Object Oriented Programming

The Swift Programming Language Part 5, Section 2

iCodeWave Community

OOP

Today’s Agenda 🎉

- Initializer
- Class vs Struct: What’s the differences?
- Class vs Struct: What’s the commons?
- Class vs Struct: When to use?

OOP

What’s initializer?

- A special method to initial properties inside a class
- **Class have** direct init function, rather than struct
- Supporting encapsulation
- Make it easier to use
- To allocate object into a memory

iCodeWave Community

Class vs Struct

What’s the differences?

- Classes are reference type, structs are value type objects
- Classes do more privilleges than structs, such as inheritance and **deinit**ializer
- Class uses reference types are not copied when they are assigned to a variable or constant, or when they are passed to a function. Rather than a copy, a reference to the same **existing** instance is used. (Memory saver)
- Struct uses value type is a type whose value is copied when it’s assigned to a variable or constant, or when it’s passed to a function

iCodeWave Community

Class vs Struct

What’s the commons?

- They both define properties to store values. If we want to create an object with some properties and behaviors, we can use both of them
- Both of them define subscripts so we can reach their values via subscript syntax
- Classes and structs have initializers to create instances with initial values
- Conforming to protocols are also in common. Some protocols may be class only so only classes can conform to that protocol, otherwise both can conform it

Class vs Struct

When to use class?

- Use classes when we need Objective-C interoperability. If we use an Objective-C API that receives data from our side, those data must be a class because Objective-C doesn’t have structs
- Another use case for classes is when we need to control identity. If we need an instance of an object through the app and we want to control its identity, classes are the solution
- Usually UIKit uses class much, because from we design UI we use UIViewController

iCodeWave Community

Class vs Struct

When to use struct?

- **Almost** anytime
- Use structs by default to represent common kinds of data. To track a portion of your code since they are value types
- Use structs when we don’t control identity. Such as Model to make blueprints of our decoded JSON values
- SwiftUI Views do always conform to struct

iCodeWave Community

Struct Example 1

```swift
struct Point {
    var x: Int
    var y: Int

    // Mutating method to modify properties
    mutating func move(x: Int, y: Int) {
        self.x += x
        self.y += y
    }
}

// Create an instance of Point
var point1 = Point(x: 10, y: 20)
var point2 = point1 // Creates a copy of point1

// Modify the copy
point2.move(x: 5, y: 5)

print("Point 1: (\(point1.x), \(point1.y))") // Output: Point 1: (10, 20)
print("Point 2: (\(point2.x), \(point2.y))") // Output: Point 2: (15, 25)
```

Class Example 1

```swift
// Define a class named Person
class Person {
    var name: String
    var age: Int

    init(name: String, age: Int) {
        self.name = name
        self.age = age
    }

    // Method to increment age
    func celebrateBirthday() {
        age += 1
    }
}

// Create an instance of Person
var person1 = Person(name: "John", age: 30)
var person2 = person1 // Both person1 and person2 reference the same instance

// Modify the age of person2
person2.celebrateBirthday()

print("Person 1: \(person1.name), \(person1.age)") // Output: Person 1: John, 31
print("Person 2: \(person2.name), \(person2.age)") // Output: Person 2: John, 31
```

Struct Example 2

```swift
struct Rectangle {
    var width: Double
    var height: Double

    var area: Double {
        return width * height
    }

    mutating func scale(by factor: Double) {
        width *= factor
        height *= factor
    }
}

var rectangle1 = Rectangle(width: 5.0, height: 3.0)
print("Area of Rectangle 1: \(rectangle1.area)") // Output: Area of Rectangle 1: 15.0


rectangle1.scale(by: 2.0)
print("Area of Rectangle 1 after scaling: \(rectangle1.area)") // Output: Area of Rectangle 1 after scaling: 60.0
```

Class Example 2

```swift
class BankAccount {
    var balance: Double

    init(balance: Double) {
        self.balance = balance
    }

    func deposit(amount: Double) {
        balance += amount
    }
}

var account1 = BankAccount(balance: 1000)
var account2 = account1 // Both account1 and account2 reference the same instance

account2.deposit(amount: 500)

print("Account 1 balance: \(account1.balance)") // Output: Account 1 balance: 1500.0
print("Account 2 balance: \(account2.balance)") // Output: Account 2 balance: 1500.0
```

Thanks For Your Attendance Today!

iCodeWave Community