Ser423 Mobile Systems

Unit 2. Life-Cycles for iOS and Android Apps
2.a Outcomes and References App Life-Cycles

2.a.1 What you should take away, and references

• **Outcomes**
  - To know and be able to explain the basic components of iOS an Android apps, including the components representing the View, Controller, and underly Model on each platform. How those components are created and the form they take in the bundle that gets installed on a mobile device.
  - To be able to describe the differences between app life-cycle and a view controller’s life-cycle
  - Know the states through which an app and its views pass as execution begins, as focus passes to different views or different apps, and methods available to the developer for coding well-behaved apps.

• **References**
  - [Busy Coders Guide to Android](#) on: Activities and Their Lifecycles.
  - Android Reference: [The Activity Lifecycle](#)
  - iOS Reference: [The App Life Cycle](#)
  - iOS Reference: [UIViewController Docs](#). Section on View Management.
2.b iOS App Components and Life Cycle

2.b.1 Model, View, and Controller App Design
2.b.2 iOS Key Objects

- **UIApplication object** - Is the basis for App scheduling, managing foreground to/from background, App creation and destruction, and manages both the event loop and other high-level app behaviors.
  - Apps may receive push notifications through the UIApplication object.

- **App Delegate object** - Is the focus of an App’s code, handling initialization, state transitions (pause, resume), high-level app events, and **Model** setup.

- **Documents and Data Model** - The underlying data and content for an App. May be files, or a database.
  - Apps can use document objects, which are custom subclasses of **UIDocument** which provides specific services for managing content, including using **iCloud**.

- **Views** and **View Controllers** - **UIViewController** is the base class for view controllers, provides default functionality for loading, and rotating Views. Views contain buttons, text fields, pickers, and lists/tables.
2.b.3 Execution States and App Delegate Methods

- States of an iOS App
  - **Not Running** - Has not been launched, or has been terminated.
  - **Inactive** - Is in the foreground (visible), but is not receiving events.
  - **Active** - Is running in the foreground and is receiving events.
  - **Background** - Is in the background (not visible) and executing code.
  - **Suspended** - Is in the background and is not executing code. Automatically moved to this state by the system.

- AppDelegate methods called by the System, which you may implement.
  - **willFinishLaunching** -- App’s first opportunity to execute launch code.
  - **didFinishLaunching** -- App’s final chance to initialize before display.
  - **applicationDidBecomeActive** -- App is about to become **foreground**.
  - **applicationWillResignActive** -- transition away from being **foreground**.
  - **applicationDidEnterBackground** -- now running in **background**
  - **applicationWillEnterForeground** -- moving from **background** to **fore**.
  - **applicationWillTerminate** -- Not called if **suspended**, only when the app is being **terminated**
2.b.4 iOS View Controller Transitions and Methods

- `viewWillAppear:`
- `viewDidAppear:`
- `viewWillDisappear:`
- `viewDidDisappear:`

States:
- Appearing
- Appeared
- Disappearing
- Disappeared
2.b.5 iOS View Controller Life Cycle Methods

- **viewDidLoad()** -- not shown in figure above. Called when the view is loaded into memory. It is only called once in the life of a view controller. It is where the app should perform view initialization, which may include model initialization, for the launch view. First call `super.viewDidLoad()`

- **viewWillAppear(_):** Called after `viewDidLoad`, but before the view appears. `viewWillAppear` is called the first time the view will appear and (unlike `viewDidLoad`) every subsequent time the view appears. For example, called as the result of a back button, tab selection in a tab bar, or on a modal dialog completion. `viewDidAppear` is called after the view appears.

- **viewWillDisappear(_):** Called just before the view disappears, usually when the user navigates away from the view. Great place to hide the keyboard, cancel network requests, save the state, or cancel existing timers. `viewDidDisappear` is called after the view is gone.

- From Apple docs for `UIViewController` class: “Not all ‘will’ callback methods are paired with only a ‘did’ callback method. You need to ensure that if you start a process in a ‘will’ callback method, you end the process in both the corresponding ‘did’ AND the opposite ‘will’ callback method.”
2.b.6 Swift for iOS View Controller Life Cycle Method

• These are not typically included in view controller templates and must be added. For example add:

```swift
override func viewWillDisappear(_ animated: Bool) {
    super.viewWillDisappear(animated)
    if self.isBeingDismissed || self.isMovingFromParentViewController {
        print(“view has been popped and will disappear“)
    }
}
```
2.c Android Activity Life Cycle

2.c.1 Android Activity Life Cycle Methods

- Android Activity and its sub-classes contain the following life-cycle methods which are called by the system as defined below and in reference documents
  - onCreate()
  - onStart()
  - onResume()
  - onPause()
  - onStop()
  - onDestroy()
2.c.2 Android Activity Life Cycle Transitions