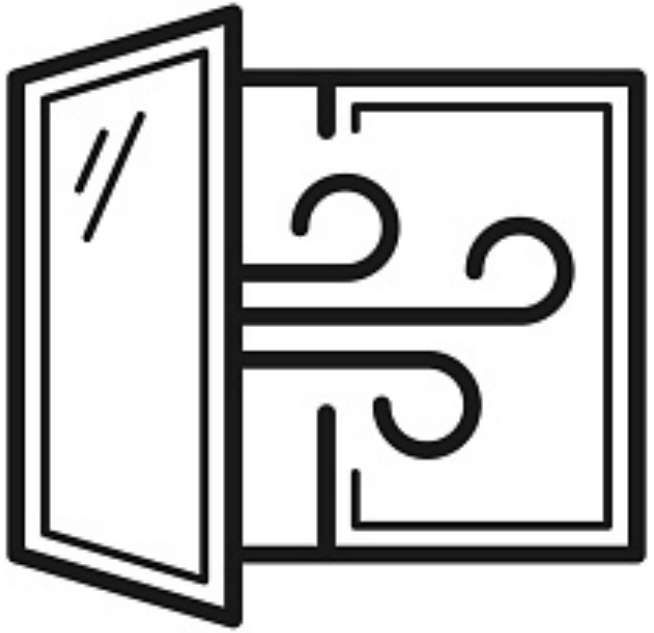
Android App Programming

Lecture 8: UX Programming 2
Fragments and Navigation
Dec 14th, 2021

Anastasia Bezerianos
Université Paris-Saclay

in class



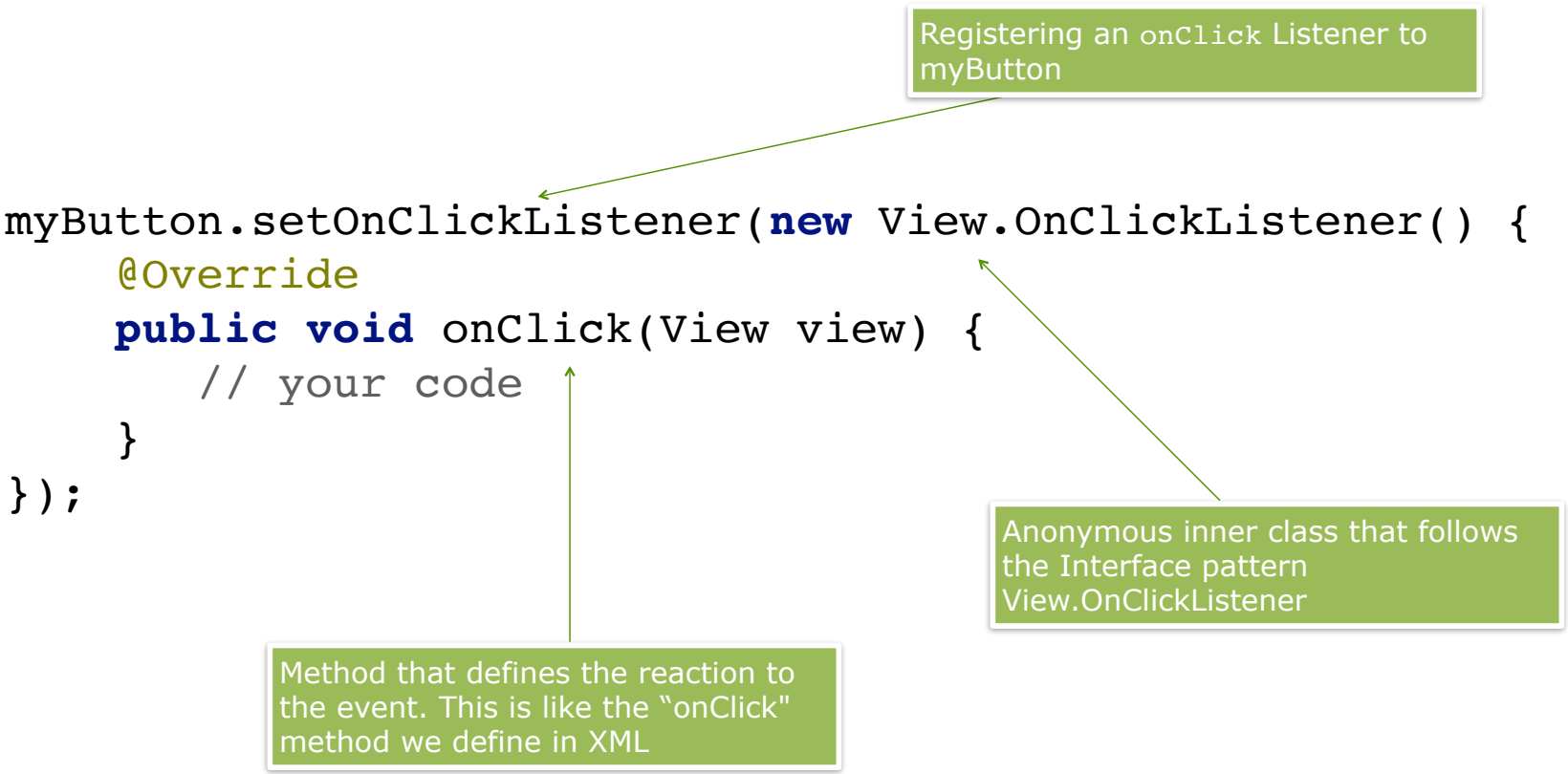
Reminder:
next session (04/01/2022)
TD noté

