragments in play at one time.

What Is a Fragment?

- A Fragment is like an Activity: it has a set of events like an Activity. It also has its own associated *View object* which defines its UI - but it has no way of displaying that *View object*.
- To display its *View object* a Fragment has to pass it on to an Activity. The Activity decides how best to display the Fragment's UI.
- If the device has enough space then Fragments can be displayed on the same screen. If the device is too small, each Fragment will be displayed on its own. The management of Fragments according to screen size is entirely up to the programmer.
- The key method in using a Fragment is

onCreateView(LayoutInflater** inflater**, ViewGroup** container**, Bundle** savedInstanceSta**te)

that returns a single View object, always a ViewGroup, with the set of View object that defines the Fragment's UI. The Activity calls this event handler when it is time for the Fragment to provide its UI for display.

The main steps on creating fragments in Android Studio:

1. Create a New Project with a Blank Activity.

2. Create a Fragment by deriving your own class from the *Fragment* class. To do this you can simply right click on the *MainActivity* class and select *New, Java* Class, that creates a new file in the project to store the class.

3. Next, edit the class to make it inherit from Fragment class:

public class myFragment ***extends Fragment****{....*

4. Use *autocomplete* code and select *onCreateView* method:

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {.....

method to make sure the layout file is inflated and displayed when the fragment is used within an activity

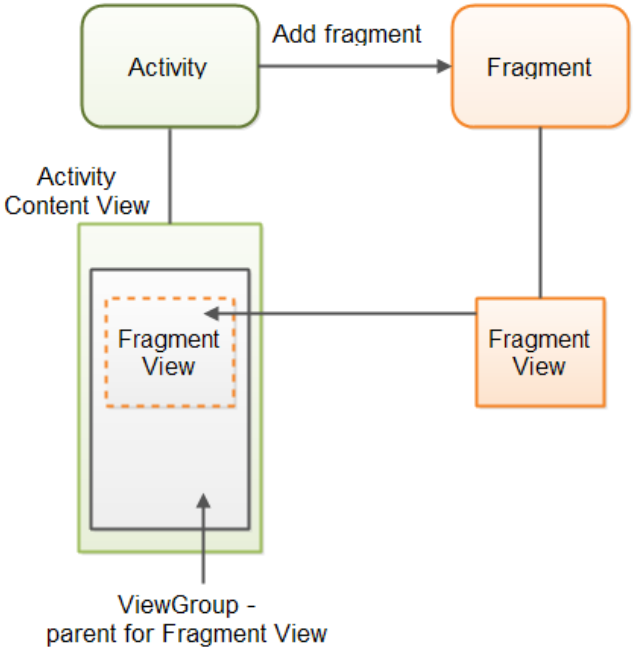
Instantiates a layout XML file into its corresponding View objects

The fragment's view is displayed inside this ViewGroup.

where it was initialized the activity

The main steps on creating fragments in Android Studio:

what happens when a fragment is added to an activity ?



a) The Activity obtains a reference to the Fragment.
b) The Activity gets a reference to the ViewGroup and the Fragment's view will be rendered inside.
c) The Activity adds the Fragment.
d) The Fragment creates its View and returns it to the Activity.
e) The View is then inserted into the ViewGroup parent, and the fragment is alive.

The main steps on creating fragments in Android Studio: *(continued)*

5. To create fragment's UI we could define an XML layout file or use the designer or create the View objects in code.

6. When the *onCreateView* is called, the Fragment will be associated with a Activity i.e. the Activity that is going to display its View. The *getActivity* method is used to return the Activity that the Fragment is associated with:

```
Button b = new Button(getActivity());
```

7. The Activity has a *FragmentManager* which is used to control how Fragment is displayed. To display a Fragment, first have to obtain the *FragmentManager* using *getFragmentManager* method. Then you have to get a *FragmentTransaction* from the *FragmentManager*.

8. You can't do anything with a Fragment unless you start a *transaction*. Within the transaction you can set up what you want to happen, usually add the Fragment to the current layout.

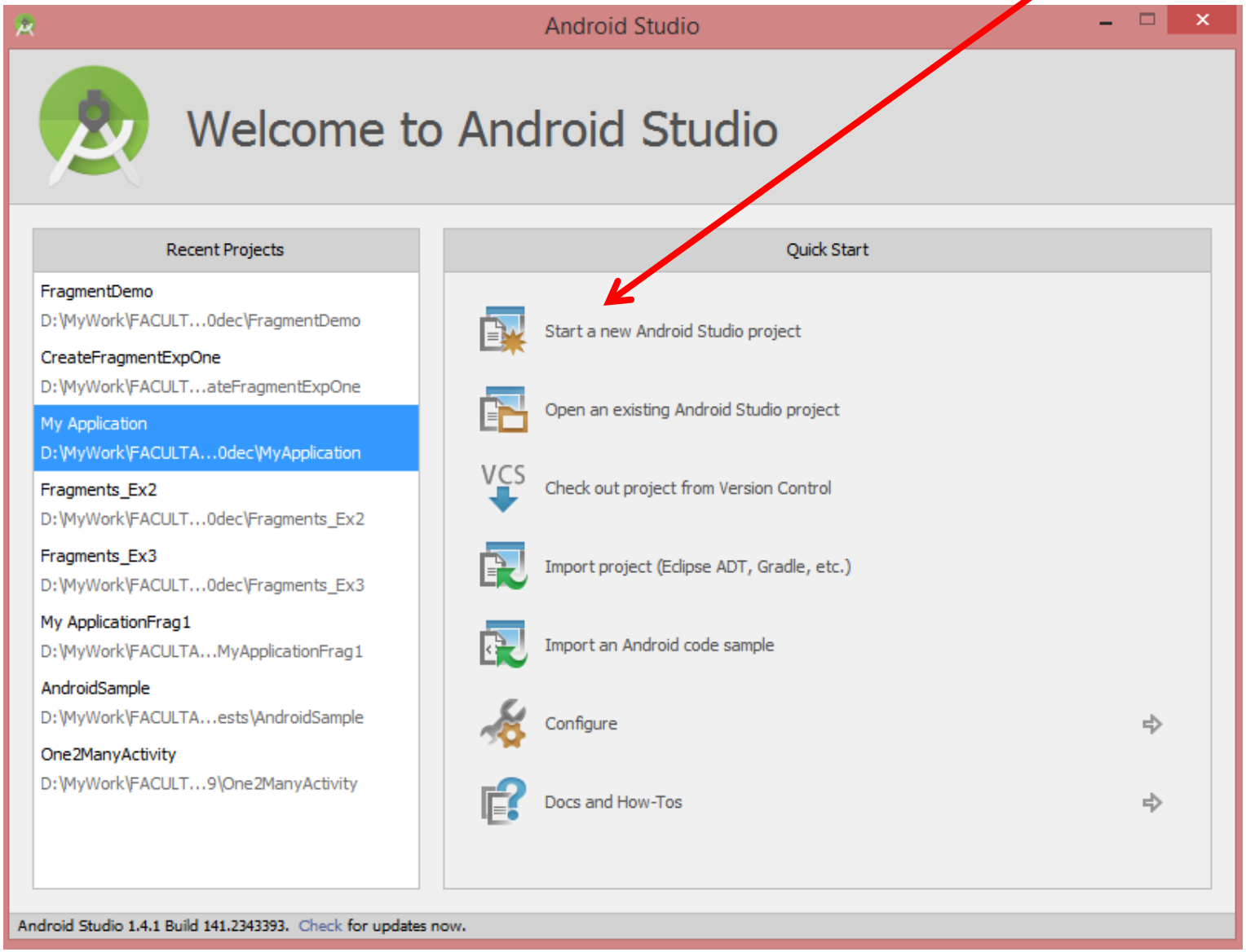
But, nothing happens until you use the *commit* method.

```
FragmentManager fm=getFragmentManager();
FragmentTransaction ft=fm.beginTransaction();
ft.add(100,frag1);      //100 Optional identifier of the container this fragment is to be placed in. If 0, it will
                        //not be placed in a container.
ft.commit();
```

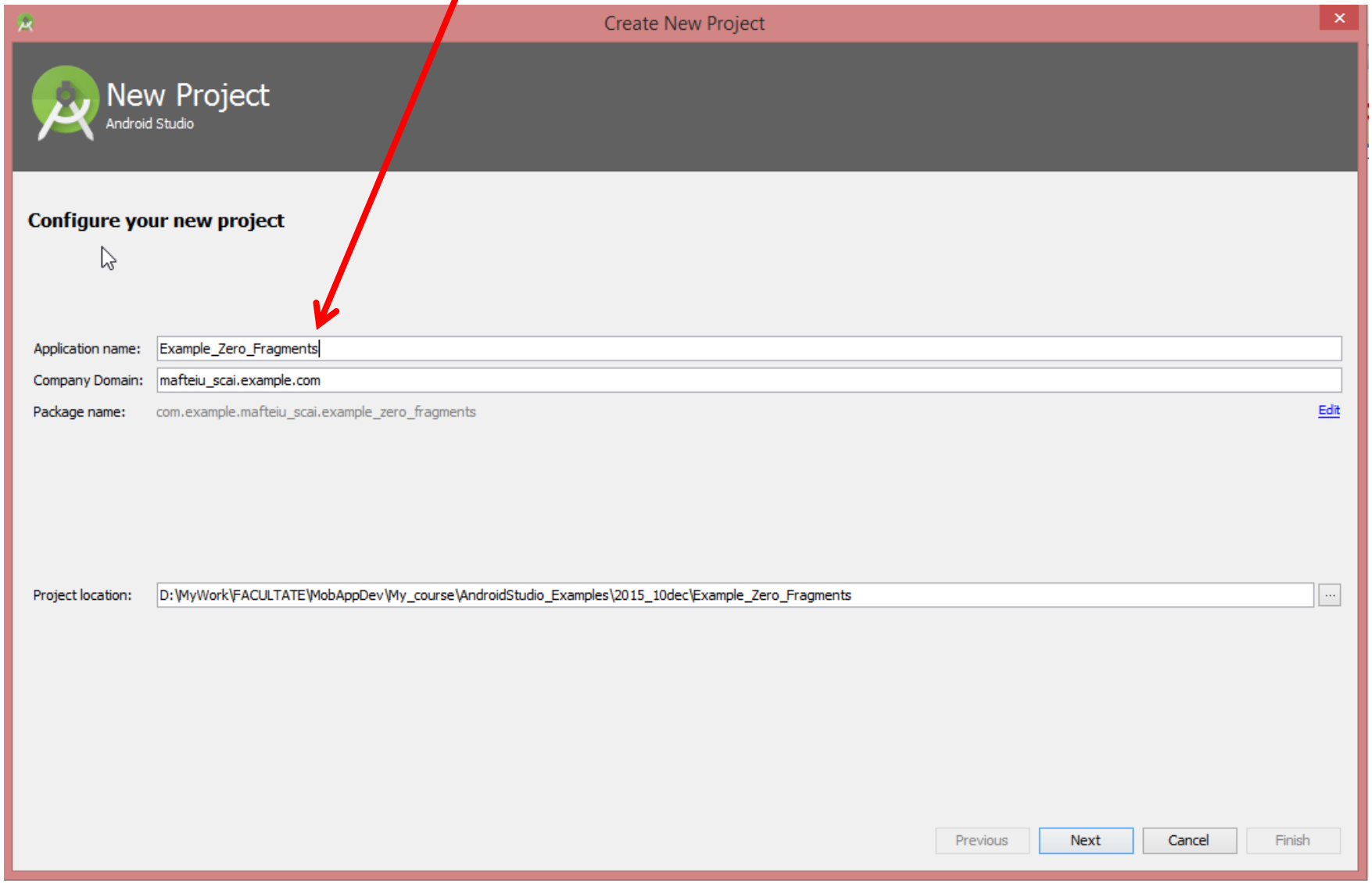
An example **step by step** to create an Activity with two fragments

