ECSE-682 Assignment-1
Basic Mobile/ Embedded Code Development

Katyayani Prakash* and Wenqi Liang†
Department of Electrical and Computer Engineering,
McGill University, Montreal, Canada
Email: *katyayani.prakash@mail.mcgill.ca, †wenqi.liang@mail.mcgill.ca,

***Abstract*—The purpose of this assignment is to show the basic lifecycle transitions for activity instance. This has been implemented by creating an application that consists of 2 activities, and 2 fragments. Different states of an activity lifecycle are displayed in the application's user interface by using toast messages. We have also deployed a counter at the top of each activity's UI to keep track of how many times a state within that activity has been called. The counter resets to zero once the activity has been destroyed. The toast notifications, and the counter, give the user of this app a hands-on idea of the activity lifecycle and it's various states.**

***Index Terms*—Activity life-cycle, Toast message, Fragment**

I. Introduction

An Activity represents a single screen in an android application with which the user can perform a single, focused task. It is usually presented to the user as a full-screen window [1]. Typically, one activity in an app is specified as the "main" activity (MainActivity.java), which is presented to the user when the app is launched. The main activity can then start other activities to perform different actions.

A. Activity Lifecycle

As a user navigates through an app, the Activity instances in the app transition through different states in their lifecycle. The Activity class provides a number of callbacks that allow the activity to know that a state has changed.

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). The system invokes each of these callbacks as an activity enters a new state.

In this project, we have demonstrated how each of these states are called, by employing toast messages for all their methods. There is also a counter at the top left corner of each activity screen, for all lifecycle states, to show how many times a callback was invoked. We have also used log messages to show the instantiation of each callback on the back-end.

II. Our Application

In this assignment, we have created a blueprint of a weather app. The Main Activity consists of a ribbon on top of the screen, with location, a search icon and drop-down menu button consisting of 2 options, both providing in-app notification. On the top left corner, there is a counter panel that consists of 6 counters, one for each state of the lifecycle, that are updated every time that callback is invoked. Below we describe the update of each counter and callback in six different use-cases.

A. Case I – Launching the app

When the app is launched, three callbacks are initiated immediately in quick succession: onCreate, onStart and onResume for the first (and Main) Activity. Since the difference between their invocation is less than a second, the three counters are updated almost simultaneously, however, toast notifications demonstrate the order in which they are called.

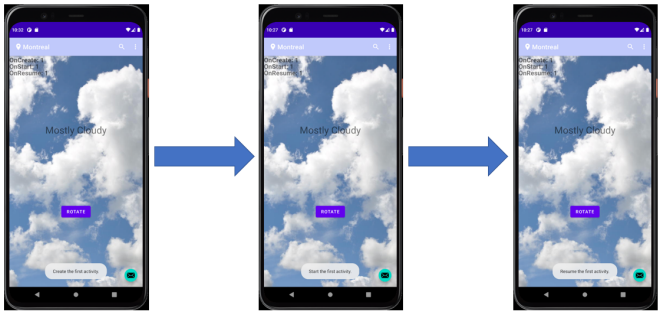

Fig. 1. Snapshots from the app in Case I[1]

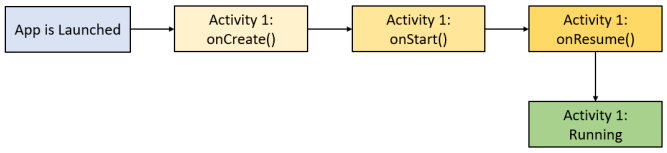

Fig. 2. Order of callbacks invoked in Case I

B. Case II – Navigating to second activity

In our app, the search button segues into the second activity. On clicking it, the first message we receive is "You clicked search button". After that, we get a series of notifications corresponding to the callback that has been invoked. The order of invocation of each method is shown in Fig. 3.

[1]Due to lack of space, these are the only in-app snapshots included here.

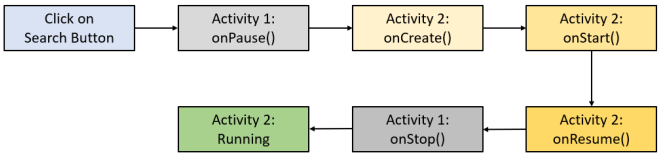

Fig. 3. Order of callbacks invoked in Case II

C. Case III – Minimize the app from second activity

If we minimize the app directly from second activity, it first pauses, and then stops the second activity. However, this activity is still running in the background (since it has not been destroyed yet). It is said to be preserved in "back stack". The back stack follows basic "last in, first out" stack logic. Thus, if we launch the app again, it will bring us back to where we left, i.e., Activity 2. A new callback, onRestart is initiated here, followed by onStart and onResume.

