# How iOS Works

Session 102

Răzvan Geangu

## Recap

## Recap

Variables

## Recap

Variables

Constants

## Recap

Variables

Constants

Conditional Statements

## Recap

Variables

Constants

Conditional Statements

Arrays

## Recap

Variables

Constants

Conditional Statements

Arrays

Dictionaries

## Recap

Variables

Constants

Conditional Statements

Arrays

Dictionaries

For Loops

## Recap

Variables

Constants

Conditional Statements

Arrays

Dictionaries

For Loops

While Loops

# Recap

Variables

Constants

Conditional Statements

Arrays

Dictionaries

For Loops

While Loops

Structs

## Recap

Variables

Constants

Conditional Statements

Arrays

Dictionaries

For Loops

While Loops

Structs

Enums

## Recap

Variables

Constants

Conditional Statements

Arrays

Dictionaries

For Loops

While Loops

Structs

Enums

Functions

## Quiz

*Clone the playground and start playing!*

## What You Will Learn

## What You Will Learn

App States

## What You Will Learn

App States

View States

## What You Will Learn

App States

View States

Protocols

## What You Will Learn

App States

View States

Protocols

Delegation

## What You Will Learn

App States

View States

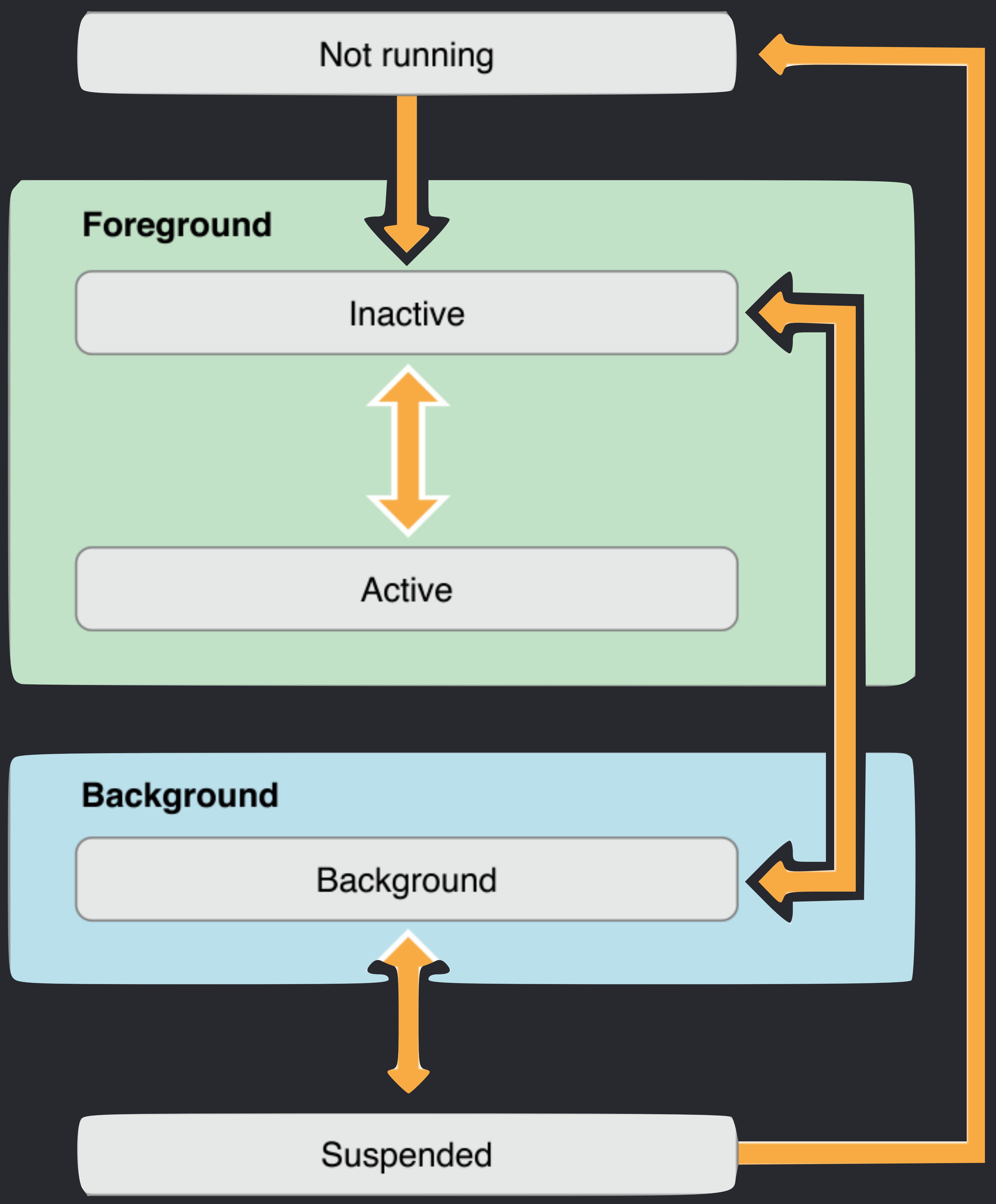
Protocols

Delegation

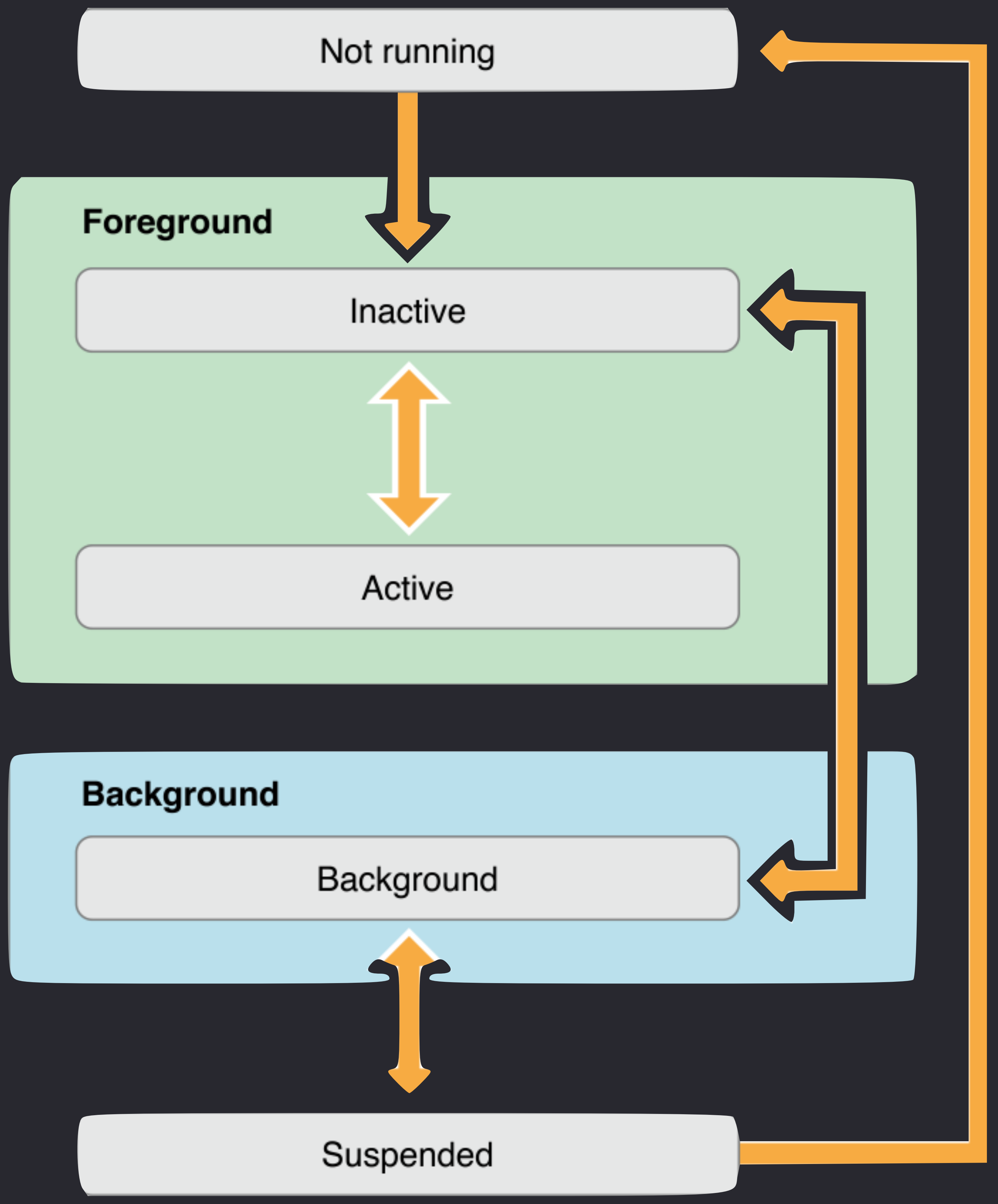
MVC

## App States

# App States


Not running
Foreground
Inactive
Active
Background
Background
Suspended

## App States



application:willFinishLaunchingWithOptions:

application:didFinishLaunchingWithOptions:

applicationDidBecomeActive:

