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: Канунникова В. В.

Студент группы 310901

Преподаватель: Усенко Ф. В.

Минск 2024

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

**Вариант 9**: протокол Книга. Создать класс Библеотека, который содержит список объектов «Книга». Выведите информацию о каждой книге в Библиотеке. Выведите все книги определённого автора. Выведите все книги определённого года выпуска

**Код программы:**

//Model

////

//Book

// протокола Book

public protocol Book {

var title: String { get set }

var author: String { get set }

var publicationYear: Int { get set }

func information() -> String // метод должен предоставлять информацию о книге

//классы или структуры, соответствующие протоколу, должны реализовать этот метод

}

////

//LibraryBook

// Класс, реализующий протокол

class LibraryBook: Book {

var title: String

var author: String

var publicationYear: Int

init?(title: String, author: String, publicationYear: Int) { // инициализатор класса, который принимает три параметра

//инициализатор является опциональным (?), что означает, что он может вернуть nil, если инициализация не удалась

guard Validator1.validate1(title: title, author: author, publicationYear: publicationYear) else {

// проверка, что все параметры инициализатора действительны

return nil

}

self.title = title

self.author = author

self.publicationYear = publicationYear

}

func information() -> String { // объявление метода, который возвращает строку

return "Название: \(title), Автор: \(author), Год публикации: \(publicationYear)"

}

}

//Library

////

//Library

import Foundation

public class Library {

private var books: [Book] = [] // массив объектов, соответствующих протоколу Book

public init() {} // специальный метод, который позволяет создать объект класса без необходимости передавать какие-либо данные

// когда не нужно ничего заранее задавать или когда мы хотим, чтобы объект сам себя настроил

public func addBook(\_ book: Book) { // добавление в библиотеку

books.append(book)

}

public func displayBookInformation() { // вывод информацию обо всех книгах в библиотеке.

for book in books { // цикл, который проходит по всем объектам book в массиве books

print(book.information())

}

}

public func booksByAuthor(\_ author: String) -> [Book] { // метод, который принимает строку author и возвращает массив книг этого автора

return books.filter { $0.author == author } // возврат нового массива, содержащего только те книги, у которых автор совпадает с переданным значением author

}

public func booksByYear(\_ year: Int) -> [Book] {

return books.filter { $0.publicationYear == year } // возврат нового массива, содержащего только те книги, у которых год совпадает с переданным значением year

}

}

/////

//Validator

import Foundation

public struct Validator1 {

public static func validate1(title: String, author: String, publicationYear: Int) -> Bool { // метод, который принимает три параметра,

//он возвращает логическое значение (Bool), указывающее, действительны ли введенные данные

return !title.isEmpty && !author.isEmpty && publicationYear > 0 && publicationYear <= Calendar.current.component(.year, from: Date())

}

}

//main

import Foundation

let library = Library() // создание экземпляра класса Library и сохранение его в константе library

@MainActor

func promptForBook() -> Book? { // ввод названия книги, автора и года издания

print("Введите название книги:")

guard let title = readLine(), !title.isEmpty else {

// считывается ввод пользователя и проверяется, что введенная строка не пуста

print("Название книги не может быть пустым.")

return nil //Если строка пуста, выводится сообщение об ошибке и возвращается nil

}

print("Введите автора книги:")

guard let author = readLine(), !author.isEmpty else {

// считываеся ввод пользователя и проверяется, что введенная строка не пуста

print("Имя автора книги не может быть пустым.")

return nil

}

print("Укажите год публикации:")

guard let yearString = readLine(), let year = Int(yearString), year > 0 else {

// считывается ввод пользователя, преобразуем в число и проверяем, что год публикации больше 0

print("Неверный год публикации.")

return nil

}

return LibraryBook(title: title, author: author, publicationYear: year) // возврат нового объекта LibraryBook, созданный с введенными данными

}

@MainActor

func addBooksToLibrary(\_ library: Library) { // добавление книги

while true { // бесконечный цикл для многократного добавления книг

if let book = promptForBook() {

library.addBook(book) // добавление книги в библиотеку

}

print("Вы хотите написать еще одну книгу? (1: да, 2: нет)")

if let response = readLine(), response.lowercased() != "1" { // // считывает ввод пользователя, если ответ не "1", выходим из цикла

break

}

}

}

@MainActor

func searchBooks(library: Library) { // поиск книг по параметрам

while true {

print("Выберите вариант поиска:")

print("1. Поиск книг по автору")

print("2. Поиск книг по году издания")

print("3. Выход в главное меню")

guard let choice = readLine(), let option = Int(choice), (option >= 1 && option <= 3) else {

