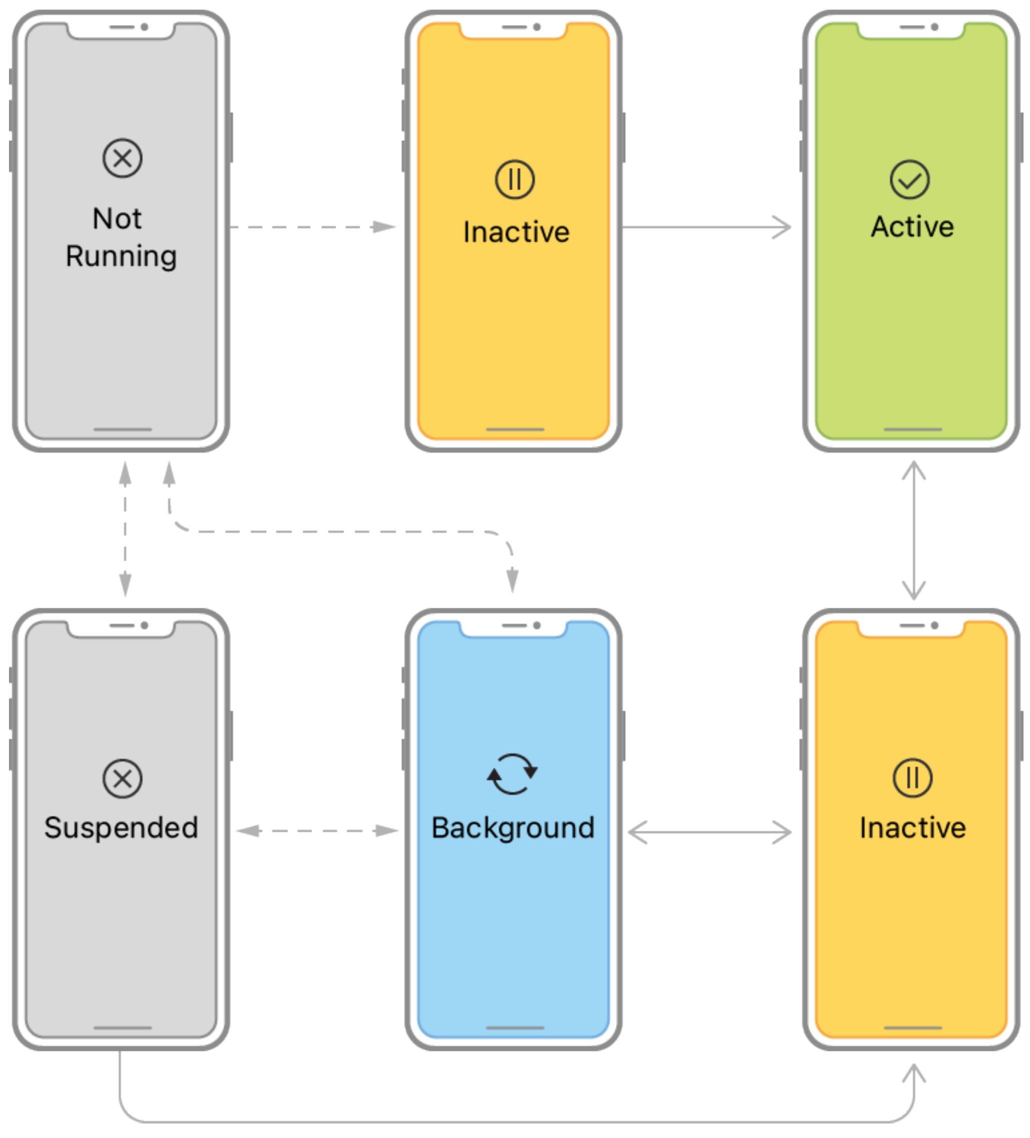
