Today’s lecture

1. Lab preparation
2. Walk through the first app
3. Basic Android UI Elements
4. Fragments

References

- This tutorial is a brief overview of some major concepts…Android is much richer and more complex
- Developer’s Guide
  - http://developer.android.com/guide/index.html
- API Reference
  - http://developer.android.com/reference/packages.html

Getting Started

- Install Android Studio (https://developer.android.com/studio/install.html)

- Install Android Virtual Device
https://developer.android.com/studio/run/managing-avds.html

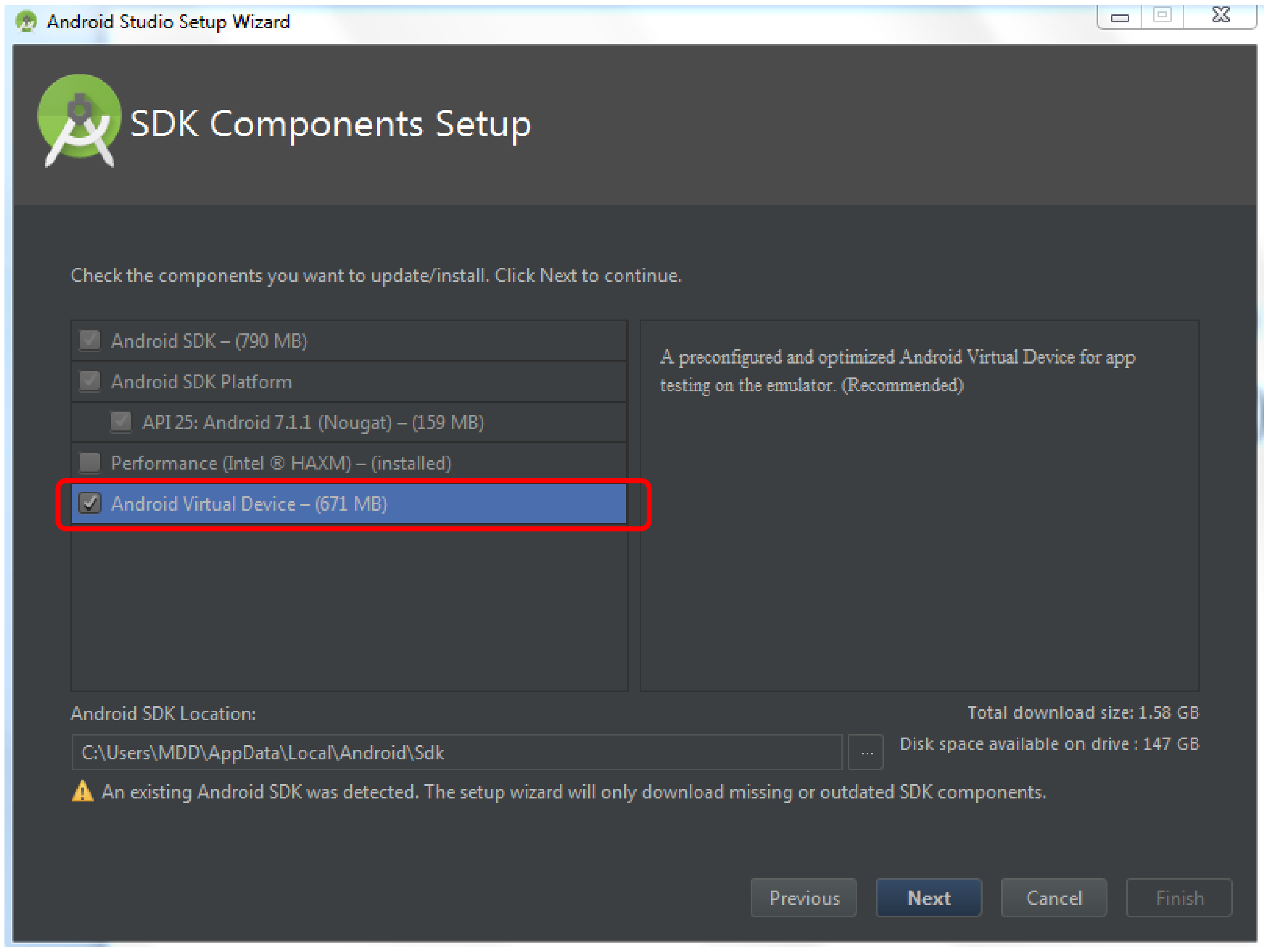
- Install drivers for Android phone
(Windows:https://developer.android.com/studio/run/oem-usb.html
*nix:https://developer.android.com/studio/run/device.html)

- Enable Android Development
https://www.kingoapp.com/root-tutorials/how-to-enable-usb-debugging-mode-on-android.htm

Installations

- Install Java Development Kit (JDK)
- Install Android Virtual Device (AVD)

Android Virtual Device



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

- [✓] Android SDK – (790 MB)
- [✓] Android SDK Platform
  - [✓] API 25: Android 7.1.1 (Nougat) – (159 MB)
- [ ] Performance (Intel ® HAXM) – (installed)
- [✓] Android Virtual Device – (671 MB)

A preconfigured and optimized Android Virtual Device for app testing on the emulator. (Recommended)

Android SDK Location: C:\Users\MDD\AppData\Local\Android\Sdk

Total download size: 1.58 GB

Disk space available on drive : 147 GB

An existing Android SDK was detected. The setup wizard will only download missing or outdated SDK components.

