iOS Assignment

Part 1: Type Check & Casting

Objective: Understand how to check types and cast objects in Swift.

Tasks:

1. Create a new Swift file named `TypeCheckAndCasting.swift`.
2. Define a base class named `Vehicle` and two subclasses `Car` and `Bike`.
3. Instantiate an array of `Vehicle` objects containing both `Car` and `Bike` instances.
4. Write a function `describeVehicles(vehicles: [Vehicle])` that iterates over the array and prints whether each vehicle is a `Car` or a `Bike` using type checking (`is`) and type casting (`as?`).

Part 2: Any Object & Any

Objective: Learn to use `Any` and `AnyObject` in Swift.

Tasks:

1. Create a new Swift file named `AnyAndAnyObject.swift`.
2. Define an array that can hold elements of any type (`Any`), including an `Int`, a `String`, and a custom class instance.
3. Write a function `describeElements(elements: [Any])` that iterates over the array and prints the type of each element.

Part 3: Delegates

Objective: Understand and implement the delegate pattern in Swift.

Tasks:

1. Create a new Swift file named `Delegates.swift`.
2. Define a protocol named `TaskDelegate` with a method `taskDidComplete`.
3. Create a class `Task` that has a delegate property conforming to `TaskDelegate`.
4. Implement a class `TaskHandler` that conforms to `TaskDelegate` and handles the completion of the task.
5. Demonstrate the delegate pattern by creating an instance of `Task`, setting its delegate to an instance of `TaskHandler`, and simulating the completion of a task.
6. Optional, try to create 3 vc and pass data from C->A.

Part 4: Class Extensions

Objective: Learn to extend the functionality of existing classes using extensions.

Tasks:

1. Create a new Swift file named `ClassExtensions.swift`.
2. Define a class `Circle` with a property `radius` and an initializer.
3. Create an extension for `Circle` that adds a computed property `area` to calculate the area of the circle.
4. Demonstrate the use of this extension by creating an instance of `Circle` and printing its area.

Study

Equatable
Identifiable
Hashable
Comparable