TD noté 2

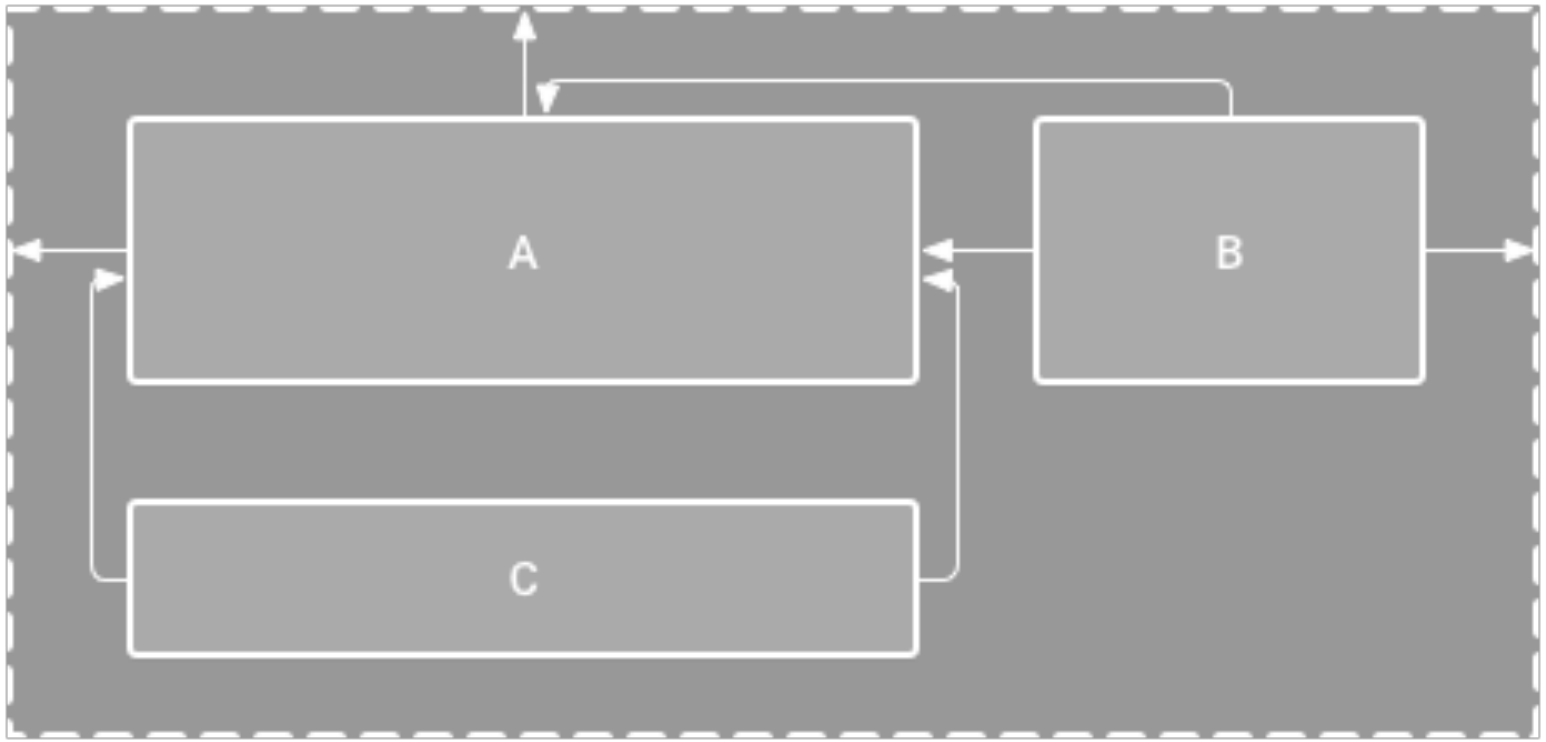
- TD noté on 04/01/2022
- You will work alone (cannot discuss with others)
- Can use any material from class (transparencies, lab solutions, videos), the book, or the internet
- BUT the final solution should be yours

