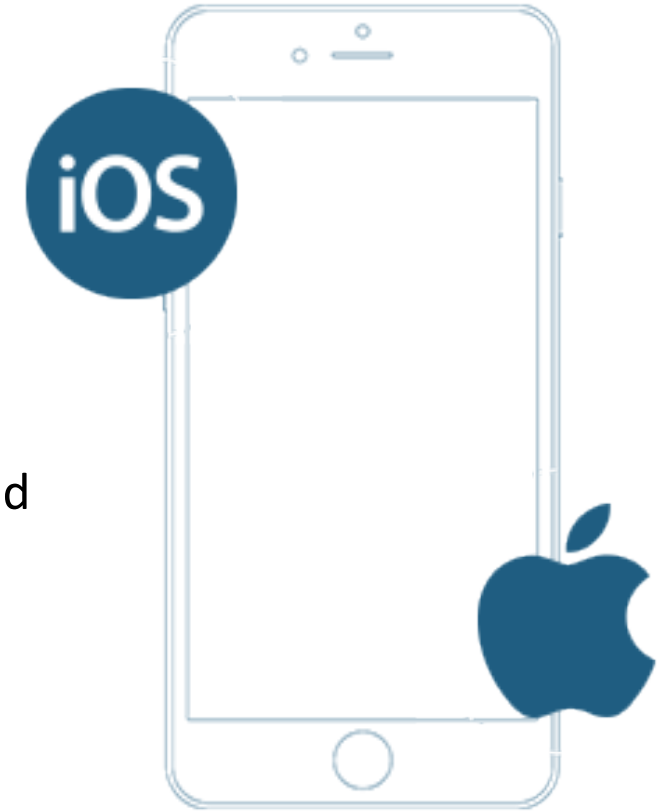
Developing IOS apps using objective-c



Presented By
Abdelrahman Sayed

Java™ Education and Technology Services

Invest In Yourself , Develop Your Career

Java™ Education and Technology Services

Lecture One

Objective-C

Agenda

- iOS Application Architecture.
- Model View Controller (MVC).
- iOS Project Structure and Design.
- Application States.
- Life Cycle.
- IBOutlets and IBActions.
- Attributes of @property