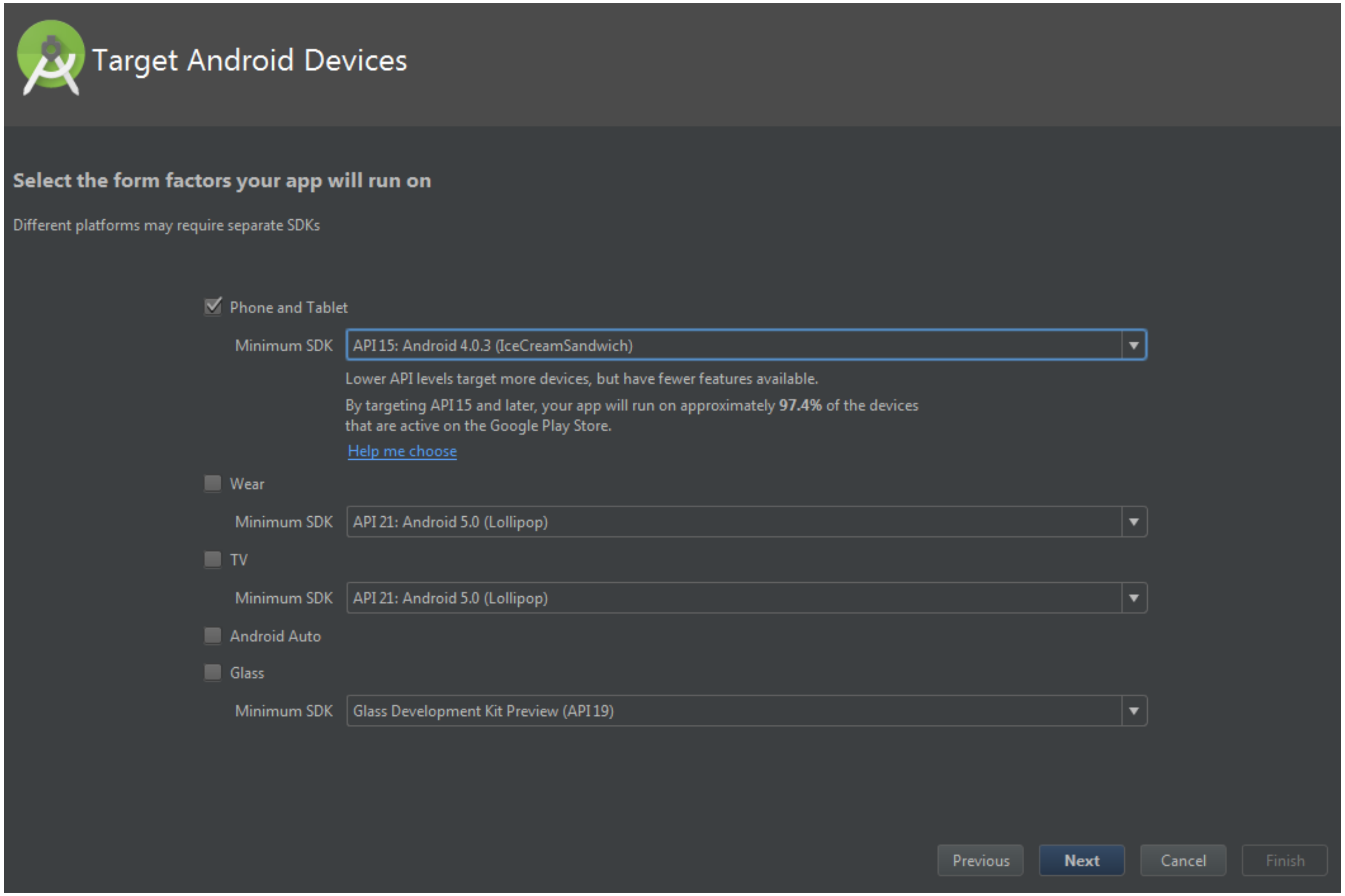
Generate the first app

- Instructions:

https://developer.android.com/training/basics/firstapp/creating-project.html

Generate the first app

Target Device: depends on your phone



Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

[✓] Phone and Tablet

Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available.

By targeting API 15 and later, your app will run on approximately **97.4%** of the devices that are active on the Google Play Store.

Help me choose

[ ] Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

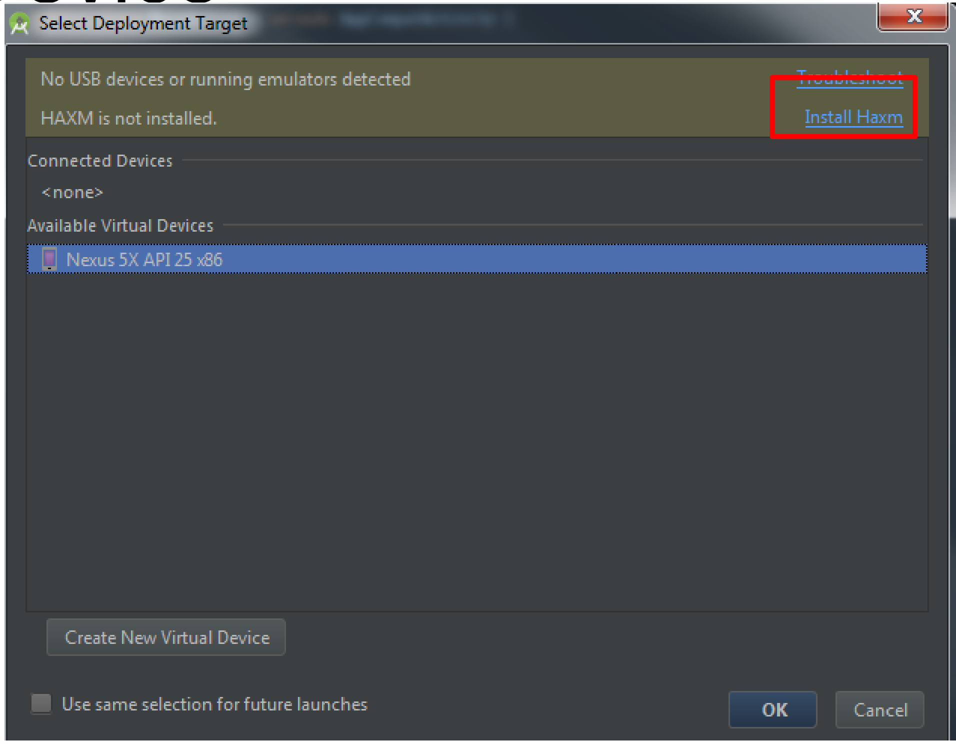
[ ] Android Auto

[ ] Glass

Minimum SDK: Glass Development Kit Preview (API 19)

Run the app

- Use Virtual Devices
- (Optional) Install Drivers
- Create New Virtual Device


Select Deployment Target

No USB devices or running emulators detected

HAXM is not installed.

Troubleshoot
Install Haxm

Connected Devices

<none>

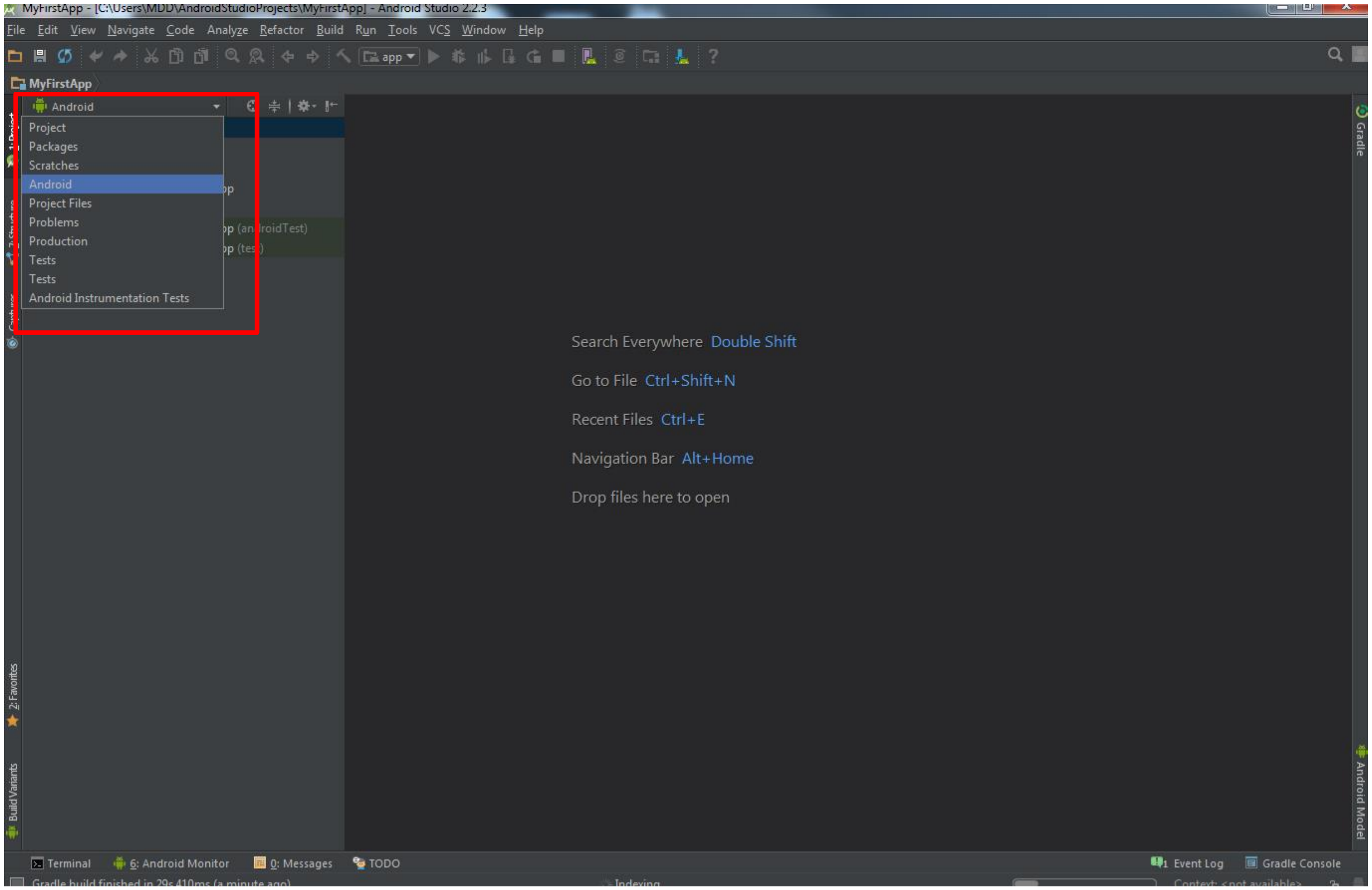
Available Virtual Devices

Nexus 5X API 25 x86

Create New Virtual Device
Use same selection for future launches
OK
Cancel

Project Components

- Go to the Android view:



