Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Учебная дисциплина «Современные языки программирования»

Отчёт

по лабораторной работе №3

«Использование языка программирования Swift: структуры и классы, методы»

Вариант 15

Выполнил студент группы 310101

Криницкий М.А.

Проверил:

Усенко Ф.В.

Минск 2024

Цель работы: разработать приложение с использованием языка программирования Swift: структуры и классы, методы.

В отличие от других языков программирования, Swift не требует создавать отдельные файлы для интерфейсов и реализаций пользовательских классов и структур. В Swift, вы объявляете структуру или класс в одном файле, и внешний интерфейс автоматически становится доступным для использования в другом коде.

Экземпляр класса традиционно называют объектом. Тем не менее, классы и структуры в Swift гораздо ближе по функциональности, чем в других языках, и многое в этой главе описывает функциональность, которую можно применить к экземплярам и класса, и структуры. В связи с этим, употребляется более общий термин – экземпляр.

Задание:

Экземпляр класса имеет координаты своего положения и угол, описывающий направление движения. Он может быть изначально поставлен в любую точку с любым направлением (конструктор), может проехать в выбранном направлении определённое расстояние и может повернуть, то есть изменить текущее направление на любое другое. Реализуйте класс автомобиля, а также класс, который будет описывать автобус. Кроме того, что имеется у автомобиля, у автобуса должны быть поля, содержащие число пассажиров и количество полученных денег, изначально равные нулю. Также должны быть методы «войти» и «выйти», изменяющие число пассажиров. Метод «move» должен увеличивать количество денег в соответствии с количеством пассажиров и пройденным расстоянием. (Листинг 1)

Листинг 1

import Foundation

class Transport {

var xTower: Double

var yTower: Double

var Direction: Double

init(xTower: Double, yTower: Double, Direction: Double) {

self.xTower = xTower

self.yTower = yTower

self.Direction = Direction

}

}

class Car: Transport {

let numberCar: String

var music: Bool

var Path: Double

init(numberCar: String, music: Bool, xTower: Double, yTower: Double, Direction: Double, Path: Double = 0.0) {

self.numberCar = numberCar

self.music = music

self.Path = Path

super.init(xTower: xTower, yTower: yTower, Direction: Direction)

}

func MoveCar(distance: Double) {

Path += distance

var x: Double = xTower

var y: Double = yTower

let b: Double = yTower - xTower \* tan(Direction \* .pi / 180)

let step: Double = (Direction >= -90 && Direction <= 90) ? 1.0 : -1.0

while (pow(x - xTower, 2) + pow(y - yTower, 2) <= pow(distance, 2)) {

print("Положение по координате x: \(x)\tПоложение по координате y: \(y)")

x += step

y = x \* tan(Direction \* .pi / 180) + b

}

print("Пройденный путь: \(Path)")

xTower = x

yTower = y

}

func turn(to newDirection: Double) {

Direction = newDirection

print("Автомобиль повернул в новое направление: \(Direction)°")

}

}

class Bus: Car {

var Amount\_of\_passengers: Int = 0

let Money\_for\_trip: Double = 0.90

var Total\_money: Double = 0 //общая сумма денег (например, за день)

var Total\_passengers: Int = 0 //общая сумма пассажиров (например, за день)

override init(numberCar: String, music: Bool, xTower: Double, yTower: Double, Direction: Double, Path: Double = 0.0) {

super.init(numberCar: numberCar, music: music, xTower: xTower, yTower: yTower, Direction: Direction, Path: Path)

}

func Enter() {

Amount\_of\_passengers += 1

Total\_passengers += 1

print("Количество пассажиров: \(Amount\_of\_passengers)")

}

func Go\_out() {

if Amount\_of\_passengers > 1 {

Amount\_of\_passengers -= 1

print("Количество пассажиров: \(Amount\_of\_passengers)")

} else {

print("Пассажиров нет")

}

}

func Move() {

Total\_money = Money\_for\_trip \* Double(Total\_passengers)\*Path

print("Общая сумма денег: \(Total\_money)")

}

}

let car1 = Car(numberCar: "1234", music: true, xTower: 2, yTower: 1, Direction: 45)

car1.MoveCar(distance: 10.0)

car1.turn(to: 225)

car1.MoveCar(distance: 10.0)

let bus1 = Bus(numberCar: "1234bus", music: false, xTower: 7, yTower: 3, Direction: 30)

for \_ in 0..<3 {

bus1.Enter()

}

bus1.MoveCar(distance: 20.0)

bus1.Move()

bus1.Go\_out()

bus1.Go\_out()

bus1.Go\_out()

bus1.Move()

Скриншоты работы программы представлены на рисунках 1,2.

