**Course: Software-Hardware Codesign - CEN 4214**

**Students: Alexandre Souza Santos, Chad Mandy, Harold Davis**

**Group 4 - Lab Report 1**

**Abstract**

The goal of this assignment is to understand the lifecycle of an activity by running and analyzing the LifeCycle App, which can be found at the developer.android.com website.

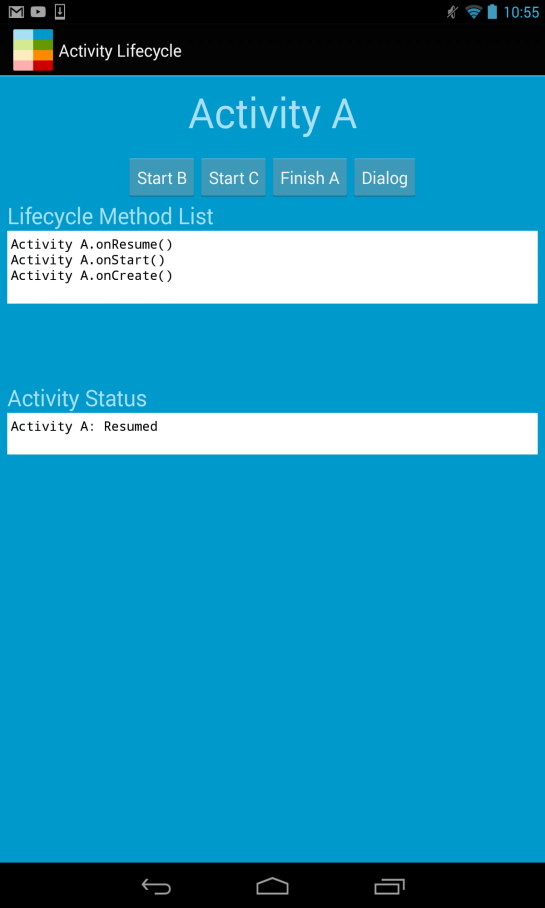
**Methods**

First, we downloaded and imported the LifeCycle app into the Eclipse. Eclipse was thus used to run the app on our Nexus 7. Then, we collected screenshots of therunning app to prove that it was working properly. The screenshots were taken on the device itself, by pressing and holding the power button at the same time until the animation and audible tone is heard.

**Results**

First, we will show the point of view of the user, and then we are going to explain each method individually.

As we can see at the figure below, when we open the LifeCycle App, the Activity A is automatically started, and the methods executed were onCreate(), onStart(), and onResume().



Next, we started Activity B by hitting the Start B bottom. We can observe that before the Activity B starts, the method onPause() is executed, pausing Activity A. Then, Activity B is started, and after this Activity A is stopped.



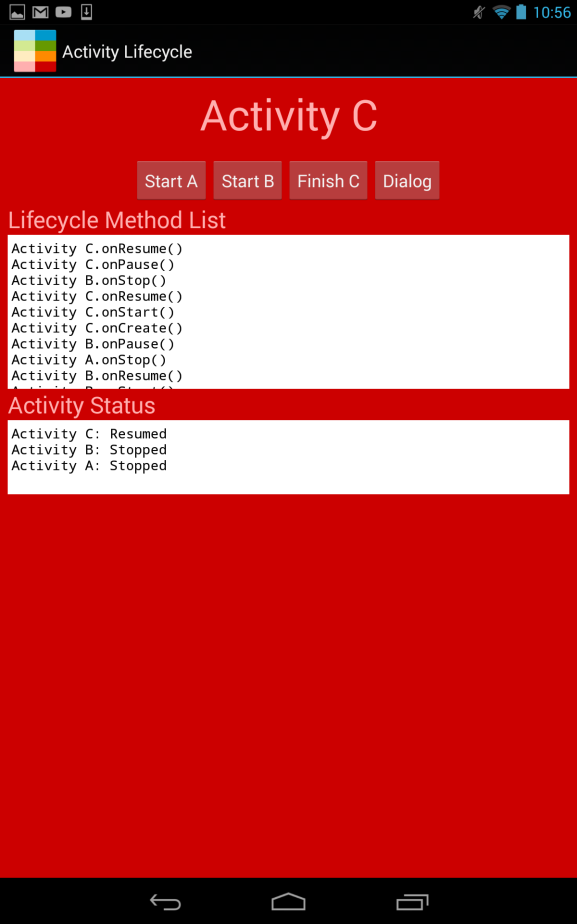
The same happens when we start Activity C. First, Activity B is paused and then Activity C is started. Hence, Activity B is stopped.



