Mobile Computing
Professor Pushpendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 14
Activity

Hello, Welcome to week 4. In this week we will go more into depth of activity so that you can understand how does an activity life cycle works, and how does your application moves from one stage of activity life cycle to another stage of activity life cycle. We have been discussing about activities from the very beginning, but today we will go into the detail that we have not covered earlier. Let us start.

Once again activity is the basic entity with the user interface with which your application is passed. So when a user navigates through out of end back to your app, the activity instances in your app transition from different states in their life cycle. For instance when your activity is start for the first time, it comes to the foreground of the system and receives user focus. During this process android system calls a series of life cycle methods on the activity in which you set up the user interface. If the user performs an action that it starts another activity or switches to another app the system calls another set of life cycle methods.

Throughout the execution of your app your activity will be moving from one stage of its life cycle to another stage of life cycle and during all these movements different call back methods will be called by the android system. Today we will learn about this call back methods as we learn about what happens when your activity moves from one stage of life cycle to different stage of life cycle. Let us start.

(Refer Slide Time: 2:20)

As you already know android system starts code in an activity instance and that there is no main method. When starting an app, the android system invokes specific call back methods that correspond to specific stages of its life cycle and a series of call back methods start up an activity. So while a sequence of call back method start up an activity, another sequence of call back methods destroy or tear down an activity, let us look at the next diagram carefully.

(Refer Slide Time: 2:59)

This is the diagram of activity of life cycle. As you see, that once you start an app, a call back method a call back method called `onCreate’ is called. This is the method that you saw in your program as the very first method when you created an app from the scratch. After the `onCreate’ method your app is in a state which we call as `Created’. From the `Created’ state another call back method `onStart’ is called and now your app is in a state called `Started’. At this time your app is visible to the user. From `Started’ another method called `onResume’ is called and now your activity is in the `Resumed’ mode.

`Resumed’ is similar to running of a process and that is why sometimes it may also be called back. `Resumed’ is a state in which your activity lives most of the time. From `Resumed’ if it is obstructed it can go to the `Paused’ state, in the `Paused’ state your activity is partially visible. For example - a dialog box comes over your activity. Again from `Paused’ state it can either go to `Resumed’ state again or if the user is closing your app your activity may go into `Stopped’ state or if the user has switched to another app your activity will again go to `Stopped’ state. So, both the conditions when the user switches on or a user wants to close your app from the `Paused’ it goes to `Stopped’. Now if the user has on the switched your application may again go into `Started’ then in `Resumed’. But if the user is stopping your allocation then the activity goes into the `Destroyed’ and a method `onDestroy’ is called.

You need to implement this activity life cycle methods carefully so that your app behaves correctly. For example - your app should not crash if the user receives a phone call or if the user switches to another app while using your app. Also your app should not consume valuable system resources because it is a mobile device when the user is not actively using it. For example, if your app is, let us say video playing app and the user for the time being moved to another app, you should stop your video player. Your app should not lose user’s progress if the user leave your app and return to it. Also your app should not crash or lose user’s progress when the user simply rotates the screen into land scape and portrait orientation. Please note that these are not special features, these are basic requirements for an android application. And in order to satisfy them you must know what life cycle methods have been called and how to implement them, so that your app behaves correctly.

(Refer Slide Time: 6:43)

Out of all the states that you saw, an activity exist only in the three states for a longer time. The other states are very much transient in nature that has activity passes through them very quickly. So, the three states in which an activity stays for a longer time are `Resumed’ - This is the state which is equivalent to running state of a process and this is the state in which your activity is in foreground and the user is interacting or can interact with it.

The second longer term state is `Paused'- In this state the activity is partially obscured by another activity. Please notice I am using the word `partially'. For example a dialog box comes out, now the activity is not in the foreground but is also not totally headed. This may happen when another activity interacts dialog box comes in for any other such thing, so that your activity is only partially obscured. The `Paused' activity does not receive any user input and cannot execute any code.

The third and the last state in which an (appli) in which your activity can live longer or a stay longer is the `Stopped' state. In this state, the activity is completely hidden and not visible to the user. For example, you have pressed the home button, now this state is considered such as your activity is in background. However, while your activity is `Stopped' the activity instance and all its state information such as member variable is retained by the android system, but it cannot execute any code.

(Refer Slide Time: 8:46)

Next, let us see a part of a manifest file. This manifest file is of a very simple application which only has one activity similar to the application that we have done. Earlier, we came to know that android programs do not have a main function and instead they can (la) start from any of the app component that is activity, services, broadcast, receives. Now, because they do not have a single entry path, how does an android system know we have to start them in the very beginning? This role is played by the manifest file. If you can go through the manifest file with you have created for your application you will see that it will have an entry similar to this activity `intent filter' inside that there would be an action item and a category item. And in the action item it would be mentioning `Main'. So, for the activity for which the `Main' is mentioned is the main activity and from this activity a program will start. But this is not sufficient; the category must also be the `Launcher' category. That is the main activity for an app must be declared in the manifest with an `intent filter' that includes the `Main' function in `Launch' category.

