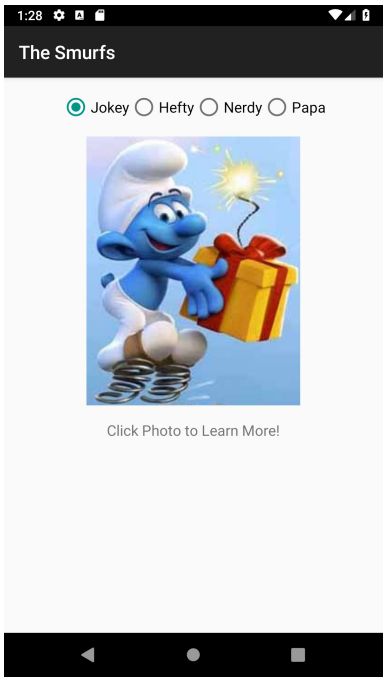
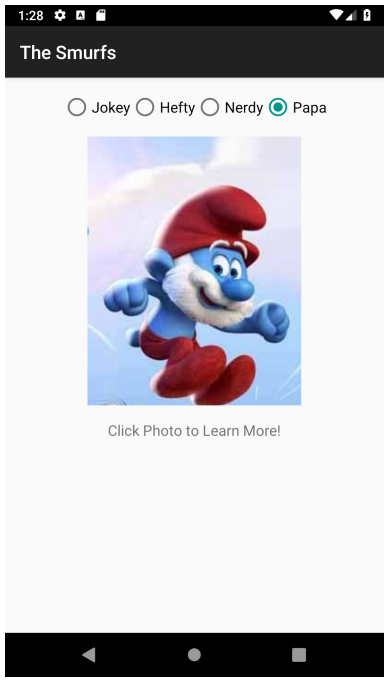
Can the students in the Aungier Street Lab Please come over to Kevin Street (me in room 1-015 or Andrei in room 3-006) as your Lab Assistant is sick today!

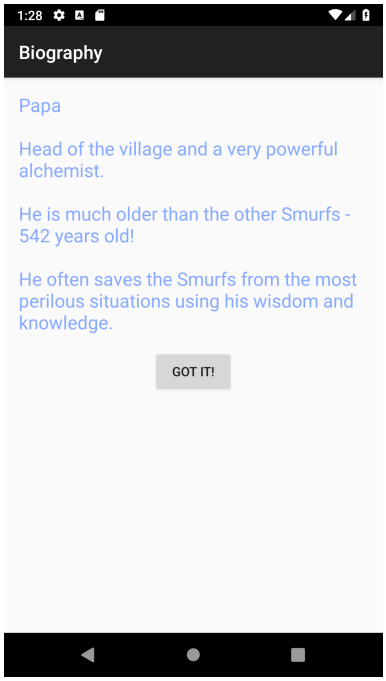
Last time we created an app that explored using **implicit** intents to move us beyond a single UI screen (**Activity**) and launch a map that displayed user entered latitude and longitude info.

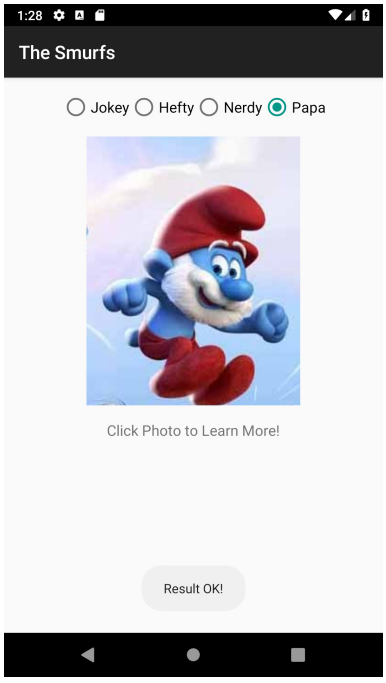
This week we will be looking at **explicit** intents and saving and restoring state as the lifecycle of our activity changes. How the app works is simple, when it launches, the Smurfs theme music plays and Jokey smurf is displayed. The user can switch between Smurfs by clicking the **radio buttons**.





When the user clicks a Smurf image, the system launches a Details Activity with biographical info.



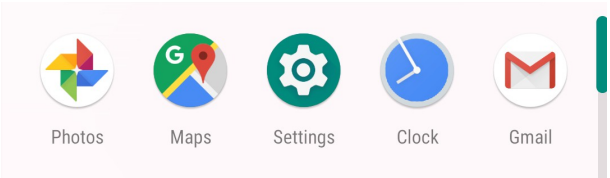


If the user clicks the “Got it” button the app returns to the Main Activity and pops a toast.