Project Components

- app > java > com.example.myfirstapp > MainActivity.java – The java codes
- app > res > layout > activity_main.xml – Defines the layout
- app > manifests > AndroidManifest.xml – Defines basic properties, like permissions, declare activities and services, and so on
- res
  - Drawables (like .png images)
  - Values (like strings)

Extensible Markup Language (XML)

- Preferred way of creating UI
  - Separates the description of the layout from any actual code that controls it
  - Can easily take a UI from one platform to another
  - Both human and machine readable

Manifest File

- Contains characteristics about application
- Specify all Activities and Services here
- Specify all permissions
- Specify some libraries, like Google Maps API

Layouts

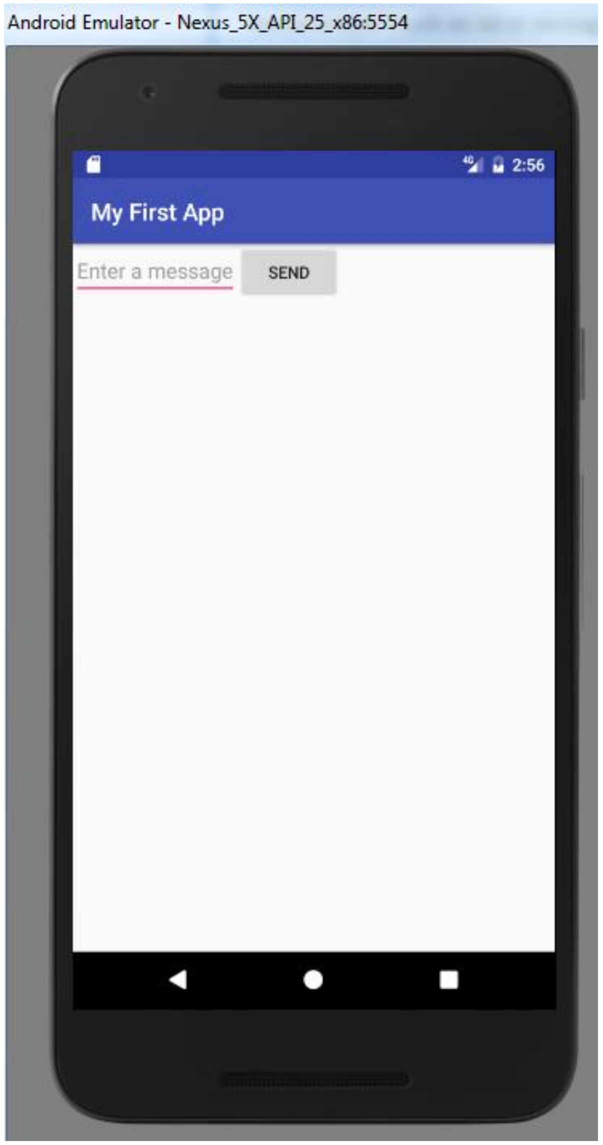
- Composed of *View* objects
- Can be specified for portrait and landscape mode
  - Use same file name, so can make completely different UIs for the orientations without modifying any code

Strings

- In res/values
  - strings.xml
- Promotes good software engineering

- Application wide available strings

Build this app



Lab 1.a Basic UI

- The app has a interface that contains a button and a text input field.
- Once the button is pushed, generate an intent that invokes a second activity
- The second activity will display the input string
- Reference: https://developer.android.com/training/basics/firstapp/creating-project.html

Implement a Basic UI

- Open “app > res > layout > activity_main.xml”
- Select the text tab. We are working with the xml codes

Implement a Basic UI (2)

Delete everything and paste the following xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:orientation="horizontal">
</LinearLayout>
```

Understand Text Field

```
<EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
```

- android:id
- android:layout_width
- android:hint

Implement a Basic UI (3)

- Add a Text Field Element:

```
<LinearLayout
    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:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
</LinearLayout>
```

Implement a Basic UI (4)

- Define the String field
- Go to : res/values/strings.xml
- Delete everything and paste:

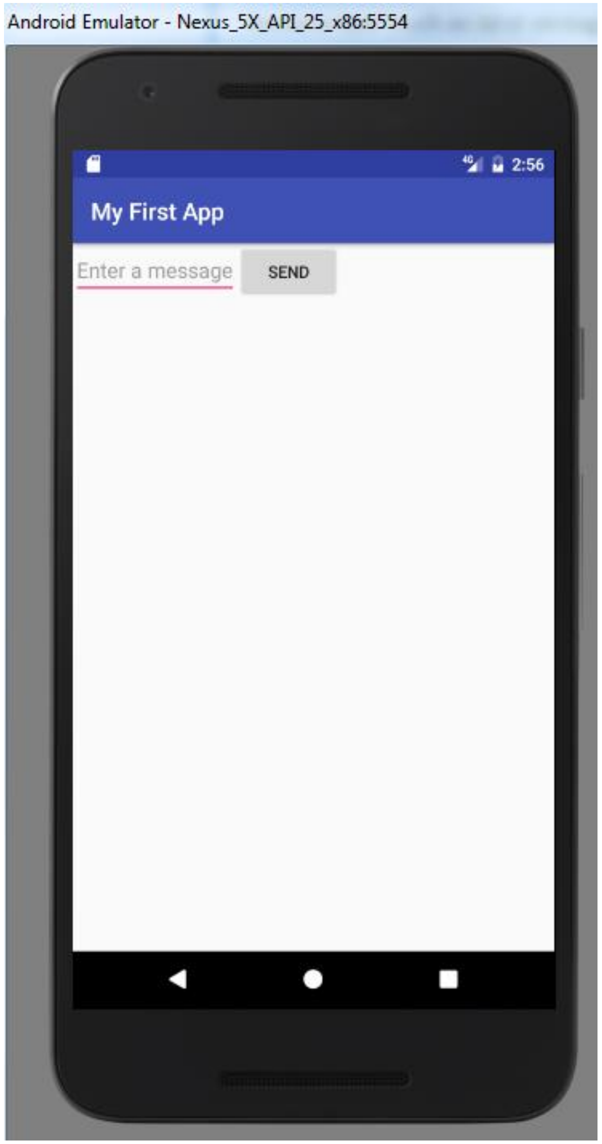
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
</resources>
```

Implement a Basic UI (5)

- Add a Button in activity_main.xml

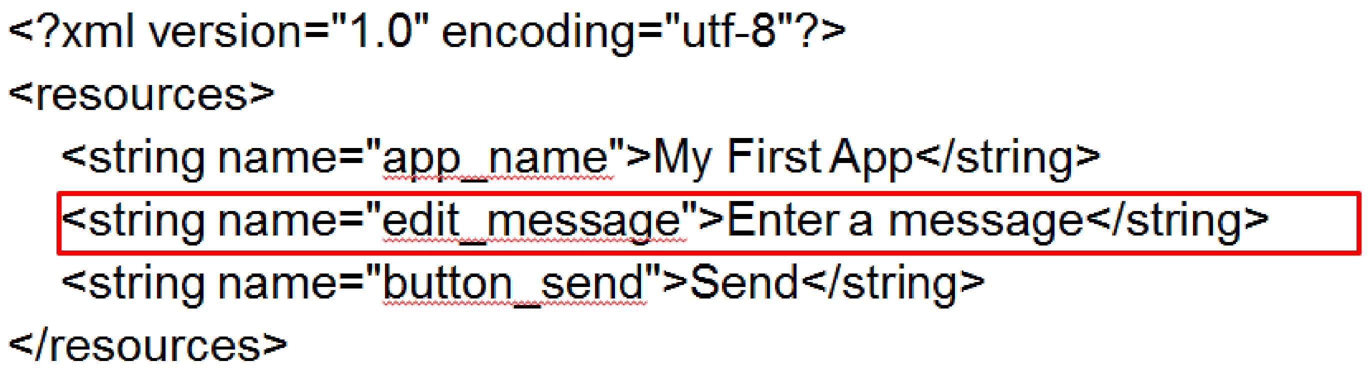
```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
        <EditText android:id="@+id/edit_message"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:hint="@string/edit_message" />
        <Button
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/button_send" />
</LinearLayout>
```

