Mobile Application Development
Aileen Pierce

ANDROID LIFECYCLE

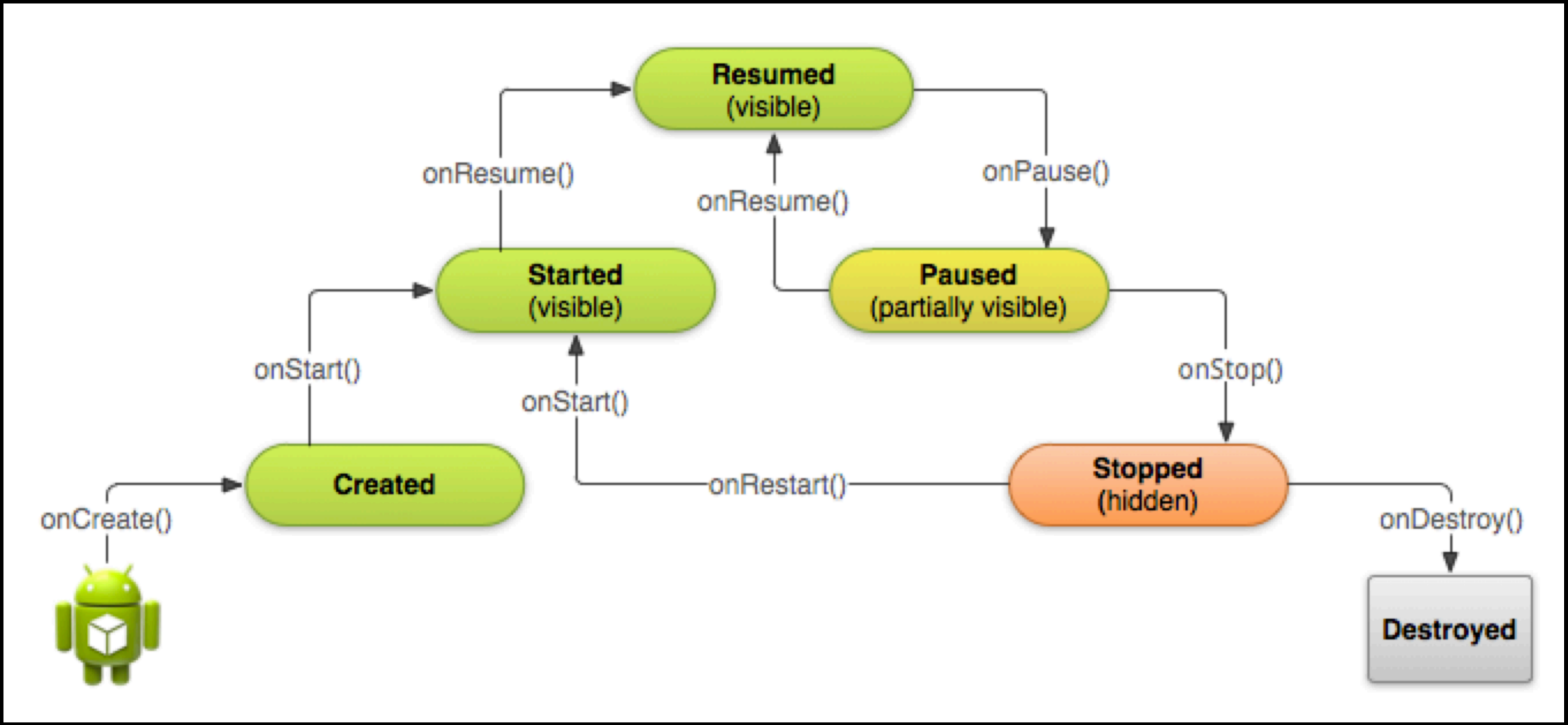
Android Activity States

- Created
  - An app's main activity has been launched
- Started
  - Activity is becoming visible
- Resumed
  - App is visible in the foreground and the user can interact with it, the running state

Android Activity States

- Paused
  - Activity is partially visible, another activity is in the foreground
  - When paused it does not receive user input and doesn’t execute any code
- Stopped
  - Activity is in the background and no longer visible
- Destroyed
  - All app processes have ended

Android Lifecycle



Android Lifecycle

- Every app has an activity that is declared as the launcher activity
  - This activity is the main entry point to the app
  - This is declared in the AndroidMainfest.xml file
  - Must have an intent-filter that includes the MAIN action and LAUNCHER category

```
<activity android:name=".FindCoffeeActivity" >
    <intent-filter>
        <action
android:name="android.intent.action.MAIN" />

        <category
android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
```

Android Lifecycle Methods

- **`onCreate()`** – activity is first created
  - Good place to do setup
  - calls **`setContentView()`** to declare the layout and configure the UI
- **`onRestart()`** – activity was stopped and is about to restart
- **`onStart()`** – activity is becoming visible
  - Followed by **`onResume()`** if the activity comes into the foreground
  - Followed by **`onStop()`** if the activity is made invisible
- **`onResume()`** – activity is in the foreground

Android Lifecycle Methods

- **`onResume()`** – activity is in the foreground
- **`onPause()`** – activity is no longer in the foreground, another activity is starting
  - Followed by **`onResume()`** if the activity returns to the foreground
  - Followed by **`onStop()`** if the activity becomes invisible
- **`onStop()`** – activity is no longer visible
  - Followed by **`onRestart()`** if the activity becomes visible again
  - Followed by **`onDestroy()`** if the activity is going to be destroyed
- **`onDestroy()`** – activity is about to be destroyed

Pausing and Resuming

- When an activity in the foreground becomes partially obscured it becomes paused
  - Stop ongoing actions
  - Commit unsaved changes (if expected)
  - Release system resources
- As long as an activity is partially visible but not in focus it remains paused
- When the user resumes the activity you should reinitialize anything you released when it paused

Stopping and Restarting

- If an activity is fully obstructed and not visible it becomes stopped
  - Switches to another app
  - Another activity is started
  - User gets a phone call
- The activity remains in memory while stopped
- If the user goes back to the app, or uses the back button to go back to the activity, it is restarted

Destroying and Recreating

- An activity is destroyed when the system decides it's no longer needed
  - User presses the back button
  - Hasn't been used in a long time
  - Needs to recover memory
- A change in device orientation results in the activity being destroyed and then recreated so the new device configuration can be loaded

Destroying and Recreating

- If the system destroys the activity due to system constraints, then although the actual Activity instance is gone, the system saves some state data in a Bundle object
  - If the user navigates back to that activity, a new instance of the activity is recreated using the data saved in the Bundle object

Destroying and Recreating

- As an activity begins to stop, the system calls the **`onSaveInstanceState(Bundle)`** method to save the current state of the activity
  - By default the Bundle instance saves information about each View object in your activity layout
  - To save additional data use the **`savedInstanceState.putxxx()`** methods
- To restore your saved state data when the activity is being recreated you can access the Bundle in the **`onCreate()`** method using the **`savedInstanceState.getxxx()`** methods