Let's begin with the END:

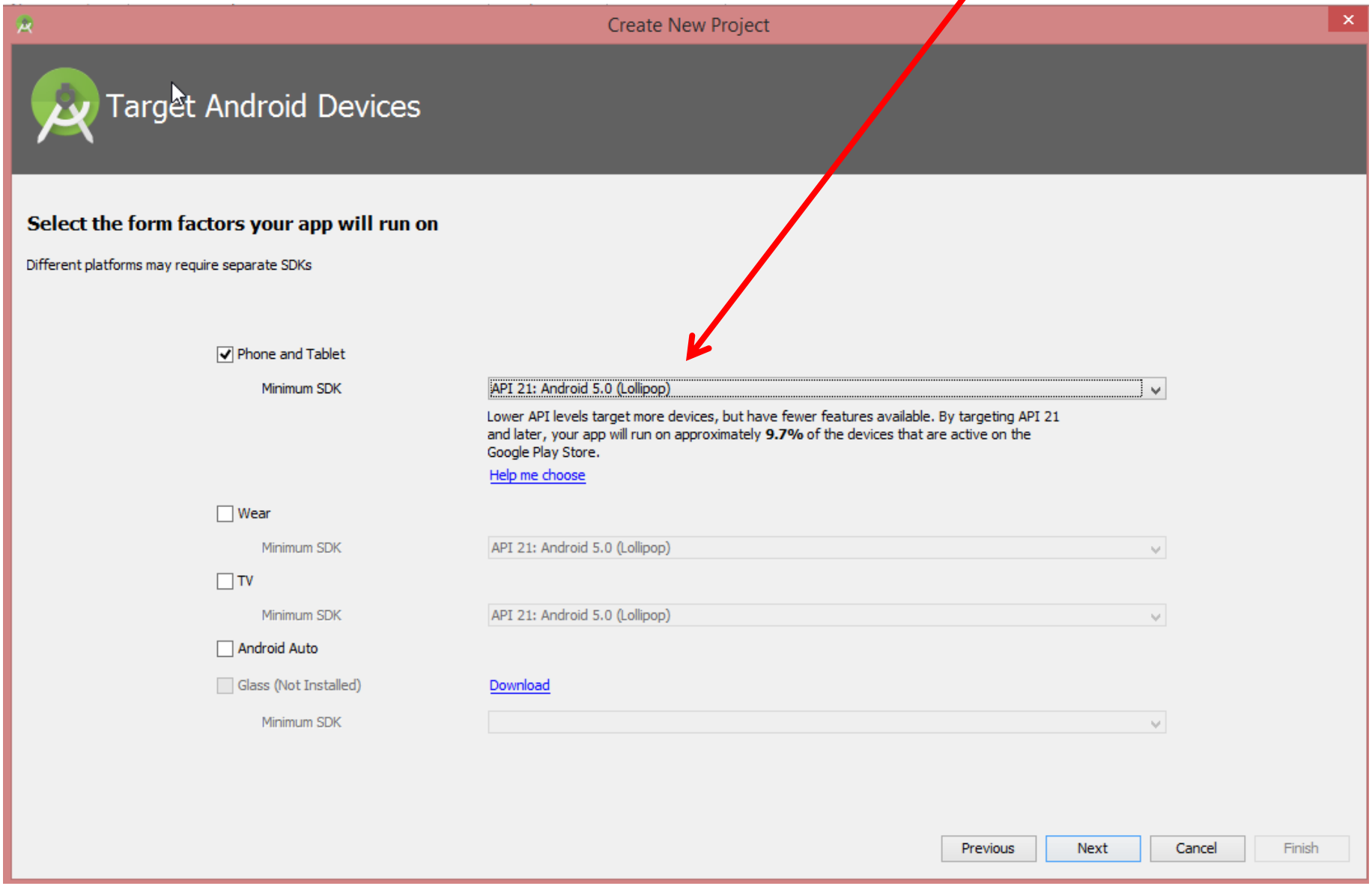
Start a new Project in Android Studio:



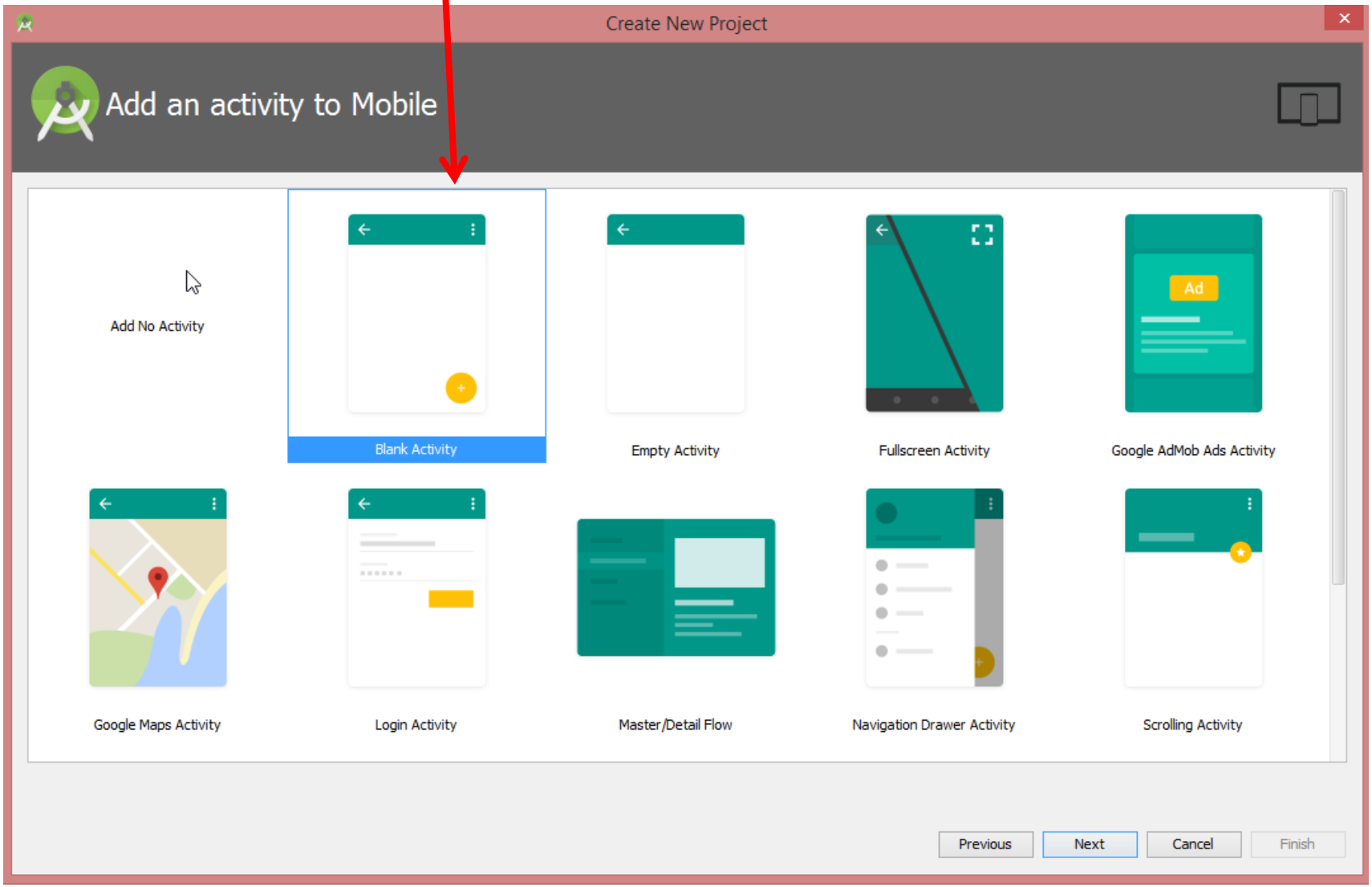
Set the application name:



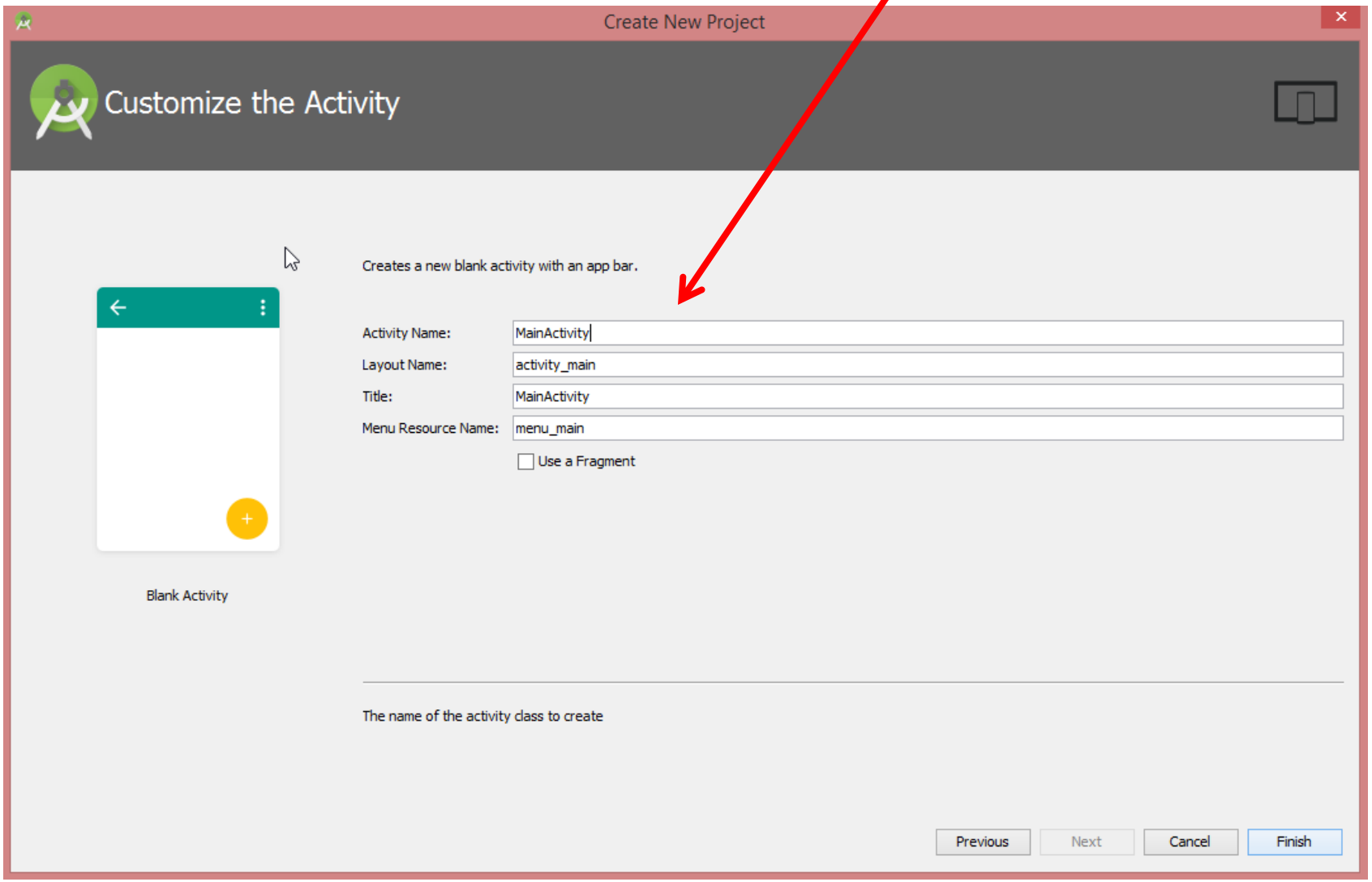
Set API level *(minimum Android 3.0 for fragments):*