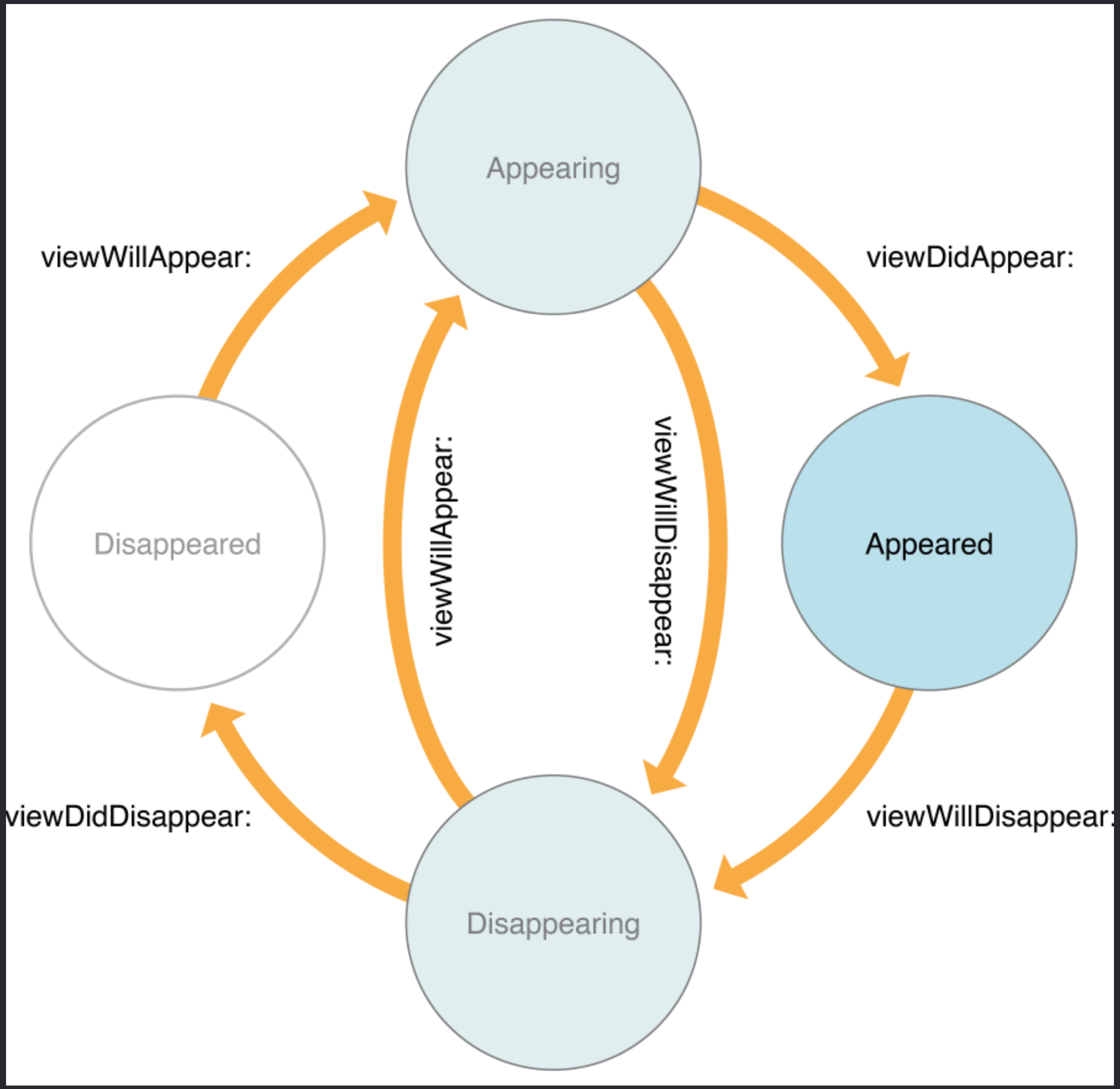
applicationDidEnterBackground:

applicationWillEnterForeground:

