Министерство образования

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Отчет по лабораторной работе №1

«Использование языка программирования Kotlin»

Выполнил: Моров Н.В.

Студент группы 310902

Проверил: Усенко Ф.В.

Минск 2024

Цель работы – Выполнить разработку приложения с использованием языка программирования Kotlin.

Задание. Создать классы Пассажир эконом класса и Пассажир первого класса, наследующие от класса Пассажир. Добавить взаимодействие класса Пассажир первого класса с Классом Стюардесса.

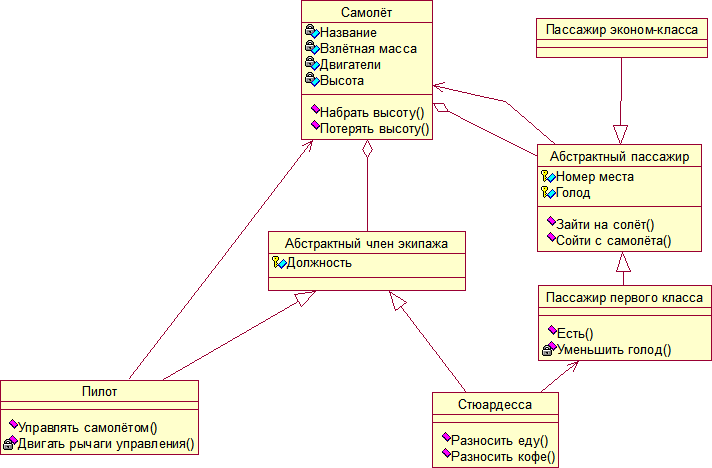


Диаграмма классов

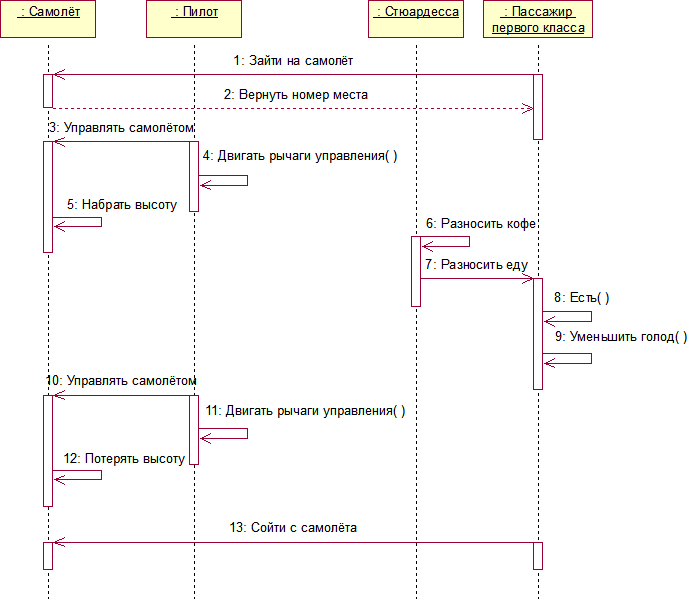


Диаграмма последовательности

**Код на языке Kotlin:**

**Main.kt**

***interface Passenger {  
 val seatNumber: Any  
 var hungry: Int  
 fun gotoPlain()  
 fun goOutPlain()  
 fun hungry()  
 fun hungryUp()  
 fun chooseFreedomSeatNumber()  
}  
 interface Plane {  
 val name: String  
 val weight: Double  
 val engine: Int  
 var height: Int  
 var numberOfPassenger: Int  
 fun heightUp()  
 fun heightLost()  
 fun pass(): Int  
}  
 interface MemberOfPlane {  
 val jobTitle: String  
 val rang: Int  
}  
  
  
fun main() {  
 val plane = AbstractPlane("Kotlinich", 4.0, 0, 0)  
 println("Саммолет прибыл к месту приема пассажиров")  
 val pilot = Pilot("Пилот", 1)  
 println("${pilot.jobTitle} взошел на борт")  
 val stewardess = Stewardess("Стюардесса", 2)  
  
