ATLS 4120: Mobile Application Development
Week 5: App Lifecycle

Project Files
The single-view app template creates the following files:

- ViewController.swift
  - view controller implementation file
- Main.storyboard
  - Storyboard file for your views
- Info.plist
  - Application configuration property list
- AppDelegate.swift
  - Every iOS application has exactly one instance of UIApplication which serves as the centralized point of control and coordination for iOS apps.
  - UIApplication is a standard part of the UIKit, and it does its job mostly behind the scenes
    - creates the UIWindow object which coordinates the presentation of views on a screen
    - ensures your app interacts properly with the system and other apps
    - responsible for the application's run loop
    - It process events and handle updates to view-based interfaces.
    - As the user interacts with a device, events related to those interactions are generated by the system and the UIApplication object is the first object to receive the event and make the decision about what needs to be done.
      - A touch event is usually dispatched to the main window object, which in turn dispatches it to the view in which the touch occurred.
    - maintains a list of all the windows(UIWindow objects) open so it can retrieve the UIView objects.
    - handles application-level functionality such as routing input to the appropriate controller class.
  - Implements the **UIApplicationDelegate** protocol which works alongside the app object to interact with the system and with other apps.
  - The application delegate lets us do things at certain predefined times on behalf of the **UIApplication** class.
    - The delegate informs the application of runtime events such as application launch, low-memory warnings, termination, etc.
  - UIApplication and UIApplicationDelegate also allow you to manage behavior that is specific to the device. You can control application response to changes in interface orientation, and temporarily suspend incoming touch events.
  - The app delegate provides methods for each stage an app goes through which gives you a chance to respond to important changes
    - Stub methods for applicationDidThis and applicationWillDoThat
  - **application(_:didFinishLaunchingWithOptions:**)
    - Tells the delegate that the launch process is almost done and the app is almost ready to run

At certain times during an application's execution, UIApplication will call specific methods on its delegate so it's useful to understand what's going on behind the scenes of your app.

App Lifecycle
https://developer.apple.com/documentation/uikit/uiapplicationdelegate

App States
https://developer.apple.com/documentation/uikit/app_and_environment/managing_your_app_s_life_cycle

- Not running
  - Not yet launched or was terminated
  - When an app is launched it moves from the not running state to the active state
  - The state of apps on a freshly rebooted device
- Inactive
  - Running in the foreground but not receiving events
  - Usually in transition to another state.
- Active
  - Running and receiving input and events
  - Normal mode for foreground apps
- Background
  - In the background and executing code
  - Most apps enter this state briefly on their way to being suspended
- Suspended
  - In the background and not executing code
  - The system moves apps to this state automatically and does not notify them before doing so.
  - While suspended, an app remains in memory and holds on to its memory. When a user goes back to an app it will pick up where it left off.
  - When a low-memory condition occurs, the system may purge suspended apps without notice to make more space for the foreground app

App StateTransitions
State transitions are accompanied by a corresponding call to an app delegate method

- App launch
  - **application(_: willFinishLaunchingWithOptions:)**
    - The app's first chance to execute code at launch time
  - **application(_: didFinishLaunchingWithOptions:)**
    - Called after the app is launched but is still in the inactive state
    - An application can be launched by tapping its icon or in order to respond to a specific type of event
    - Good place for initialization
  - **applicationDidBecomeActive(_:)**
    - Called when the application comes to the foreground, including at launch time.
    - Use it to perform any tasks that are common to the two transitions.
    - The app's chance to prepare to run as the foreground app
- App launching into foreground
  - Once the app is in the foreground the UIViewController class loads the interface from the interface files and connects all the outlets and actions.
  - **viewDidLoad()**
    - Gets called when the view has been created and you are sure that all the outlets are in place.