(Refer Slide Time: 11:05)

As told earlier to you intent is nothing but message object and the `intent filter' intents to filter all the messages except the one that satisfy this category, this category of `Launcher'. You can now all look at the manifest file of your program and verify it. Here we are. I am using an earlier version of our application then we only had two buttons. Let us look at our manifest part. And as you see we have an entry of `Main' and `Launcher' which is similar to what we were seeing, what this manifest file is telling that the quiz activity is the activity that must be launched first.

(Refer Slide Time: 11:34)

Now, let us start from the very beginning. When if the user first interacts with your app that is by double clicking the icon on an android phone, the system creates the new instance of activity by calling its own crate method if your application starts the single activity. Or in case a new activity or incase an activity needs to be called by let us say any other app, the system will create the new instance of that activity by calling `onCreate’ method. So, the `onCreate’ method is the first method that gets hold. The `onCreate’ method must perform the basic applications start up logic that should happen only once for the entire life of activity. For example, defining user interface or Instantiating class-scope variables, once the `onCreate’ finishes execution, `onStart’ and `onResume’ methods are called in quick succession. An activity never resides in the `Created’ or `Started’ states as told you earlier.

(Refer Slide Time: 12:41)

At the time of finishing the activity or destroying the activity `onDestroy’ is the last callback. So, `onCreate’ is the first call back and `onDestroy’ is the last call back. The system calls this method on an activity as the final signal that activity instance is being completely removed from the system. Now, you may be tempted to release your resources in the `onDestroy’ but that is usually not a good idea. You should do this clean up before `onDestory’ has called and as we saw in the diagram that before `onDestroy’, `onPaused’ and `onStop’s are called. Let us go back to this cycle quickly and check this.

(Refer Slide Time: 13:30)

Yes. So, before `onDestroy’, `onPaused’ and `onStop’s are called. So, in the `onPause’ and `onStop’ you should do the cleanup. However sometimes it creates some background threads during `onCreate’ or other long running resources. Now, those resources and those threads should be killed in the `onDestroy’. Please also note that `onDestroy’ can also be called directly from `onCreate’ by calling finish. This is used when an activity operates as a temporary decision maker to launch another activity.

(Refer Slide Time: 14:16)

Now, we have seen the first call back method and the last call back method. Now let us come to some of the middle call back methods. An activity is paused when it is partially obstructed by another activity, but remains visible. For example, a semitransparent activity opens on top, now in this case if an activity is visible, then it is paused but if it is fully obstructed and not visible then it stops. So, `onPause' is called when an activity enters the paused state as we saw in the diagram. `onPause' method allows us to pause ongoing actions that should not continue in a paused state. For example, if your app is about playing avideo, you may want to pause playing of the video if the your application is in a paused state because, that means that your activity is not fully visible to the user.

(Refer Slide Time: 15:50)

`onPause' also allows us to persist any information that should be permanently saved in case the user continues to leave your app. Let us go back to the diagram again and try to understand this. As you see that the `onPause' is being called from an activity which was in `Resumed' state. Now, if the user wants to come back that is fine, but sometimes if the user or every time when the user will be leaving your app again `onPause' will be called just before `onStop' and `onDestroy'. In such cases `onPause' is just on the route of when you are when the user is going to leave your app. And that is why whenever your app is in paused state, you should persist any information that you need later. Because the user may be on the path to leave your app. `onPause' is also the first indication that a user is trying to leave your activity.

(Refer Slide Time: 16:56)

Now, a user can always come back from the `Paused to the `Resumed' state and if the user returns to your activity from the `Paused' state the system resumes it calls down this method. Here are some recommended actions when you are pausing an activity. Please note that these actions are only a guideline and you will have to decide action depending on the application that you are down loading. For example, when you are pausing an activity, you should stop animation or any other ongoing actions that consumes CPU. The reason is that you may not know for how long your application will be in the `Paused' state and you would not like to consume CPU which in turn consumes battery and which should be preserved in all scenarios. You should also not commit unsaved changes, if and only if you want these changes to be permanently saved for example, a draft email. If the user was drafting an email and your application goes into a `Paused' state, you should save the draft because often the user wants to come back to the draft and start where they left.

This is also a good place to release system resources such as broadcast receivers, handles to sensors or any resources that may affect battery life, while your activity is paused. It was when your activity is paused the user does not need them. More over when the activity is paused, it may be well on a path to stop and destroy and that is why this is a good place to release system resources.

The activity instance is kept resident in memory when your activity is paused and it is recalled when the activity resumes. What this means is that in the paused state and coming back to the resumed state you do not need to re-initialize components that were created during any of the call back methods leading up to the `Resumed' state. Let me repeat, during any of the call back methods leading up to the `Resumed' state. What I am saying is that if you have any such component you do not need to re-initialize it, android system will take care of it. However, if there are other things which happened in the `Resumed' state you need to take care of that. Also during `onPause' you should avoid performing CPU-intensive task for example, writing to a data base. Just during performance of your application and it can slow the visible transition to the next activity. If you need to do such heavy CPU-intensive work, then please do that during `onStop'.

That is it for the today's lecture, in the next next lecture we will start with Resuming an activity. These series of lectures together will tell you about the complete life cycle of activity, so you should watch them either together or in quick succession so that you understand the entire concept.

Thank you.