Java™ Education and Technology Services

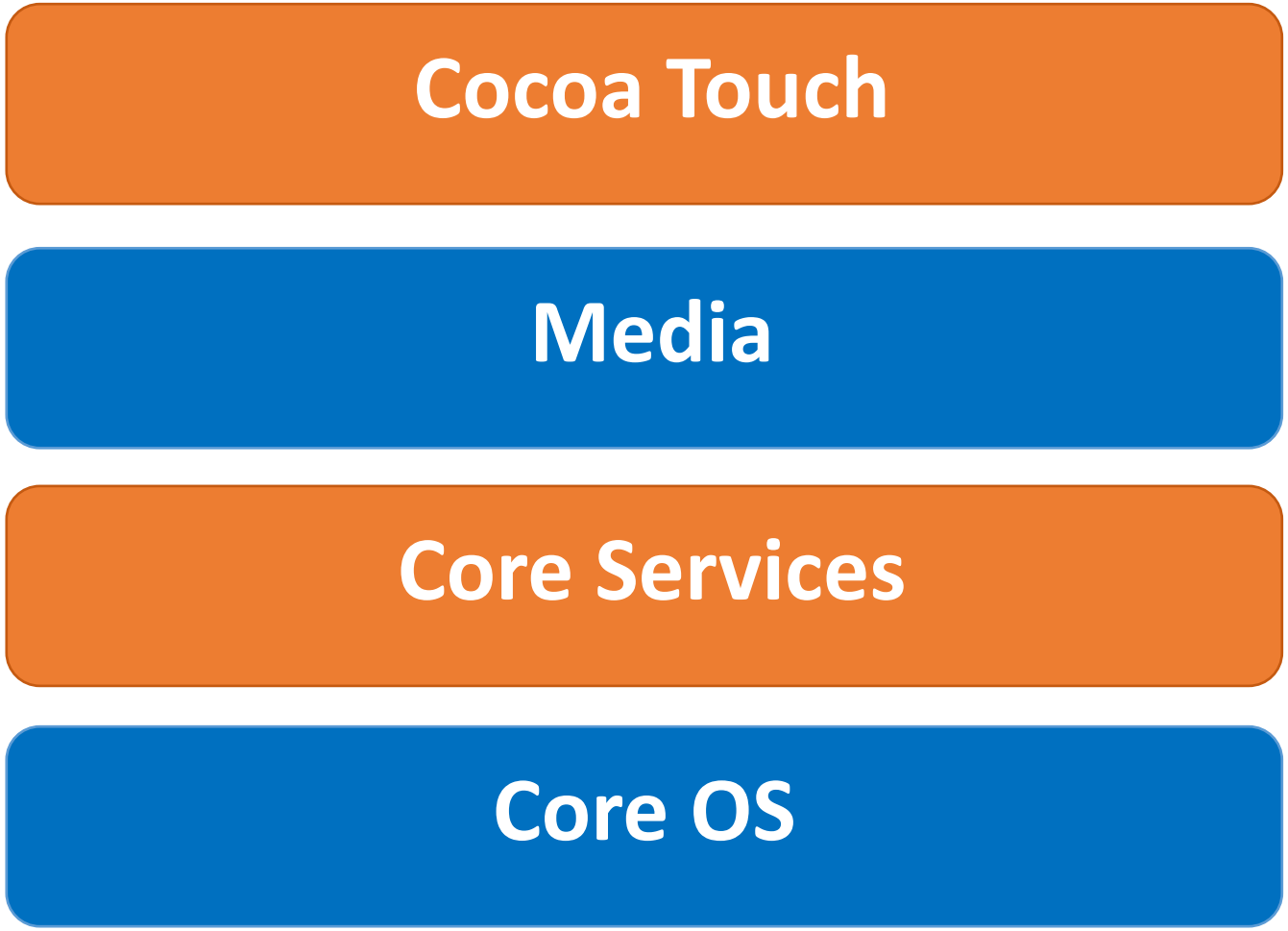
Objective-C

Java™ Education and Technology Services

iOS Application Architecture

Objective-C

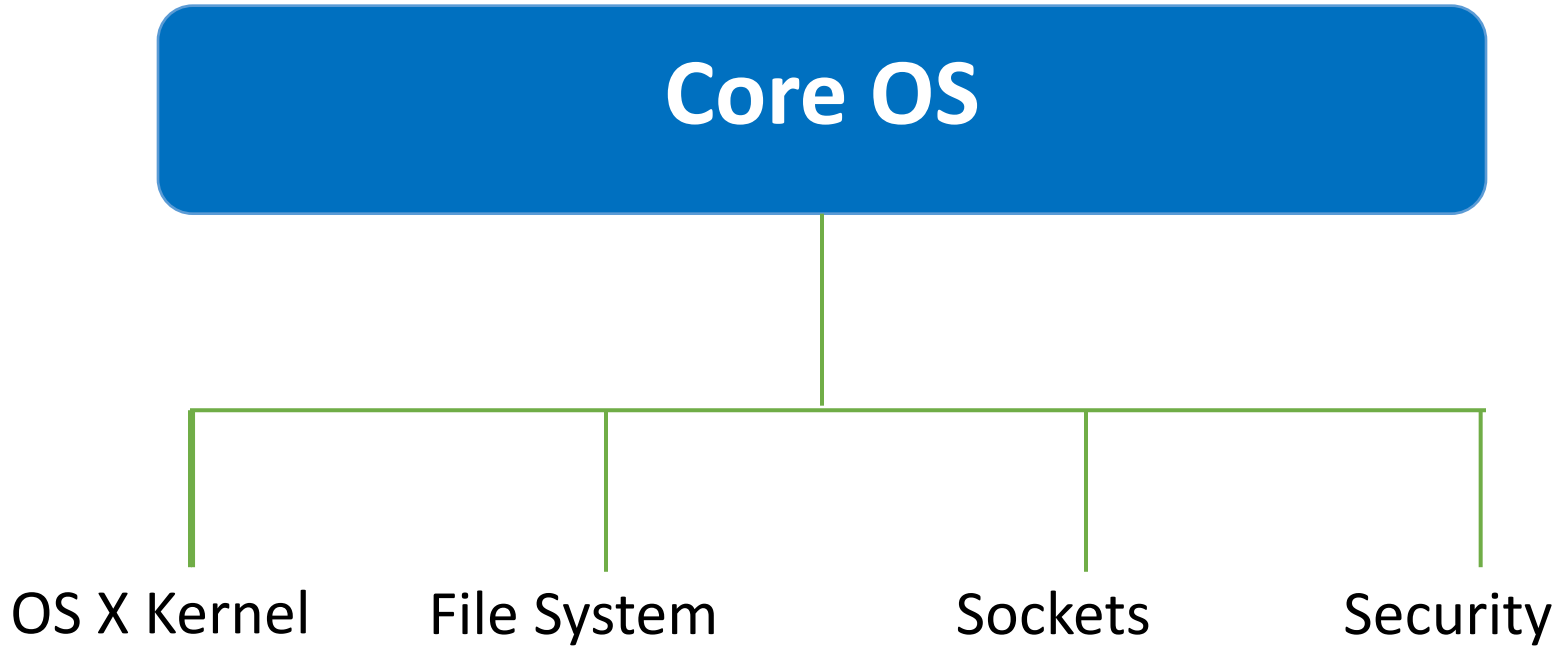
Architecture



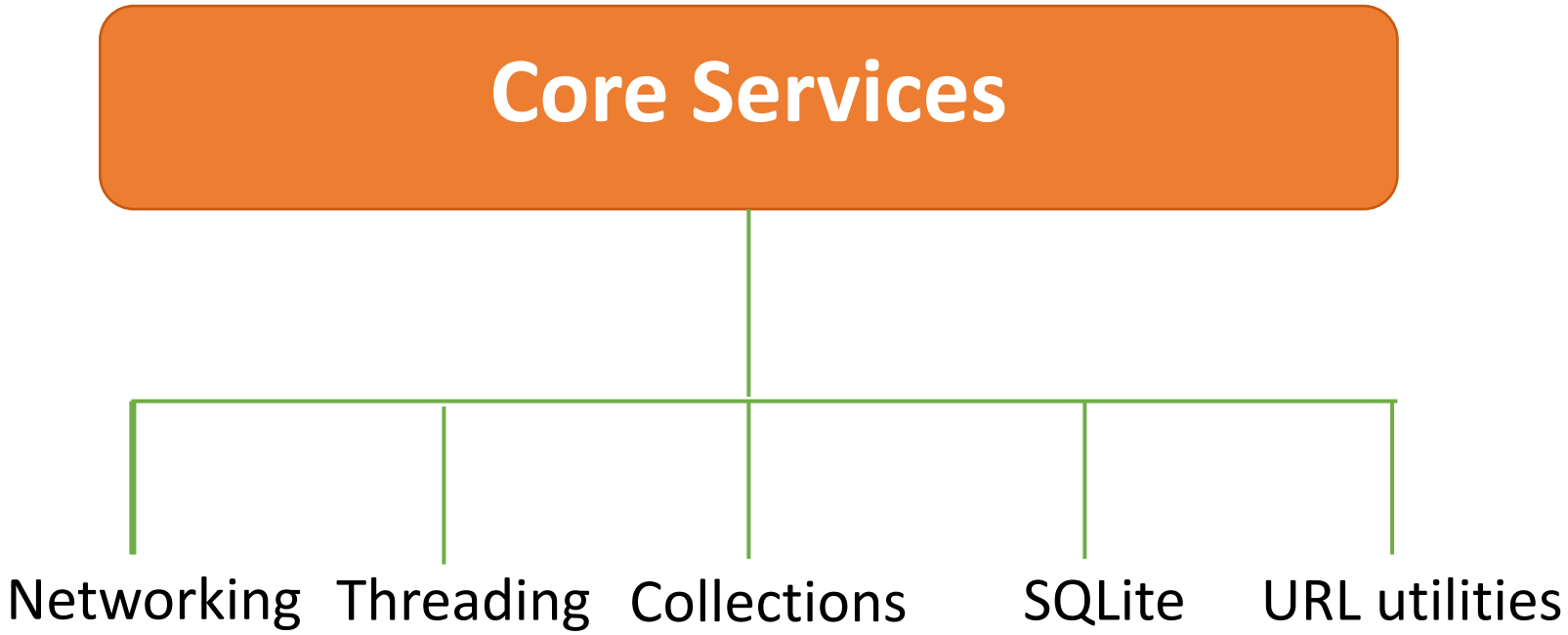
Java™ Education and Technology Services

Objective-C

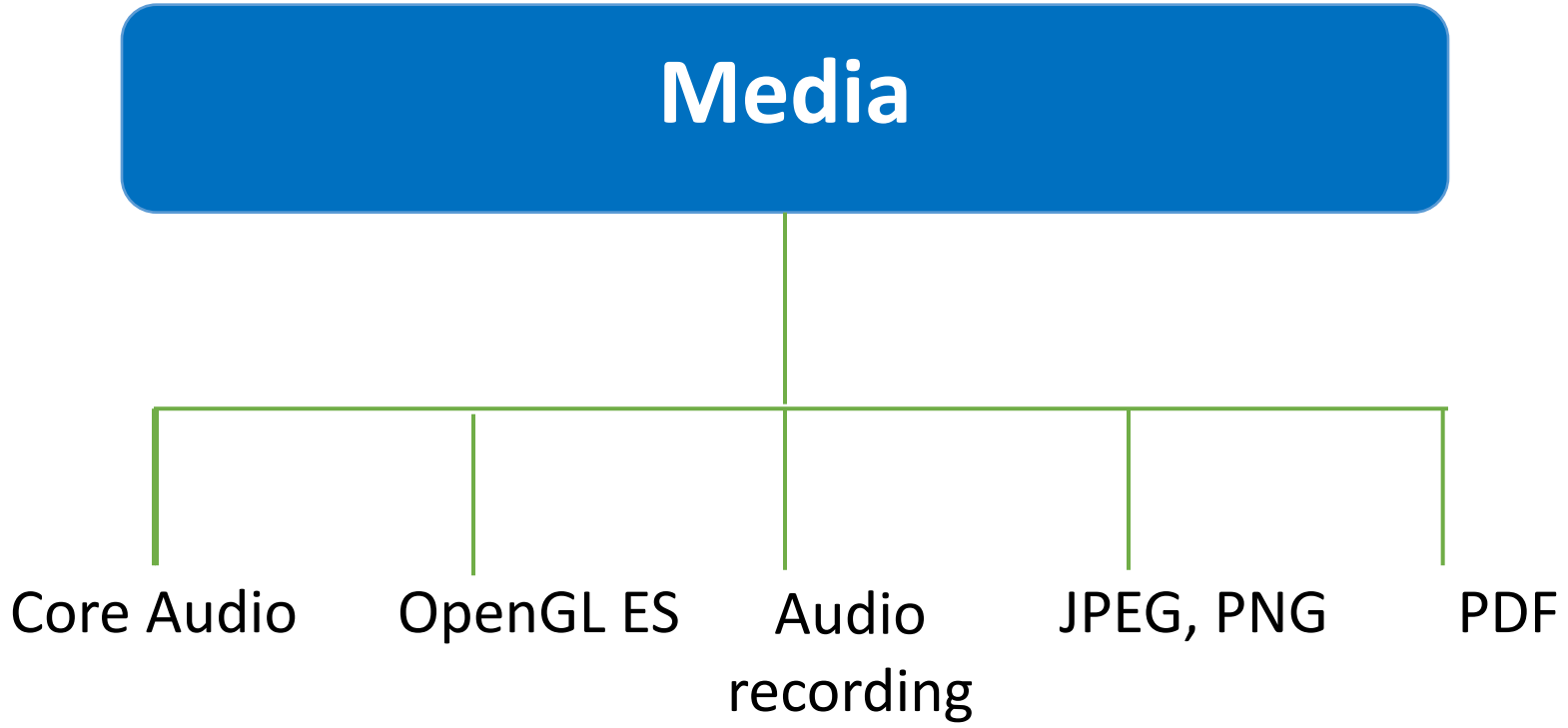
Core OS



Core Services



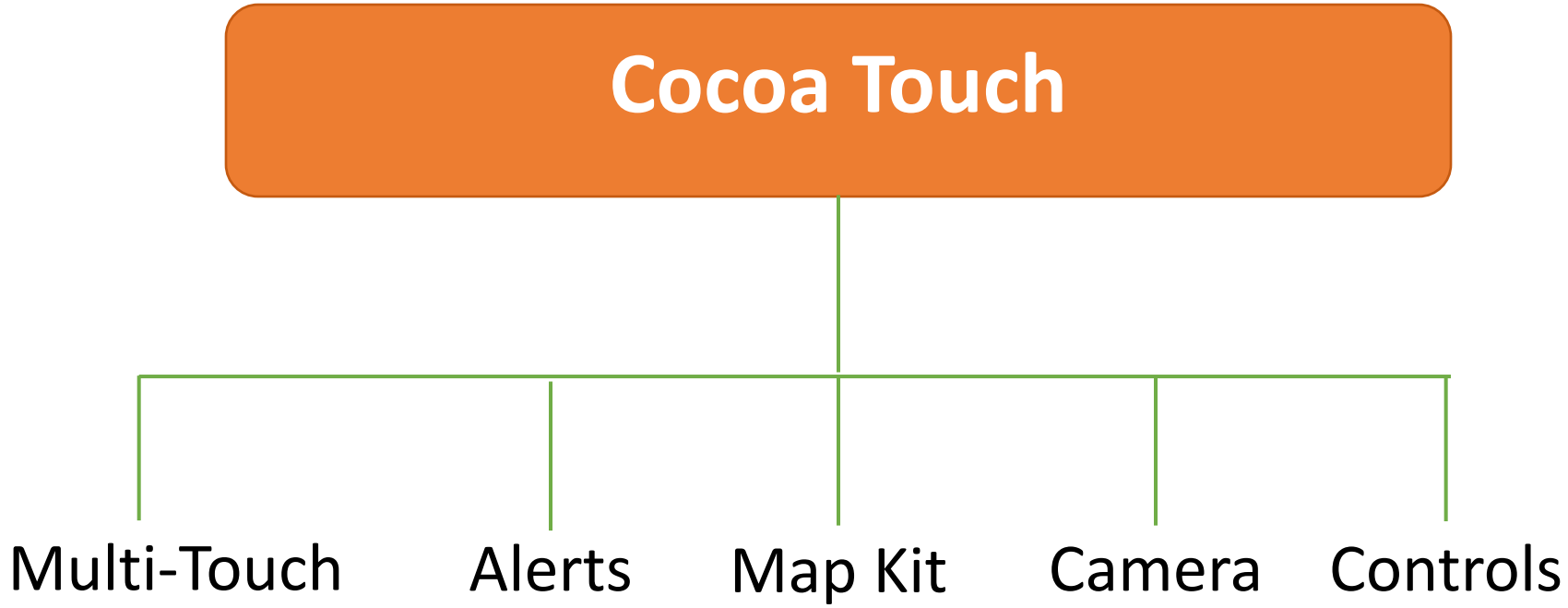
Media



Java™ Education and Technology Services

Objective-C

Cocoa Touch



Objective-C

Java™ Education and Technology Services

Model View Controller
MVC

Objective-C

MVC

- It gives you the ability to change business rules without affecting GUI

- Also GUI could be changed without affecting business

Model

- What your application do
- It’s not connected directly to the UI
- Many different UIs could have the same model