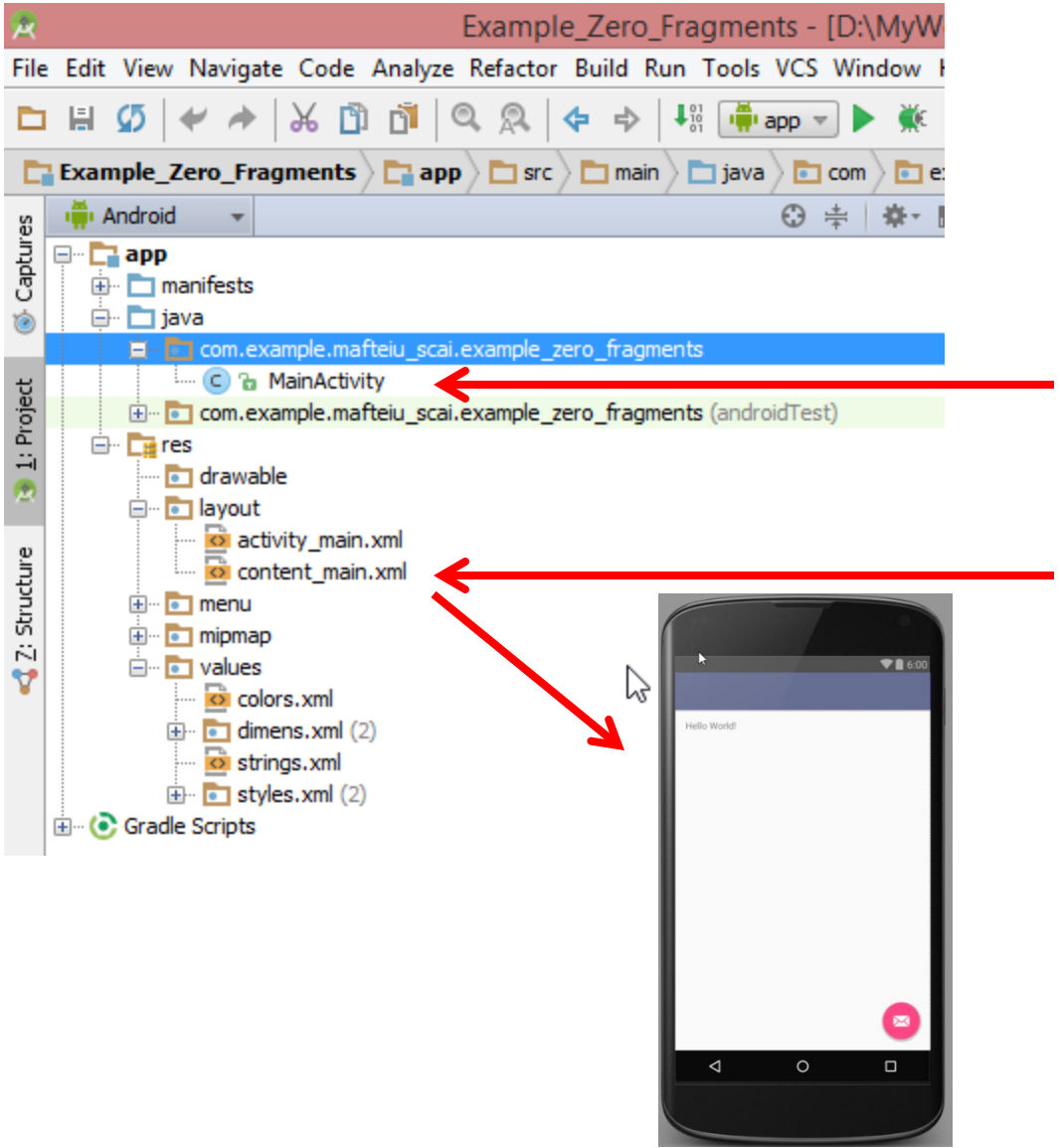
Select "Blank Activity":



Set activity and layout names for Main Activity:



Files generated by Android Studio:

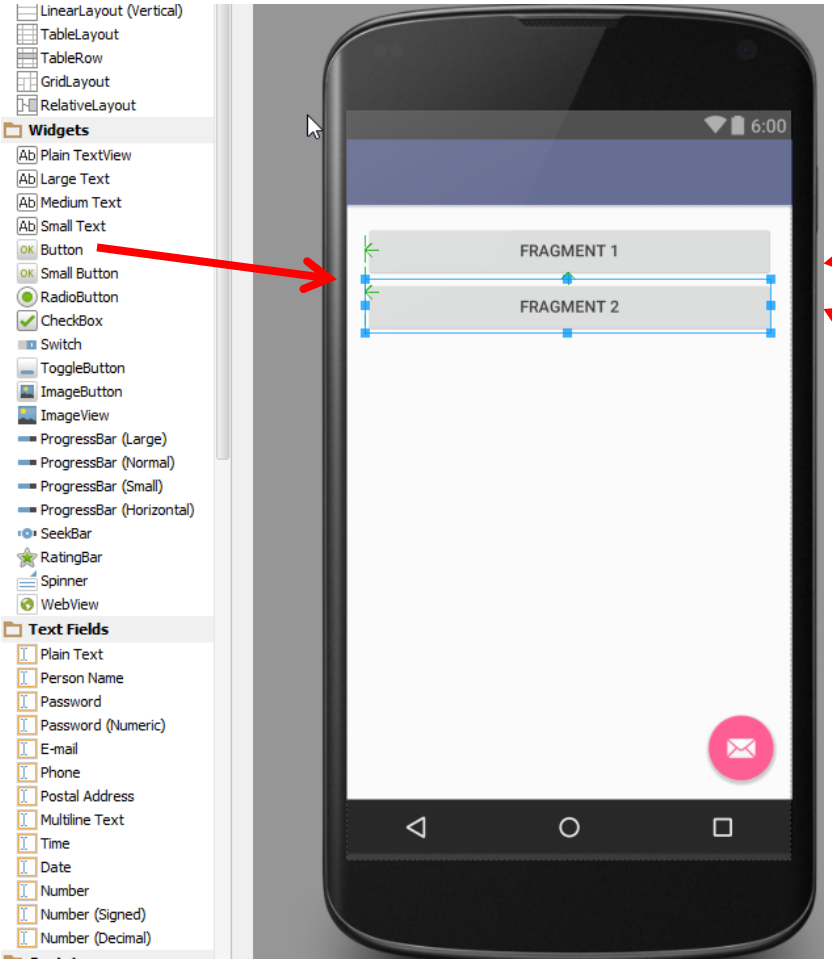


MainActivity.java

Layout for MainActivity

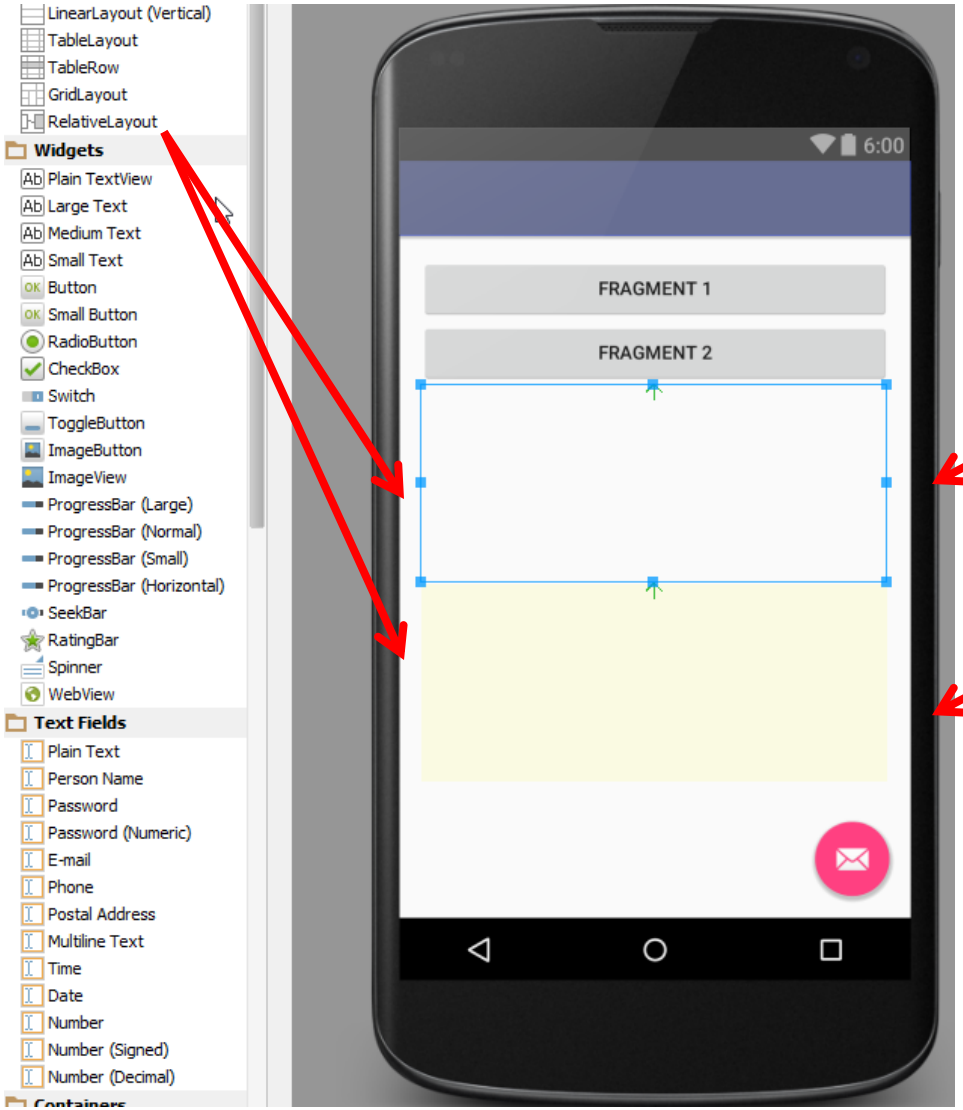
Add two buttons in MainActivity layout, used to call fragments:

*(could be used **Design** option for content_main.xml. And, delete "Hello World" resource)*




```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Fragment 1"
    android:id="@+id/b1"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true" />

<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Fragment 2"
    android:id="@+id/b2"
    android:layout_below="@+id/b1"
    android:layout_alignStart="@+id/b1" />
```

Insert two* RelativeLayout *s, used by fragments' views *(content_main.xml)*




```
<RelativeLayout
    android:id="@+id/fr1_id"
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:layout_below="@+id/b2">
</RelativeLayout>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:id="@+id/fr2_id"
    android:layout_below="@+id/fr1_id"
</RelativeLayout>
```

Let's add functionality!