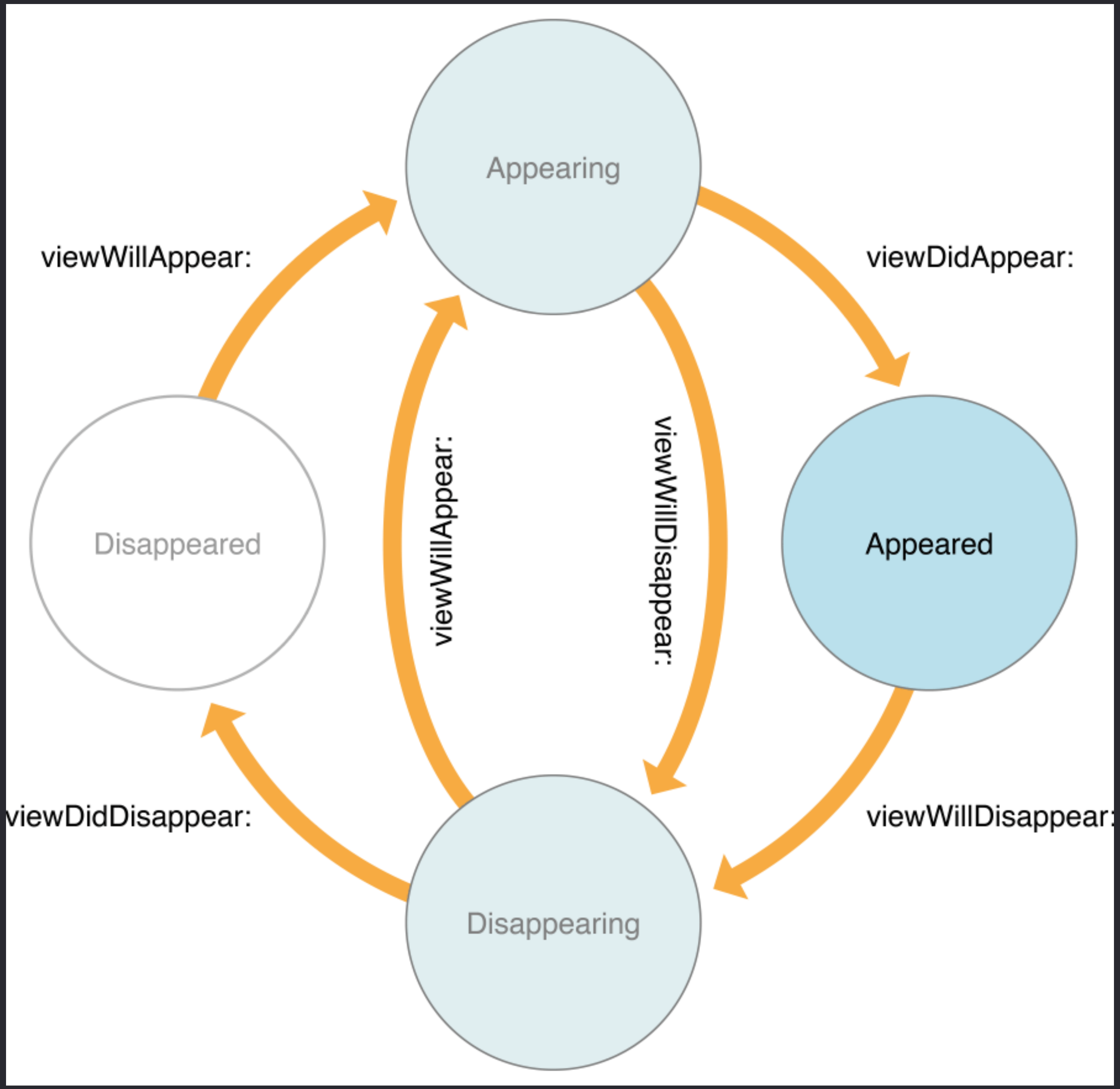
applicationWillTerminate:

## View States

# View States


Appearing
viewWillAppear:
viewDidAppear:
Disappeared
viewWillAppear:
viewWillDisappear:
Appeared
viewDidDisappear:
viewWillDisappear:
Disappearing

## View States



viewDidLoad:

viewWillAppear:

viewDidAppear:

viewWillDisappear:

viewDidDisappear:

## Protocols

## Protocols

A protocol defines a blueprint of methods, properties, and other requirements that suit a particular task or piece of functionality. It can be adopted by a class, structure or enumeration.

## Protocols

A protocol defines a blueprint of methods, properties, and other requirements that suit a particular task or piece of functionality. It can be adopted by a class, structure or enumeration.

```swift
protocol RandomNumberGenerator {
    var lastRandomNumber: Double { get }
    func random() -> Double
    mutating func changeSomething()
}
```

## Protocols

## Protocols

```
class Game: RandomNumberGenerator, OtherProtocol {
    var lastRandomNumber: Double = 0.0
    func random() -> Double { ... }
    mutating func changeSomething() { ... }
}
```

## Delegation

## Delegation

Delegation is a design pattern that enables a class or structure to hand off some of its responsibilities to an instance of another type.

## Delegation

Delegation is a design pattern that enables a class or structure to hand off some of its responsibilities to an instance of another type.

```
protocol DiceGame {
    func play()
}
protocol DiceGameDelegate {
    func didStartGame(_ game: DiceGame)
    func didEndGame(_ game DiceGame)
}
```

## Delegation

## Delegation

```swift
class SnakesAndLadders: DiceGame {
    weak var delegate: DiceGameDelegate?
    func play() {
        delegate?.gameDidStart(self)
        // Play game
        delegate?.gameDidEnd(self)
    }
}

class DiceGameTracker: DiceGameDelegate {
    myGame.delegate = self
    func didStartGame(_ game: DiceGame) { ... }
    func didEndGame(_ game: DiceGame) { ... }
}
```

## Video

*Stanford CS193P Fall 2017 -18*

## Demo

KCL TECH SOCIETY