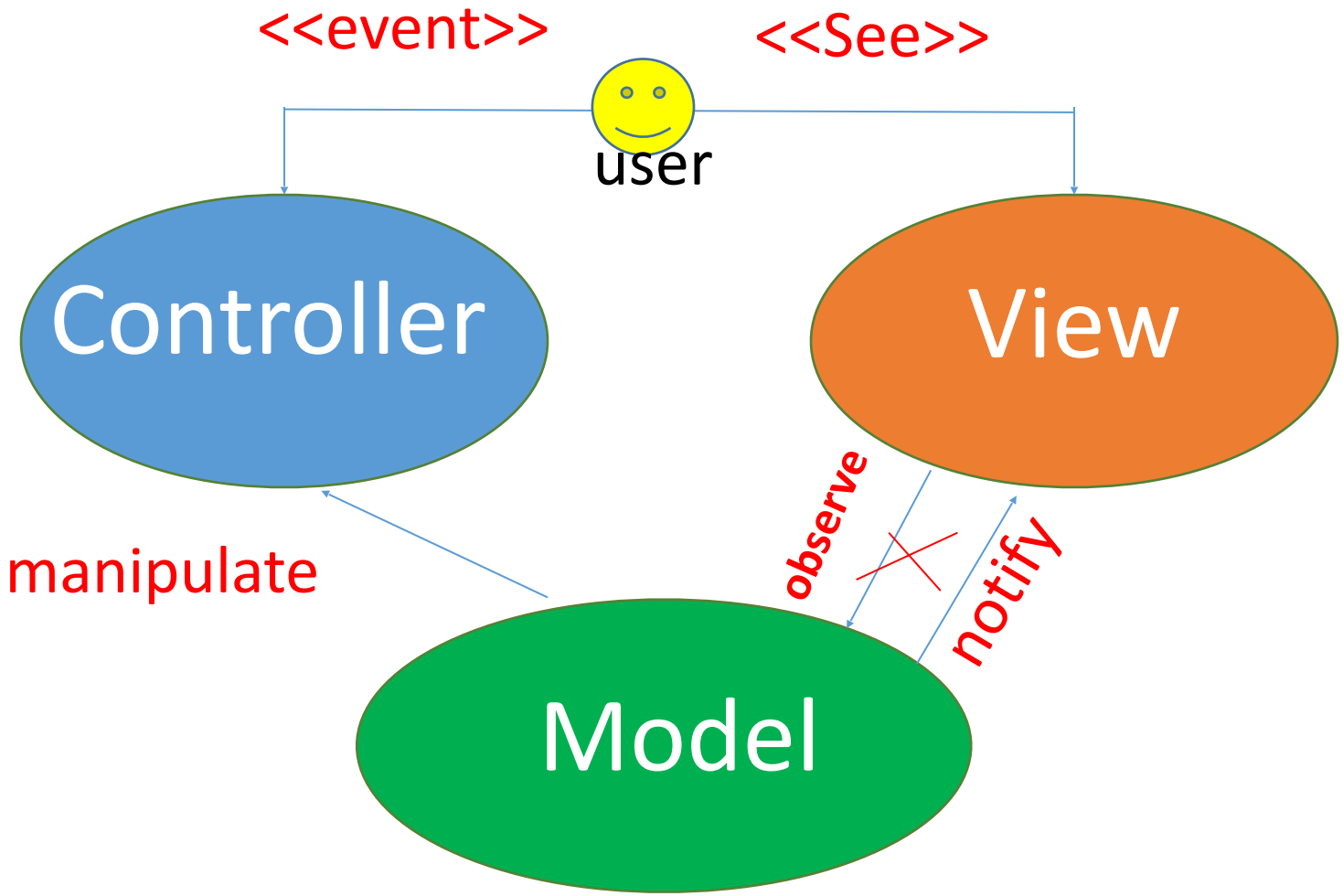
View

- The representation of the application to the user
- Should not hold any data
- Enables the user to interact with data

Controller

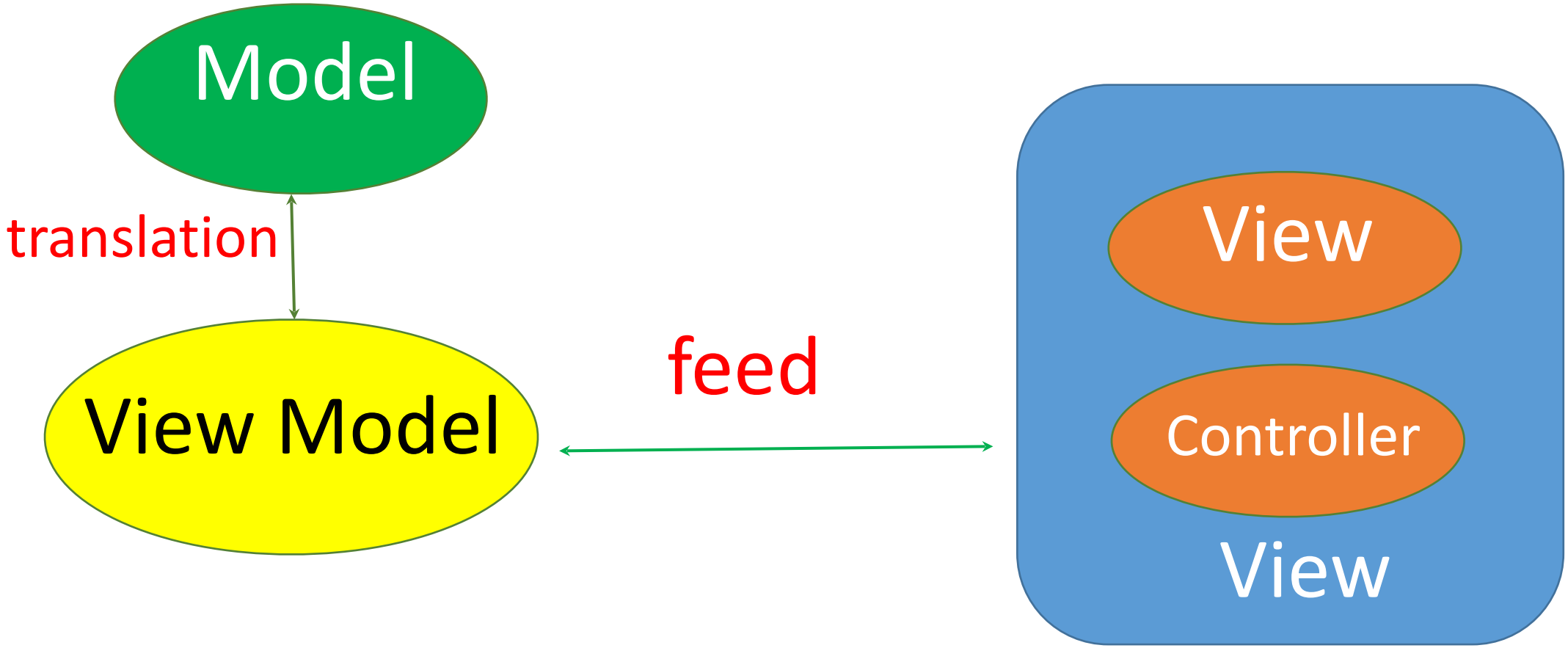
- It is the communicator between the model and view
- Updates model when view requires
- Updates view when model changes

