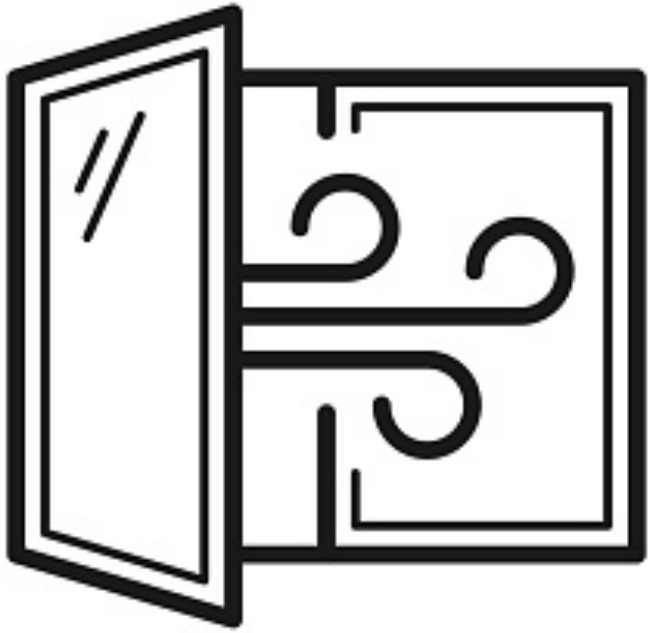
Android App Programming

Lecture 9: Preparing for the Project
Custom Drawing
Building Web APIs
Jan 04, 2022