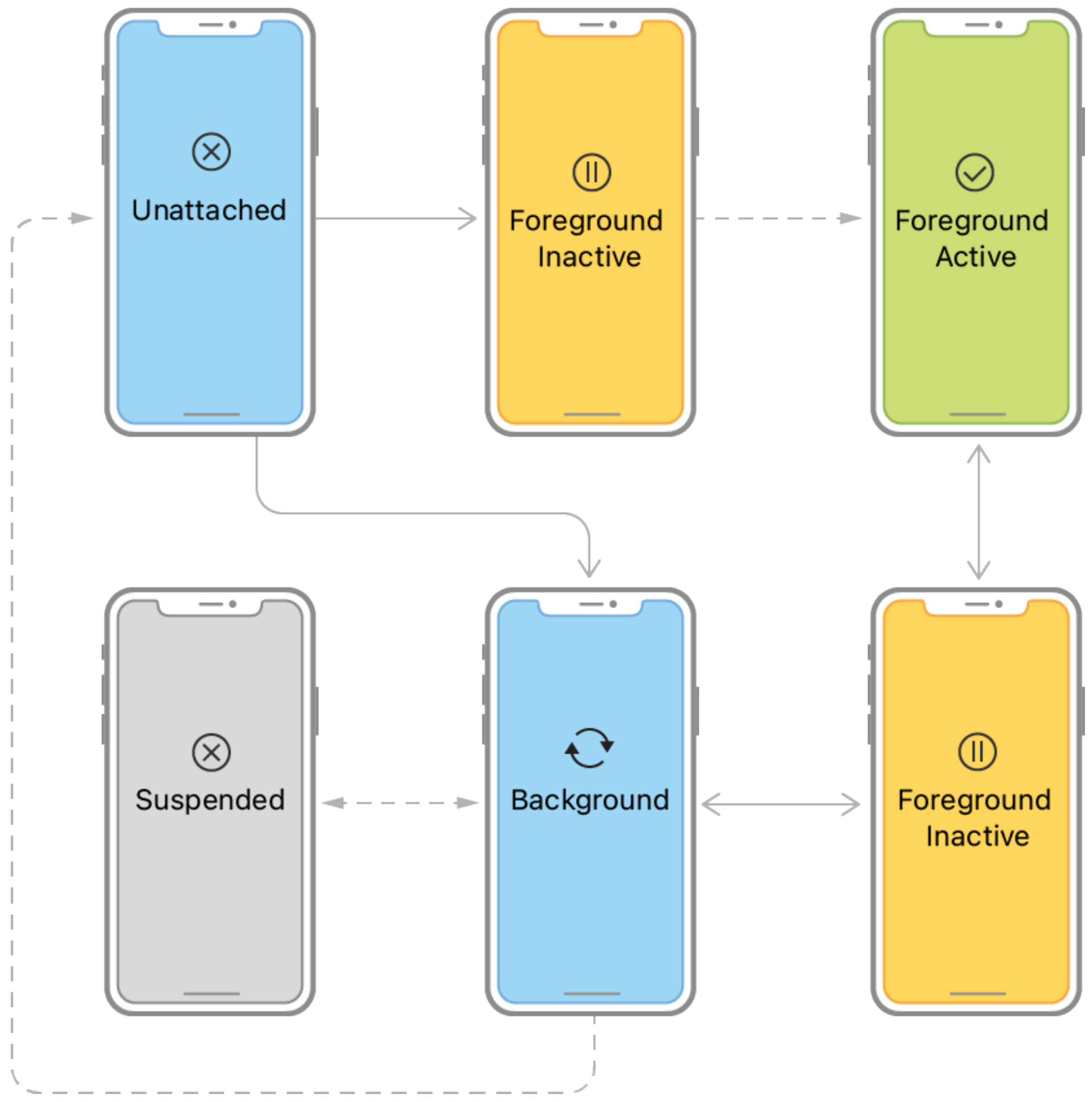
App Lifecycle