Run the app


Android Emulator - Nexus_5X_API_25_x86:5554
2:56
My First App
Enter a message
SEND

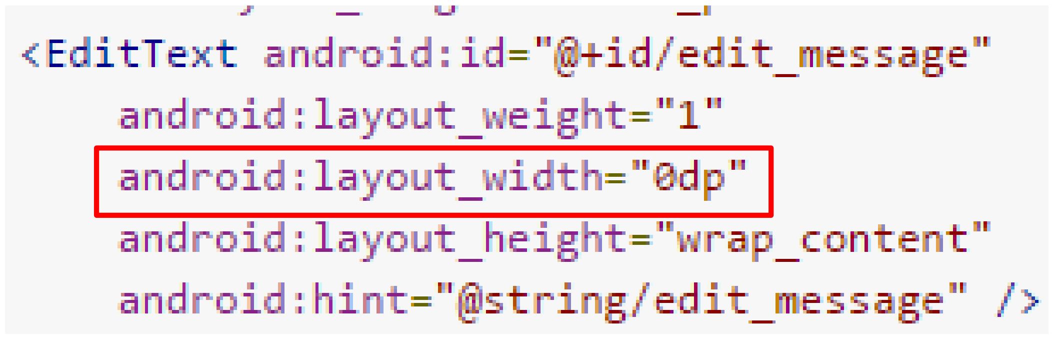
Play around UI

- Change the string field



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
</resources>
```

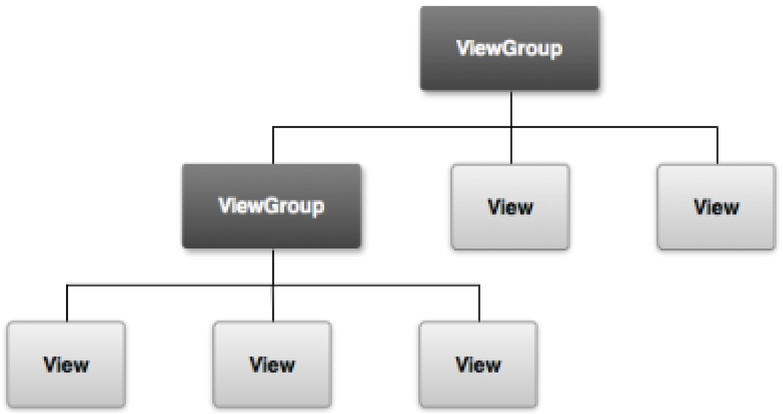
- Change width of text field



```
<EditText android:id="@+id/edit_message"
    android:layout_weight="1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message" />
```

Understand Basic UI Elements

- ViewGroup and View:
  - View is a basic element. ViewGroup is the organization of Views or ViewGroups
  - Text field, Button are views
  - LinearLayout defines a ViewGroup



Control the UI

- Goal: know how to push a button, generate a message, and generate a new screen.

Control the UI

- Add a field to the Button View

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Control the UI (2)

- Go to java > com.example.myfirstapp > MainActivity.java , insert the following code:

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

    /** Called when the user clicks the Send button */
    public void sendMessage(View view) {
        // Do something in response to button
    }
}
```

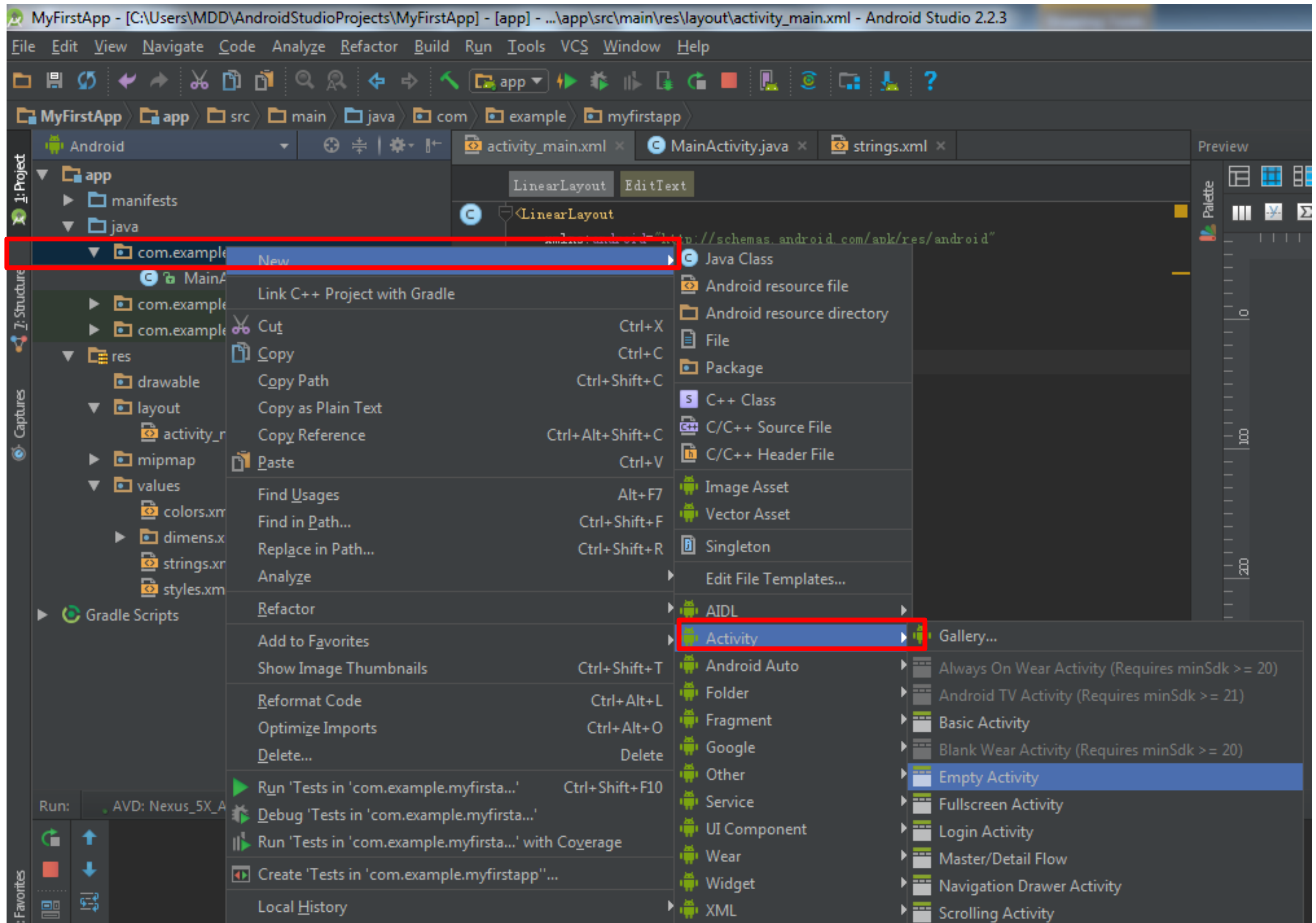
Control the UI (3)