        - Good place for initialization, populating the user interface with data, or other set up
        - It is also a good place where to start some background activity where you need to have the user interface in place at the end.
        - This method is called only **once** in the lifetime of a view controller, so you use it for things that need to happen only once.
    - **viewWillAppear()**
        - View is about to be put on the screen
        - Good place to update the user interface with data that might have changed while the view controller was not on the screen.
        - Don't do anything here that takes a long time
        - This method is called **every** time a view controller comes on screen.
    - **viewDidAppear()**
        - This method gets called after the view controller appears on screen.
        - You can use it to start animations in the user interface, to start playing a video or a sound, or to start collecting data from the network.
- Transitioning to the inactive state
    - Interruptions cause an app to go into an inactive state.
        - When an alert-based interruption occurs, such as an incoming phone call, the app moves temporarily to the inactive state so that the system can prompt the user about how to proceed.
        - When a user quits an app its process is not terminated, just moved to the background
    - The app and its objects are still in memory and are not recreated when the app is relaunched
    - **applicationWillResignActive(_:)** is called when an app is about to move from the active to inactive state
    - All ongoing tasks should be paused
    - The app remains in this state until the user dismisses the alert. At this point, the app either returns to the active state or moves to the background state.
- Transitioning to the background
    - When the user returns to the Home screen, or the system launches another app, the foreground app transitions to the inactive state and then to the background state.
    - An app can also be launched into the background instead—usually to handle some type of background event. Instead of your app being made active, it enters the background state to handle the event and then is suspended shortly afterward.
        - When launching into the background, the system still loads your app's user interface files but it does not display the app's window.
    - **applicationWillResignActive(_:)**
        - The app is transitioning away from being in the foreground
    - **applicationDidEnterBackground(_:)**
        - The app is now running in the background and may be suspended at any time.
        - Where your app should free all resources that can be re-created later, save all user data, close network connections, and so on.
        - when applicationDidEnterBackground(_:) is called, you can safely assume that applicationWillResignActive(_:) has also been recently called.
        - It has about 5 seconds
- Transitioning to the foreground
    - **applicationWillEnterForeground(_:)**

        - The app is moving out of the background and back into the foreground, but it is not yet active.
        - should re-create whatever was torn down in applicationDidEnterBackground(_:), such as reloading user data, reestablishing network connections, and so on.
    - **applicationDidBecomeActive(_:)**
        - When applicationWillEnterForeground(_:) is called, you can assume that applicationDidBecomeActive(_:) will soon be called as well.
- Termination
    - **applicationWillTerminate(_:)**
        - The app is being terminated
        - Seldom, if ever, used
        - Perform any needed cleanup, save user data or state information
        - Has 5 seconds to clean up
        - Not called if the app is currently suspended
        - Can terminate due to memory constraints
        - Must be prepared for your app to be killed without any notification.
- Memory
    - **applicationDidReceiveMemoryWarning(_:)**
        - Sent low-memory notifications
        - Implement the **didReceiveMemoryWarning()** method in your view controller class to release views or other controllers
        - Not freeing up enough memory will result in **applicationWillTerminate(_:)** being called and your app terminating

There are also matching notifications for each state change.

<u>App refresh</u>
Open up one of your previous projects such as beatles (beatles reset).
Run the app and change some of the UI controls.
Return to the home screen and then go back into the app.
It keeps the data you had from last time.
Is that always the behavior you want?
What if you want it to reset?

Create a method in ViewController.swift that will reset the UI.

```
func refreshUI(){
    titleLabel.text="The Beatles"
    imageControl.selectedSegmentIndex = -1
    capitalSwitch.isOn = false
    beatlesImage.image=UIImage(named: "Beatles_Abbey_Road")
}
```

Now go into AppDelegate.swift.
Look at the class reference for `UIApplicationDelegate`
The method `applicationWillEnterForeground(_:)` is called as part of the transition from the background to the inactive state.
That's a great place to reset fields as the app comes from the background to the foreground.

```
func applicationWillEnterForeground(_ application: UIApplication!) {
    // Called as part of the transition from the background to the
```

```
inactive state; here you can undo many of the changes made on entering the
background.
        let vc = application.keyWindow!.rootViewController as! ViewController
        vc.refreshUI()
    }
```

The first line is defining an instance of our ViewController class by accessing the applications root view controller(type UIViewController) and casting it as ViewController, a subclass of UIViewController. Then we call the refresh() method.

Also works to reset these in
`func applicationDidBecomeActive(_:)` This method is called to let your application know that it moved from the inactive to active state.