UIKit and SwiftUI Lifecycle Events

App Lifecycle



Before iOS 13



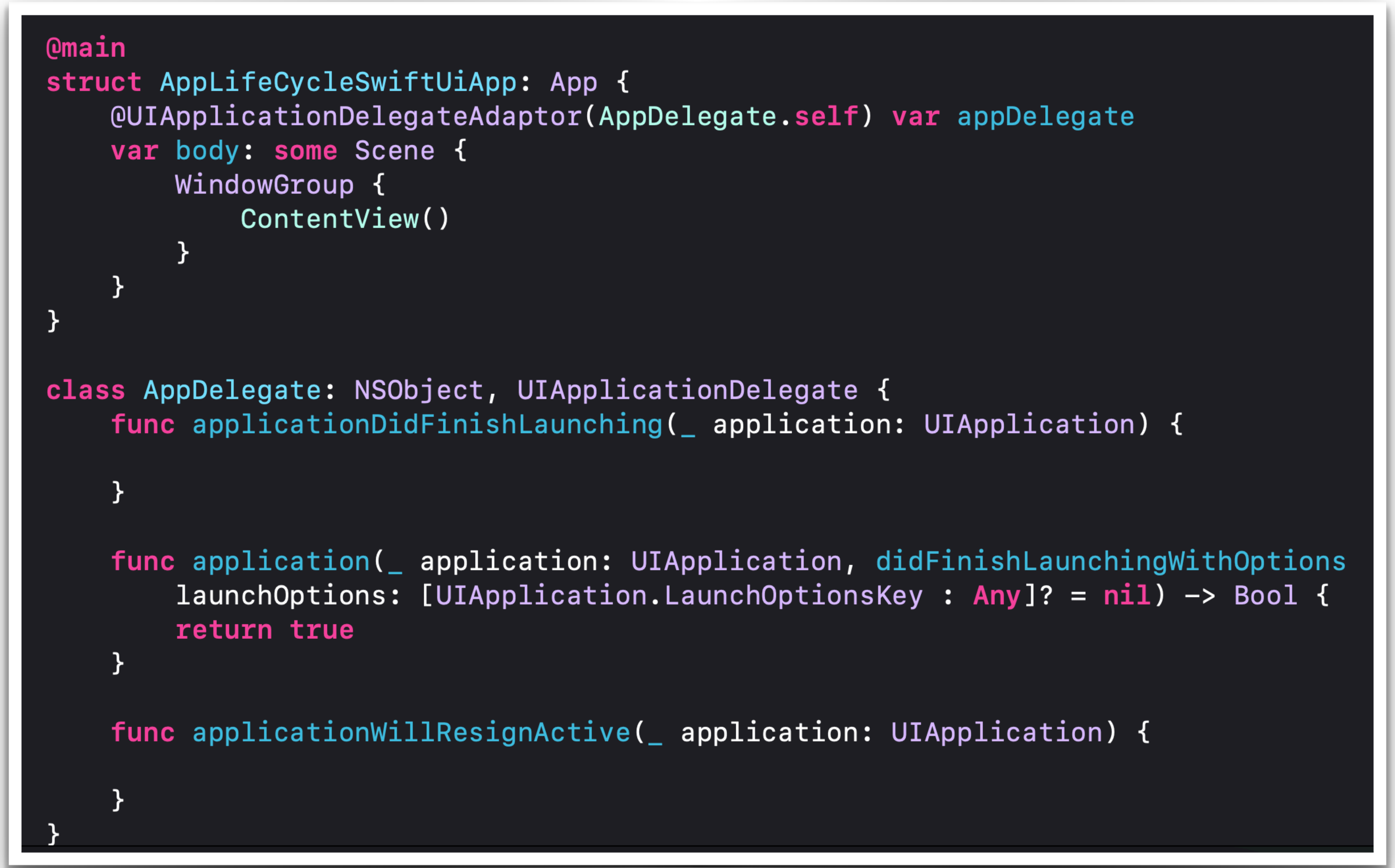
After iOS 13

- Solid Line - User initiated action
- Dashed line - System initiated action

AppDelegate vs SceneDelegate

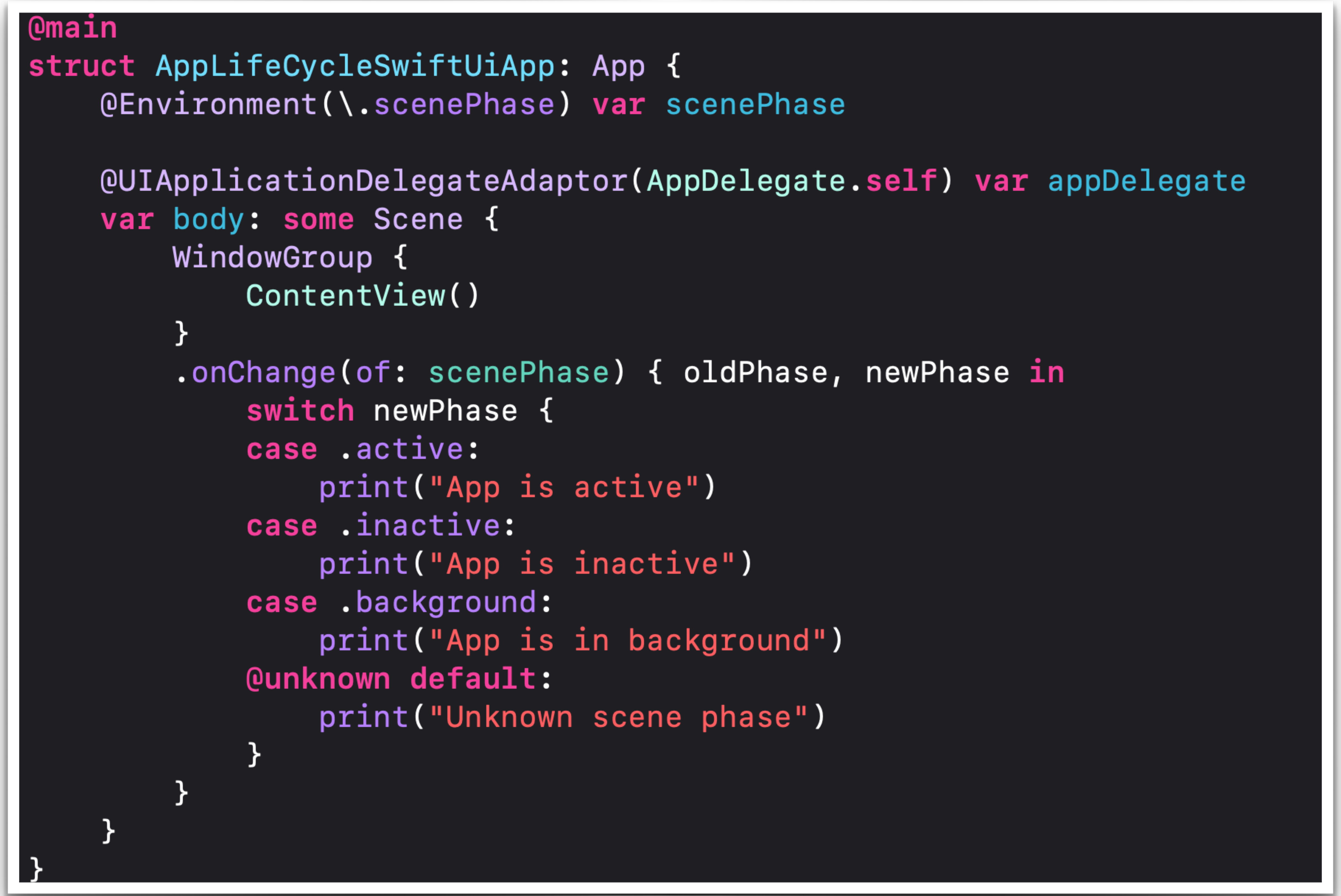
AppDelegate Methods	SceneDelegate Methods	Description
application(_:didFinishLaunchingWithOptions:)		Called when the app launches. Used for app-wide setup
applicationWillTerminate(_:)		Called when the app is about to terminate.
	scene(_:willConnectTo:options:)	Called when a new scene session is being created. Used for scene-specific setup
	sceneDidDisconnect(_:)	Called when a scene is being released by the system.
applicationDidBecomeActive(_:)	sceneDidBecomeActive(_:)	Called when the app/scene becomes active.
applicationWillResignActive(_:)	sceneWillResignActive(_:)	Called when the app/scene is about to become inactive.
applicationDidEnterBackground(_:)	sceneDidEnterBackground(_:)	Called when the app/scene enters the background.
applicationWillEnterForeground(_:)	sceneWillEnterForeground(_:)	Called when the app/scene is about to enter the foreground.
application(_:handleEventsForBackgroundURLSession:completionHandler:)	scene(_:handleEventsForBackgroundURLSession:completionHandler:)	Handles events for background URL sessions
application(_:configurationForConnecting:options:)`		Configures and returns a UISceneConfiguration object.
application(_:didDiscardSceneSessions:)		Called when the user discards a scene session.

SwiftUI using AppDelegate methods



```swift
@main
struct AppLifeCycleSwiftUiApp: App {
    @UIApplicationDelegateAdaptor(AppDelegate.self) var appDelegate