Last lecture recap

Adding a listener



Constraint Layout



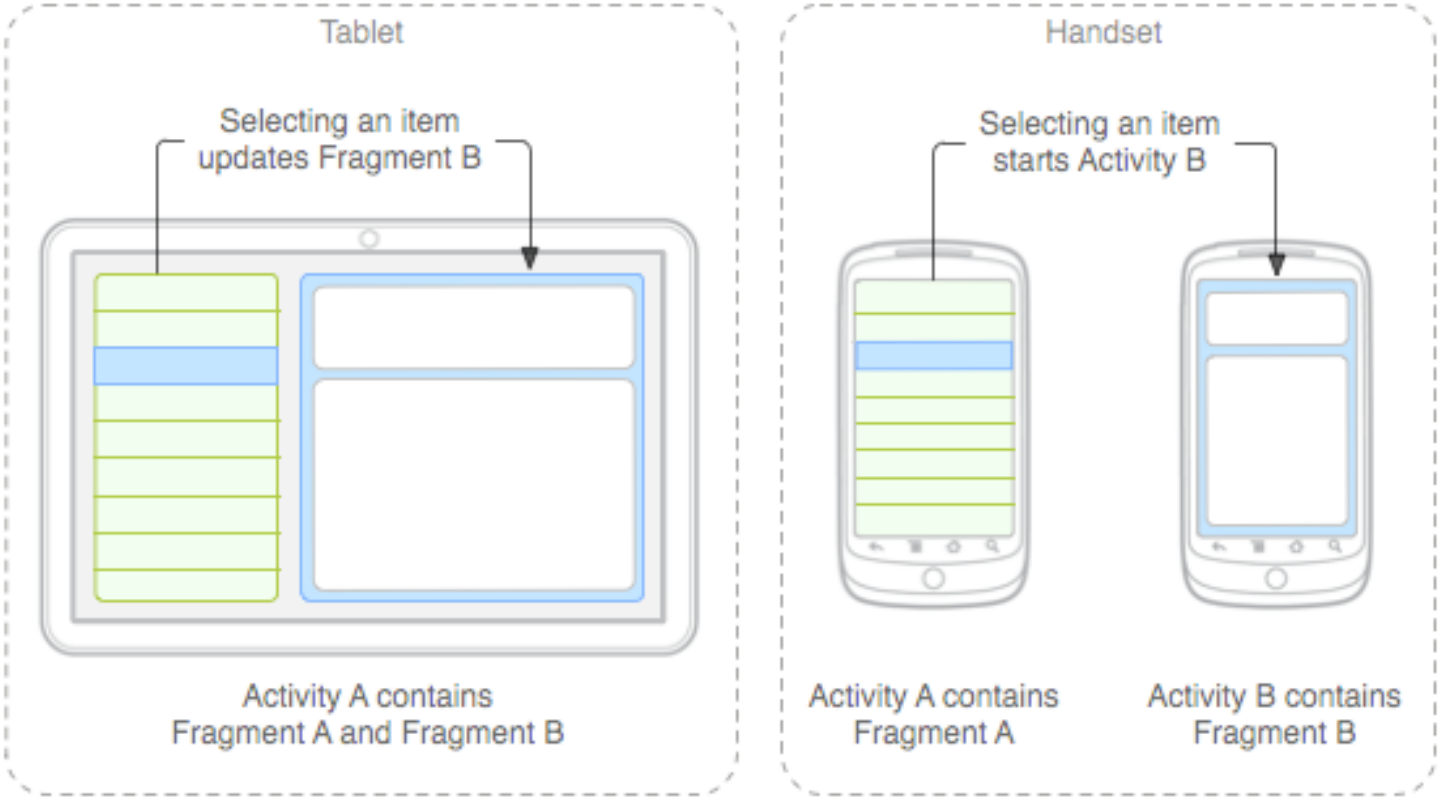
UX Programming 2

Fragments and Navigation

Fragments

- modular user interface units
- sub-activities with a lifecycle, class and layout
- embedded into one or more activities
- added during design and reused by activities
- added at runtime to create dynamic interfaces

