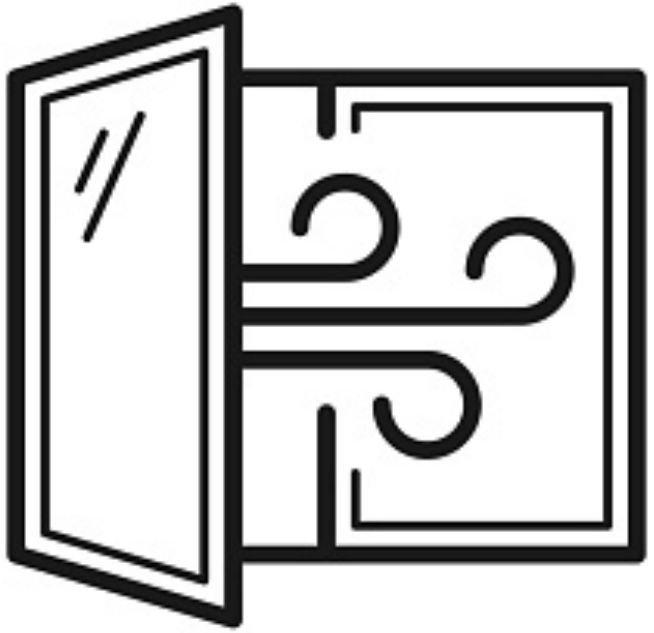
# Android App Programming

Lecture 2: Intents
October 12, 2021

Anastasia Bezerianos
Université Paris-Saclay

course based in part on that of Thomas Nowak

# in class



# Last lecture recap

# R

- Dynamically generated class during compilation
- Links XML layout files to Java
- Contains all assets of the application (layout files, images, strings, etc.)
- Field names defined in layout editor
- Ex: `R.id.toolbar`

# Activities

- Single block of user interaction
- In the simplest case, one UI screen
- Can be a window or part of the screen
- Activities are by default independent from each other
- To display something, set its content:
- `setContentView(View)`

# Views

- Basic UI component
- E.g., Button, TextView, ImageView, menus, etc.
- Has a set of mutable properties (e.g., text size)
- Can register listeners:
- `button.setOnClickListener(this);`

# ADB and LOG

# ADB

- **Android Debug Bridge (adb)**
- lets you communicate with a device (virtual or physical)
- one such tool is the **Logcat**
- (also command line if interested) under `my*android_sdk*/platform-tools/`

# Logcat & LOG

- Logcat is a window in Android Studio displays system messages
- You can create your own msgs to add to the Logcat
- Create a tag in your program

```
private static final String TAG = "MyActivity";
```

- Use it to print msgs to Logcat, for example

```
Log.d(TAG,"i am a debug msg");
```

# Intents

# Intents

- Mechanism to launch Activities from other Activities
- To change Activity: need to create object of type `Intent`, then call the method `startActivity` with that object
- ex :

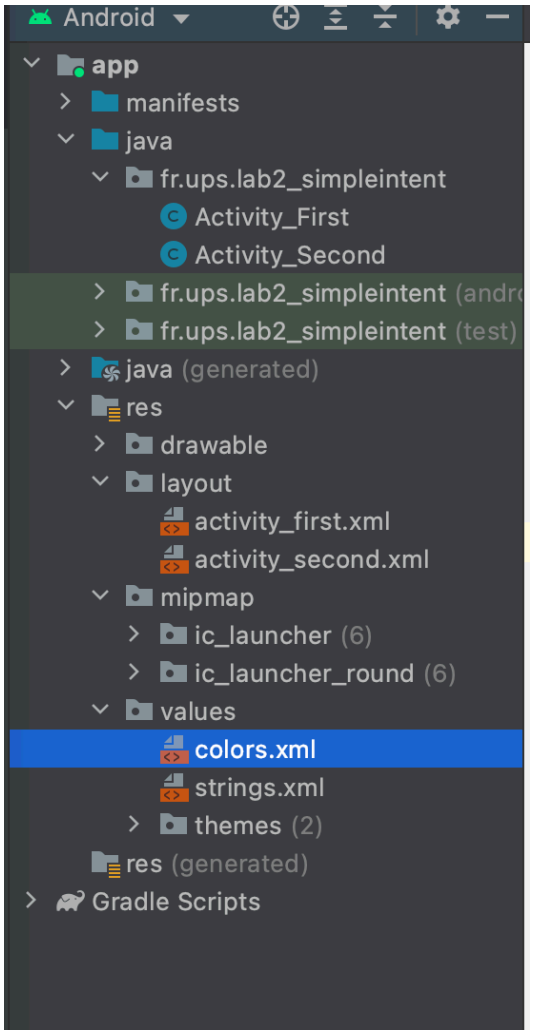
```
Intent i = new Intent(this,
                      OtherActivity.class);
startActivity(i);
```