- Create a Intent
  - Intent: an object that deliver message in run time between separate components, such as two activities.

```
public final static String EXTRA_MESSAGE =
"com.example.myfirstapp.MESSAGE";
public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.edit_message);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
```

Control the UI (3)

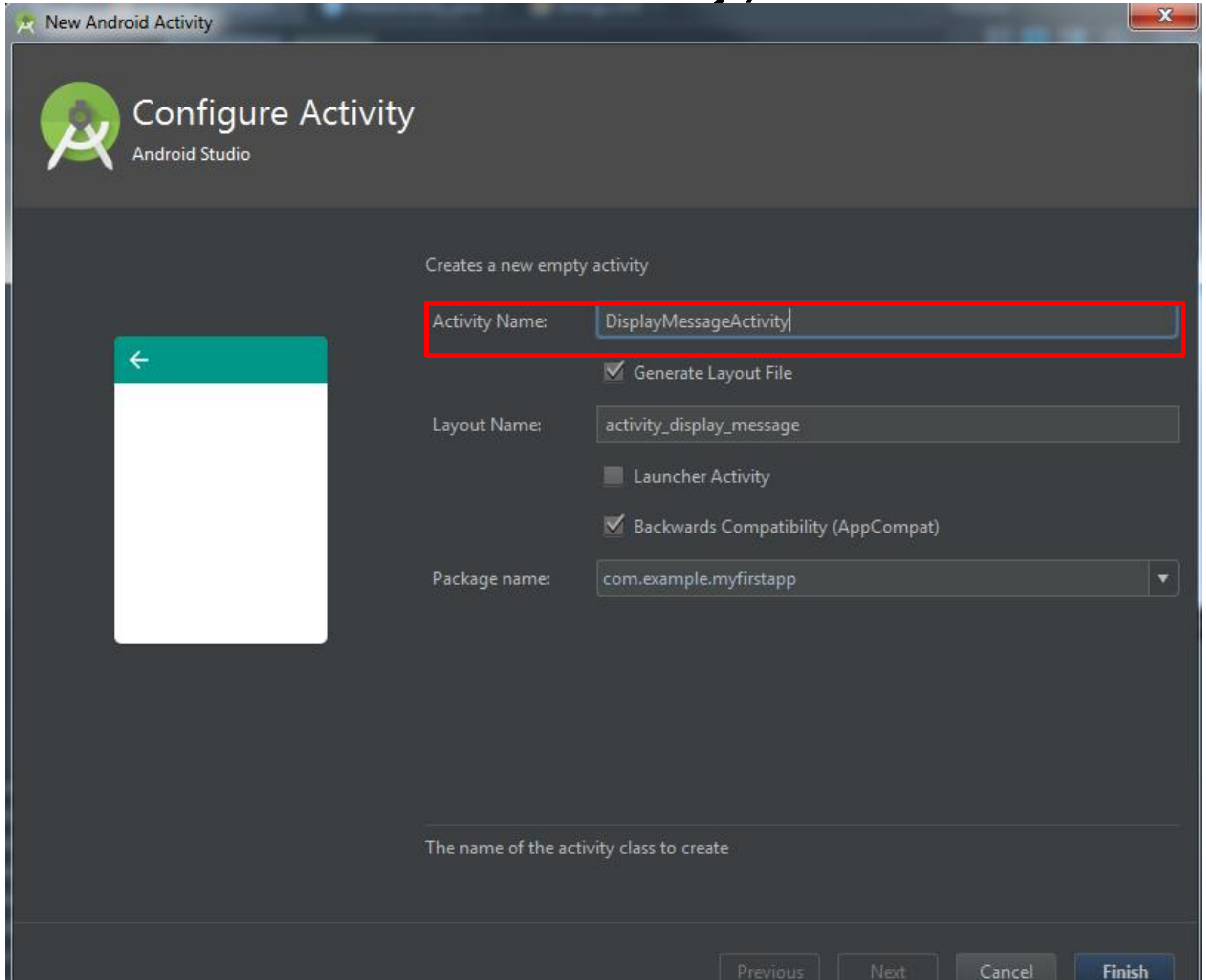
- Create a new Activity


MyFirstApp - [C:\Users\MDD\AndroidStudioProjects\MyFirstApp] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 2.2.3

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
```

Control the UI (3)

- Create a new Activity, Enter the name



Creates a new empty activity

Activity Name: DisplayMessageActivity

☑ Generate Layout File

Layout Name: activity_display_message

☐ Launcher Activity

☑ Backwards Compatibility (AppCompat)

Package name: com.example.myfirstapp

The name of the activity class to create

Control the UI (3)

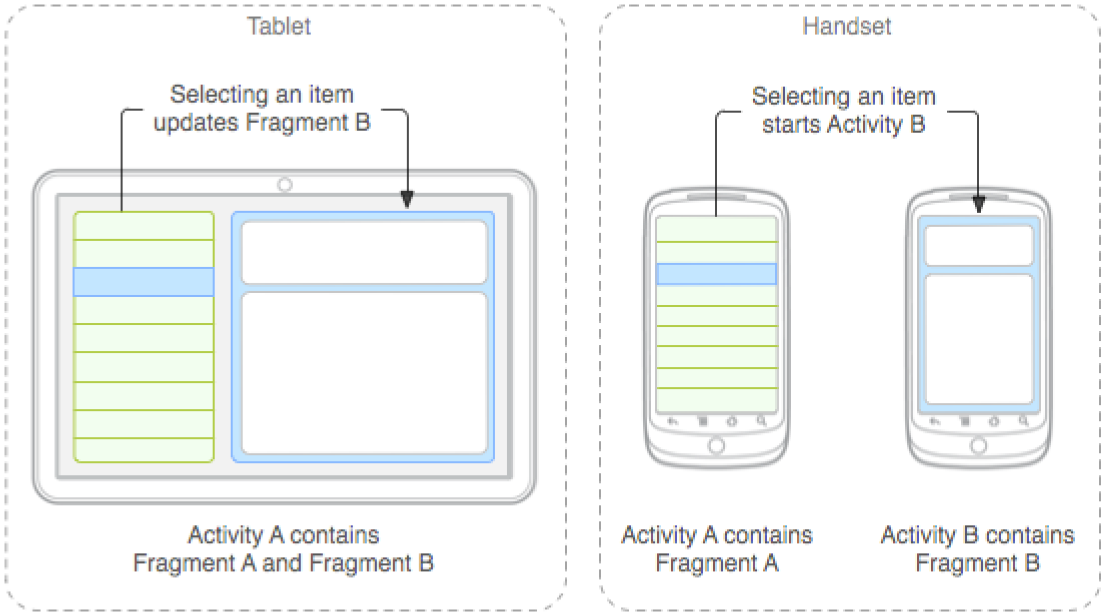
In DisplayMessageActivity.java, paste the following:

```
public class DisplayMessageActivity extends AppCompatActivity {

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

        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
        TextView textView = new TextView(this);
        textView.setTextSize(40);
        textView.setText(message);