Fragments


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

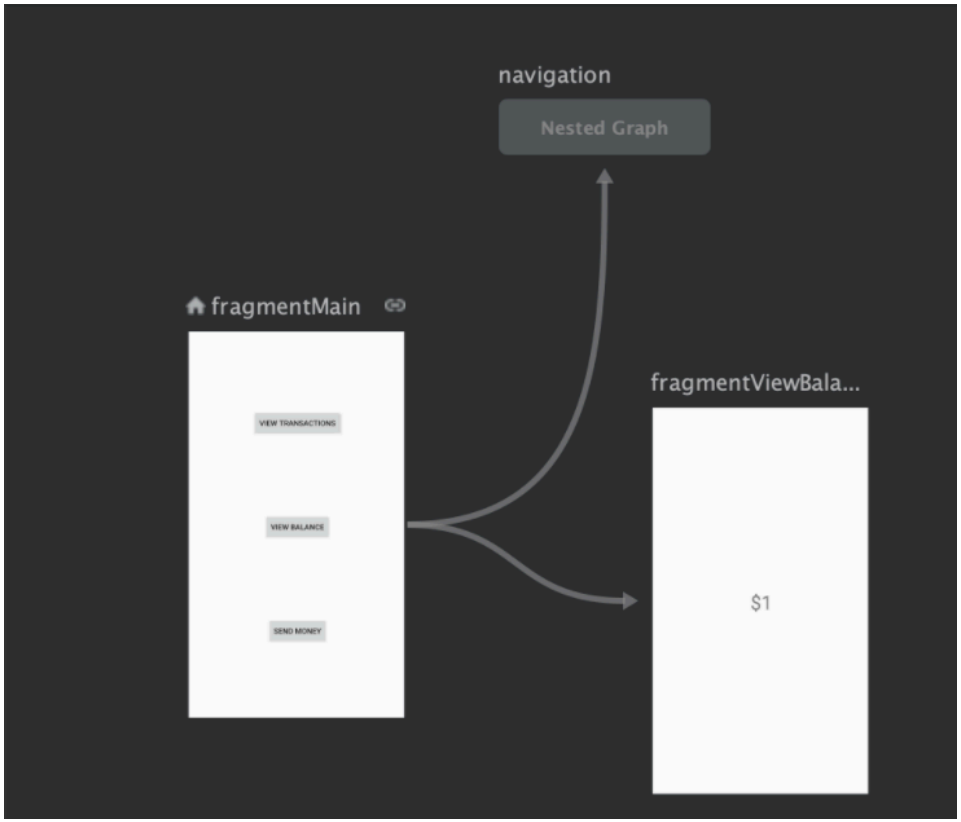
Fragments

- extend the class `FragmentActivity`
- have `onCreateView()` method that returns a View
- In an Activity you can add & manage Fragments with the
  - `FragmentManager` and
  - `FragmentTransaction` classes

Fragments

- Chapter 30 of the book has an example on Fragments (optional class in the Lab)

Navigation


navigation
Nested Graph
fragmentMain
VIEW TRANSACTIONS
VIEW BALANCE
SEND MONEY
fragmentViewBala...
$1

Navigation

- Navigation Architecture component comes with Android
- Assumes a single Activity with many Fragments (each activity has each own navigation graph)
  - Navigation graph: XML info on destinations & paths
  - NavHost: empty container that displays a destinations (usually the class `NavHostFragment`)
  - NavController: object that manages the transitions

Navigation

- Navigation editor to create / edit the navigation graph
- Each NavHostFragment class has its own NavController, accessible with `findNavController()`
- Can use Bundles to communicate information across Fragments
- Need to add dependencies into build.gradle files

Lab 8

- Exercise 1 & 2: Navigation & Communication

build.gradle Project

```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath “com.android.tools.build:gradle:7.0.2"

        def nav_version = "2.3.5"
        classpath "androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version"
   }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Adding global navigation support.
Don’t forget to Sync

build.gradle Module

```
plugins {
    id 'com.android.application'
    id 'androidx.navigation.safeargs'
}

android {
    compileSdk 30

    defaultConfig {
        applicationId "fr.ups.lab8_simplenavigation"
        minSdk 21
        targetSdk 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.0'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.2'

    implementation 'androidx.navigation:navigation-fragment:2.3.5'
    implementation 'androidx.navigation:navigation-ui:2.3.5'

    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

Needed to pass data across fragments through the navigation

Navigation libraries.
Don't forget to Sync

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView2"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/navigation_graph"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Tells the activity this Fragment is a NavigationHostFragment that will be showing different fragments as the user navigates

Tells the activity there is a NavHost

and the name of the navigation graph (another xml file)

MainActivity.java

```
ckage fr.ups.lab8_simplenavigation;

import androidx.appcompat.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);
    }
}
```

navigation_graph.xml

```
<?xml version="1.0" encoding="utf-8"?>
<navigation 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:id="@+id/navigation_graph"
    app:startDestination="@id/firstFragment">

    <fragment
        android:id="@+id/firstFragment"
        android:name="fr.ups.lab8_simplenavigation.ToolbarFragment"
        android:label="fragment_first"
        tools:layout="@layout/fragment_toolbar" >
        <action
            android:id="@+id/action_toolbarFragment_to_textFragment"
            app:destination="@id/secondFragment" />
    </fragment>
    <fragment
        android:id="@+id/secondFragment"
        android:name="fr.ups.lab8_simplenavigation.TextFragment"
        android:label="fragment_second"
        tools:layout="@layout/fragment_text" />
</navigation>
```

starting destination Fragment

action that connects the FirstFragment to the Second (one way)

fragment_toolbar.xml

```
<?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">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/seekBar1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="17dp"
        android:text="@string/change_text" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:inputType="text"
        android:minHeight="48dp"
        tools:ignore="SpeakableTextPresentCheck">

        <requestFocus />
    </EditText>
    <SeekBar
        android:id="@+id/seekBar1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/editText1"
        android:layout_marginTop="14dp"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

ToolbarFragment.java

