### **Lab 6:** **Enumerations in Swift**

Enumerations, or enums, in Swift are powerful tools for defining a group of related values. This lab exercise will cover defining enums with associated values, raw values, and methods.

**Part 1: Basic Enums**

1. **Simple Enum:**

* Define an enum called Direction with cases for north, south, east, and west.

enum Direction {

case north, south, east, west

}

1. **Using Enum Cases:**

* Create a variable of type Direction and assign it a value.
* Print the value of the variable.

var currentDirection: Direction = .north

print("Current direction: \(currentDirection)")

**Part 2: Enums with Associated Values**

1. **Enum with Associated Values:**

* Define an enum called Measurement with cases for length(Double), width(Double), and height(Double).

enum Measurement {

case length(Double)

case width(Double)

case height(Double)

}

1. **Using Enum with Associated Values:**

* Create instances of the Measurement enum with different values.
* Print the values of these instances.

let length = Measurement.length(5.0)

let width = Measurement.width(3.0)

let height = Measurement.height(10.0)

print("Length: \(length)")

print("Width: \(width)")

print("Height: \(height)")

**Part 3: Enums with Raw Values**

1. **Enum with Raw Values:**

* Define an enum called Planet with raw values for the names of planets in our solar system.

enum Planet: String {

case mercury = "Mercury"

case venus = "Venus"

case earth = "Earth"

case mars = "Mars"

case jupiter = "Jupiter"

case saturn = "Saturn"

case uranus = "Uranus"

case neptune = "Neptune"

}

1. **Using Enum with Raw Values:**

* Access the raw value of each enum case and print it.

print("Raw value of Earth: \(Planet.earth.rawValue)")

**Part 4: Enums with Methods**

1. **Enum with Method:**

* Define an enum called Operation with cases for basic arithmetic operations (add, subtract, multiply, divide).
* Add a method called perform that takes two Double parameters and returns the result of the operation.

enum Operation {

case add, subtract, multiply, divide

func perform(\_ a: Double, \_ b: Double) -> Double {

switch self {

case .add:

return a + b

case .subtract:

return a - b

case .multiply:

return a \* b

case .divide:

guard b != 0 else {

fatalError("Cannot divide by zero")

}

return a / b

}

}

}

1. **Using Enum with Method:**

* Create instances of the Operation enum and call the perform method with different values.

let addition = Operation.add.perform(3, 2)

let subtraction = Operation.subtract.perform(5, 2)

let multiplication = Operation.multiply.perform(4, 3)

let division = Operation.divide.perform(10, 2)

print("Addition: \(addition)")

print("Subtraction: \(subtraction)")

print("Multiplication: \(multiplication)")

print("Division: \(division)")

**Part 5: Practical Examples**

1. **HTTP Status Codes:**

* Define an enum called HTTPStatusCode with cases for common HTTP status codes (success, clientError, serverError) and associated values for the status code and message.

enum HTTPStatusCode {

case success(code: Int, message: String)

case clientError(code: Int, message: String)

case serverError(code: Int, message: String)

}

1. **Using Enum for HTTP Status Codes:**

* Create instances of the HTTPStatusCode enum and print their associated values.

let success = HTTPStatusCode.success(code: 200, message: "OK")

let clientError = HTTPStatusCode.clientError(code: 404, message: "Not Found")

let serverError = HTTPStatusCode.serverError(code: 500, message: "Internal Server Error")

print("Success: \(success)")

print("Client Error: \(clientError)")

print("Server Error: \(serverError)")

**Summary**

This exercise covers the basics of enums in Swift, including defining enums with associated values, raw values, and methods. By completing these tasks, you will become familiar with using enums to model groups of related values and operations in Swift. Experiment with additional operations and modifications to further enhance your understanding of enums.