        ViewGroup layout = (ViewGroup) findViewById(R.id.activity_display_message);
        layout.addView(textView);
    }
}
```

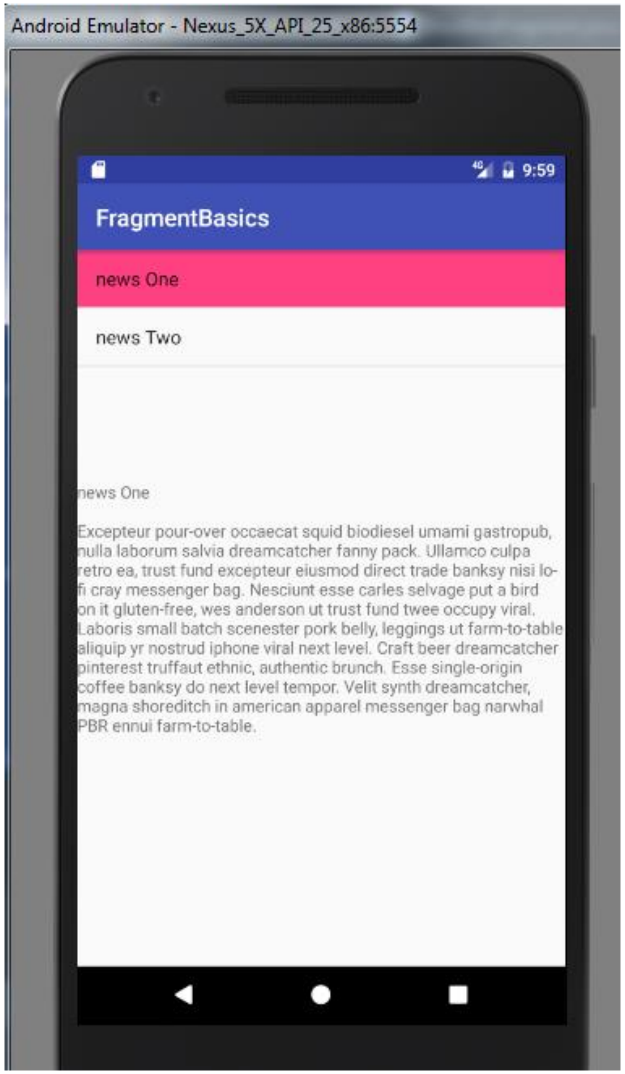
Run the App

Fragments

- A Fragment represents a portion of user interface in an Activity.
- can combine multiple fragments in a single activity, or reuse a fragment in multiple activities.
- Example?


Tablet
Selecting an item updates Fragment B
Activity A contains Fragment A and Fragment B
Handset
Selecting an item starts Activity B
Activity A contains Fragment A
Activity B contains Fragment B

Let’s do this



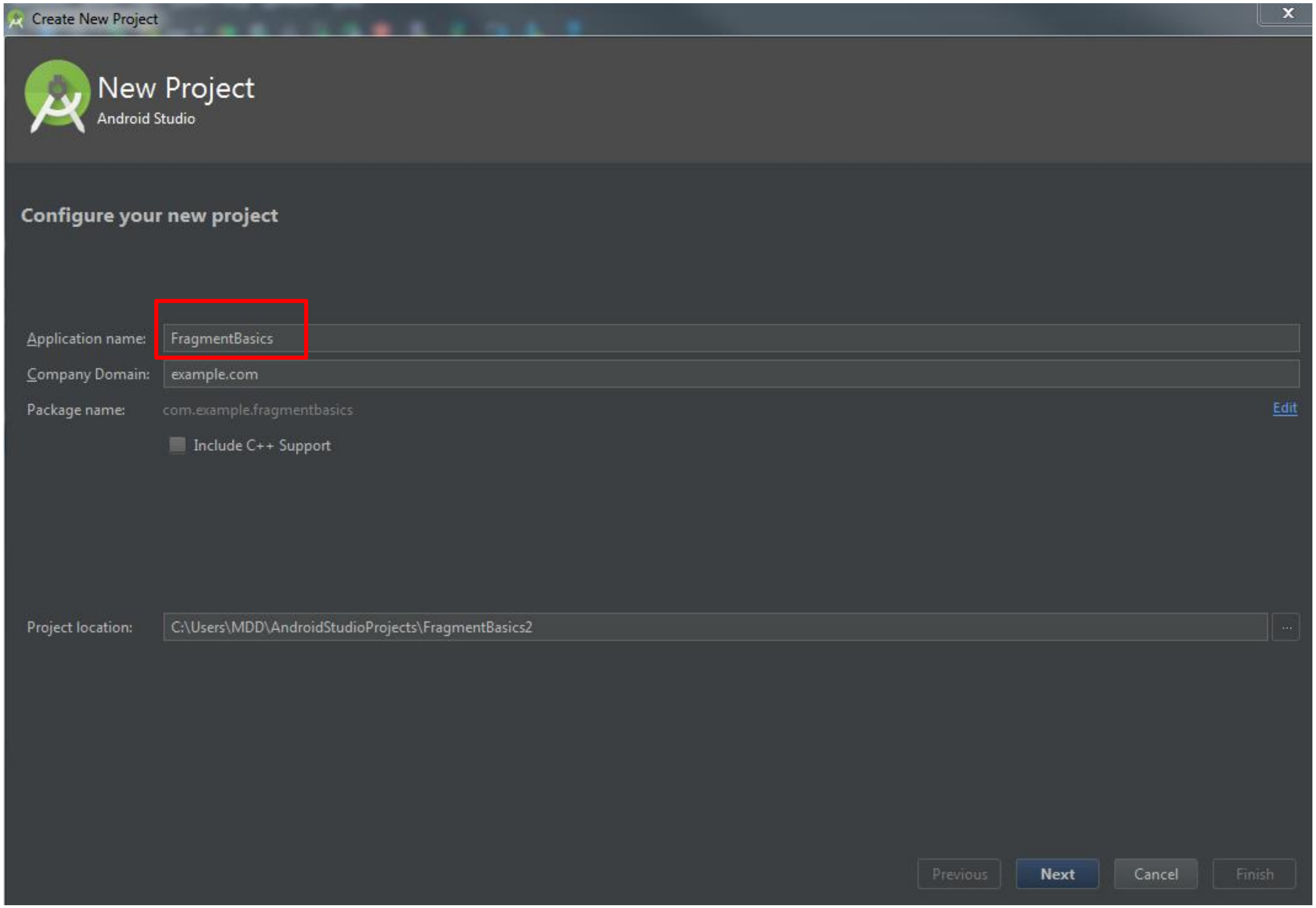
news One

Excepteur pour-over occaecat squid biodiesel umami gastropub, nulla laborum salvia dreamcatcher fanny pack. Ullamco culpa retro ea, trust fund excepteur eiusmod direct trade banksy nisi lo-fi cray messenger bag. Nesciunt esse carles selvage put a bird on it gluten-free, wes anderson ut trust fund twee occupy viral. Laboris small batch scenester pork belly, leggings ut farm-to-table aliquip yr nostrud iphone viral next level. Craft beer dreamcatcher pinterest truffaut ethnic, authentic brunch. Esse single-origin coffee banksy do next level tempor. Velit synth dreamcatcher, magna shoreditch in american apparel messenger bag narwhal PBR ennui farm-to-table.

Lab 1.b FragmentBasics

- Create two fragments. One is a list fragment that contains all news headlines, the other is a fragment that displays the news article
- Display the two fragments inside a single activity
- Reference: https://developer.android.com/training/basics/fragments/index.html

Build the first app with fragments



Application name: FragmentBasics

Company Domain: example.com

Package name: com.example.fragmentbasics

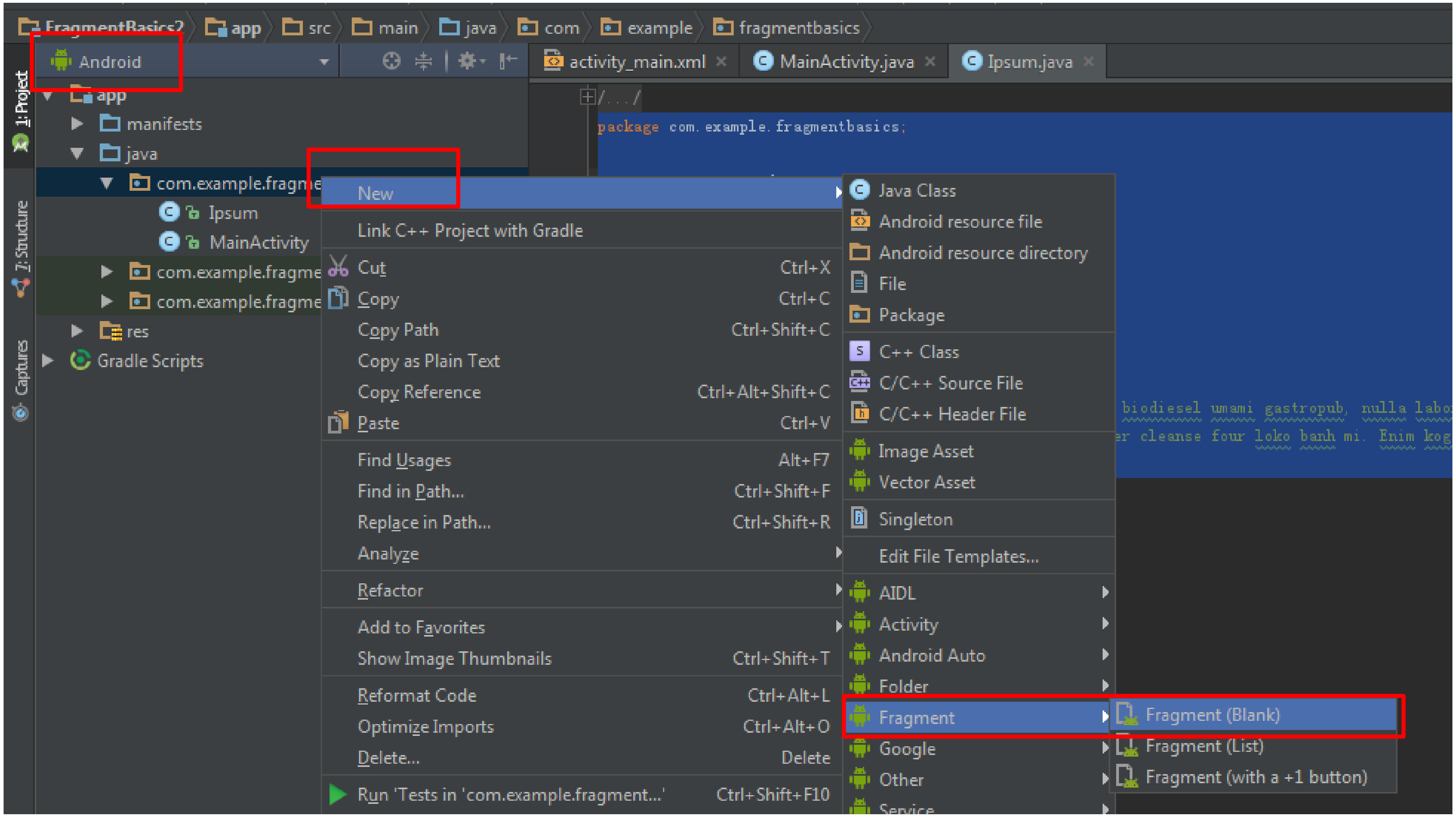
Include C++ Support

Project location: C:\Users\MDD\AndroidStudioProjects\FragmentBasics2

- Create Ipsum.java class, copy the following codes as the news:

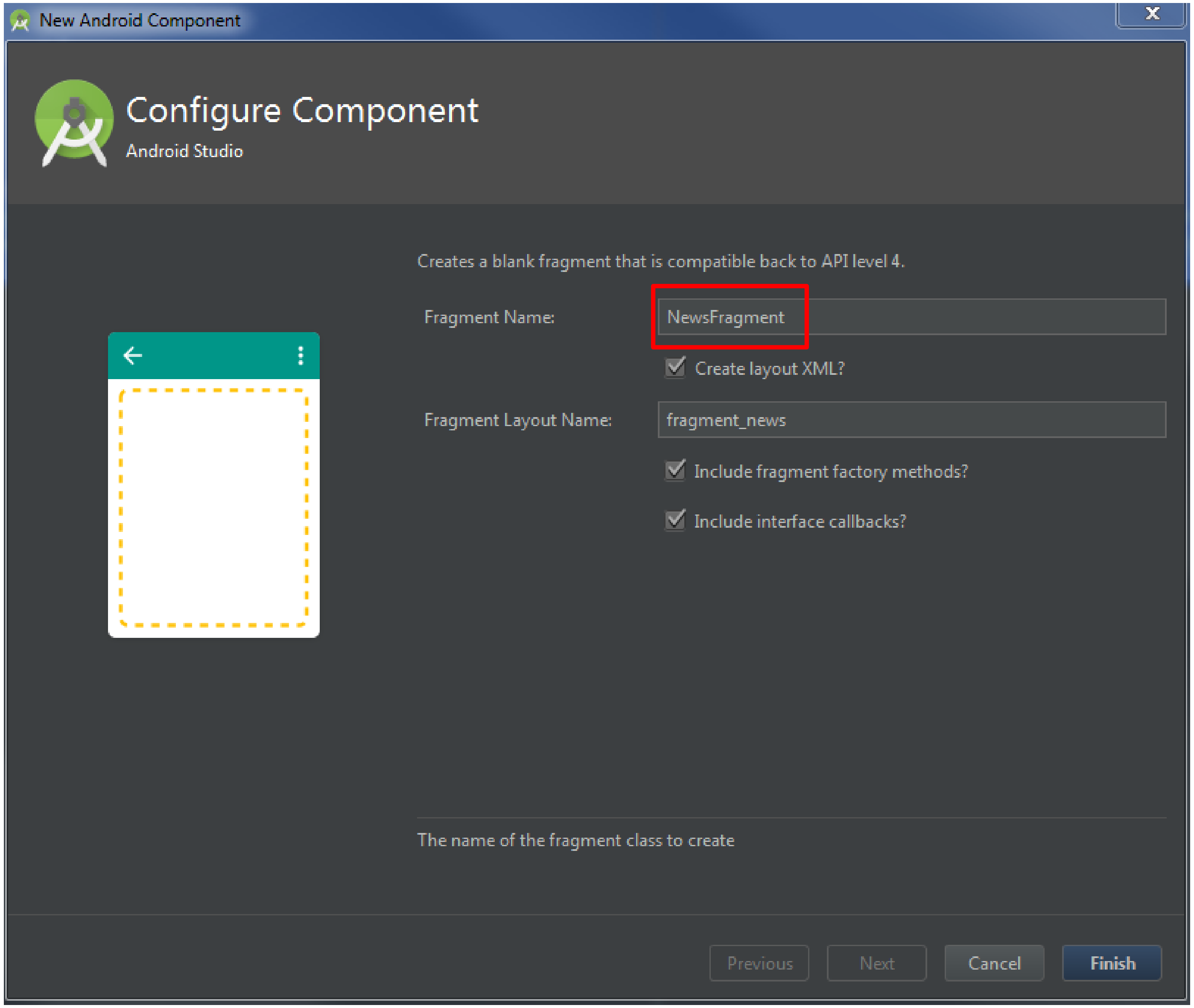
```
package com.example.fragmentbasics;