# Intents
# Coding lab

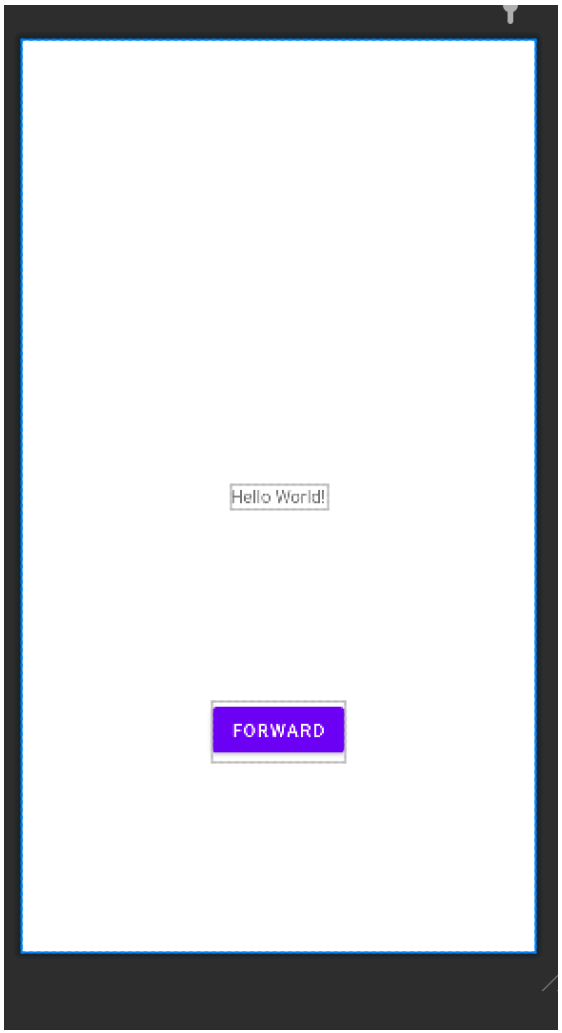
**Lab 2: Ex 1, part 1**

# Coding lab

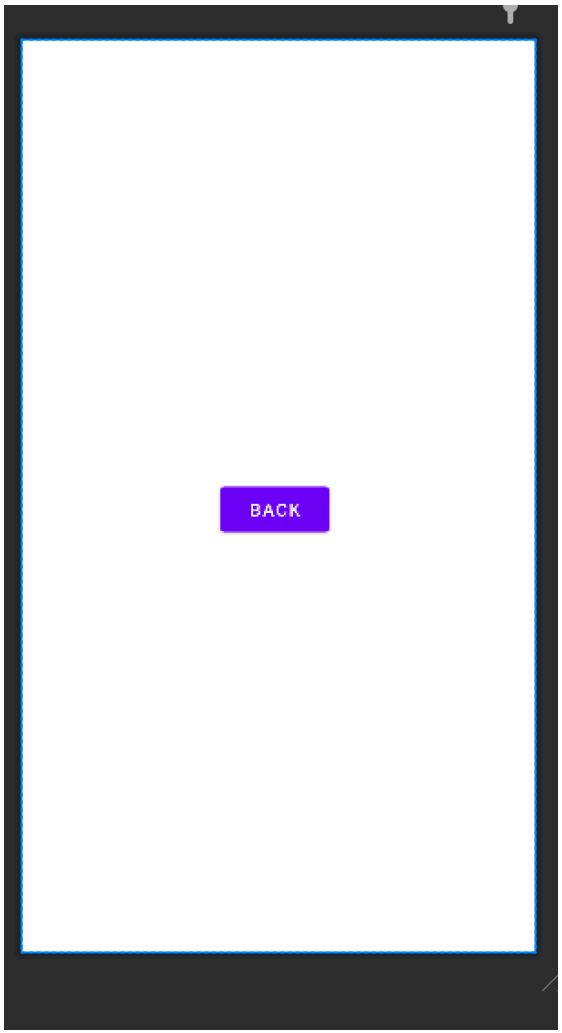
1. Create a new Empty activity that has a button
2. Add another activity that has a button
3. Add a method on onClick in the first (main) activity
4. 
```
Intent i = new Intent(this,
                      OtherActivity.class);
startActivity(i);
```



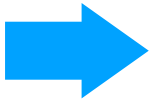

## activity_first.xml




## activity_second.xml

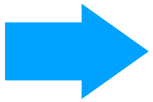


```
public class Activity_First extends AppCompatActivity {

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

    public void buttonCallSecondActivity(View view) {
        Intent second_i = new Intent (this, Activity_Second.class);
        startActivity(second_i);
    }
}
```



---

```
public class Activity_Second extends AppCompatActivity {

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

    public void buttonCallFirstActivity(View view) {
        Intent first_i = new Intent (this, Activity_First.class);
        startActivity(first_i);
    }
}
```



