Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

Современные языки программирования

Отчет по лабораторной работе №4

«Использование языка программирования Swift: наследование, протоколы»

Выполнил: Чиж А. В.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

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

**Вариант** **2** Создать класс, который проверяет число и определяет, что это – простое число, число Фибоначчи, комплексное число, целое, вещественное. Каждый метод относится к отдельному протоколу.

**Код программы:**

import Foundation

class numberCheck: simpleProtocol, fibonacciProtocol, complexProtocol, intProtocol, realProtocol {

    var number: Double

    init (number: Double) {

        self.number = number

    }

}

let number = numberCheck(number: validate.getNumber())

print("Is simple: \(number.isSimple())")

print("Is fibonacci: \(number.isFib())")

print("Is complex: \(number.isComplex())")

print("Is integer: \(number.isInt())")

print("Is real: \(number.isReal())")

import Foundation

protocol simpleProtocol {

    var number: Double { get set }

}

extension simpleProtocol {

    func isSimple() -> Bool {

        guard number >= 2 && number == floor(number) else { return false }

        let intNumber = Int(number)

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

            if intNumber % i == 0 {

                return false

            }

        }

        return true

    }

}

protocol fibonacciProtocol {

    var number: Double { get set }

}

extension fibonacciProtocol {

    func isFib() -> Bool {

        var n: Int

        if (number == floor(number)) {

            n = Int.init(number)

        } else {

            return false

        }

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

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

            return s \* s == x

        }

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

    }

}

protocol complexProtocol {

    var number: Double { get set }

}

extension complexProtocol {

    func isComplex() -> Bool {

        return number.isNaN

    }

}

protocol intProtocol {

    var number: Double { get set }

}

extension intProtocol {

    func isInt() -> Bool {

        return number == floor(number)

    }

}

protocol realProtocol {

    var number: Double { get set }

}

extension realProtocol {

    func isReal() -> Bool {

        if (number == floor(number)) {

            return false

        }

        return !number.isNaN && !number.isInfinite

    }

}

import Foundation

class validate {

    public static func getNumber() -> Double {

        var x: Double?

        var s: String?

        func isComplex(s: String) -> Bool {

            let complexRegex = "^([-+]?[0-9]\*\\.?[0-9]+)([-+][0-9]\*\\.?[0-9]+)i$"

            let regex = try! NSRegularExpression(pattern: complexRegex, options: .caseInsensitive)

            let range = NSRange(location: 0, length: s.utf16.count)

            return regex.firstMatch(in: s, options: [], range: range) != nil

        }

        while true {

            print("Введите число: ")

            s = readLine()

            if (s == nil){

                print("Ошибка ввода!")

                continue

            } else {

                if (isComplex(s: s!)) {

                    return Double.nan

                }

                if (Double.init(s!) != nil) {

                    return Double.init(s!)!

                } else {

                    print("Ошибка ввода!")

                    continue

                }

            }

        }

    }

}

Результат работы программы представлен на рисунке 1

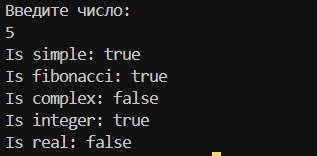


Рисунок 1 – результат работы программы

**Ответы на вопросы к лабораторной работе:**

1. Что такое наследование?

Наследование в Swift — это механизм, который позволяет создавать новые классы на основе уже существующих. Это позволяет повторно использовать код и расширять функциональность базовых классов.

1. Что такое протоколы?

Протоколы в Swift — это набор требований, которые определяют методы, свойства и другие требования, которые должны реализовывать классы, структуры или перечисления. Протоколы позволяют задавать общие интерфейсы для различных типов, обеспечивая гибкость и возможность полиморфизма.

1. Для чего служат протоколы?

Протоколы используются для определения общих интерфейсов, которые могут быть реализованы различными типами. Это позволяет создать гибкий и модульный код, где различные типы могут взаимодействовать друг с другом через общие интерфейсы.

1. Что такое расширение протоколов?

Расширение протоколов в Swift — это механизм, который позволяет добавлять новую функциональность к протоколам, не изменяя их исходное определение. С помощью расширений можно добавлять реализацию методов, свойств и других требований к протоколам, что делает код более гибким и модульным.

**Вывод**: Изучена работа языка *Swift* и выполнена разработка приложения с использованием наследования и протоколов.