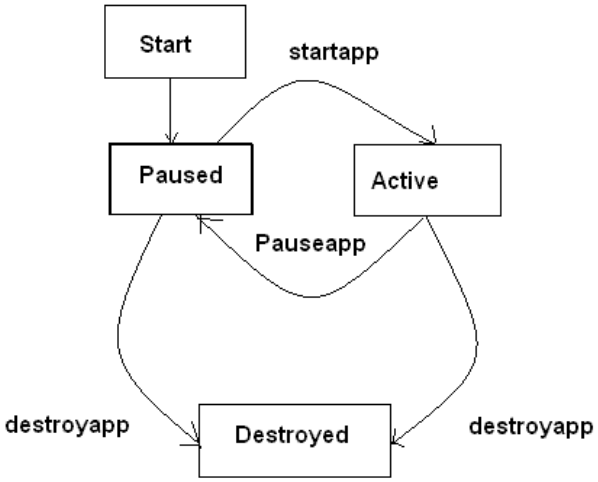
Midlet basic lifecycle and states

When a user clicks on the MIDlet icon the MIDlet begins execution.Following are the steps that follow:

1. The AMS first calls the zero-argument constructor to create a new instance of the MIDlet.
2. When the constructor returns, the AMS places the MIDlet in the Paused state.
3. To shift the MIDlet to the Active state the AMS calls the midlet.startApp() method.
   The startApp is the method where the main application logic begins..like in a typical game a thread is started and the game loop begins.
4. A transition from the Active state back to the Paused state occurs whenever the AMS calls midlet.pauseApp().
5. The MIDlet may shift from Paused to Active or back any number of times during its execution, each time on a call to startApp() or pauseApp().
   Usually pauseApp gets called when there is a device interrupt like call on a device or maybe you inserted the charger plug ..etc. **Note**. pauseApp is never called by the system on Nokia developer platforms.

   The basic states can be summarized from the figure below.



6. The MIDlet may enter the Destroyed state from either Paused or Active, on a call to midlet.destroyApp().
7. A MIDlet may voluntarily enter the Paused state by calling midlet.notifyPaused().
8. In a similar fashion, it may call midlet.notifyDestroyed() to inform the AMS that the MIDlet can now be considered Destroyed. The destroyApp() method is actually called with a boolean argument. If this boolean is true, the MIDlet will be in the Destroyed state when destroyApp() returns. If this boolean is false, however, the MIDlet can request not to enter the Destroyed state by throwing a MIDletStateChangeException.
9. Finally, a call to midlet.resumeRequest() will tell the AMS that the MIDlet is interested in entering the Active state. While it's idle in most ways, a MIDlet in the Paused state may handle asynchronous events such as timers and callbacks.

For more details click here

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.