First, modify MainActivity.java, ie delete unnecessary items. After these MainActivity.java must be:

mple > mafteiu_scai > example_zero_fragments > MainActivity

MainActivity.java × | content_main.xml ×

```
package com.example.mafteiu_scai.example_zero_fragments;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {

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

Let's add functionality!

Second, add buttons B1 and B2 in *MainActivity.java*, (class Button). Use *Alt-Enter* to automatic import the corresponding library.

After, link these to id/b1 and id/b2 from *content_main.xml*

MainActivity.java × content_main.xml ×

```
package com.example.mafteiu_scai.example_zero_fragments;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;

public class MainActivity extends Activity {
    Button B1,B2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        B1 = (Button)findViewById(R.id.b1);
        B2 = (Button)findViewById(R.id.b2);

    }
}
```

Instantiation classes and add methods to be invoked when buttons B1 and B2 are clicked in MainActivity

```
B1 = (Button)findViewById(R.id.b1);
B2 = (Button)findViewById(R.id.b2);
B1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

    }
});
B2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

    }
});
```

Now, create classes for fragments

(FragmentOne.java and FragmentTwo.java)*:* java->com.example…<right click> new->java class:

Android
app
manifests
java
com.example.mafteiu_scai.example_zero_fragments
FragmentOne
MainActivity
com.example.mafteiu_scai.example_zero_fragments (androidTest)
res
drawable
layout
activity_main.xml
content_main.xml
menu
mipmap
values
colors.xml
dimens.xml (2)
strings.xml
styles.xml (2)
Gradle Scripts

MainActivity.java | FragmentOne.java | content_main.xml

```
package com.example.mafteiu_scai.example_zero_fragments;

/**
 * Created by mafteiu-scai on 12/8/2015.
 */
public class FragmentOne {
}
```

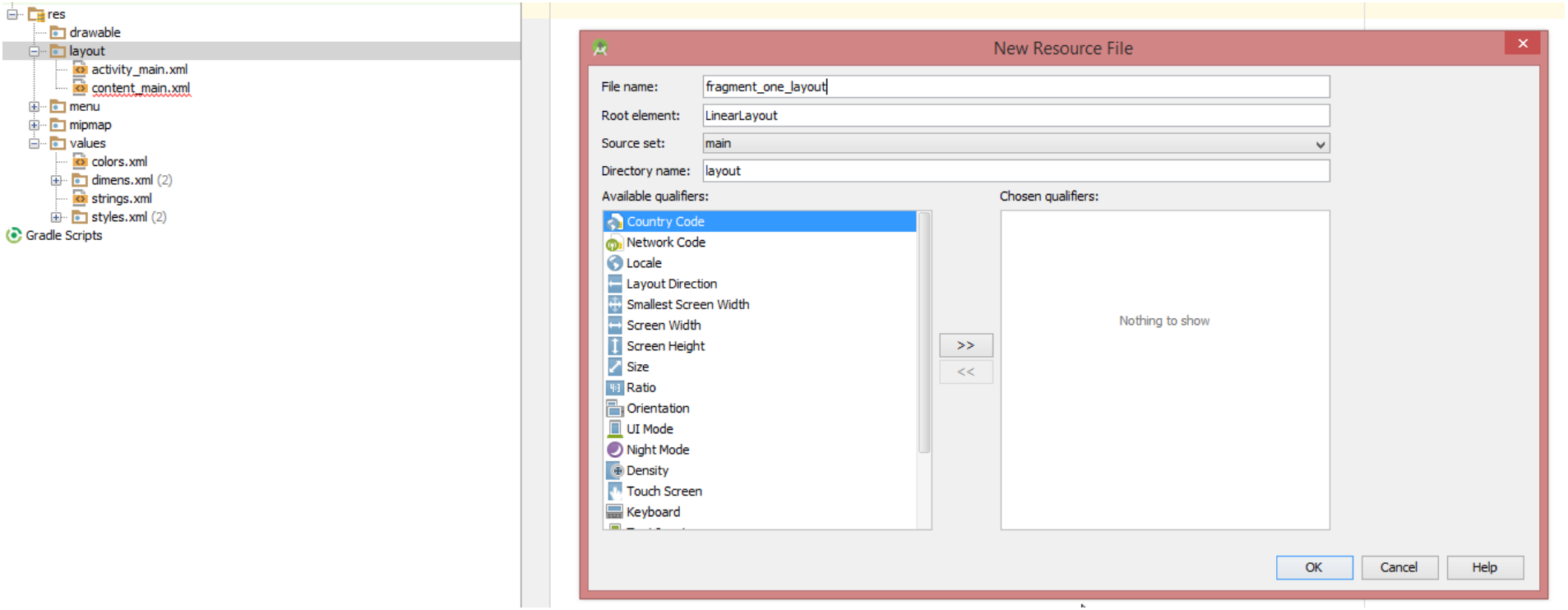
Create New Class
Name: FragmentTwo
Kind: Class
OK Cancel

MainActivity.java | FragmentOne.java | FragmentTwo.java | content_main.xml

```
package com.example.mafteiu_scai.example_zero_fragments;

/**
 * Created by mafteiu-scai on 12/8/2015.
 */
public class FragmentOne {
}
```

Now, create layouts for fragments

(fragment_one_layout.xml and fragment_two_layout.xml)*: r*es/layout/new…:



Now, modify layout for fragment one:

MainActivity.java | FragmentOne.java | fragment_one_layout.xml | fragment_two_l

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:orientation="vertical"
    android:background="#BB0A0A">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Fragment One"
        android:id="@+id/textView"
        android:layout_gravity="center_horizontal"
        android:textColor="#d3f0e7"
        android:textSize="50dp" />

</LinearLayout>
```

Similar, modify layout for fragment two:

MainActivity.java | FragmentOne.java | fragment_one_layout.xml | fragment_two_layout.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:orientation="vertical"
    android:background="#55AABB">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Fragment Two"
        android:id="@+id/textView2"
        android:layout_gravity="center_horizontal"
        android:textColor="#f4ec77"
        android:textSize="50dp" />

</LinearLayout>
```

Modify classes for fragments *(extend Fragment class from Android and add View for these)*

MainActivity.java | FragmentOne.java | fragment_one_layout.xml | fragment_two_layout.xml | FragmentTwo.java

```java
package com.example.mafteiu_scai.example_zero_fragments;
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 mafteiu-scai on 12/8/2015.
 */
public class FragmentOne extends Fragment{

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

Modify classes for fragments *(extend Fragment class from Android and add View for these)*

MainActivity.java × | FragmentOne.java × | fragment_one_layout.xml × | fragment_two_layout.xml × | FragmentTwo.java ×

```java
package com.example.mafteiu_scai.example_zero_fragments;

/**
 * Created by mafteiu-scai on 12/8/2015.
 */

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;

public class FragmentTwo extends Fragment {
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.fragment_two_layout, container, false);
        return v;
    }
}
```

Now, complete methods *OnClick* for B1 and B2 buttons into *MainActivity.java*

```
B1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FragmentManager FM = getFragmentManager();
        FragmentTransaction FT = FM.beginTransaction();
        FragmentOne F1 = new FragmentOne();
        FT.add(R.id.fr1_id, F1);
        FT.commit();
    }
});
B2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FragmentManager FM = getFragmentManager();
        FragmentTransaction FT = FM.beginTransaction();
        FragmentTwo F2 = new FragmentTwo();
        FT.add(R.id.fr2_id, F2);
        FT.commit();
    }
});
```

The app is ready but…don't respond to device's back button to close an activity*. For this, we must to add an AndroidStudio method to back (pop) in app's stack:*

```
B1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FragmentManager FM = getFragmentManager();
        FragmentTransaction FT = FM.beginTransaction();
        FragmentOne F1 = new FragmentOne();
        FT.add(R.id.fr1_id, F1);
        FT.addToBackStack("f1");
        FT.commit();
    }
});
B2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FragmentManager FM = getFragmentManager();
        FragmentTransaction FT = FM.beginTransaction();
        FragmentTwo F2 = new FragmentTwo();
        FT.add(R.id.fr2_id, F2);
        FT.addToBackStack("f2");
        FT.commit();
    }
});
```