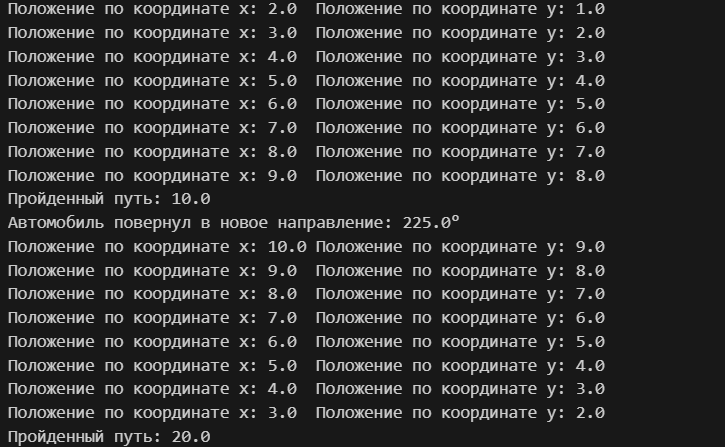


Рисунок 1 – Скриншот работы программы задания 1

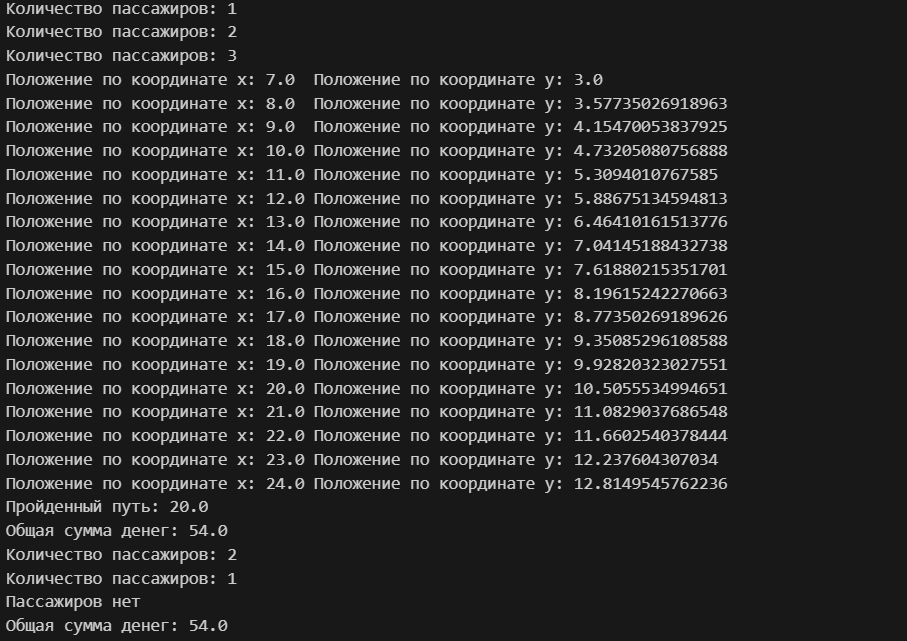


Рисунок 2 – Скриншот работы программы задания 2

**Вывод**

В лабораторной работе реализована система классов для моделирования транспортных средств на языке Swift, включающая базовый класс Transport и производные классы Car и Bus. Класс Car обеспечивает функциональность для движения и изменения направления автомобиля, в то время как класс Bus добавляет управление пассажирами и расчет доходов от поездок. Реализация демонстрирует принципы объектно-ориентированного программирования, такие как наследование и инкапсуляция, позволяя эффективно управлять состоянием и поведением различных типов транспортных средств.