Intercommunications

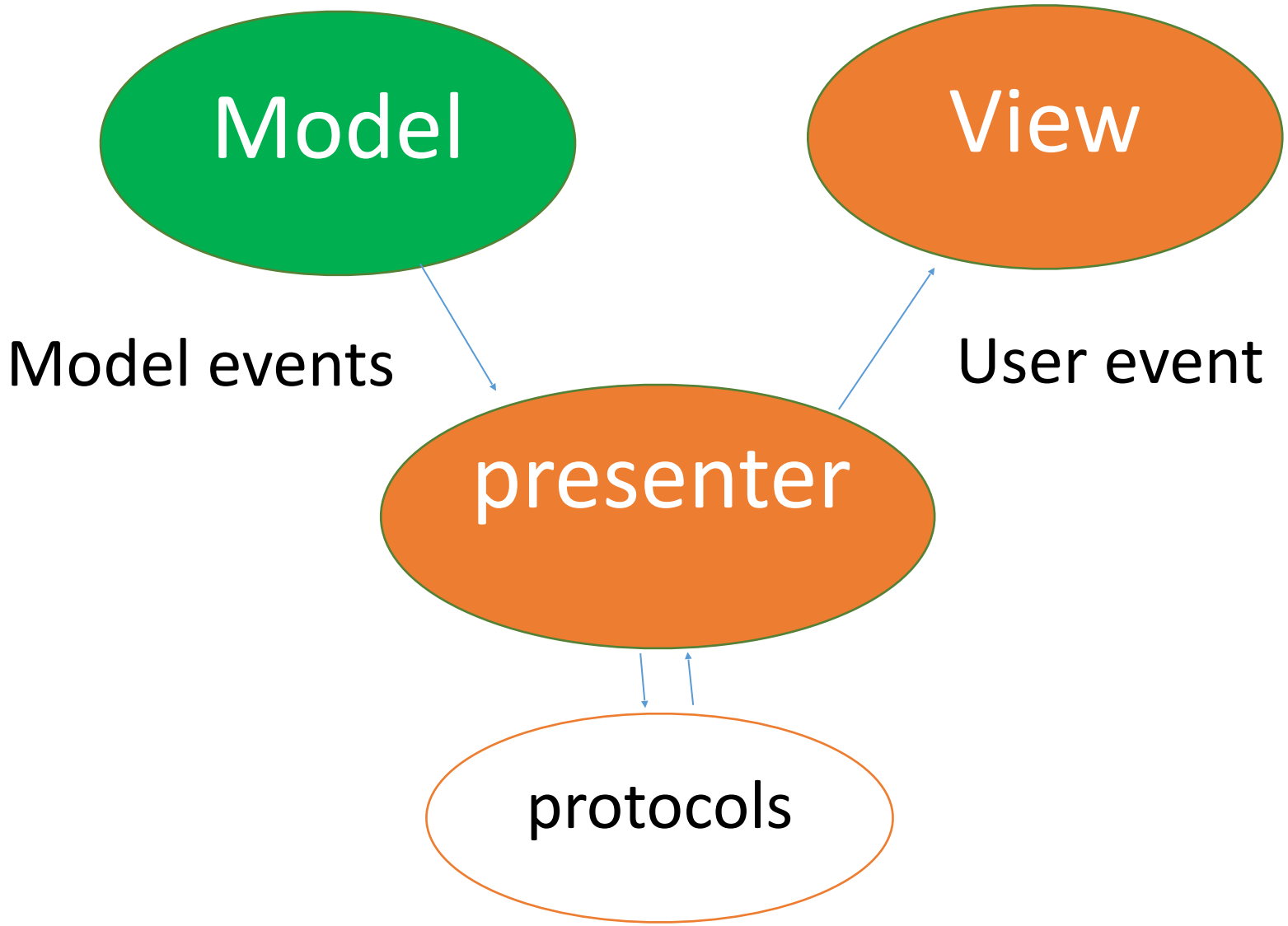


Objective-C

MVVM



MVP



Java™ Education and Technology Services

Objective-C

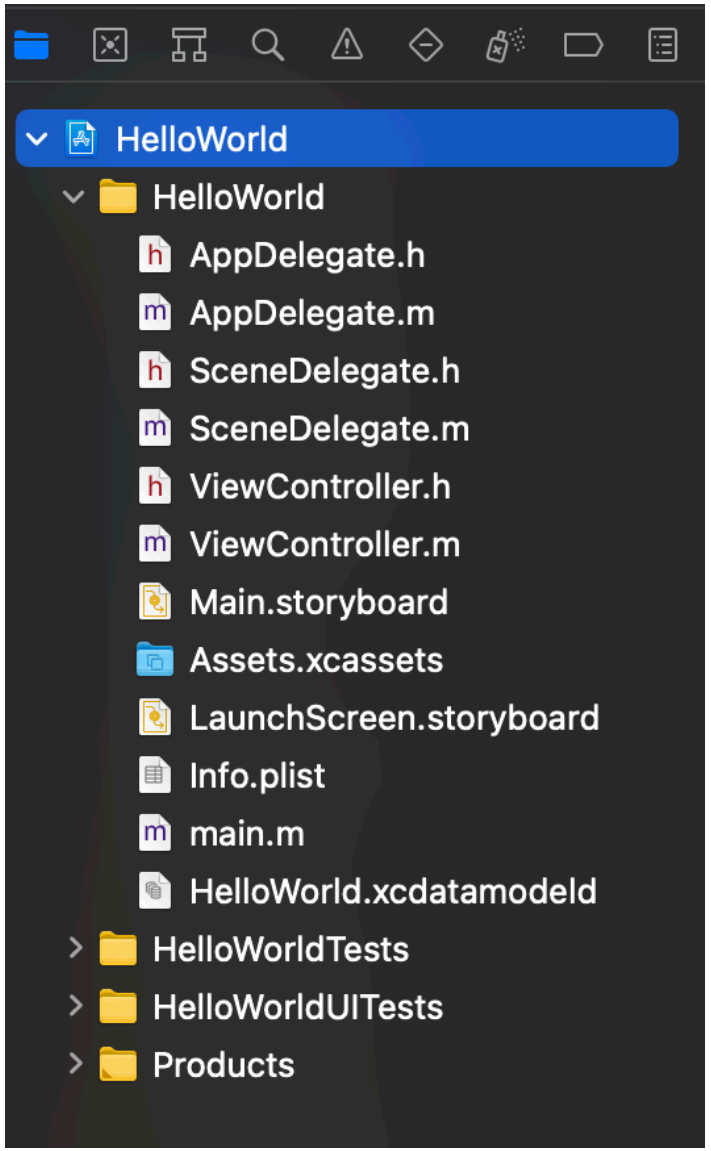
Hello World Demo

Java™ Education and Technology Services

iOS Project Structure and Design

Objective-C

Main Files


HelloWorld
HelloWorld
AppDelegate.h
AppDelegate.m
SceneDelegate.h
SceneDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
main.m
HelloWorld.xcdatamodeld
HelloWorldTests
HelloWorldUITests
Products

Main Files

- Main.storyboard
- ViewController.h
- ViewController.m
- AppDelegate.h
- AppDelegate.m
- SceneDelegate.h
- SceneDelegate.m
- LaunchScreen.storyboard

Main.storyboard

- It’s used to build your application’s user interface (View in MVC) using drag and drop
- No code is needed to build the user interface
- It’s created by the IB
- Interface Builder (IB) was an associated application with XCode 3.2, while in XCode 4 it’s one of its features
- It generates events automatically during runtime which will be sent to the associated method on the View Controller

Main Files

- Main.storyboard
- ViewController.h
- ViewController.m
- AppDelegate.h
- AppDelegate.m
- SceneDelegate.h
- SceneDelegate.m
- LaunchScreen.storyboard

Java™ Education and Technology Services

Objective-C

ViewController

- It acts as the controller in the MVC model
  (Communicates between Model and View)

- A view controller manages a set of views that make up a portion of your application's user interface.

- It contains:
  - Reference of each view component (outlet)
  - An action for each event

Java™ Education and Technology Services

Objective-C

Main Files

- Main.storyboard
- ViewController.h
- ViewController.m
- AppDelegate.h
- AppDelegate.m
- SceneDelegate.h
- SceneDelegate.m
- LaunchScreen.storyboard

AppDelegate

- It’s the core class in the application
- It’s auto generated in the application
- Every iOS application must contain only one application delegate

Java™ Education and Technology Services

Objective-C

AppDelegate Cont.