 val p1 = FirstClassPassenger(plane.pass(), 0)  
 p1.gotoPlain()  
 p1.chooseFreedomSeatNumber()  
 p1.hungryUp()  
 val p2 = FirstClassPassenger(plane.pass(), 0)  
 p2.gotoPlain()  
 p2.chooseFreedomSeatNumber()  
 p2.hungryUp()  
 val p3 = FirstClassPassenger(plane.pass(), 0)  
 p3.gotoPlain()  
 p3.chooseFreedomSeatNumber()  
 p3.hungryUp()  
 val p4 = FirstClassPassenger(plane.pass(), 0)  
 p4.gotoPlain()  
 p4.chooseFreedomSeatNumber()  
 p4.hungryUp()  
 val p5 = passengereconommy(plane.pass(),0)  
 p5.gotoPlain()  
 p5.chooseFreedomSeatNumber()  
 p5.hungryUp()  
 pilot.managePlane(plane,2)  
 pilot.open()  
 stewardess.giveCoffee()  
 stewardess.giveFood(p1)  
 stewardess.giveFood(p2)  
 stewardess.giveFood(p3)  
 stewardess.giveFood(p4)  
 p1.hungryUp()  
 p2.hungryUp()  
 p3.hungryUp()  
 p4.hungryUp()  
 p5.hungryUp()  
 pilot.managePlane(plane,1)  
 pilot.open()  
 p1.goOutPlain()  
 p2.goOutPlain()  
 p3.goOutPlain()  
 p4.goOutPlain()  
 p5.goOutPlain()  
 println("${pilot.jobTitle} перестал управлять самолетом и сошел с борта")  
}***

**passenger econommy class.kt**

***class passengereconommy(seatNumber: Any, hungry: Int): AbstractPassenger(seatNumber, hungry) {  
}***

**AbstractMemberOfPlane.kt**

***abstract class AbstractMemberOfPlane( val jobTitle: String, val rang: Int = 0)  
  
class Pilot(jobTitle: String, rang: Int) : AbstractMemberOfPlane(jobTitle, rang) {  
 fun managePlane(plane: AbstractPlane,number:Int) {  
 println("Пилот управляет самолетом")  
 if(number == 1){  
 plane.heightLost()  
 }  
 else if( number == 2){  
 plane.heightUp()  
 }  
 else{  
 println("команда пилота не распознана")  
 }  
 }  
 public fun open():Int{  
 val number = 0  
 switchLever()  
 return number  
 }  
 private fun switchLever(): Int {  
 val number = 0  
 println("Пилот двигает рычаги управления")  
 return number  
 }  
}***

**AbstractPassenger.kt**

***open class AbstractPassenger(protected val seatNumber: Any, public var hungry: Int = 0) {  
 fun gotoPlain() {  
 println("Пассажир вошел на борт")  
 }  
  
 fun goOutPlain() {  
 println("Пассажир покинул место $seatNumber и сошел с борта")  
 }  
  
 fun hungry() {  
 if (hungry >= 30 && hungry < 60) {println("Пассажир $seatNumber голодает")}  
 else if (hungry >= 60) {println("Пассажир $seatNumber сходит с ума от голода")}  
 }  
  
 fun hungryUp() {  
 hungry += 10  
 }  
  
 fun chooseFreedomSeatNumber() {  
 println("Пассажир занял место $seatNumber")  
 }  
}***

