Object Oriented Programming

The Swift Programming Language Part 5, Section 3

iCodeWave Community

Principals of OOP & Encapsulation

Today's Agenda 🤯

iCodeWave Community

OOP

What's the principals of OOP?

- There are 4 pillars of OOP that consist of: encapsulation, inheritance, abstraction, polymorphism

OOP

Encapsulation

- Involves bundling the data of attributes and methods (functions) that operate on the data within a single unit, often referred to as a class
- To hide the internal details of a class and provide a well-defined interface for interacting with it
- Achieved through the use of access control modifiers, such as **private**, **fileprivate**, **internal**, and **public**
- Encapsulation dictate the level of visibility and access a class, property, or method has within the codebase

iCodeWave Community

Encapsulation

Modifiers: Private

- Accessible only within the same source file where it's defined
- This is the most restrictive access level
- Typically used for properties, methods, or types that are meant to be used only within a single type (such as a class or struct)
- Cannot be accessed from outside the enclosing declaration

iCodeWave Community

Encapsulation

Modifiers: Private – Example Code

```swift
class PrivateExample {
  // User can't access this
  private var privateProperty: Int = 0

  // User can't access this
  private func privateMethod() {
    print("This is a private method")
  }

  // Public method to demonstrate access to private members
  func accessPrivate() {
    print("Accessing private property: \(privateProperty)")
    privateMethod()
  }
}

let privateObject = PrivateExample()
privateObject.accessPrivate() // Output: Accessing private property: 0
```

iCodeWave Community

Encapsulation

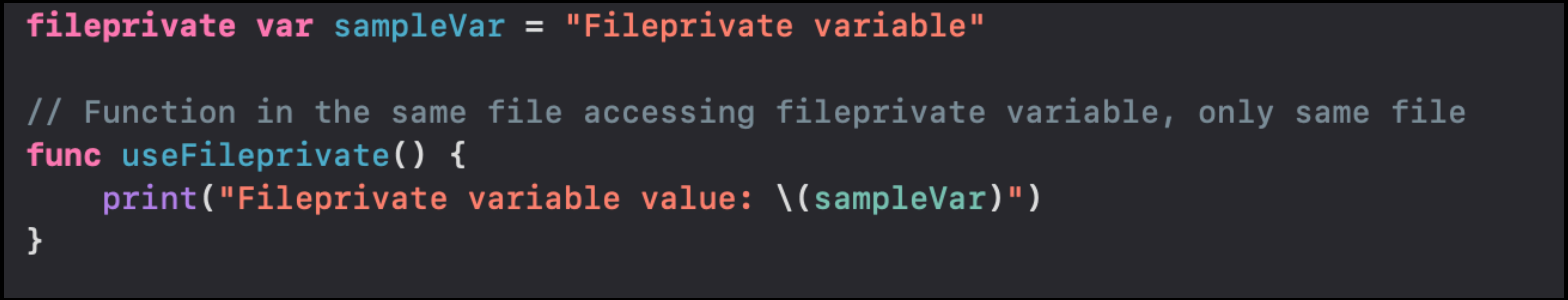
Modifiers: Fileprivate

- Accessible only within the same Swift file where it's defined
- Similar to **private**, but allows access from other types defined in the same file
- Useful for restricting access to members within a specific file

iCodeWave Community

Encapsulation

Modifiers: Fileprivate – Example Code



```swift
fileprivate var sampleVar = "Fileprivate variable"

// Function in the same file accessing fileprivate variable, only same file
func useFileprivate() {
    print("Fileprivate variable value: \(sampleVar)")
}
```

Encapsulation

Modifiers: Internal

- Default access level if no access level modifier is specified
- Accessible within the same module (a compiled bundle of code)
- Not accessible from outside the module, which means it's invisible to code in other modules

Encapsulation

Modifiers: Internal – Example Code

```swift
internal class InternalExample {
    var internalProperty: String = "Internal Property"

    // Internal method accessible within the module
    internal func internalMethod() {
        print("Internal method called")
    }
}
```

Encapsulation

Modifiers: Public

- Default modifier even not declared
- Accessible from any other module that imports the module in which the entity is defined
- Provides the highest level of access
- Used when you want the entity to be available for use by code outside the module

iCodeWave Community

Encapsulation

Modifiers: Public – Example Code

```swift
struct PublicExample { // using public or not will give default public access
    public var publicProperty: Double

    // Public method accessible from other modules
    public func publicMethod() {
        print("Public method called")
    }
}
```

iCodeWave Community

Thanks For Your Attendance Today!

iCodeWave Community