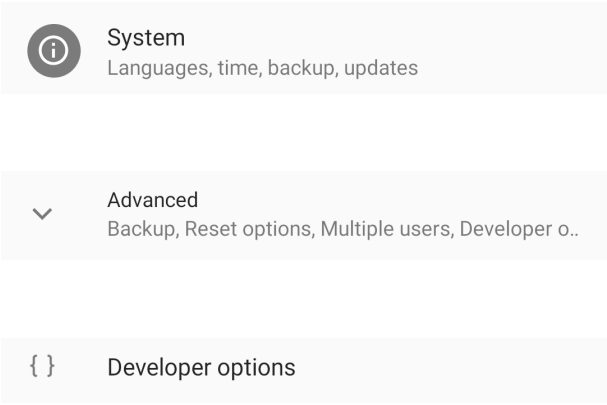
Based on the same procedure used the last couple of weeks create a new project, you can call the application “My Fifth App”.

N.B. For this lab we would like to study what happens when an Activity gets destroyed. For this reason we need to make a change to the Emulator settings. To do this follow these instructions:

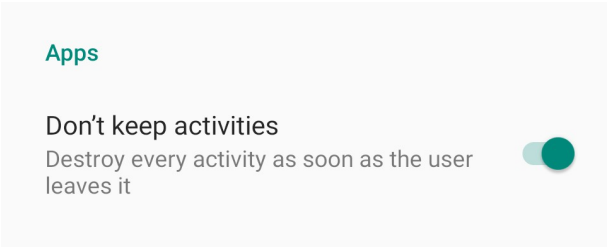
In the Emulator launch the settings app



Click on System → Advanced → Developer options



Then scroll all the way down to Apps and **turn on** Don't keep activities



If you are not seeing developer options try this:

Navigate to Settings → System → About emulated device → Build number

Click that build number seven times, and you’ll eventually see a notification that says “Congrats, you are now a developer”

Just ask the Lab Assistant if you are having problems with any of this.

Part 1 Project Set Up

Java

There are two activity files for this app MainActivity.java (where you will be writing code) and DetailsActivity.java (already completed). Copy these files into your project.

Resources

Under **res** copy the .png image files for the smurfs into the **drawable** folder and the .mp3 theme tune file into the **raw** folder (you may need to create this directory). The two layout files activity_main and activity_details are already completed. These files make use of information in the colors, dimens and strings xml files available in the **Other XML files** folder.

Part 2 Logging LifeCycle Methods

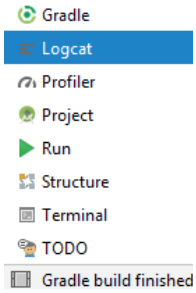
To keep track of when the system is calling the Lifecycle methods in the Main Activity we will be logging these events to **Logcat**.

In MainActivity.java

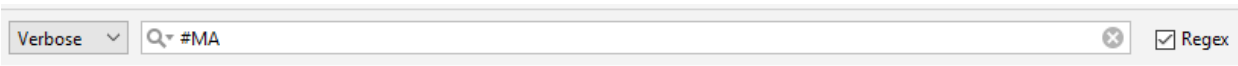
1) Add the following Lifecycle methods:

   onPause, onResume, onRestart, onStart, onStop and onDestroy

2) The first line of each Lifecycle method should contain a call to the method in the **superclass**.
3) Import android.util.Log
4) Include a call to the method Log.d in each method of MainActivity.java. The arguments for this method are: a String tag e.g. "#MA" and a String **msg** "theNameOfTheMethod" e.g. below, I have accessed **Logcat** by clicking the button in the bottom left corner of Android Studio



You can **filter** for your logs by entering the tag and checking the Regex **checkbox**:



In this example logging calls have been included in the first three lifecycle methods, which produces the following output containing the tag and method name in **Logcat** when the app launches:

```
com.example.lab.smurfs D/#MA: onCreate
com.example.lab.smurfs D/#MA: onStart
com.example.lab.smurfs D/#MA: onResume
```

5) The MediaPlayer (used to play the Smurfs theme music) has already been created and is referenced by the **player** variable. In onResume check if **player** is **not** equal to null and if this is true, set looping equal to true and start the **player**.
6) In onPause stop the **player**, as we don't want to hear the music when reading the biography info.

Part 3 Explicit Intent with Result

When the user clicks on an image of a Smurf in the Main Activity we would like to launch the **Details** Activity and pass it the **id** of the checked radio button (0, 1, 2, or 3), so that it knows which biography to show.

This code should be written in the method onClickSmurfImage

1) Since the RadioGoup that looks after the radio buttons is defined in the layout file, we need to find it by it's **id** and assign the returned value to a variable of type RadioGroup.
2) Then, ask the RadioGroup variable to get the id of the checked radio button and assign the returned value to an integer variable e.g. you can call this variable **id**.
3) Create an explicit intent to launch the Details Activity.
4) Use putExtra to add the **id** to the intent with the name “smurf_id”
5) To launch the Details Activity, instead of calling startActivity, call the startActivityForResult method and include ***REQUEST_CODE_DETAILS_ACTIVITY*** as the second parameter in this call.

Part 4 Saving Instance State

Should the Main Activity get destroyed we would like to restore it to the state in which the user left it. In this case we would like to keep track of the **id** of RadioGroup button that was checked.

This code should be written in the method onSaveInstanceState

1) Put the int id of the current checked radio button in the bundle and give it the key “id”