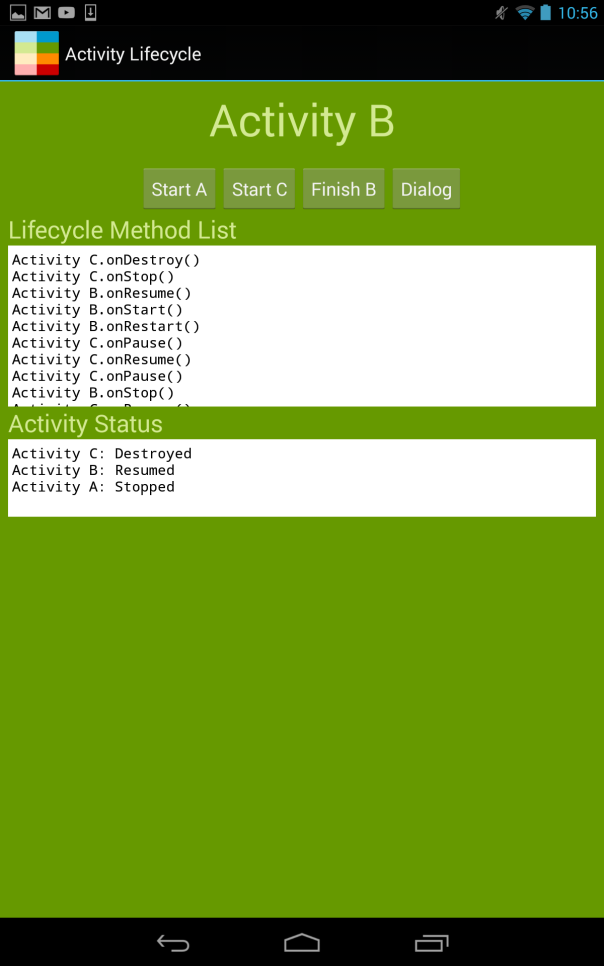
When we click on the bottom Dialog, Activity C is paused and a small window pops up.



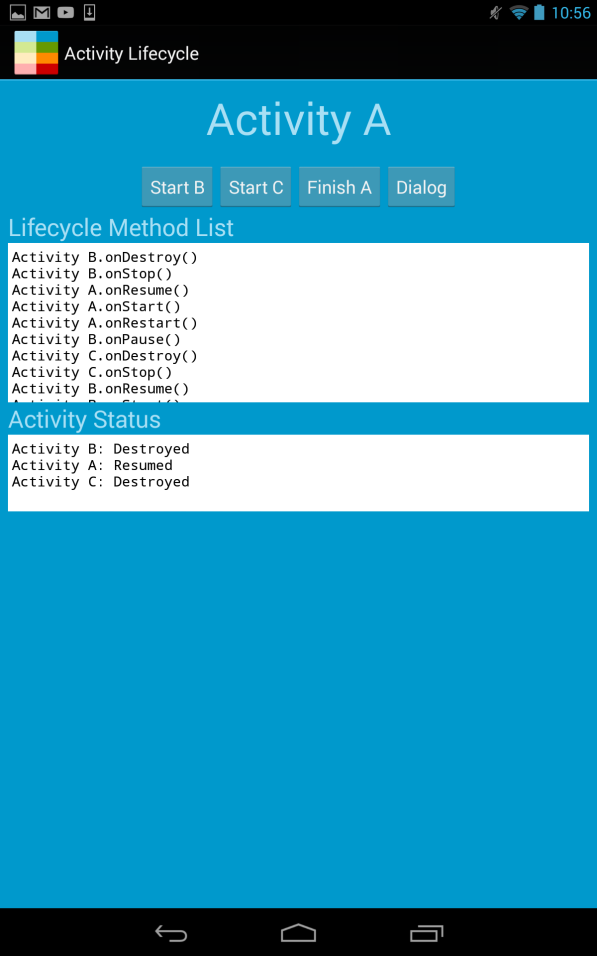
After closing this window, Activity C is resumed.



To continue, when we finish Activity C, Activity B is restarted, and then C is stopped and destroyed.



Lastly, the same happens when we finish Activity B. A is restarted and B is stopped and destroyed.



**Discussion**

Creating an activity

when an activity is created, it is loaded in memory for the first time because it does not yet exist there. This is when any static resources that the application needs such as layouts and images are loaded.