**AbstractPlane.kt**

***class AbstractPlane(private val name: String, private val weight: Double, private val engine: Int, private var height: Int, private var numberOfPassenger: Int = 0) {  
 fun heightUp() {  
 height += 100  
 println("Самолет поднялся на высоту $height метров от уровня моря")  
 }  
  
 fun heightLost() {  
 height -= 100  
 println("Самолет опустился на высоту $height метров от уровня моря")  
 }  
  
 fun pass(): Int {  
 numberOfPassenger++  
 return numberOfPassenger  
 }  
}***

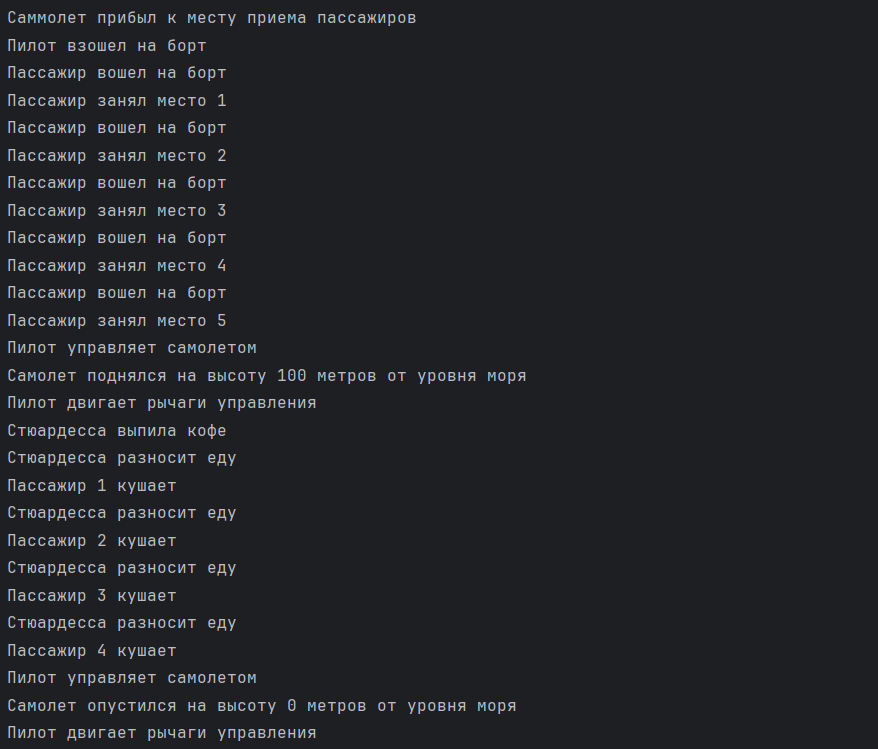
**FirstClassPassenger.kt**

***open class FirstClassPassenger(seatNumber: Any, hungry: Int) : AbstractPassenger(seatNumber, hungry) {  
 private fun toEat() {  
 hungry -= 30  
 }  
 public fun open(){  
 println("Пассажир $seatNumber кушает")  
 toEat()  
 }  
}***

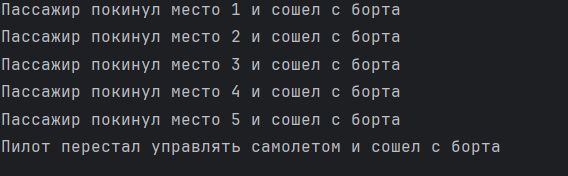
**Stewardess.kt**

***open class Stewardess(jobTitle: String, rang: Int) : AbstractMemberOfPlane(jobTitle, rang) {  
 fun giveFood(firstClassPassenger: FirstClassPassenger) {  
 println("$jobTitle разносит еду")  
 firstClassPassenger.open()  
 }  
 fun giveCoffee() {  
 println("$jobTitle выпила кофе")  
  
 }  
  
 }***

**Скриншот работающей программы:**

****

**Рисунок 1 - Результат выполнения программы**

****

**Рисунок 2 - Результат выполения программы**

**Вывод: В этой лабораторной работе мы получили базовые знания о языке программирования Kotlin и написали программу на этом языке, разбив ее функционал на несколько пакетов с учетом логики и сделав валидацию вводимых значений.**