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: Усов А.М.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

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

Задание 15: Протокол Создания уникальных чисел. Реализовать метод, который генерирует уникальное число. Создать класс, который будет иметь в себе генератор. Добавить несколько различных алгоритмов создания уникальных чисел. Создать несколько уникальных последовательностей чисел.

Листинг кода:

Файл main.swift:

import Generator

print("Выбирете алгоритм:")

print("1. Алгоритм инкремента")

print("2. Алгоритм Фибоначчи")

print("3. Алгоритм по дате")

var opt = -1;

while opt < 1 || opt > 3 {

    if let input = readLine() {

        if let number = Int(input) {

            opt = number

        }else{

            print("Введите число")

        }

    }else{

        print("Введите число")

    }

}

var generator: GeneratorBase

var algorithm: UniqueNumberGenerateAlgorithm

if opt == 1 {

    algorithm = UniqueNumberAlg1(step: 1)

    generator = Generator(algorithm: algorithm)

}

else if opt == 2 {

    algorithm = UniqueNumberAlg2()

    generator = Generator(algorithm: algorithm)

}

else {

    algorithm = UniqueNumberAlg3()

    generator = Generator(algorithm: algorithm)

}

while true{

    print("выбирете действие:")

    print("1. Сгенерировать число")

    print("2. Сгенерировать несколько чисел")

    print("3. Выход")

    var opt = -1;

    while opt < 1 || opt > 3 {

        if let input = readLine() {

            if let number = Int(input) {

                opt = number

            }else{

                print("Введите число")

            }

        }else{

            print("Введите число")

        }

    }

    if opt == 3 {

        break

    }

    if opt == 1 {

        print(generator.generate())

    }

    if opt == 2 {

        print("Введите количество:")

        var count = -1

        while count < 1 {

            if let input = readLine() {

                if let number = Int(input) {

                    count = number

                }else{

                    print("Введите число")

                }

            }else{

                print("Введите число")

            }

        }

        print(generator.generate(count: count))

    }

}

Файл Generator.swift

import Foundation

public protocol UniqueNumberGenerateAlgorithm {

    func generate() -> Int

}

// like increment

public class UniqueNumberAlg1: UniqueNumberGenerateAlgorithm {

    private var currentNumber: Int = 0

    private var step: Int = 1

    public init(step: Int) {

        self.step = step

    }

    public func generate() -> Int {

        currentNumber += step

        return currentNumber

    }

}

// like Fibonacci

public class UniqueNumberAlg2: UniqueNumberGenerateAlgorithm {

    private var currentNumber: Int = 1

    private var previousNumber: Int = 1

    public init(){}

    public func generate() -> Int {

        let newNumber = currentNumber + previousNumber

        previousNumber = currentNumber

        currentNumber = newNumber

        return newNumber

    }

}

// by date

public class UniqueNumberAlg3: UniqueNumberGenerateAlgorithm {

    private var lastNumber: Int = 0

    public init(){}

    public func generate() -> Int {

        let date = Date()

        let calendar = Calendar.current

        let components = calendar.dateComponents([.second], from: date)

        let newNumber = components.second! + lastNumber

        lastNumber = newNumber

        return newNumber

    }

}

public protocol GeneratorBase {

    var algorithm: UniqueNumberGenerateAlgorithm { get }

    func generate() -> Int

    func generate(count: Int) -> [Int]

}

public class Generator: GeneratorBase {

    public let algorithm: UniqueNumberGenerateAlgorithm

    public init(algorithm: UniqueNumberGenerateAlgorithm) {

        self.algorithm = algorithm

    }

    public func generate() -> Int {

        return algorithm.generate()

    }

    public func generate(count: Int) -> [Int] {

        var result: [Int] = []

        for \_ in 0..<count {

            Thread.sleep(forTimeInterval: 1)

            result.append(algorithm.generate())

        }

        return result

    }

}

Результат работы программы представлен на рисунке 1.

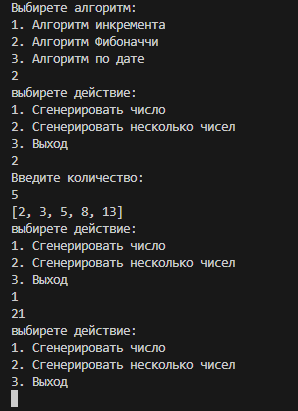


Рисунок 1 – Результат выполнения программы

Вывод: в ходе выполнения лабораторной работы номер 4 по языку программирования swift мы познакомились с методами проектирования и связей такие как наследование, реализация абстракций и их отношения. В ходе работы познакомились с необходимыми конструкциями языка для реализации абстракций.