```java
package fr.ups.lab8_simplenavigation;

import android.os.Bundle;
import androidx.fragment.app.Fragment;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.fragment.NavHostFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.SeekBar;
import android.widget.TextView;

public class ToolbarFragment extends Fragment implements SeekBar.OnSeekBarChangeListener {

    private static int seekvalue = 10;
    private static EditText edittext;

    public ToolbarFragment() {
        // Required empty public constructor
    }

    public static ToolbarFragment newInstance(String param1, String param2) {
        ToolbarFragment fragment = new ToolbarFragment();
        Bundle args = new Bundle();
        fragment.setArguments(args);
        return fragment;
    }

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

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_toolbar, container, false);

        edittext = (EditText) view.findViewById(R.id.editText1);
        final SeekBar seekbar = (SeekBar) view.findViewById(R.id.seekBar1);
        seekbar.setOnSeekBarChangeListener(this);
        final Button button = (Button) view.findViewById(R.id.button1);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                buttonClicked(v);
            }
        });
       return view;
    }

    public void buttonClicked (View view) {
        Bundle bundle = new Bundle();
        bundle.putString("resizetext", edittext.getText().toString());
        bundle.putInt("resizevalue", seekvalue);
        Navigation.findNavController(view).navigate(R.id.action_toolbarFragment_to_textFragment, bundle);
    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
        seekvalue = i;
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
    }
}
```

Class extends Fragment but also implements an OnSeekBarListener.
This forces the class to create the methods that handle events on the SeekBar: onProgressChanged, on StartTrackingTouch, onStopTrackingTouch

Automatically created method

"Inflating" a view means taking the layout XML and parsing it to create the view and viewgroup Java objects.

Tell the seeker bar that the code for the OnSeekBarChange events are in "this" (the object of this class)

NEW. Used this listener implementation to
(i) create a Bundle to pass the info from EditText and Seekbar; and
(ii) Use the Navigation class to go to the Text Fragment, using the action

fragment_text.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/fragment_two"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

TextFragment.java

```
package fr.ups.lab8_simplenavigation;

import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link SecondFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class TextFragment extends Fragment {

    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String resizetext = "resizetext";
    private static final String resizevalue = "resizevalue";

    private String TextEdit_text;
    private int TextEdit_fontsize;

    private static TextView textview;

    public TextFragment() {
        // Required empty public constructor
    }


    public static TextFragment newInstance(String param1, int param2) {
        TextFragment fragment = new TextFragment();
        Bundle args = new Bundle();
        args.putString(resizetext, param1);
        args.putInt(resizevalue, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            TextEdit_text = getArguments().getString(resizetext);
            TextEdit_fontsize = getArguments().getInt(resizevalue);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment

        View view = inflater.inflate(R.layout.fragment_text, container, false);
        textview = (TextView) view.findViewById(R.id.textView1);

        changeTextProperties(TextEdit_fontsize, TextEdit_text );

        return view;
    }

    public void changeTextProperties(int fontsize, String text)
    {
        textview.setTextSize(fontsize);
        textview.setText(text);
    }
}
```

Will use these names to access the msg from the Bundle

I will store their values here

Automatically created method to read the Bundle, not needed

Automatically created method, I am retrieving info from Bundle

"Inflating" a view means taking the layout XML and parsing it to create the view and viewgroup Java objects.

A public method to change the size of the font

Bindings

(*aside note*, since Android Studio 3.6)

Bindings

- So far we accessed a widget (eg *myButtonFoo)* from Activities using `findViewById(R.id.myButtonFoo)`
- Since Android Studio 3.6 an alternative way (optional) is using a View Binding
- View Bindings are classes which are automatically generated by Android Studio for each XML layout file

Bindings

- Both Bindings and `findViewById()` are valid choices.
- Bindings can help avoid errors BUT they require you to change your compilation options
- Edit *build.gradle* (Module: app) file:

```
android {
    buildFeatures {
       viewBinding = true
    }
```

Bindings

```
public class MainActivity extends AppCompatActivity {

  private ActivityMainBinding binding;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    binding = ActivityMainBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();
    setContentView(view);
}

// Now we have a reference to the binding and can access the views by name
public void convertCurrency(View view) {