- The AppDelegate is responsible for the application lifecycle and setup
- It’s responsible for handling critical system messages such as: applicationDidFinishLaunching method.

Java™ Education and Technology Services

Objective-C

Main Files

- Main.storyboard
- ViewController.h
- ViewController.m
- AppDelegate.h
- AppDelegate.m
- SceneDelegate.h
- SceneDelegate.m
- LaunchScreen.storyboard

SceneDelegate

- From iOS 13, the responsibilities of AppDelegate have been split between AppDelegate and SceneDelegate. This is the result of new multi-window support feature that is introduced with iPad-OS and that splits the work of AppDelegate into two

- The SceneDelegate is responsible for what is shown on the screen (Windows or Scenes) handle and manage the way your app is shown.

Main Files

- Main.storyboard
- ViewController.h
- ViewController.m
- AppDelegate.h
- AppDelegate.m
- SceneDelegate.h
- SceneDelegate.m
- LaunchScreen.storyboard

Launchscreen.stroyboard

- A launch screen appears instantly when your app starts up.
- The launch screen is quickly replaced with the first screen of your app, giving the impression that your app is fast and responsive.

Java™ Education and Technology Services

Application States

Objective-C

Application States

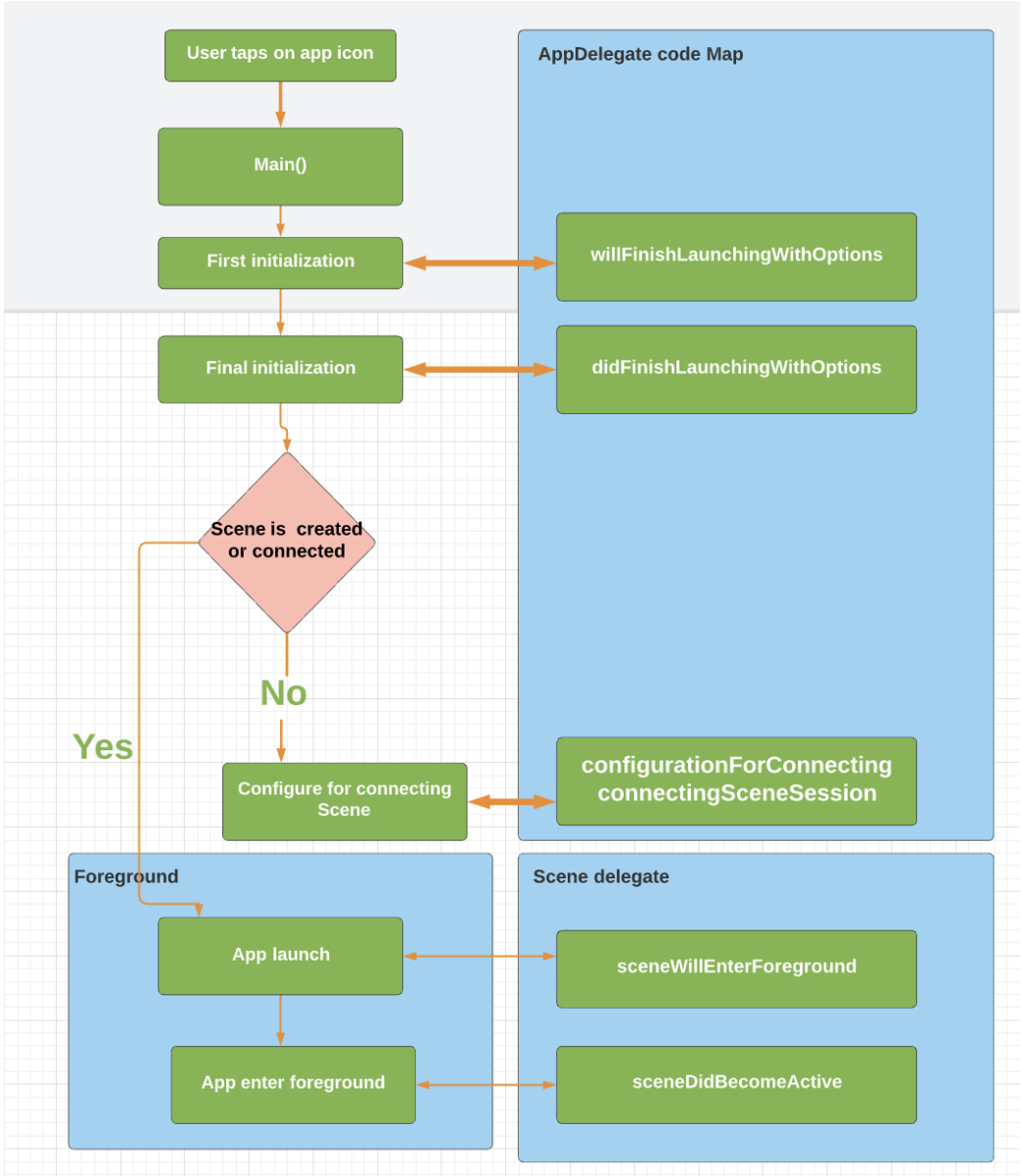
State	Description
Not Running	The app has not been launched or was running but was terminated by the system
Inactive	The app is running in the foreground but is currently not receiving events. (It may be executing other code though.) An app usually stays in this state only briefly as it transitions to a different state
Active	The app is running in the foreground and is receiving events. This is the normal mode for foreground apps
Background	The app is in the background and executing code. Most apps enter this state briefly on their way to being suspended. However, an app that requests extra execution time may remain in this state for a period of time
Suspended	The app is in the background but is 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 but does not execute any code