* The onCreate() function serves as an initial entry point for an activity and anything that needs to happen when an activity starts for the first time it its life cycle should be placed here.
* The onCreate() is called when a user navigates to an activity before it is loaded in memory. If there is not enough memory, than it is possible for a paused or stopped activity to be destroyed to free up space. After the activity is loaded it is placed into the activity stack and any previous activity will be paused.
* OnCreate loads any static resources that the activity needs including any previous saved bundled state from other sessions.

Starting an activity

when an activity has been loaded into memory its onStart() function is called which places the activity on the top of the activity stack and applies any layouts or views to the activity so that the user can see the activity loading.

* the onStart() function acts as an entry point for any activity that has been stopped and restarted with onRestart(). This is where the activity will return to when a user navigates away and comes back.
* During onStart, content is loaded into the layouts and views so the activity becomes visible in this state, but it is not yet interactable. Only after onResume will the user be able to interact with the activity. Code placed here should be as minimal as possible to reduce amount of time the user waits on an activity.

Pausing an activity

When a user leaves an activity, either to switch to another activity or to close the activity, the first method that is called is the onPause() method. When the activity is in this state, the activity is unable to receive any user input nor execute any code. The activity goes into the background, while another semi-transparent activity jumps partially into the foreground to take its place.

* While in this state, the onPause() callback is typically used to suspend any ongoing action that can take up CPU cycles, i.e. animations, videos, transition effects, etc.
* Saving the user’s work is also recommended in case of accidental interruptions, such as the user hitting the home button by mistake. Saved information should only be for something that is important, such as editing documents, drafting emails, editing photos, or creating excel sheets. Typically this is reserved for when a user expects something to be auto-saved.
* If the activity is paused, any system resource that it is currently using may be released to free up the resources and save battery life. A user is not going to need the Camera if they switch from their Instagram app to Google Chrome.
* Keeping the amount of operations done when onPause() is called will help create a faster and smoother transition from one activity to the next.

Something worth noting is that when an activity becomes paused, the instance of the Activity is automatically stored in memory and is recalled when said activity resumes.

In the Activity Lifecycle app, the onPause() method specifically stores the status of the app and then prints it on the app.

Resuming an activity

Sometimes referred to as the running state, the resumed state is when a paused activity goes from the background to the foreground and is now able to be interacted with. This stat calls the onResume() method. Now because this activity is called when the activity enters the foreground, this also includes when the activity is first created so make sure you plan for that.

* The onResume() method is essentially the counterpart to the onPause() method, anything you do in onPause(), it is generally recommended to do the opposite in onResume().
* Like the name suggests, any animation, effect or video that you had previously paused should now be resumed and allowed to run/play again.
* Restoring saved information to the appropriate location should happen in this method, i.e. retrieving all of the saved cells in an excel spreadsheet and placing them in the appropriate places.
* If a system resource was released, it should be re-initialized here. If you released the camera, you should initialize it to be used again.

In the Activity Lifecycle app, the onResume() method specifically stores the status of the app and then prints it on the app.

Stopping/Restarting an activity

The method onStop() is called when the activity is not visible to the user anymore. It happens when the current activity is being destroyed or when another activity occupies the foreground completely. This method is follow by the method onRestart()when this activity returns to the screen, and by the method onDestroy() when this one is being destroyed.

There are some situations in which the activity is stopped and restarted:

* When the user switches from the current app to another, the activity in the foreground is stopped. When the user returns to the first app, it restarts.
* When the user starts a new activity in the app, the previous activity it is stopped. Then, when the user click on the Back button, the first activity is restarted.
* When the user receives a phone call while using the app.

In general, when a activity is stopped, the Activity object is kept in memory and is recalled when the activity resumes.

How it was said before, the method onRestart() is only called when your activity comes back to the foreground from the stopped state, and it can be used to perform special restoration work. After an activity is restarted, the method onStart() is also called, what happens when your activity becomes visible.

Destroying an activity

The method onDestroy() performs a final cleanup before an activity is destroyed. It can be called when an activity is finished on purpose, or when the systems needs to recover the system memory.

* OnDestroy does not nessesarily remove the application from memory, it just removes it from the activity stack. This can improve performance because the CPU will not be busy with memory cleanup when the space is not needed. It will be completely removed if another activity needs the memory.
* OnDestroy may also be called on any paused or stopped activity if an activity with higher priority needs the memory

**Conclusion**

After analyzing the LifeCycle App we could better understand how the life cycle of an activity works while the app is running. The Graphic User Interface is very friendly, in which we could handle easily. The Methods List and the Activity Status were very helpful because we could see what methods were being called and the status of the activities at each time we performed an action. In brief, this assignment was a good opportunity to improve our team skills, and we are prepared for the next ones.

**References**

http://developer.android.com/