     EditText dollarText = findViewById(R.id.dollarText);
     TextView textView = findViewById(R.id.textView);

    if (!binding.dollarText.getText().toString().equals("")) {
        …
    }
}
```

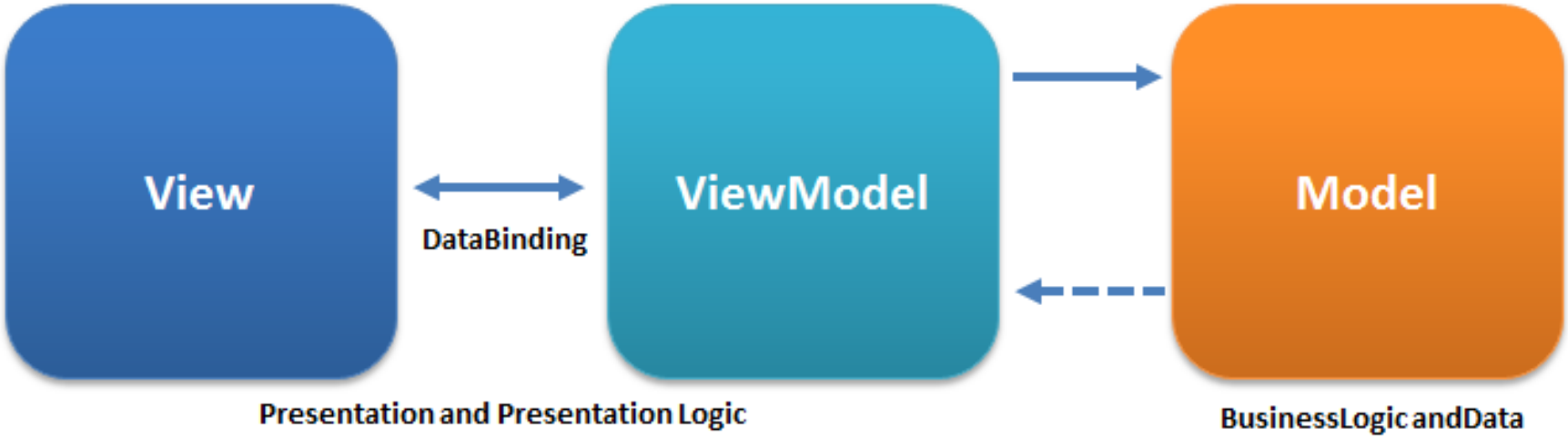
Bindings

- Many of the Project examples use Bindings
- For example Basic Activity, Bottom Navigation, etc.

More under https://developer.android.com/topic/libraries/view-binding

MVVM

- MVVM code organization
- stands for Model, View, ViewModel



https://www.journaldev.com/20292/android-mvvm-design-pattern

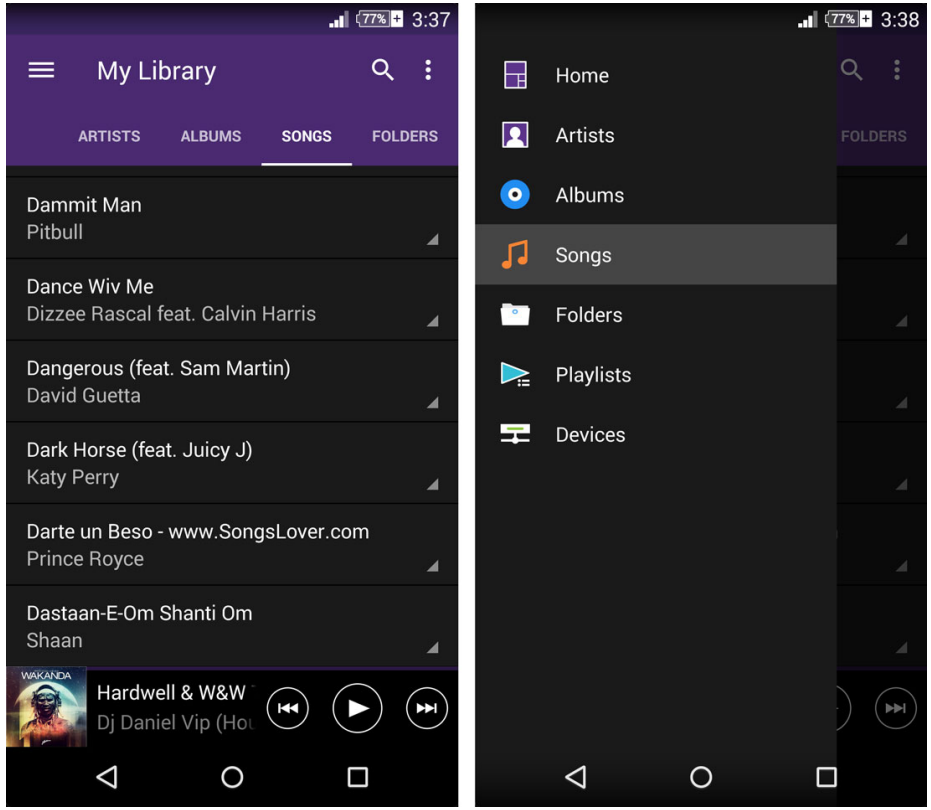
MVVM

- MVVM stands for Model, View, ViewModel.
- Model: Data of the application. It cannot directly talk to the View. Generally, we try to pass the data to the ViewModel through *Observables*.
- View: It represents the UI of the application devoid of any Application Logic. It *observes* the ViewModel.