Anastasia Bezerianos
Université Paris-Saclay

in class

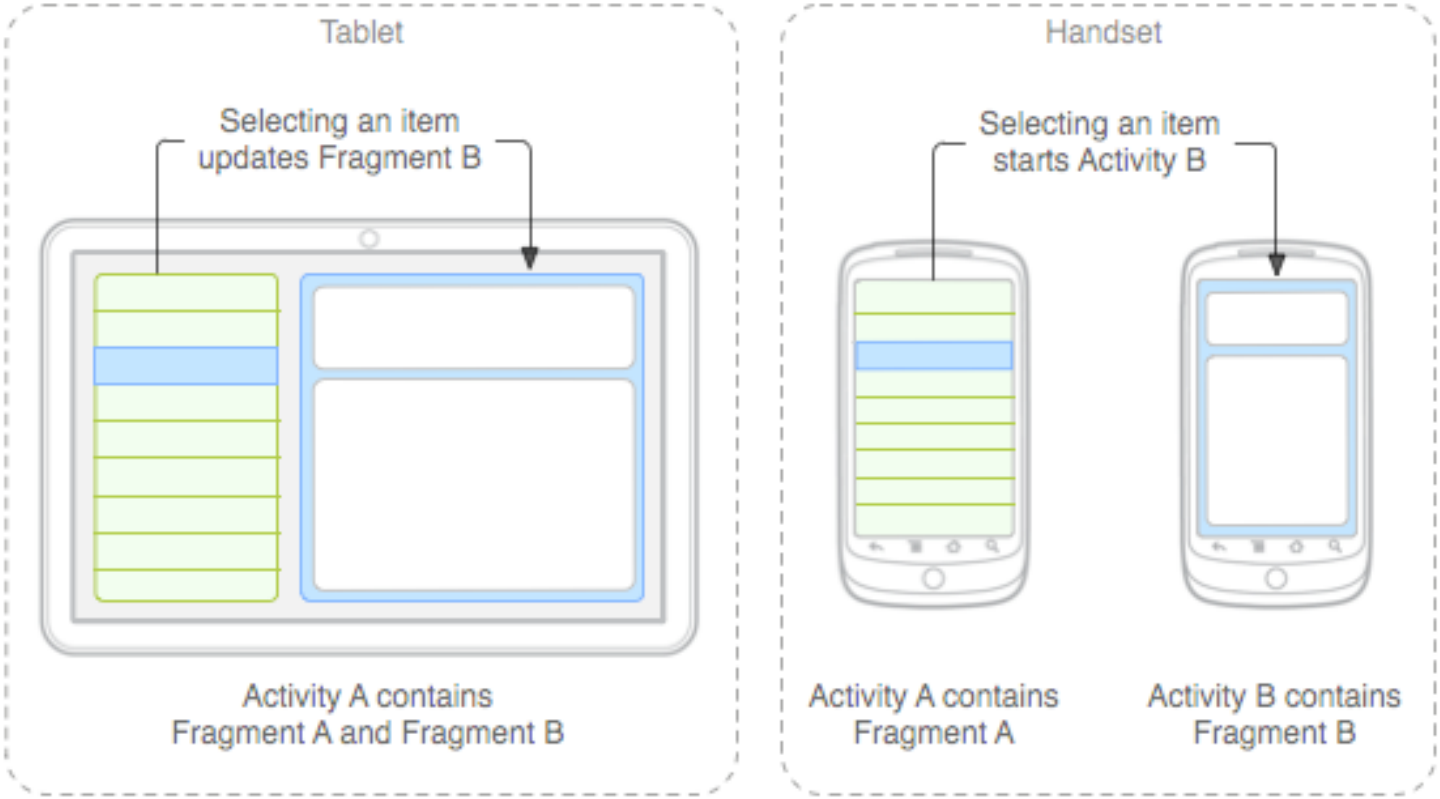


NO TD noté 2

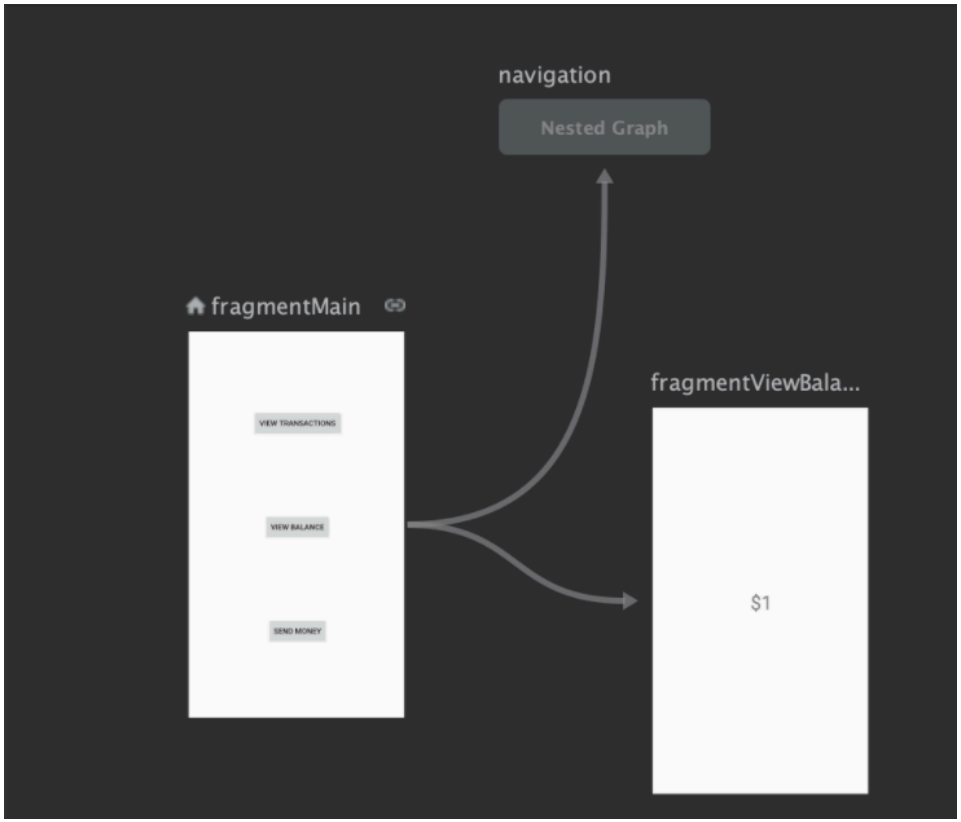
- 50% Individual work
  - Quiz 10%
  - TD noté 1 40%
- 50% Project (with Chantal Keller & Thomas Nowak)