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}

class AppDelegate: NSObject, UIApplicationDelegate {
    func applicationDidFinishLaunching(_ application: UIApplication) {

    }

    func application(_ application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {

    }
}
```

Lifecycle states in SwiftUI



```swift
@main
struct AppLifeCycleSwiftUiApp: App {
    @Environment(\.scenePhase) var scenePhase

    @UIApplicationDelegateAdaptor(AppDelegate.self) var appDelegate
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .onChange(of: scenePhase) { oldPhase, newPhase in
            switch newPhase {
            case .active:
                print("App is active")
            case .inactive:
                print("App is inactive")
            case .background:
                print("App is in background")
            @unknown default:
                print("Unknown scene phase")
            }
        }
    }
}
```

Real-life use cases for handling app states

App state	Action/Behaviour
.active	Resume game, media playback or ongoing tasks
	Log when the app becomes active for usage analytics.
	Resume or initiate network requests like downloads/uploads.
	Check if data syncing is needed when the app is back.
	Restart heavy tasks or processes that were stopped.
	Unlock the app or remove screen blur (for sensitive apps).
.inactive	Save the current state or data.
	Blur sensitive information or lock the app when not in use.
	Pause media playback, games, or ongoing processes.
.background	Stop GPS tracking, processing, or machine learning tasks.
	Continue syncing or uploading tasks in the background.