Java™ Education and Technology Services

Life Cycle

Objective-C

Life Cycle



Java™ Education and Technology Services

Objective-C

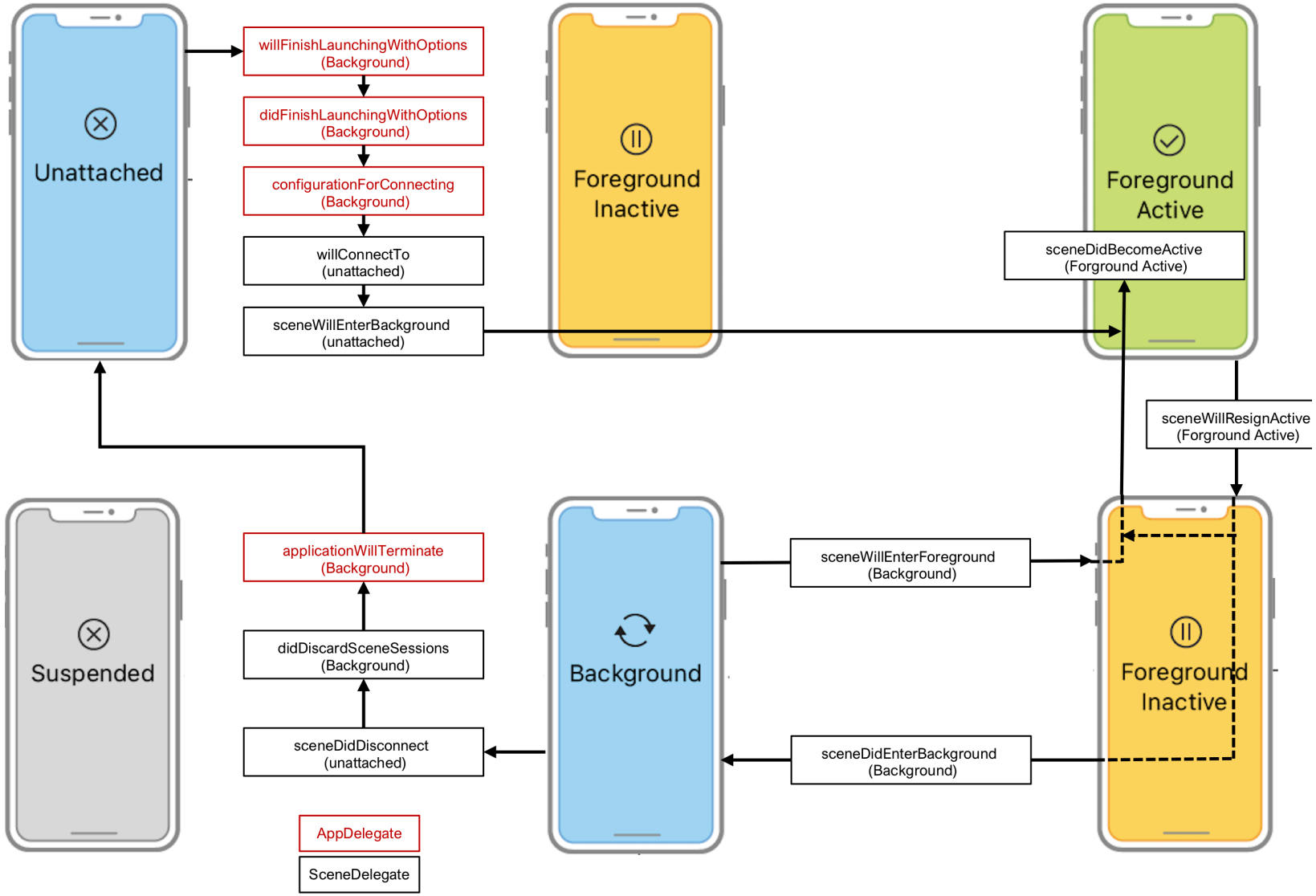
Life Cycle Cont.

```
int main(int argc, char *argv[])
{
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([JETSAppDelegate class]));
    }
}
```

Java™ Education and Technology Services

Objective-C

Life Cycle Cont.



Objective-C

Life Cycle Cont.

- The AppDelegate methods

```
1. func application(_:didFinishLaunchingWithOptions:) -> Bool
2. func application(_:configurationForConnecting:options:) -> UISceneConfiguration
3. func application(_:didDiscardSceneSessions:)
```

are used to:

- Initialize the application’s critical data structures
- Handle the external services like push notification registrations, location services, app termination and more..

Life Cycle Cont.