# Bundles

# Bundles

- Mechanism for Intents to pass data to activities: **Bundles**
- Can put data(*) in and retrieve it
- ex:

```
bundle.putFloat(“pi”, 3.14);

bundle.getFloat(“pi”);
```

**(*) Float, Integer, String, …**
**but also any object of a class that Implemets Serializable**

# Passing Data between Activities

- Every `Intent` has a bundle called “Extra”
- Access via `putExtra` or `getExtra`
- You can send multiple data through one Extra bundle

# Passing Data between Activities (1st => 2nd)

- Put Extras in the sender activity in the Intent

```
Intent i = new Intent(this, ActivityB.class);
i.putExtra("myString", "This is a message for ActivityB”);
i.putExtra("pie", 3.14)
```

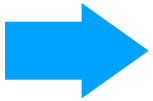
- In the receiving activity get access to the bundle it has received first, and then get data.

```
Bundle extras = getIntent().getExtras();
if (extras != null) {
    String myString = extras.getString("myString");
    Float myPie = extras.getFloat("pie");
}
```

# Bundles 1st=>2nd
# Coding lab

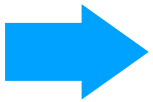
**Lab 2: Ex 1, part 2**

```
public class Activity_First extends AppCompatActivity {

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

    public void buttonCallActivitySecond(View view) {
        Intent second_i = new Intent (this, Activity_Second.class);
        second_i.putExtra("a_name", 3.14);
        startActivity(second_i);
    }
}
```



---

```
public class Activity_Second extends AppCompatActivity {

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

        Bundle extras = getIntent().getExtras();
        if (extras != null){
          float myFloat = extras.getFloat("a_name");
        }
    }

    public void buttonCallActivityFirst(View view) {
        Intent first_i = new Intent (this, Activity_First.class);
        startActivity(first_i);
    }
}
```



# Passing Data
# Inverse direction

- Inverse direction (will be soon DEPRICATED)
  - ~~In the caller (sender) `startActivityForResult`~~
  - ~~need to implement `onActivityResult` in caller (sender)~~
  - ~~need to call `setResult` in `finish` method in callee (receiver)~~

# Passing Data between Activities (1st <= 2nd)

- Inverse direction (new recommended approach)
- Activity 1st:
  - Use the Activity Result API on the sender side
  - Create an `ActivityResultLauncher`
  - Create a callback `ActivityResultCallback` and put in onActivityResult (default method) the code to handle results
  - Register a callback/listener `registerForActivityResult` and `.launch`
  - read more in https://developer.android.com/training/basics/intents/result

# Passing Data between Activities (cont'd)

- Inverse direction (new recommended approach)
- 2nd activity
  - On the callee/ receiver side …
  - Create a new Intent (without an activity)
  - Add data to its bundle (Extra)
  - Assign the intent as the result of the activity `setResult(RESULT_OK, Intent)`
  - terminate the receiver activity with `finish()`

# Bundles A<=B
# Coding lab

**Lab 2: Ex 1, part 3**

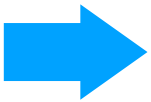
# (reminder) Java

- Event listenerAnonymous inner class:

```
addListener(Event, Listener);

addListener(Event, new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                …;
                }
            });
```