Last lecture recap

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

Navigation


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

Working on Canvas

Canvas

- Class for drawing directly pixels
- Uses a 2D graphics library
- Uses Touch events

Lab 9

- Exercise 1: Canvas
- (start with an Empty project and add a class following the tutorial https://medium.com/@huseyinozkoc/android-canvas-and-create-custom-view-c6ed11fcc42f

myView.java
(from this tutorial)

```java
package fr.ups.myapplication;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;

import androidx.annotation.Nullable;

public class myView extends View {

    private Path path = new Path();
    private Paint paint = new Paint();

    //Constructor
    public myView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);
        setFocusable(true);
        setFocusableInTouchMode(true);
        PaintSettings();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawPath(path, paint);
    }

    // Get x and y and follow user motion events
    public boolean onTouchEvent(MotionEvent event) {
        float pointX = event.getX();
        float pointY = event.getY();
        // Checks for the event that occurs
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                // Starts a new line in the path
                path.moveTo(pointX, pointY);
                break;
            case MotionEvent.ACTION_MOVE:
                // Draws line between last point and this point
                path.lineTo(pointX, pointY);
                break;
            default:
                return false;
        }

        invalidate();
        return true;
    }

    private void PaintSettings() {

        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.BLUE);
        paint.setStrokeWidth(10);
    }
}
```

Class that extends View (all UI elements)

Class Paint (defines drawing properties like color, style, etc).
Class Path draws paths between points.

Constructor: define some parameters

onDraw comes with the Canvas and helps us create custom Views

EventListener for Touch events
(reacts to touch down and touch move)

Uses methods from class Paint and class Path

calls the redrawing of the View

Helper function that defines drawing parameters (the Paint class has a lot more)

MainActivity.java

```
package fr.ups.myapplication;

import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;

import android.graphics.Color;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

    private void configureLayout(){
        ConstraintLayout myLayout = new ConstraintLayout(this);
        myLayout.setBackgroundColor(Color.rgb(3, 132, 252));

        myView mycustomview = new myView(getApplicationContext(), null);

        myLayout.addView(mycustomview);

        setContentView(myLayout);
    }
}
```

