### **Lab 9:** **Protocol in Swift**

Protocols in Swift define a blueprint of methods, properties, and other requirements that suit a particular task or piece of functionality. This lab exercise will cover creating protocols, conforming to protocols, and using protocol-oriented programming concepts.

**Part 1: Basic Protocols**

1. **Simple Protocol:**

* Define a protocol called Greeter with a single method requirement sayHello().

protocol Greeter {

func sayHello()

}

1. **Class Conforming to Protocol:**

* Create a class called Person that conforms to the Greeter protocol by implementing the sayHello() method to print "Hello!".

class Person: Greeter {

func sayHello() {

print("Hello!")

}

}

1. **Using Protocol:**

* Create an instance of the Person class and call the sayHello() method.

let person = Person()

person.sayHello()

**Part 2: Protocols with Properties**

1. **Protocol with Property Requirement:**

* Define a protocol called Describable with a property requirement description of type String.

protocol Describable {

var description: String { get }

}

1. **Struct Conforming to Protocol:**

* Create a struct called Book that conforms to the Describable protocol by implementing the description property to return a book's title.

struct Book: Describable {

var title: String

var description: String {

return "Title: \(title)"

}

}

1. **Using Protocol Property:**

* Create an instance of the Book struct and access its description property.

let book = Book(title: "Swift Programming")

print(book.description)

**Part 3: Protocol Inheritance**

1. **Inheriting Protocols:**

* Define a protocol called Animal with a method requirement makeSound().
* Define another protocol called Pet that inherits from Animal and adds a property requirement name of type String.

protocol Animal {

func makeSound()

}

protocol Pet: Animal {

var name: String { get }

}

1. **Class Conforming to Inherited Protocols:**

* Create a class called Dog that conforms to the Pet protocol by implementing the makeSound() method to bark and providing a name property.

class Dog: Pet {

var name: String

init(name: String) {

self.name = name

}

func makeSound() {

print("\(name) barks!")

}

}

1. **Using Inherited Protocols:**

* Create an instance of the Dog class and call the makeSound() method.

let dog = Dog(name: "Buddy")

dog.makeSound()

**Summary**

This lab exercise covered the basics of using protocols in Swift, including defining protocols, conforming to protocols, protocol inheritance, and protocol extensions. By completing these tasks, you've become familiar with creating blueprints for behavior and properties in Swift, enabling flexible and reusable code. Experiment with additional protocols and conformances to further explore the capabilities of Swift protocols.