**Protocol Extension**

Protocol extension are used to implement the methods subscripts and computed property of that protocol

Protocol extensions can add implementations to conforming types but can’t make a protocol extend or inherit from another protocol. Protocol inheritance is always specified in the protocol declaration itself.

Protocol requirements with default implementations provided by extensions are distinct from optional protocol requirements. Although conforming types don’t have to provide their own implementation of either, requirements with default implementations can be called without optional chaining.

Demystifying Swift protocol extensions with default parameters

A close up of a map

Description automatically generated

A close up of a sign

Description automatically generated

**Two extension of protocol with different implementation**

**-Compile time error**

**When implementing class conform to protocol both had same method**

**-Compile time error on implementing class or structure**

**When same methods is implemented in their extension**

**-Compile time error**

**When there is no method name in protocol but extension had the implementation**

**You can use that method if implementing class extending that protocol**

**How to contain the extension using where use of self in where**

**extension** **ErrorHandler** **where** **Self**: **UIViewController** {

**func** **handle**(error: Error) {

**let** alert = **UIAlertController**(title: nil, message: error.localizedDescription, preferredStyle: .alert)

**let** action = **UIAlertAction**(title: "OK", style: .cancel, handler: nil)

alert.addAction(action)

present(alert, animated: true, completion: nil)

}

}

From?