- The SceneDelegate methods

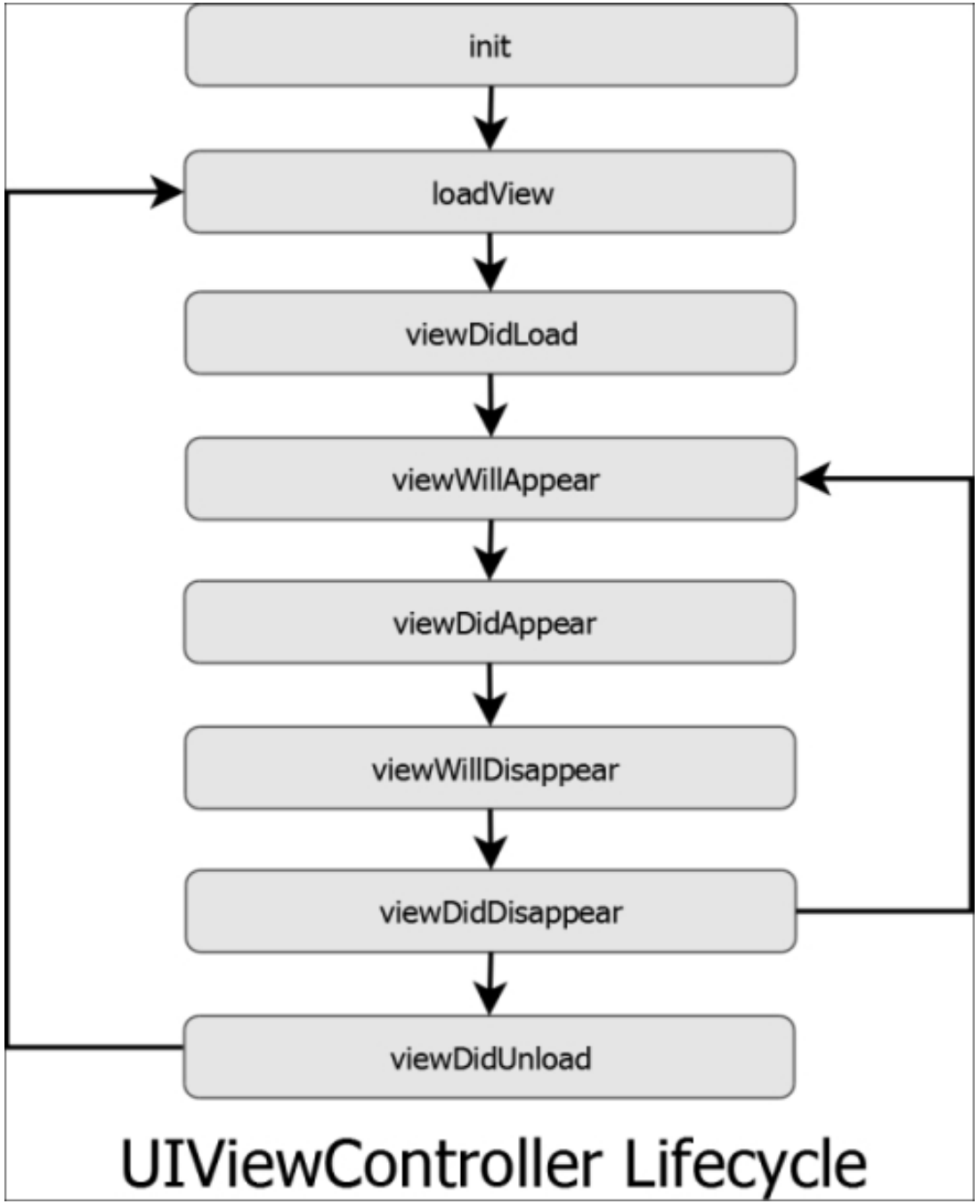
```
1. scene(_:willConnectTo:options:)
2. sceneDidDisconnect(_:)
3. sceneDidBecomeActive(_:)
4. sceneWillResignActive(_:)
5. sceneWillEnterForeground(_:)
6. sceneDidEnterBackground(_:)
```

are used to:

- Be responsible for what's displayed on the screen in terms of UI and data.
- create new UIWindow, set the root view controller and make this window the key window to be displayed.
- Handle the transitions from background to foreground and vice versa.

Objective-C

View LifeCycle



Life Cycle Demo

Java™ Education and Technology Services

IBOutlets and IBActions

Objective-C

IBOutlet

- It is used to identify that this element should appear in the interface builder for LINKING with some elements

- You don’t have to define all the GUI components as IBOutlets, you have to define components that you need to access for read or write

IBAction

- It is used to identify that this method is a call back method, called by the system upon GUI event.
- It makes this method appear in the interface builder for LINKING with some events
- You define only GUI event actions methods as IBAction
- IBAction method can serve more than an event

Call Back Methods

- Methods created by the developer and called by the system upon certain events
- Call back methods are used in iOS development much
- They are used in IBActions and delegate methods

IBOutlet and IBAction Demo

Attributes of @property

- atomic.
- nonatomic.
- retain.
- copy.
- readonly.
- readwrite.
- strong.

atomic

- It is the default behaviour.
- If an object is declared as atomic then it becomes thread-safe.
- Thread-safe means, at a time only one thread of a particular instance of that class can have the control over that object.

Non atomic

- It is not thread-safe.
- You can use the nonatomic property attribute to specify that synthesized accessors simply set or return a value directly, with no guarantees about what happens if that same value is accessed simultaneously from different threads.
- For this reason, it’s faster to access a nonatomic property than an atomic one.

retain

- is required when the attribute is a pointer to an object.
- The setter method will increase retain count of the object, so that it will occupy memory in autorelease pool.

copy

- If you use copy, you can't use retain.
- Using copy instance of the class will contain its own copy.

Read only

- If you don't want to allow the property to be changed via setter method, you can declare the property read only.

Read write

- Is the default behaviour.
- You don't need to specify readwrite attribute explicitly.

strong

- The compiler will take care for any object that you assign to the property and will not be destroyed as long as you point to it with strong reference

- With weak reference you will not control objects life time

Lab Exercise

1.Hello World

- Create Hello World application with Button and Label that shows Hello World message when you press the button.

Java™ Education and Technology Services

Objective-C

2.Life Cycle

- Use AppDelegate methods to view the life cycle of your application.

3.Copying Text

- Create an application with Button ,text field and label that shows the entered text in the text field on the label.

4.Your Colleagues

- Create an application that switches between your adjacent colleagues names using Next and Previous buttons.

- Make it cyclic.

5.Validation

- Create an application to validate whether the user’s input is text or number using two buttons

Java™ Education and Technology Services

Objective-C

6.Simple Calculator

- Create a simple calculator to perform the basic operations (addition, subtraction, multiplication, division)