- ViewModel: It acts as a link between the Model and the View. It's responsible for transforming the data from the Model. It provides data streams to the View. It also uses hooks or callbacks to update the View. It'll ask for the data from the Model.

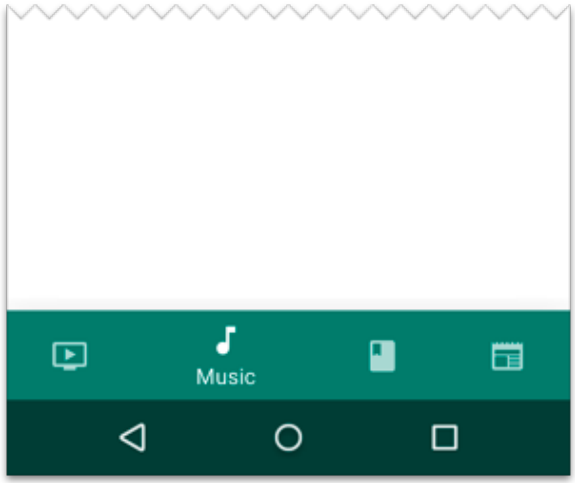
Navigation + Fragments

- Navigation, Fragments and Binding can be combined to create more complex apps
- Example projects types like: Tabbed Activity, Bottom Navigation, Navigation Drawer, …

Navigation + Fragments

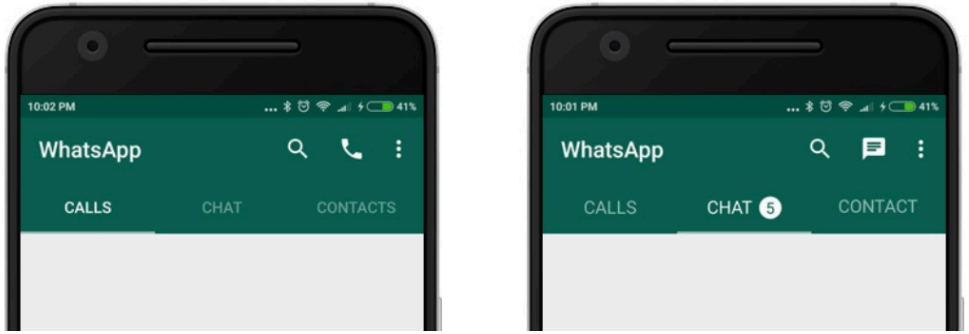


Navigation Drawer

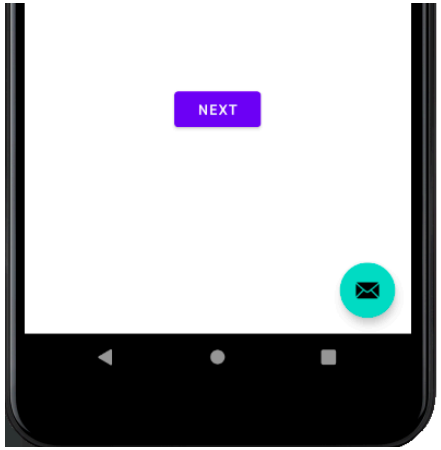


Bottom Navigation

Navigation + Fragments



Tabbed Activity



Basic Activity

Lab 8

- Exercise 3: Use existing Project types
- Optional use them to revisit Lab7-Ex3

Lab 8

- Exercise Optional: Fragments
  - Follow the tutorial from the book on Chapter 30 (your reading of Chapter 29 that contains the basics)
  - !!!! *See on the webpage for notes about the book tutorial that you need to adjust*

Fragments

- Start a manager and assign a transaction