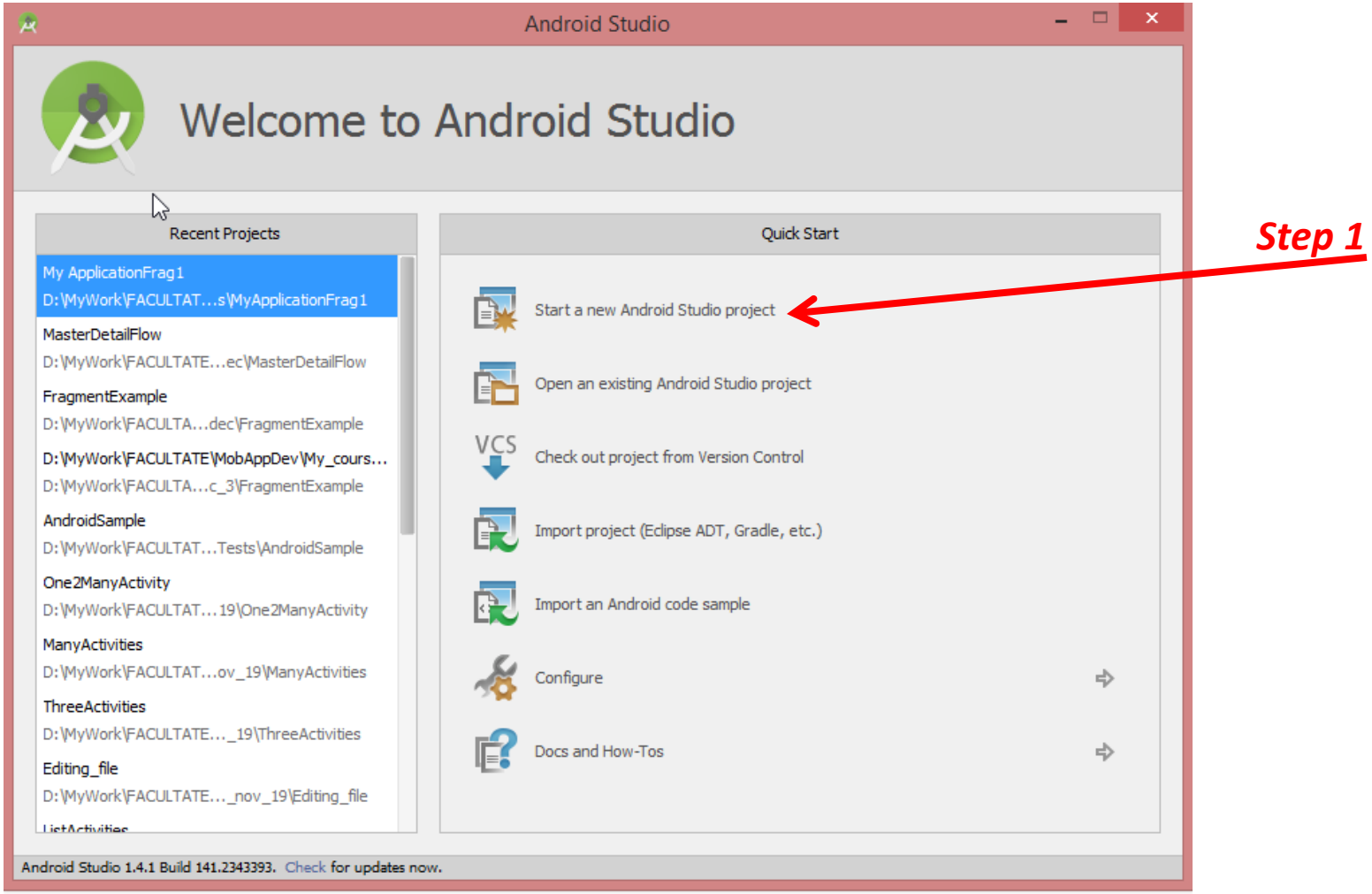
Oh,

I finally finished,

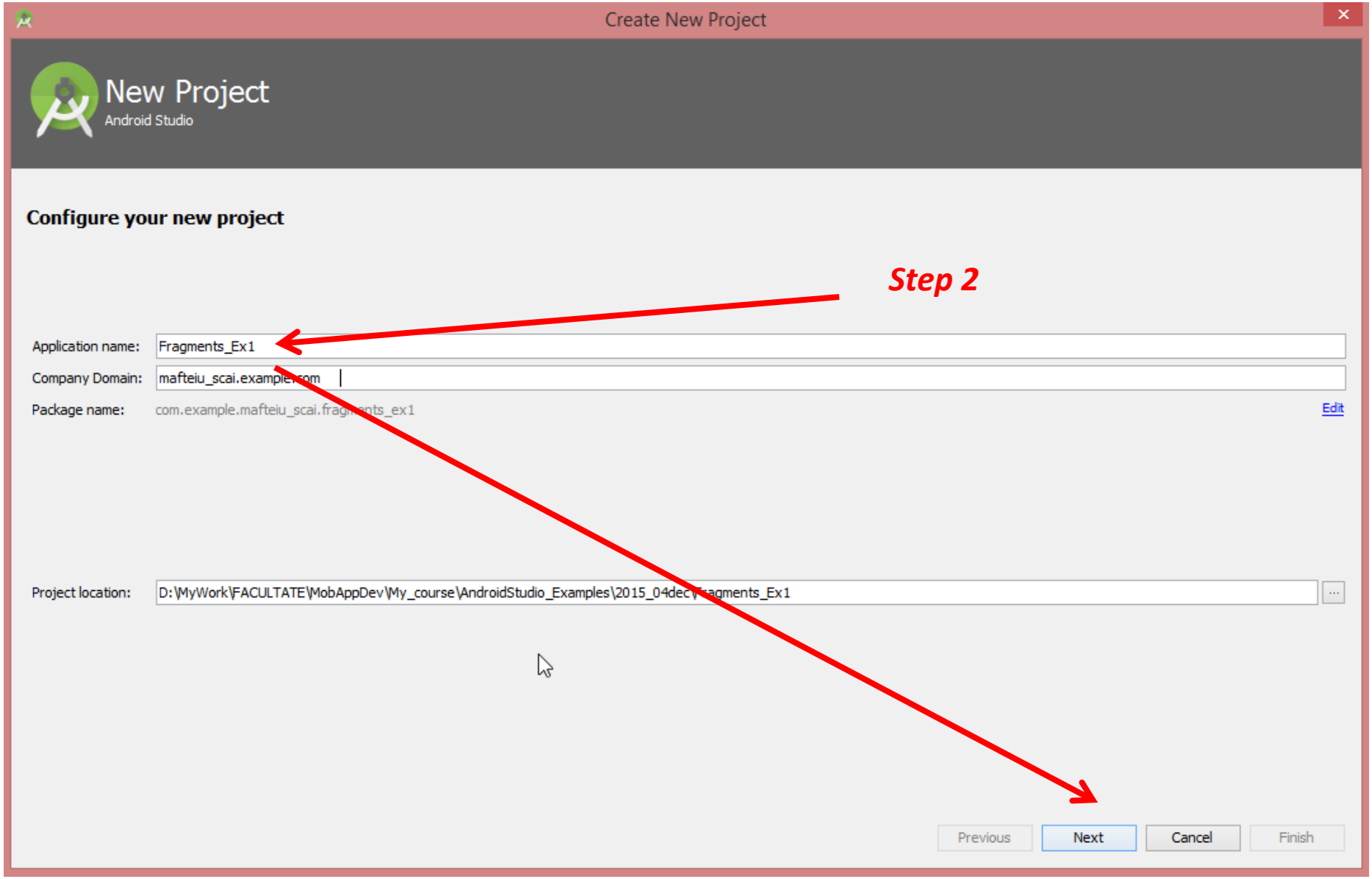
so

let's go to the next example

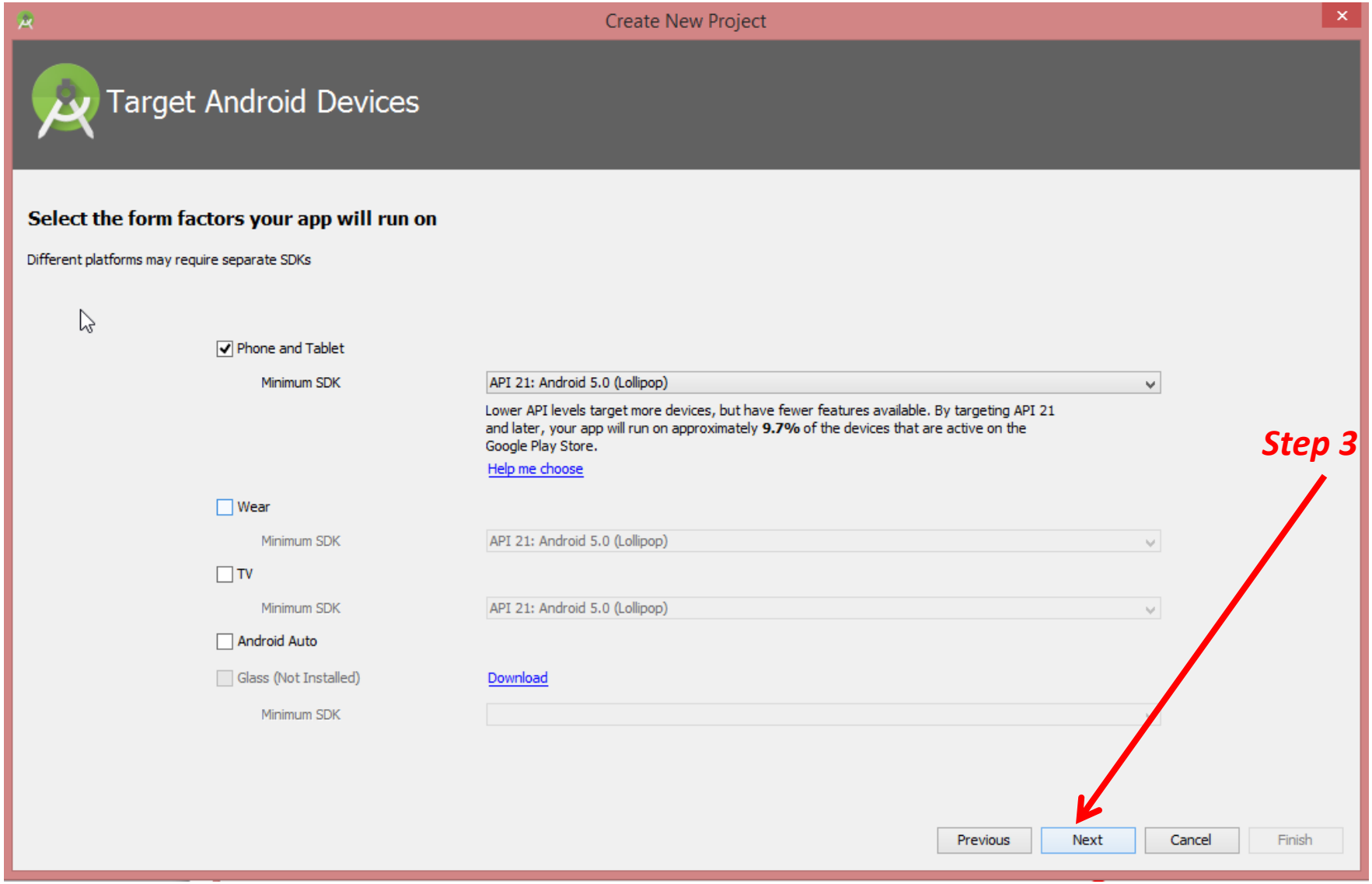
An application with fragments auto-generated by Android Studio