// считывает ввод пользователя и проверяем, что введенное значение корректно

print("Неверный выбор.")

continue

}

if option == 1 {

print("Введите имя автора:")

if let author = readLine(), !author.isEmpty { // проверка, что введенная строка не пуста

let books = library.booksByAuthor(author) // поиск необходимых книг

print("Книги автора \(author):")

for book in books { // цикл, который проходит по всем книгам в массиве books

print(book.information())

}

} else {

print("Имя автора не может быть пустым.")

}

} else if option == 2 {

print("Укажите год публикации:")

if let yearString = readLine(), let year = Int(yearString) { // проверка, что ввод не является nil и присваивание его переменной yearString

let books = library.booksByYear(year) // возврат массива книг, опубликованных в данном году, и присваивает его переменной books

print("Книги, опубликованные в \(year):")

for book in books { // цикл, который проходит по всем книгам в массиве books

print(book.information())

}

} else {

print("Неверный год публикации.")

}

} else if option == 3 { // выход

break

}

print("\nХотите выполнить другой поиск или выйти в главное меню? (1: Поиск, 2: Выход)")

// запрос на повторный поиск или выход

if let response = readLine(), response == "2" {

break // выход из цикла при вводе "2"

}

}

}

@MainActor

func viewAllBooks(library: Library) { // вывод всех книг

print("\nИнформация обо всех книгах в библиотеке:")

library.displayBookInformation()

}

На рисунке 1 представлен ввод информации о книге.

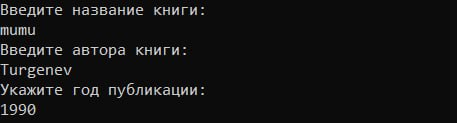


Рисунок 1 – Ввод информации о книге

На рисунке 2 представлен вывод всех книг в библиотеке.

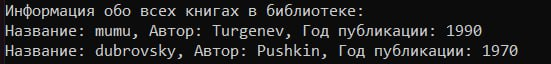


Рисунок 2 – Ассортимент библиотеки

На рисунке 3 показан поиск книги по автору.

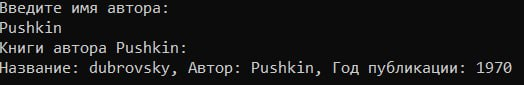


Рисунок 3 – Поиск книги по автору

На рисунке 4 продемонстрирован поиск книги по году публикации.

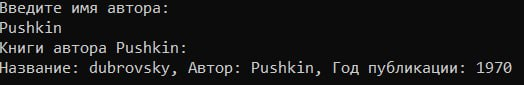


Рисунок 4 – Поиск книги по году публикации

**Ответы на вопросы к лабораторной работе:**

1. Что такое наследование?

Наследование позволяет одному классу (подклассу) получать свойства и методы другого класса (суперкласса).

2. Что такое переопределение? Что можно переопределять?

Переопределение позволяет подклассу предоставлять свою реализацию свойств, методов и сабскриптов суперкласса. Переопределять можно свойства, методы и сабскрипты.

3. Что такое протоколы?

Протоколы определяют требования к методам, свойствам и другим элементам, которые должны быть реализованы в классах, структурах и перечислениях.

4. Для чего служат протоколы?

Протоколы используются для обеспечения общих интерфейсов для классов, структур и перечислений, что позволяет унифицировать и структурировать код.

5. Что такое расширение протоколов?

Расширение протоколов позволяет добавлять методы и свойства в протоколы для всех типов, которые соответствуют этому протоколу.

6. Какой синтаксис у протоколов?

protocol ИмяПротокола {

// требования протокола

}

7. Что из себя представляет делегирование?

Делегирование – это паттерн, где один объект делегирует выполнение задачи другому объекту. Оно часто используется для передачи ответственности за выполнение задач между объектами.

**Вывод**: был изучен язык программирования Swift, а именно наследование и протоколы, и использован для разработки программы.