Layout created dynamically (programmatically) as in Lab 7

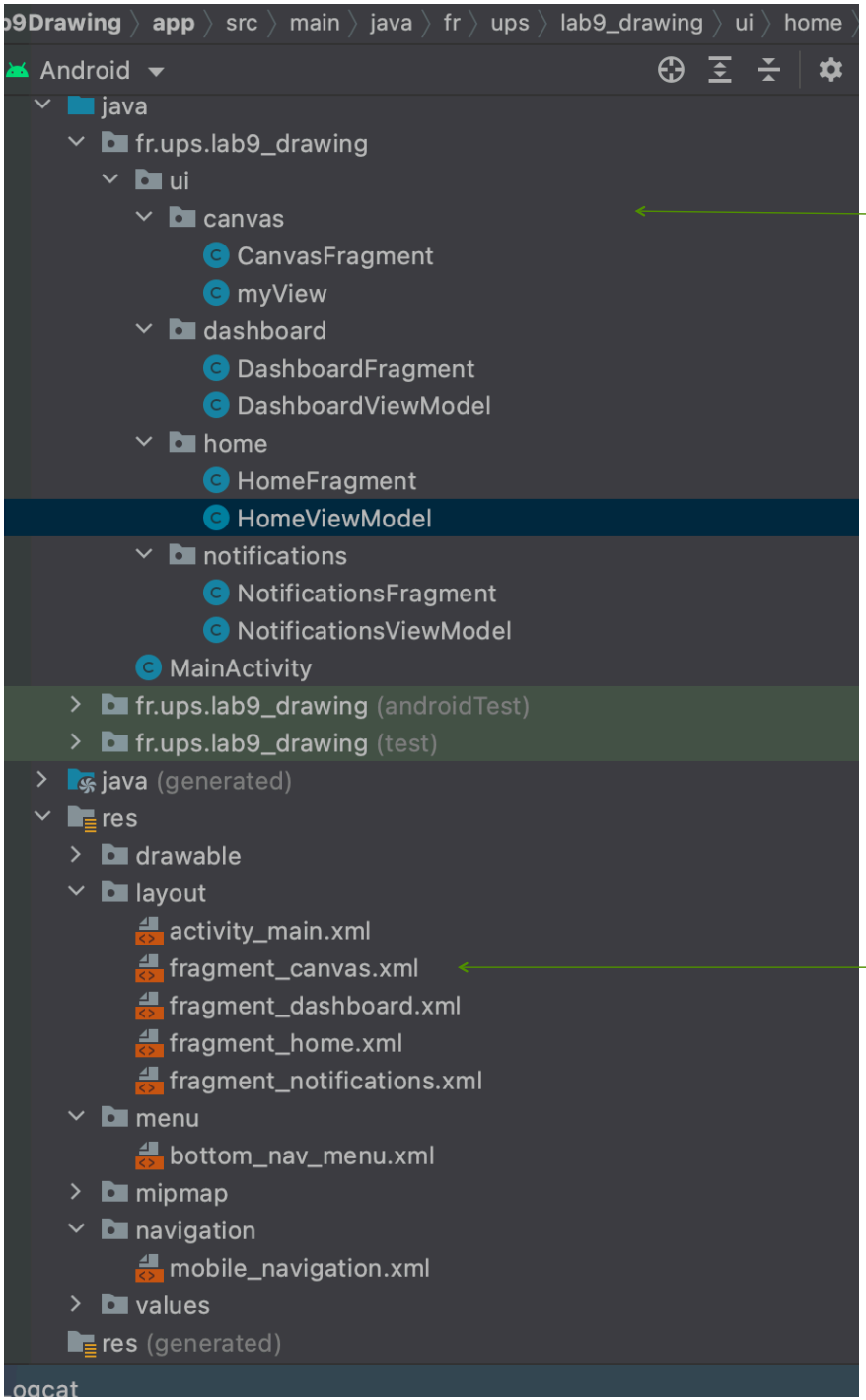
Creating a View of class myView

Adding the custom View to my layout

set the content of my MainActivity to be the layout

Lab 9

- Exercise 2: Canvas (or any other View) inside a Fragment



I created a package canvas and added my Fragment class and myView class (from Ex 1)

my canvas fragment

bottom_nav_menu.xml

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

    <item
        android:id="@+id/navigation_home"
        android:icon="@drawable/ic_home_black_24dp"
        android:title="@string/title_home" />

    <item
        android:id="@+id/navigation_dashboard"
        android:icon="@drawable/ic_dashboard_black_24dp"
        android:title="@string/title_dashboard" />

    <item
        android:id="@+id/navigation_notifications"
        android:icon="@drawable/ic_notifications_black_24dp"
        android:title="@string/title_notifications" />

    <item
        android:id="@+id/navigation_canvas"
        android:icon="@drawable/ic_notifications_black_24dp"
        android:title="Canvas" />

</menu>
```

my canvas fragment

bottom_nav_menu.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/mobile_navigation"
    app:startDestination="@+id/navigation_home">

    <fragment
        android:id="@+id/navigation_home"
        android:name="fr.ups.lab9_drawing.ui.home.HomeFragment"
        android:label="@string/title_home"
        tools:layout="@layout/fragment_home" />

    <fragment
        android:id="@+id/navigation_dashboard"
        android:name="fr.ups.lab9_drawing.ui.dashboard.DashboardFragment"
        android:label="@string/title_dashboard"
        tools:layout="@layout/fragment_dashboard" />

    <fragment
        android:id="@+id/navigation_notifications"
        android:name="fr.ups.lab9_drawing.ui.notifications.NotificationsFragment"
        android:label="@string/title_notifications"
        tools:layout="@layout/fragment_notifications" />

    <fragment
        android:id="@+id/navigation_canvas"
        android:name="fr.ups.lab9_drawing.ui.canvas.CanvasFragment"
        android:label="Canvas"
        tools:layout="@layout/fragment_canvas" />
</navigation>
```

navigation to my Fragment

fragment_canvas.xml

my canvas fragment xml, code from Android studio

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.canvas.CanvasFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="My Canvas Fragment" />