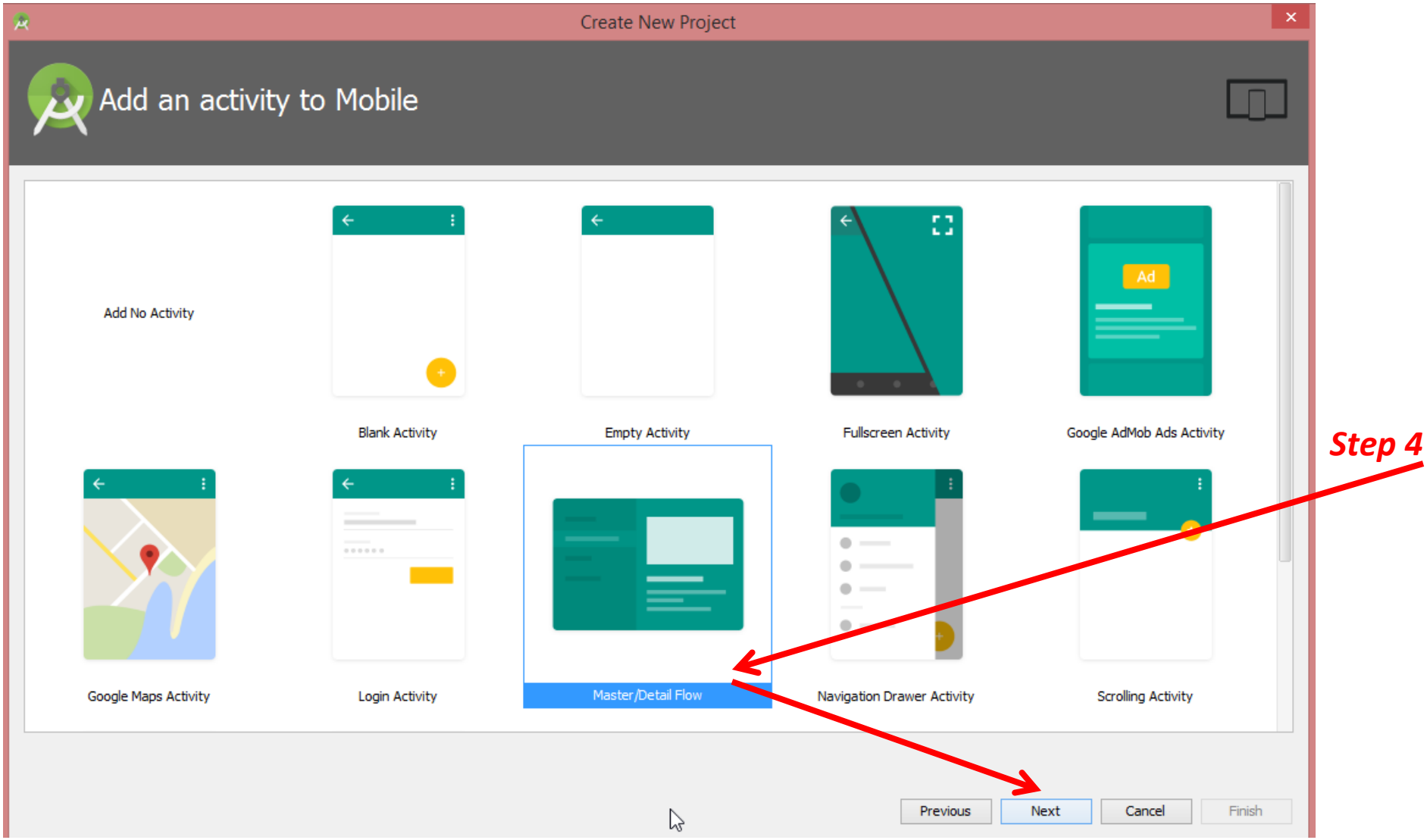
Mobile App Development 11, Android Studio ***Working with Fragments*** - Example 1



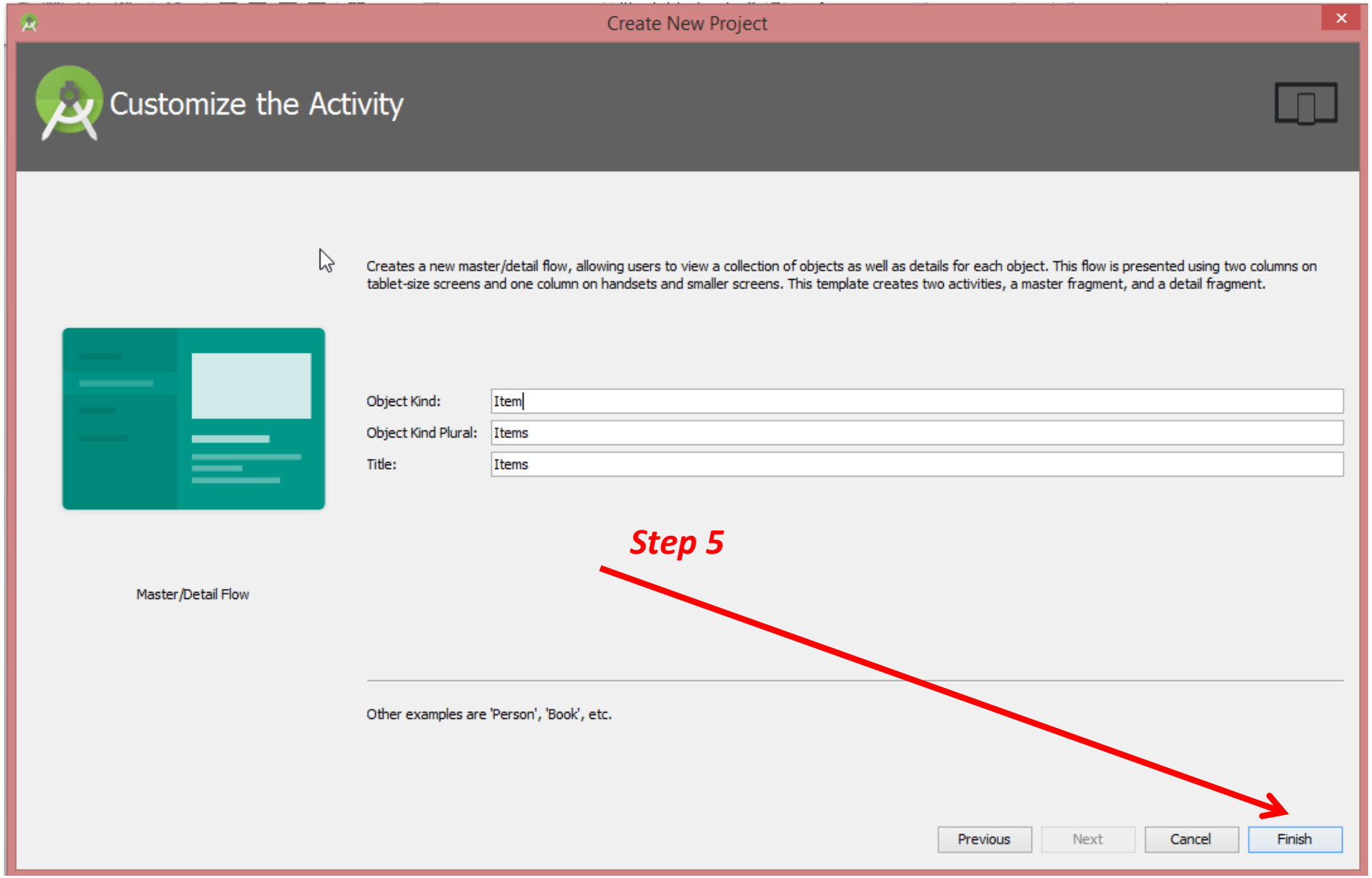
Mobile App Development 11, Android Studio ***Working with Fragments*** - Example 1



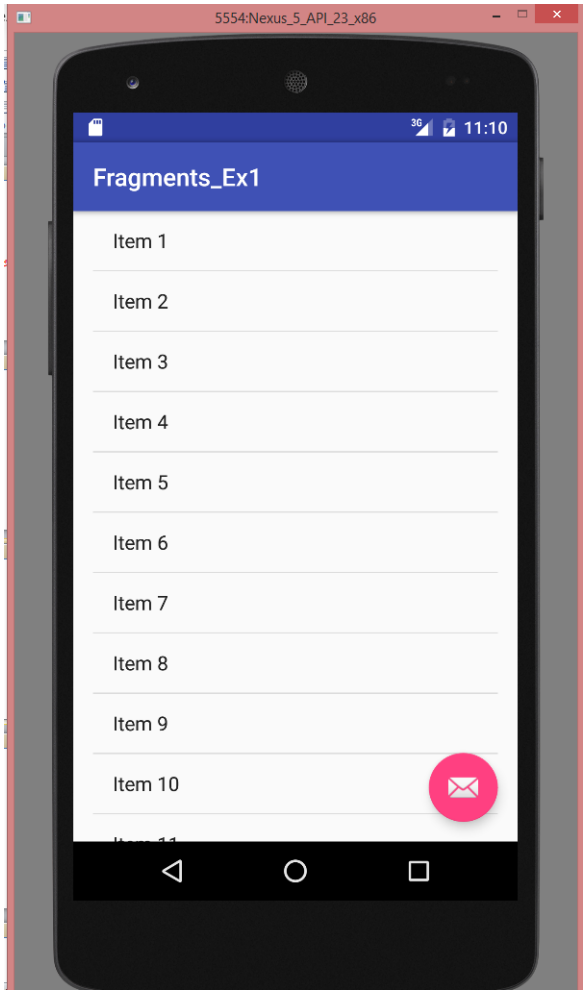
Mobile App Development 11, Android Studio ***Working with Fragments*** - Example 1