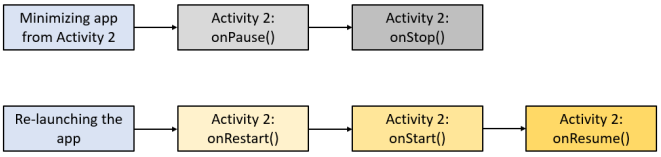

Fig. 4. Order of callbacks invoked in Case III

D. Case IV – Going back to first activity

On pressing the back button, we navigate back to the Main Activity. In doing so, the second activity is first paused, then stopped and destroyed, while first activity is now restarted, started and resumed. It is noteworthy that the second activity is stopped and destroyed after the first one is up and running.

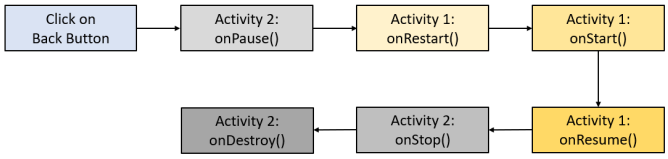

Fig. 5. Order of callbacks invoked in Case IV

E. Case V – Minimize the app from first activity

This case is similar to third case, and as expected, we see the callbacks mirroring the lifecyle of Activity 2 in Case III.

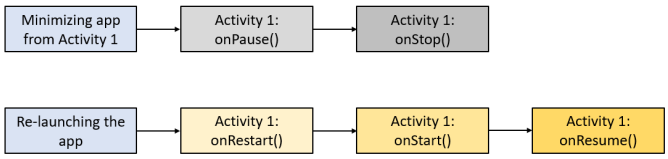

Fig. 6. Order of callbacks invoked in Case V

F. Case VI – Killing the app

When the app is killed, the main activity is paused, stopped and finally destroyed. If we kill the app directly from the second activity without first navigating to the main one, first second activity will be destroyed, and then the first one.

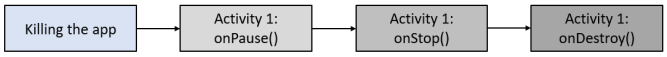

Fig. 7. Order of callbacks invoked in Case VI

III. Results

Below tables summarize the counter count for each callback state of both activities, after every use-case.

TABLE I
ACTIVITY 1 COUNTER VALUES AFTER EVERY CASE

	Counter Values					
	onCreate	onStart	onResume	onPause	onStop	onRestart
Case I	1	1	1	0	0	0
Case II	1	1	1	1	1	0
Case III	1	1	1	1	1	0
Case IV	1	2	2	1	1	1
Case V	1	3	3	2	2	2
Case VI*	1	3	3	3	3	2

TABLE II
ACTIVITY 2 COUNTER VALUES AFTER EVERY CASE

	Counter Values					
	onCreate	onStart	onResume	onPause	onStop	onRestart
Case I	0	0	0	0	0	0
Case II	1	1	1	0	0	0
Case III	1	2	2	1	1	1
Case IV*	1	2	2	2	2	1

*onDestroy() is also invoked in this step, terminating the activity completely and resetting all counters to zero. However, since it's the last step, it's invocation is never visible in UI. It can be viewed from the backend in log messages.

IV. Additional App Feature : Fragments

Aside from displaying the activity lifecycle, we have also incorporated fragments in the first activity. The home screen is Activity 1, Fragment 1. On pressing the button Rotate, the homepage rotates 90°, which is the second fragment. The Back button in second fragments brings back to first fragment.

V. Conclusion

In this assignment, we have built an android app from scratch, consisting of 2 activities (and 2 fragments). Incorporation of toast messages and callback counters provides the user with real-time visual feedback of lifecycle transitions through various callback states.

REFERENCES

[1] "The Activity Lifecycle — Android Developers", *Android Developers*, 2021. [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle. [Accessed: 25- Sep- 2021].

[2] "Android fundamentals 02.1: Activities and intents — Android Developers", *Android Developers*, 2021. [Online]. Available: https://developer.android.com/codelabs/android-training-create-an-activity?index=..%2F..%2Fandroid-training#1. [Accessed: 01- Oct- 2021].

[3] "Toasts overview — Android Developers", *Android Developers*, 2021. [Online]. Available: https://developer.android.com/guide/topics/ui/notifiers/toasts#java. [Accessed: 01- Oct- 2021].

[4] MindOrks, *Understanding Activity Lifecycle with Example Use Cases — Android Development For Beginners*. 2020. Available : https://www.youtube.com/watch?v=RiFui-i-s-o&t=286s. [Accessed: 30-Sep- 2021]

[5] Lecture Slides of ECSE-682: Topics in Computers and Circuits by Prof. Zeljko Zilic