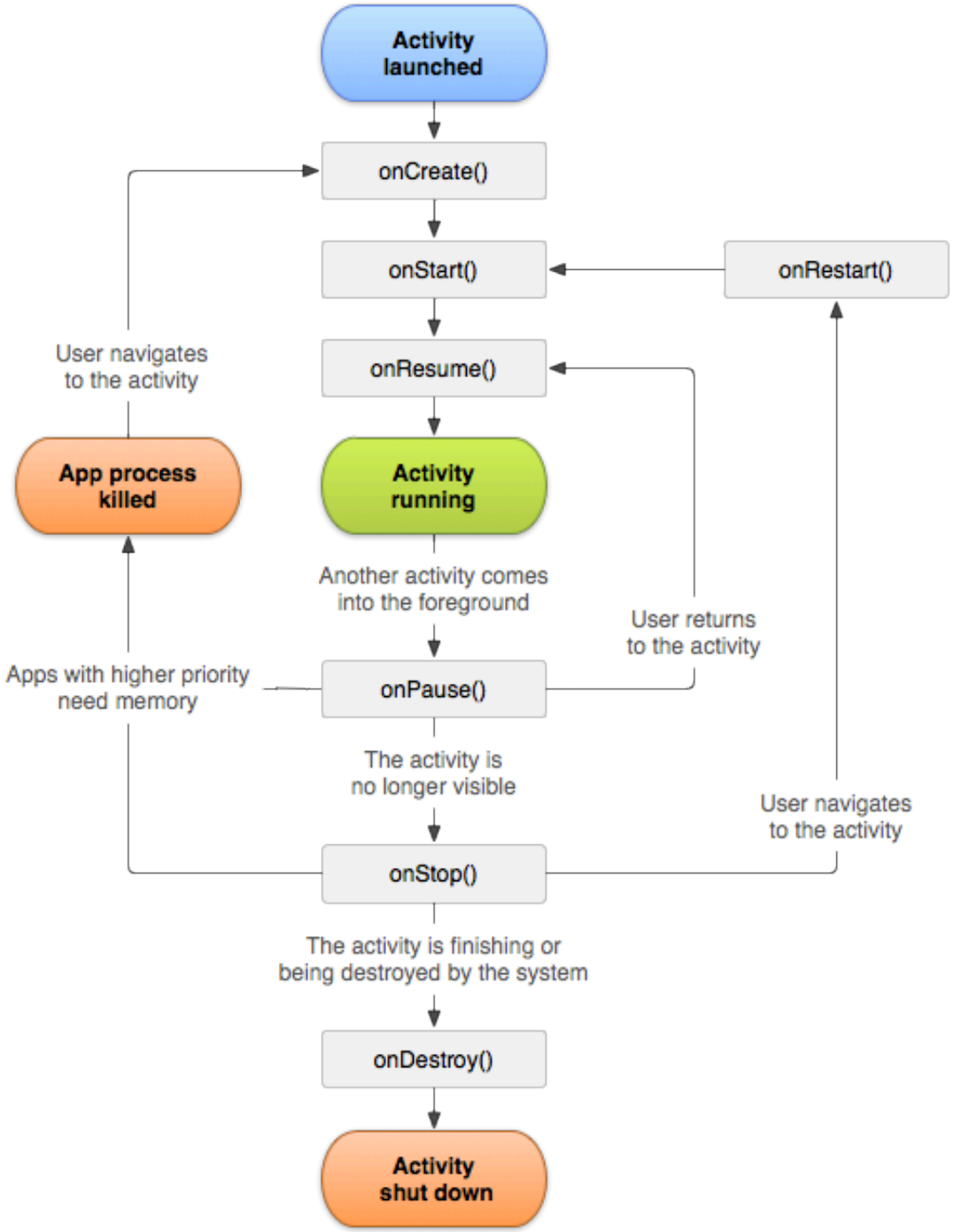
```
public class Activity_First extends AppCompatActivity {

    ActivityResultLauncher<Intent> mStartForResult;

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

        mStartForResult = registerForActivityResult(
                            new ActivityResultContracts.StartActivityForResult(),
                            new ActivityResultCallback<ActivityResult>() {
                              @Override
                              public void onActivityResult(ActivityResult result) {
                                if (result.getResultCode() == Activity.RESULT_OK) {
                                  Intent intent = result.getData();
                                  // Handle the Intent results
                                  float result =
                                   intent.getExtras().getFloat("myOtherFloat");
                                }
                              }
                            });
                }

    public void buttonCallActivitySecond(View view) {
        Intent results_activity_i = new (this, Activity_Second.class);
        results_activity_i.putExtras("a_name",)
        mStartForResult.launch(results_activity_i);
    }
}
```

```
public class Activity_Second extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
       Bundle extras = getIntent().getExtras();
       if (extras != null){
        float myFloat = extras.getFloat("a_name");
       }
    }

    public void buttonCallActivityFirst(View view) {
      Intent returnIntent = new Intent();
      // notice that it is not attached to an activity
      returnIntent.putExtra("myOtherFloat",2.718);
      setResult(RESULT_OK, returnIntent);
      finish();
    }
}
```



# Activity Lifecycle

- Activities can be:
  - foreground
  - visible, but not foreground
  - neither
- Activities that are not foreground can be killed by the system to free resources (priority to non-visible)


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

# Activity Lifecycle

- Activities that are killed and restarted come with their initial state (as defined in `onCreate`)
- To save and restore some dynamic state, the methods `onCreate` and `onSaveInstanceState` are called by the system upon creation and pausing. Both have a `Bundle` as parameter
- **BUT** sometimes the system kills activities for memory (see our lab examples). So it is better to store important state information externally (eg. databases)

# Lab 2

1. (done) Simple app with 2 activities and communication
2. Redo last week's game with multiple activities

Upload your code (exercise 1-2) in a zip file

# Advance reading for next week

- Next topic:
- Web services - 5min **test next week**
- https://www.androidauthority.com/use-remote-web-api-within-android-app-617869/

# 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

```
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

- 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

- Many of the Project activity examples that newer Android Studio versions provide actually use Bindings
- For recent Android Studios, try Basic Activity or Bottom Navigation activity to see it in action in (other aspects, like Fragments, we will discuss later)

# Questions