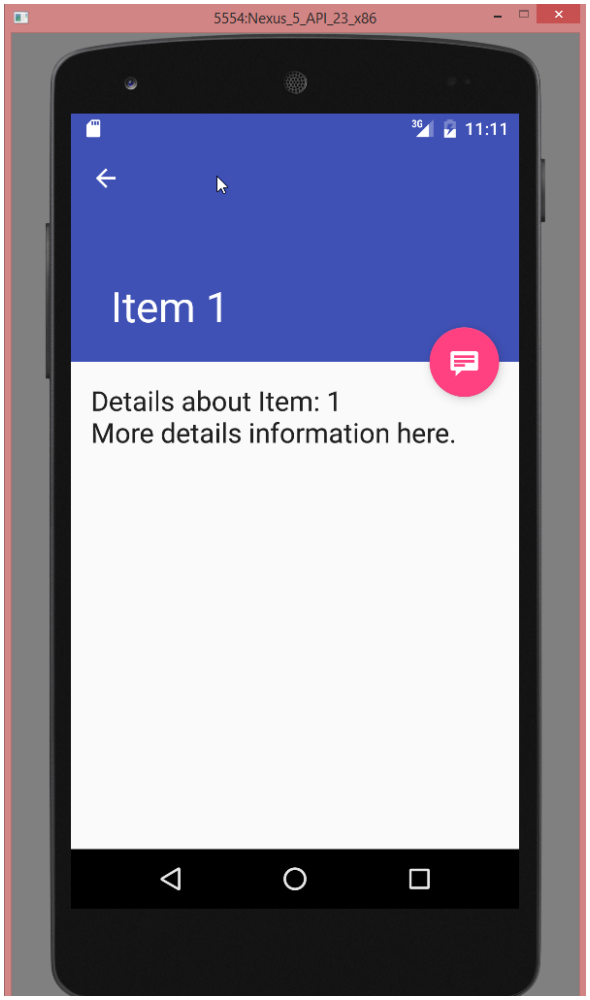


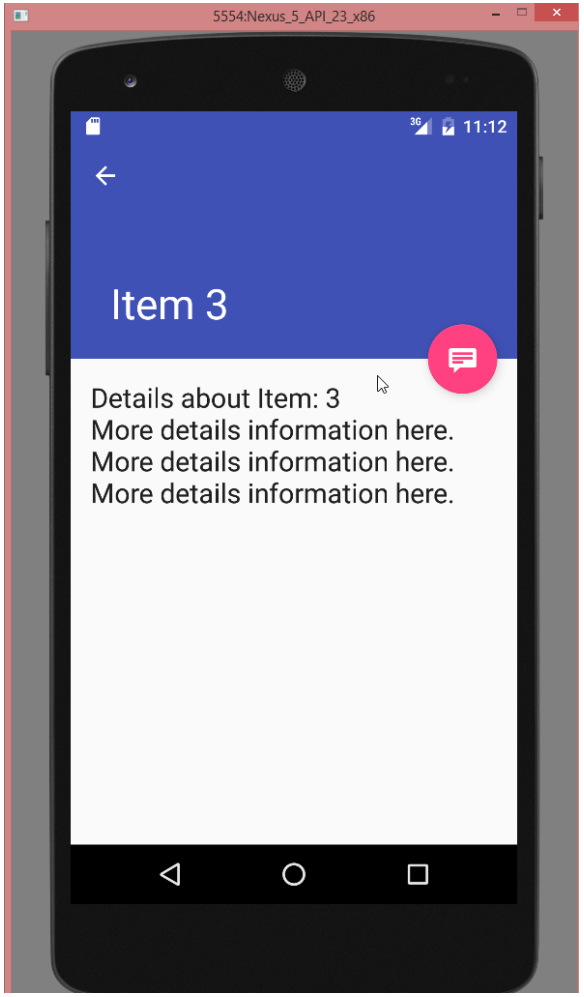
Mobile App Development 11, Android Studio ***Working with Fragments*** - Example 1



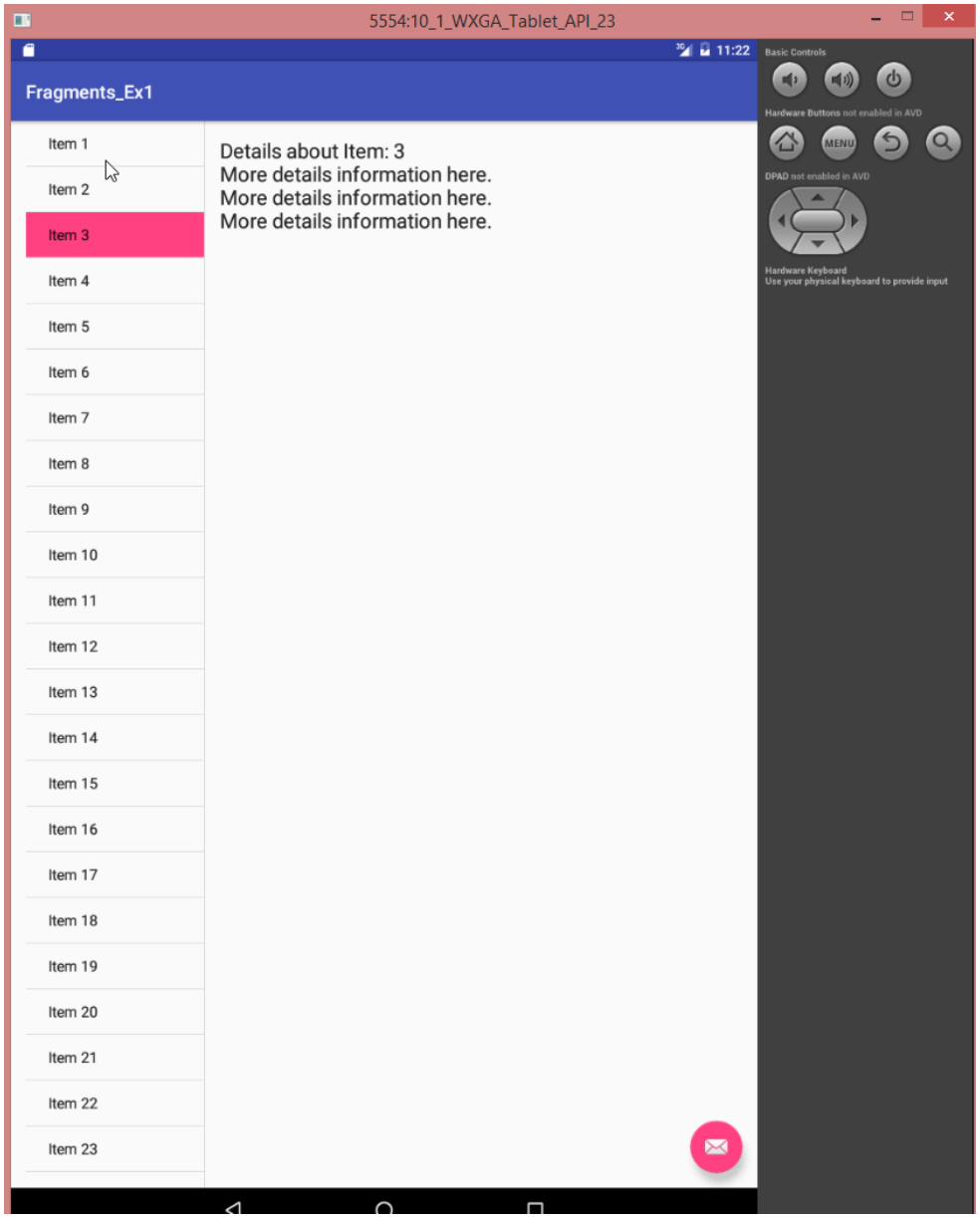
After these operations, for smartphones:







After these operations, for tablets:

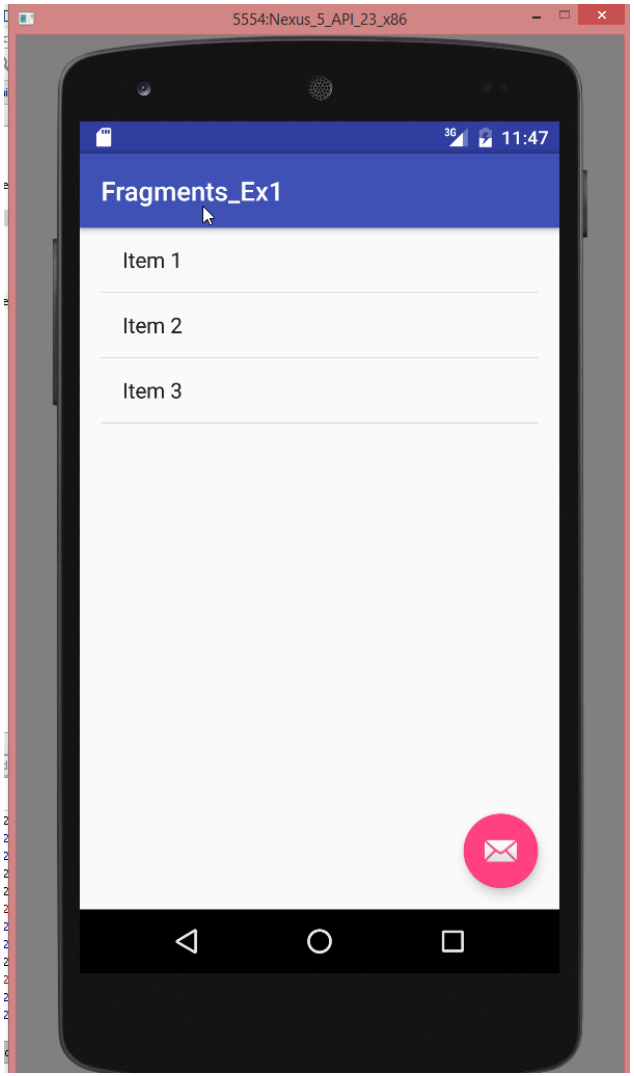


Let's adapt it to our requirements

From the beginning, we will reduce the number of items (originally 25) to 3:

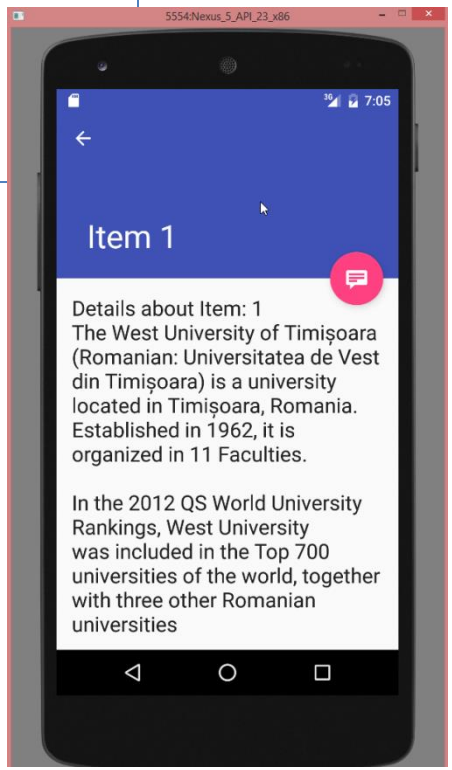
Open file "DummyContent.java" and replace *25* with *3* for *COUNT* value.

```
public class DummyContent {

    /**
     * An array of sample (dummy) items.
     */
    public static List<DummyItem> ITEMS = new ArrayList<~>();

    /**
     * A map of sample (dummy) items, by ID.
     */
    public static Map<String, DummyItem> ITEM_MAP = new HashMap<~>();

    private static final int COUNT = 3; //old value was 25;
```



Also, in file "DummyContent.java" you can change the text displayed in items:

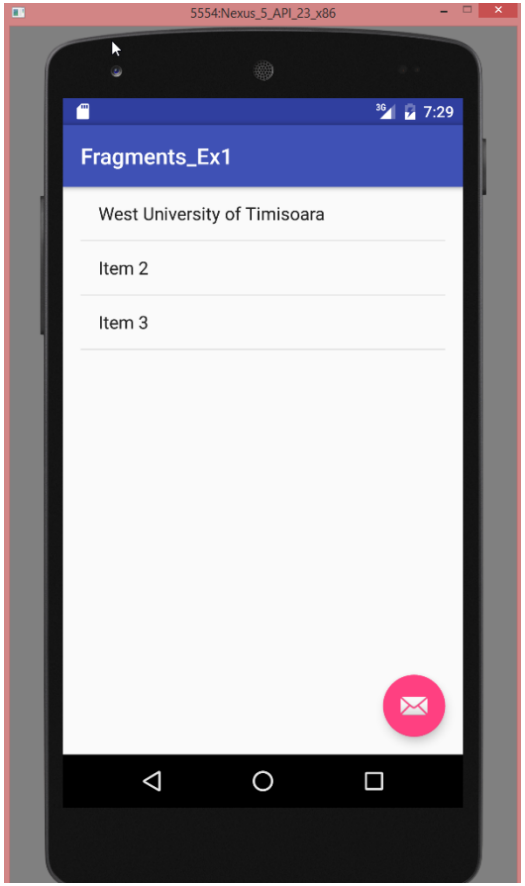
```
private static String makeDetails(int position) {
    StringBuilder builder = new StringBuilder();
    builder.append("Details about Item: ").append(position);
    /*for (int i = 0; i < position; i++) {
        builder.append("\nMore details information here.");
    }*/
    if(position==1)
        builder.append("\nThe West University of Timișoara (Romanian: Universitatea de Vest din Timișoara) " +
                "is a university located in Timișoara, Romania. " +
                "Established in 1962, it is organized in 11 Faculties. " +
                "In the 2012 QS World University Rankings, West University was included in the " +
                "Top 700 universities of the world, together with three other Romanian universities");
    if(position==2)
        builder.append("\nInformation 2" );
    if(position==3)
        builder.append("\nInformation 3.");
    return builder.toString();
}
```