public class Ipsum {
    static String[] Headlines = {
        "Article One",
        "Article Two"
    };
    static String[] Articles = {
        "Article One\n\nExcepteur pour-over occaecat squid biodiesel umami gastropub, nulla laborum salvia dreamcatcher fanny pack. Ullamco culpa retro ea, trust fund excepteur eiusmod direct trade banksy nisi lo-fi cray messenger bag. Nesciunt esse carles selvage put a bird on it gluten-free, wes anderson ut trust fund twee occupy viral. Laboris small batch scenester pork belly, leggings ut farm-to-table aliquip yr nostrud iphone viral next level. Craft beer dreamcatcher pinterest truffaut ethnic, authentic brunch. Esse single-origin coffee banksy do next level tempor. Velit synth dreamcatcher, magna shoreditch in american apparel messenger bag narwhal PBR ennui farm-to-table.",
        "Article Two\n\nVinyl williamsburg non velit, master cleanse four loko banh mi. Enim kogi keytar trust fund pop-up portland gentrify. Non ea typewriter dolore deserunt Austin. Ad magna ethical kogi mixtape next level. Aliqua pork belly thundercats, ut pop-up tattooed dreamcatcher kogi accusamus photo booth irony portland. Semiotics brunch ut locavore irure, enim etsy laborum stumptown carles gentrify post-ironic cray. Butcher 3 wolf moon blog synth, vegan carles odd future."
    };
}
```

- Create a new fragment