</FrameLayout>
```

MainActivity.java

```java
package fr.ups.lab9_drawing;

import android.os.Bundle;
import android.view.View;
import android.widget.LinearLayout;

import com.google.android.material.bottomnavigation.BottomNavigationView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import android.view.LayoutInflater;

import fr.ups.lab9_drawing.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity {

    private ActivityMainBinding binding;

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

        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        BottomNavigationView navView = findViewById(R.id.nav_view);
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(
                R.id.navigation_home, R.id.navigation_dashboard, R.id.navigation_notifications,
                R.id.navigation_canvas)
                .build();

        NavController navController = Navigation.findNavController(this, R.id.nav_host_fragment_activity_main);
        NavigationUI.setupActionBarWithNavController(this, navController, appBarConfiguration);
        NavigationUI.setupWithNavController(binding.navView, navController);


    }

}
```

Most of this code is created by default by Android studio

Inflate to use binding in the navigation (Lab 8)

The menu

added here my own Fragment

Setup of Navigation controller (see Lab 8)

CanvasFragment.java

```java
package fr.ups.lab9_drawing.ui.canvas;

import android.graphics.Color;
import android.os.Bundle;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import fr.ups.lab9_drawing.R;

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

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public CanvasFragment() {
        // Required empty public constructor
    }
    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment CanvasFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static CanvasFragment newInstance(String param1, String param2) {
        CanvasFragment fragment = new CanvasFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

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

        ConstraintLayout myLayout = new ConstraintLayout(container.getContext());
        myLayout.setBackgroundColor(Color.rgb(3, 132, 252));

        myView mycustomview = new myView(container.getContext(), null);
        myLayout.addView(mycustomview);

        return myLayout;

        //myView mycustomview = new myView(container.getContext(), null);
        //return mycustomview;
    }
}
```

Most of this code is created by default by Android studio
(see class and Lab 8)

Create a Layout (which is a View but also a container), add my custom myCanvas to it

and return the Layout (ie this Fragment will show the layout)

Alternative: only show the custom View

Building Web APIs

RESTful APIs (week 3)

- We will use a REST API, i.e., that follows the REpresentational State Transfer architecture [*]
- they are stateless and separate the client and server (their implementation can be done independently)
- server / client only know the format of msg
- REST-compliant systems are called RESTful systems

communicate (week 3)

1. Allow your app to go online (in `AndroidManifest.xml`)

2. Use a library or client to do the HTTP request towards the API (e.g., Volley, Retrofit 2, Ion, …)

3. Read JSON data and turn them into a Java object using a library (e.g., org.json, GSON) [*]
   - You should have a class to store these objects, often called POJOs (Plain Old Java Objects)

4. Setup & Make the Http request.
   - Use a callback function to wait for response

5. … do stuff with the data we read

[*] these libraries do de-serialization

- What about creating our own server / online repository ?

Node.js

- JavaScript runtime for web **servers**
- reactive (i.e., asynchronous) programming
- can scale to massively parallel programs
- used for web sites as well as for web services

Lab 9

- Exercise 3
  - Preparing for the project: GitHub and Node.js
  - build a basic RESTful web API in Node.js

setup

(instructions in the tutorial webpage)

- 1. Create a GitHub account
- 2. Locally install Node.js
- 3. Create a Heroku account
- 4. Create your first service

1. GitHub

- Code repository and versioning control
- Uses Git (software for tracking changes in any set of files)
- Good for your projects (especially if working with others)
- (can also host websites)
- Check if machines have Git for Windows installed in your machine ("git --version"). Let Anastasia know if not
- Create an account in https://github.com/

2. Node.js

- Install following instructions from the Lab page

3. Heroku

- A platform for deploying apps / services online
- We will use it to host our Node.js server online
- Create an account in https://www.heroku.com/

4. First service

- Follow tutorial to create and deploy a simple service online

- https://javascript.plainenglish.io/hosting-node-js-app-on-heroku-in-less-than-5-mins-bc7ce244c8d0

Questions