Mobile App Development 11, Android Studio ***Working with Fragments*** - Example 1

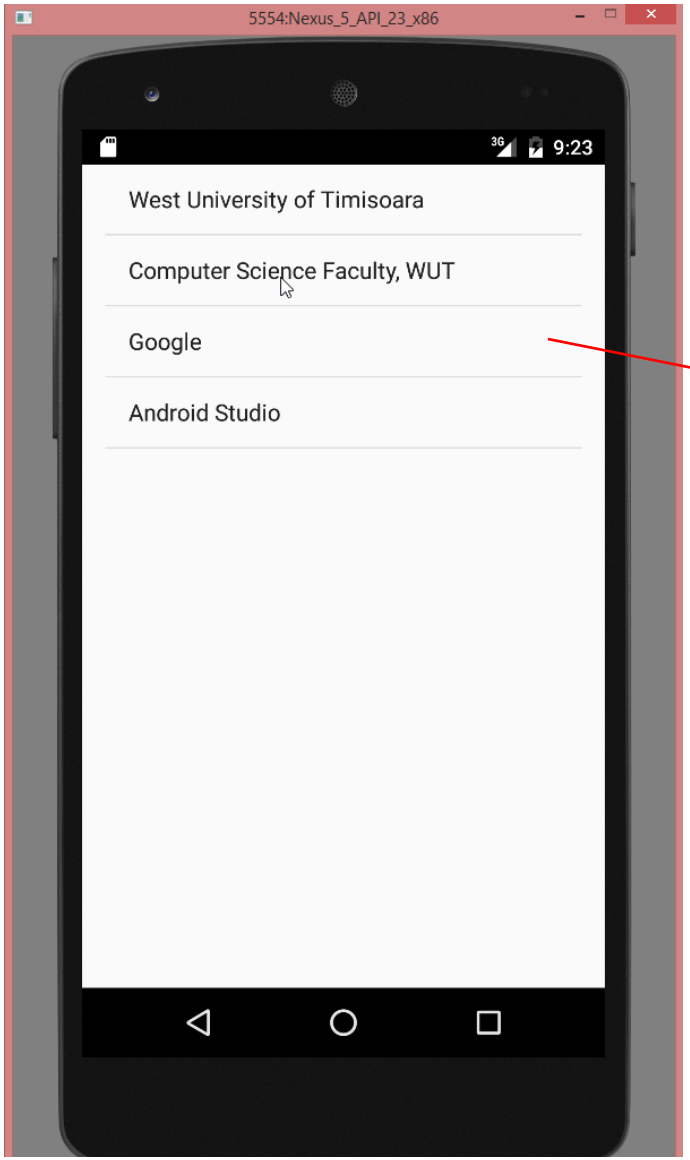
More, in file "DummyContent.java" you can change the items text:

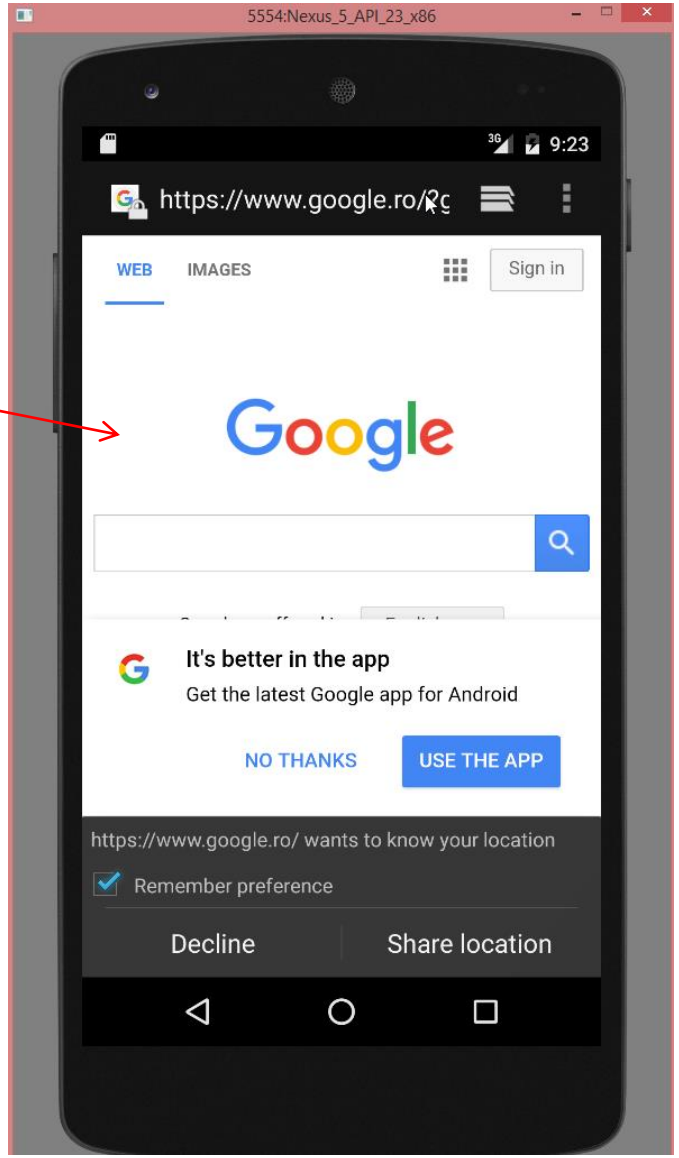
```
private static DummyItem createDummyItem(int position) {

    if(position==1)
        return new DummyItem(String.valueOf(position), "West University of Timisoara", makeDetails(position));
    else
        return new DummyItem(String.valueOf(position), "Item " + position, makeDetails(position));
}
```



See the project in Fragments_Ex1

Something similar, but for web access in items - *See project Fragments_Ex2*





Mobile App Development 11, Android Studio ***Working with Fragments*** – Example 2

5554:10_1_WXGA_Tablet_API_23

9:33

West University of Timisoara

Computer Science Faculty, WUT

Google

Android Studio

Developers

Show navigation

Android Studio Overview

Android Studio is the official IDE for Android application development, based on IntelliJ IDEA. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle-based build system
- Build variants and multiple `apk` file generation
- Code templates to help you build common app features
- Rich layout editor with support for drag and drop theme editing
- `lint` tools to catch performance, usability, version compatibility, and other problems
- ProGuard and app-signing capabilities
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine
- And much more

If you're new to Android Studio or the IntelliJ IDEA interface, this page provides an introduction to some key Android Studio features.

For specific Android Studio how-to documentation, see the pages in the Workflow section, such as Managing Projects from Android Studio and Building and Running from Android Studio. For a summary of the latest changes to Android Studio, see the Android Studio Release Notes.

Project and File Structure

Android project view

By default, Android Studio displays your project files in the *Android* project view. This view shows a flattened version of your project's structure that provides quick access to the key source files of Android projects and helps you work with the Gradle-based build system. The *Android* project view:

- Shows the most important source directories at the top level of the module hierarchy.
- Groups the build files for all modules in a common folder.
- Groups all the manifest files for each module in a common folder.
- Shows resource files from all Gradle source sets.
- Groups resource files for different locales, orientations, and screen types in a single group per resource type.

Basic Controls

Hardware Buttons not enabled in AVD

MENU

DPAD not enabled in AVD

Hardware Keyboard
Use your physical keyboard to provide input

Some comments about *DummyContent.java*

1. *Items are "placed" into an array:* **public static** List<DummyItem> *ITEMS* = **new** ArrayList<DummyItem>();

2. *To construct a **Map** is used **HashMap** (a Map implementation):*

public static Map<String, DummyItem> *ITEM_MAP* = **new** HashMap<String, DummyItem>();

3. *Name of items and their fragments:*

```
static {
   // Add 4 sample items.
   addItem(new DummyItem("1", "West University of Timisoara",
         "http://www.uvt.ro"));
   addItem(new DummyItem("2", "Computer Science Faculty, WUT",
         "http://www.info.uvt.ro"));
   addItem(new DummyItem("3", "Google",
         "http://www.google.com"));
   addItem(new DummyItem("4", "Android Studio",
         "http://developer.android.com/tools/studio/index.html"));
}
private static void addItem(DummyItem item) {
   ITEMS.add(item);
   ITEM_MAP.put(item.id, item);
}
```

Some comments about *WebsiteDetailFragment.java*

1. The class for fragments is a derived from Fragment class (import android.support.v4.app.Fragment)

```
public class WebsiteDetailFragment extends Fragment {
  /**
   * The fragment argument representing the item ID that this fragment represents.
   */
  public static final String ARG_ITEM_ID = "item_id";
```

2. To create fragment activity for each item in part:

```
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
  View rootView = inflater.inflate(R.layout.fragment_website_detail, container, false);

  // Show the dummy content as text in a TextView.
  if (mItem != null) {
    ((WebView) rootView.findViewById(R.id.website_detail))
        .loadUrl(mItem.website_url);
  }

  return rootView;
}
```

Some comments about *WebsiteListActivity.java*

1. *The detail container view will be present only in the large-screen layouts and the activity will be in two-pane mode:*

```
public void onItemSelected(String id) {
   if (mTwoPane) {
      Bundle arguments = new Bundle();
      arguments.putString(WebsiteDetailFragment.ARG_ITEM_ID, id);
      WebsiteDetailFragment fragment = new WebsiteDetailFragment();
      fragment.setArguments(arguments);
      getSupportFragmentManager().beginTransaction()
            .replace(R.id.website_detail_container, fragment)
            .commit();

   } else {
      // In single-pane mode, simply start the detail activity  for the selected item ID.
      Intent detailIntent = new Intent(this, WebsiteDetailActivity.class);
      detailIntent.putExtra(WebsiteDetailFragment.ARG_ITEM_ID, id);
      startActivity(detailIntent);
   }
}
```

Mobile App Development 11, Android Studio ***Working with Fragments – Example 3***

See the Fragment_Ex3 project

Ta-Ta for now!