```
package com.example.fragmentbasics;
```

- Configure the new fragment



New Android Component

Configure Component

Android Studio

Creates a blank fragment that is compatible back to API level 4.

Fragment Name: NewsFragment

☑ Create layout XML?

Fragment Layout Name: fragment_news

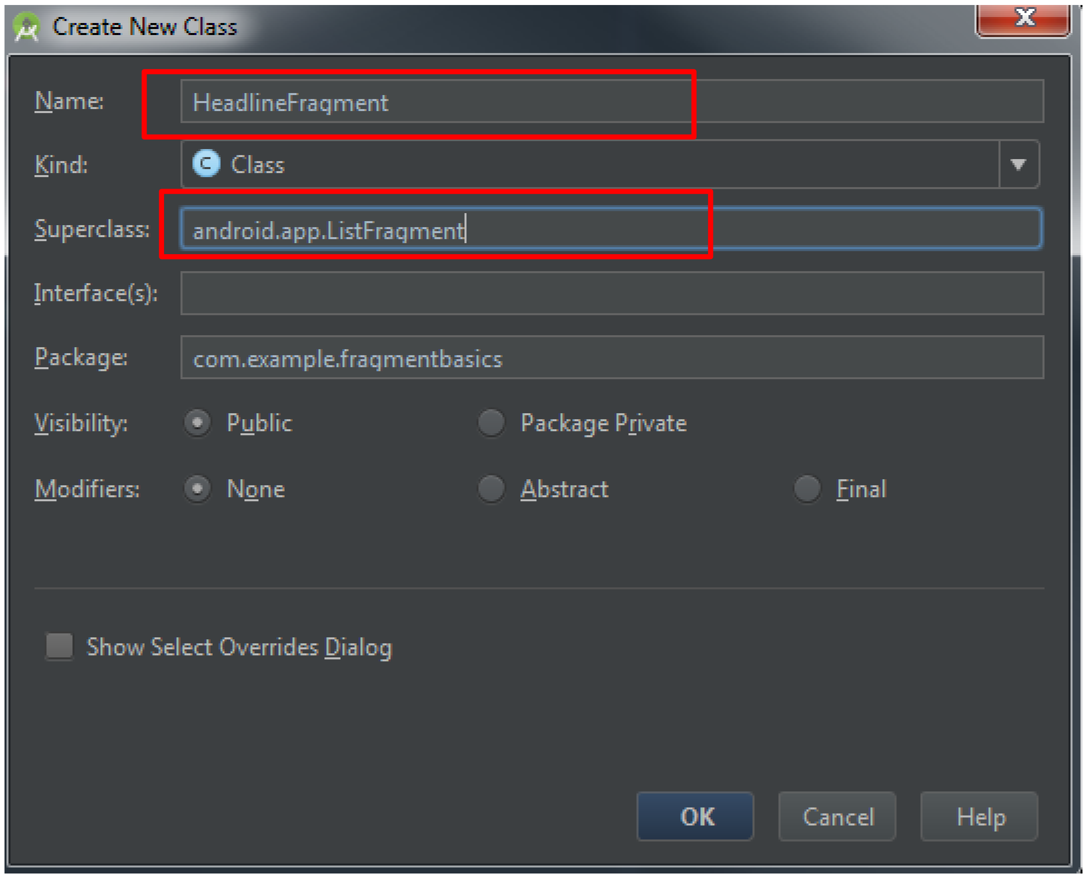
☑ Include fragment factory methods?

☑ Include interface callbacks?

The name of the fragment class to create

```
public class NewsFragment extends Fragment{
    final static String ARG_POSITION = "position";
    int mCurrentPosition = -1;


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

        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_news, container, false);
    }

    @Override
    public void onStart() {
        super.onStart();

        // During startup, check if there are arguments passed to the fragment.
        // onStart is a good place to do this because the layout has already been
        // applied to the fragment at this point so we can safely call the method
        // below that sets the article text.
        Bundle args = getArguments();
        if (args != null) {
            // Set article based on argument passed in
            updateArticleView(args.getInt(ARG_POSITION));
        } else if (mCurrentPosition != -1) {
            // Set article based on saved instance state defined during onCreateView
            updateArticleView(mCurrentPosition);
        }
    }

    public void updateArticleView(int position) {
        TextView article = (TextView) getActivity().findViewById(R.id.news);
        article.setText(Ipsum.Articles[position]);
        mCurrentPosition = position;
    }


    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

        // Save the current article selection in case we need to recreate the fragment
        outState.putInt(ARG_POSITION, mCurrentPosition);
    }
}
```

- Create a second list fragment: HeadlineFragment.java



Create New Class

Name: HeadlineFragment

Kind: Class

Superclass: android.app.ListFragment

Interface(s):

Package: com.example.fragmentbasics

Visibility: [✓ Public] Package Private

Modifiers: [✓ None] Abstract Final

Show Select Overrides Dialog

- Define an Interface in HeadlineFragment

```
public class HeadlineFragment extends ListFragment {
    OnHeadlineSelectedListener mCallback;
    // Container Activity must implement this interface
    public interface OnHeadlineSelectedListener {
        public void onArticleSelected(int position);
    }
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);

        // This makes sure that the container activity has implemented
        // the callback interface. If not, it throws an exception
        try {
            mCallback = (OnHeadlineSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                    + " must implement OnHeadlineSelectedListener");
        }
    }

    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        // Notify the parent activity of selected item
        mCallback.onArticleSelected(position);

        // Set the item as checked to be highlighted when in two-pane layout
        getListView().setItemChecked(position, true);
    }
}
```

- Implement onCreate and onStart for HeadlineFragement class

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    int layout = android.R.layout.simple_list_item_activated_1;
    setListAdapter(new ArrayAdapter<String>(getActivity(), layout, Ipsum.Headlines));
}

@Override
public void onStart() {
    super.onStart();

    if (getFragmentManager().findFragmentById(R.id.news_fragment) != null) {
        getListView().setChoiceMode(ListView.CHOICE_MODE_SINGLE);
    }
}
```

- In the MainActivity.java, implement the interface for the list fragment

```
public class MainActivity extends AppCompatActivity implements
HeadlineFragment.OnHeadlineSelectedListener{

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

    public void onArticleSelected(int position) {

        NewsFragment newsFragment = (NewsFragment)
getSupportFragmentManager().findFragmentById(R.id.news_fragment)
;
        newsFragment.updateArticleView(position);

    }
}
```

- Organize these two fragments in activity_main.xml

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

<fragment android:name="com.example.fragmentbasics.HeadlineFragment"
    android:id="@+id/headlines_fragment"
    android:layout_weight="1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<fragment android:name="com.example.fragmentbasics.NewsFragment"
    android:id="@+id/news_fragment"
    android:layout_weight="2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

</LinearLayout>
```

Run the app

- Play around different layouts
  – Table Layout
  – RelativeLayout
  – FrameLayout
- Run the app in the phone

Assignment

- Finish both lab 1.a and 1.b. Demonstrate them during lab 2.
- Extra credits: change of phone orientations. Build an app with two fragments (a headline fragment and a news fragment). The app should display only a single fragment when the phone is held in portray position, and display both fragments at the same time when the phone is held in landscape position.

Lab 1.a Basic UI

- The app has a interface that contains a button and a text input field.
- Once the button is pushed, generate an intent that invokes a second activity
- The second activity will display the input string
- Reference: https://developer.android.com/training/basics/firstapp/creating-project.html

Lab 1.b FragmentBasics

- Create two fragments. One is a list fragment that contains all news headlines, the other is a fragment that displays the news article
- Display the two fragments inside a single activity
- Reference: https://developer.android.com/training/basics/fragments/index.html