```
FragmentManager fM = getSupportFragmentManager();
FragmentTransaction transaction = frM.beginTransaction();
```

- A transaction can be used to:
  - add a Fragment, replace it, remove it, addToBackStack (for when the user goes back in navigation)
- at the end commit the transaction `transaction.commit()`

Fragments

- Fragments can communicate via the activity they belong to

Fragments

- Communication: Activity => Fragment
- done via the Fragment ID
- `getSupportFragmentManager().findFragmentById(…)`

Fragments

- Communication: Fragment => Activity requires:
  - Fragment: Override the onAttach() method and store a link to the Activity that attached (used) the Fragment
  - Fragment: define a listener interface (pattern)
  - Activity: implement a listener interface of the type defined by the Fragment

Lab 8 - optional

FragmentActivity.java

my package name (also directory in my disk)

This class now extends FragmentActivity
BUT
It also implements a listener ToolbarListener that I have created inside my class ToolbarFragment.

This forces my class FragmentExampleActivity to provide the method `onButtonClick`

This Listener that waits for events is how the ToolBarFragment will send information to the Activity.

```
package fr.ups.lab8_ex1;

import androidx.fragment.app.FragmentActivity;
import android.os.Bundle;

public class FragmentExampleActivity extends FragmentActivity
        implements ToolbarFragment.ToolbarListener {

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

    @Override
    public void onButtonClick(int fontsize, String text)
        TextFragment textFragment = (TextFragment)
                getSupportFragmentManager().findFragmentById(R.id.fragmentContainerView2);

        textFragment.changeTextProperties(fontsize, text);
    }
}
```

This method will be activated by the Fragment ToolbarFragment

Access the method changeTextProperties of my TextFragment

This is how the Activity sends information to the TextFragment.

We access the activity's FrameManager (through the method getSupportFrameManager) and ask it to find a specific frame (the one contained inside my fragmentContainerView2)

activity_fragment_example.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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=".FragmentExampleActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView"
        android:name="fr.ups.lab8_ex1.ToolbarFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/fragmentContainerView2"
        android:name="fr.ups.lab8_ex1.TextFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/fragmentContainerView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

A FrameContainerView, I have added here my ToolbarFragment.
The layout is the middle of the Activity

A FrameContainerView, I have added here my TextFragment.
The layout is the middle of the Activity horizontally, and vertically between the previous container and the bottom

ToolbarFragment.java

```
package fr.ups.lab8_ex1;

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.SeekBar;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class ToolbarFragment extends Fragment implements SeekBar.OnSeekBarChangeListener {

    private static int seekvalue = 10;
    private static EditText edittext;

    ToolbarListener activityCallback;

    public interface ToolbarListener {
        public void onButtonClick(int position, String text);
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        try {
            activityCallback = (ToolbarListener) context;
        } catch (ClassCastException e) {
            throw new ClassCastException(context.toString()
                    + " must implement ToolbarListener");
        }
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view =  inflater.inflate(R.layout.toolbar_fragment,
                container, false);

        edittext = (EditText) view.findViewById(R.id.editText1);

        final SeekBar seekbar = (SeekBar) view.findViewById(R.id.seekBar1);
        seekbar.setOnSeekBarChangeListener(this);

        final Button button = (Button) view.findViewById(R.id.button1);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                buttonClicked(v);
            }
        });

        return view;
    }

    public void buttonClicked (View view) {
        activityCallback.onButtonClick(seekvalue,
                edittext.getText().toString());
    }

    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
        seekvalue = i;
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) { }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {}
}
```

Class extends Fragment but also implements an OnSeekBarListener.
This forces the class to create the methods that handle events on the SeekBar: onProgressChanged, on StartTrackingTouch, onStopTrackingTouch

This is how this Fragment will communicate with the Activity that contains it.
We override the method onAttach to store a link to the context (ie the Activity that called the onAttach)

"Inflating" a view means taking the layout XML and parsing it to create the view and viewgroup Java objects.

Tell the seeker bar that the code for the OnSeekBarChange events are in "this" (the object of this class)

Send a msg to the Activity that contains this Fragment.
We use the stored link to the Activity to call the Activity's method onButtonClick that expects two parameters, the seekvalue and the text

toolbar_fragment.xml

```
<?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">

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/seekBar1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="17dp"
        android:text="@string/change_text" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:inputType="text"
        android:minHeight="48dp"
        tools:ignore="SpeakableTextPresentCheck">

        <requestFocus />
    </EditText>
    <SeekBar
        android:id="@+id/seekBar1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/editText1"
        android:layout_marginTop="14dp"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

TextFragment.java

```
package fr.ups.lab8_ex1;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.SeekBar;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

public class TextFragment extends Fragment  {

    private static TextView textview;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.text_fragment,
                container, false);

        textview = (TextView) view.findViewById(R.id.textView1);

        return view;
    }

    public void changeTextProperties(int fontsize, String text)
    {
        textview.setTextSize(fontsize);
        textview.setText(text);
    }
}
```

"Inflating" a view means taking the layout XML and parsing it to create the view and viewgroup Java objects.

A public method to change the size of the font

text_fragment.xml

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

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/fragment_two"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

Advance Reading

- Next week's topic: building web services
- https://en.wikipedia.org/wiki/Representational_state_transfer

- (no quiz as we'll have a TD note)

Questions