Министерство образования

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Отчет по лабораторной работе №4

«Использование языка программирования Swift: наследование, протоколы»

Выполнил: Моров Н.В.

Студент группы 310902

Проверил: Усенко Ф.В.

Минск 2024

Цель работы – Выполнить разработку приложения с использованием языка программирования Swift: наследование, протоколы.

1. Задание. Создать класс, который проверяет число и определяет, что это – простое число, число Фибоначчи, комплексное число, целое, вещественное. Каждый метод относится к отдельному протоколу.

**Код на языке Swift:**

**Package.swift**

**// swift-tools-version:5.7**

**import PackageDescription**

**let package = Package(**

**name: "Nikita4",**

**targets: [**

**.executableTarget(**

**name: "MainModule",**

**dependencies: ["ValidationModule", "NumberAnalyzerModule"]**

**),**

**.target(**

**name: "ValidationModule",**

**dependencies: []**

**),**

**.target(**

**name: "NumberAnalyzerModule",**

**dependencies: ["ValidationModule"]**

**),**

**]**

**)**

**NumberAnalyzerModule\NumberAnalyzer.swift**

**import Foundation**

**import ValidationModule**

**public class NumberAnalyzer: NumberAnalyzerProtocol {**

**public init() {}**

**public func isPrime(\_ number: Int) -> Bool {**

**guard number > 1 else { return false }**

**for i in 2..<Int(sqrt(Double(number))) + 1 {**

**if number % i == 0 { return false }**

**}**

**return true**

**}**

**public func isFibonacci(\_ number: Int) -> Bool {**

**func isPerfectSquare(\_ x: Int) -> Bool {**

**let s = Int(sqrt(Double(x)))**

**return s \* s == x**

**}**

**return isPerfectSquare(5 \* number \* number + 4) || isPerfectSquare(5 \* number \* number - 4)**

**}**

**public func isComplex(\_ real: Double, \_ imaginary: Double) -> Bool {**

**return imaginary != 0**

**}**

**public func isInteger(\_ number: Double) -> Bool {**

**return floor(number) == number**

**}**

**public func isReal(\_ number: Any) -> Bool {**

**return number is Double || number is Float**

**}**

**}**

**NumberAnalyzerModule\Protocols.swift**

**import Foundation**

**public protocol PrimeChecker {**

**func isPrime(\_ number: Int) -> Bool**

**}**

**public protocol FibonacciChecker {**

**func isFibonacci(\_ number: Int) -> Bool**

**}**

**public protocol ComplexNumberChecker {**

**func isComplex(\_ real: Double, \_ imaginary: Double) -> Bool**

**}**

**public protocol NumberTypeChecker {**

**func isInteger(\_ number: Double) -> Bool**

**func isReal(\_ number: Any) -> Bool**

**}**

**public protocol NumberAnalyzerProtocol: PrimeChecker, FibonacciChecker, ComplexNumberChecker, NumberTypeChecker {}**

**MainModule\main.swift**

**import Foundation**

**import NumberAnalyzerModule**

**import ValidationModule**

**do {**

**let analyzer: NumberAnalyzerProtocol = NumberAnalyzer()**

**let inputNumber = 13**

**try Validation.validateNumber(inputNumber)**

**print("Анализ числа \(inputNumber):")**

**print("Простое число: \(analyzer.isPrime(inputNumber))")**

**print("Число Фибоначчи: \(analyzer.isFibonacci(inputNumber))")**

**let realPart = 4.5**

**let imaginaryPart = 1.2**

**print("Анализ числа \(realPart) + \(imaginaryPart)i")**

**print("Комплексное число: \(analyzer.isComplex(realPart, imaginaryPart))")**

**print("Целое число: \(analyzer.isInteger(realPart))")**

**print("Вещественное число: \(analyzer.isReal(realPart))")**

**} catch let error as ValidationError {**

**print("Ошибка валидации: \(error.description)")**

**} catch {**

**print("Неизвестная ошибка: \(error)")**

**}**

**ValidationModule\InputValidation.swift**

**import Foundation**

**public enum ValidationError: Error, CustomStringConvertible {**

**case invalidNumber**

**public var description: String {**

**switch self {**

**case .invalidNumber:**

**return "Некорректное число. Пожалуйста, введите число."**

**}**

**}**

**}**

**public class Validation {**

**public static func validateNumber<T>(\_ input: T) throws -> T where T: LosslessStringConvertible {**

**guard let \_ = Double(String(describing: input)) else {**

**throw ValidationError.invalidNumber**

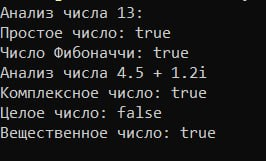
**}**

**return input**

**}**

**}**

**Скриншоты работающей программы:**

****

**Рисунок 1 - Результат выполнения программы**

**Вывод: В этой лабораторной работе мы разработали приложение на языке программирования Swift, используя наследование и протоколы. Функционал приложения был разбит на несколько пакетов, учитывая логику работы, и была сделана валидация вводимых значений, работа программы происходила с экземплярами протоколов, а не экземплярами классов.**