### **Lab 8:** **Class in Swift**

Classes in Swift are fundamental building blocks for creating flexible and reusable code. This lab exercise will cover creating classes with properties, methods, initializers, inheritance, and demonstrating their usage.

**Part 1: Basic Classes**

1. **Simple Class:**

* Define a class called Person with properties for name (String) and age (Int).

class Person {

var name: String

var age: Int

init(name: String, age: Int) {

self.name = name

self.age = age

}

}

1. **Using Class:**

* Create an instance of the Person class with your own name and age.
* Print the properties of the created instance.

let person1 = Person(name: "John", age: 30)

print("Name: \(person1.name), Age: \(person1.age)")

**Part 2: Classes with Methods**

1. **Class with Method:**

* Extend the Person class with a method called introduce that prints a personalized introduction.

class Person {

var name: String

var age: Int

init(name: String, age: Int) {

self.name = name

self.age = age

}

func introduce() {

print("Hello, my name is \(name) and I am \(age) years old.")

}

}

1. **Using Class Method:**

* Create an instance of the Person class and call the introduce method.

let person2 = Person(name: "Alice", age: 25)

person2.introduce()

**Part 3: Inheritance**

1. **Subclassing:**

* Define a subclass called Student that inherits from the Person class and adds a new property studentID (String).

class Student: Person {

var studentID: String

init(name: String, age: Int, studentID: String) {

self.studentID = studentID

super.init(name: name, age: age)

}

}

1. **Using Subclass:**

* Create an instance of the Student class with your own name, age, and student ID.

Print the properties of the created instance.

let student1 = Student(name: "Bob", age: 20, studentID: "S123456")

print("Name: \(student1.name), Age: \(student1.age), Student ID: \(student1.studentID)")

**Part 4: Practical Examples**

1. **Bank Account Class:**

* Define a class called BankAccount with properties for accountNumber (String) and balance (Double).
* Add methods deposit(amount:) and withdraw(amount:) to deposit and withdraw money from the account.

class BankAccount {

var accountNumber: String

var balance: Double

init(accountNumber: String, balance: Double) {

self.accountNumber = accountNumber

self.balance = balance

}

func deposit(amount: Double) {

balance += amount

print("Deposited \(amount) into account \(accountNumber). New balance: \(balance)")

}

func withdraw(amount: Double) {

guard balance >= amount else {

print("Insufficient funds to withdraw \(amount) from account \(accountNumber).")

return

}

balance -= amount

print("Withdrawn \(amount) from account \(accountNumber). New balance: \(balance)")

}

}

1. **Using Bank Account Class:**

* Create an instance of the BankAccount class and perform deposit and withdrawal operations.

let account1 = BankAccount(accountNumber: "123456", balance: 1000.0)

account1.deposit(amount: 500.0)

account1.withdraw(amount: 200.0)

**Summary**

This lab exercise covered the basics of using classes in Swift, including defining classes with properties, methods, initializers, and inheritance. By completing these tasks, you've become familiar with creating custom data types and implementing functionality within them. Experiment with additional properties, methods, and inheritance relationships to further explore the capabilities of Swift classes..