**vНЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ** **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ** **«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ** **“СИНЕРГИЯ”»**

|  |  |  |
| --- | --- | --- |
| **Факультет/Институт** |  | Программирование |
|  |  | (наименование факультета/ Института) |
| **Направление/специальность** |  | Информационные системы и программирование |
| **подготовки:** |  | (код и наименование направления /специальности подготовки) |
| **Форма обучения:** |  | Очная |
|  |  | (очная, очно-заочная, заочная) |
|  |  |  |

**Отчет по практической работе №5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **на тему** |  | Проектирование классов | | |
|  |  | (наименование темы) | | |
|  |  |  | | |
| **по дисциплине** | | |  | Разработка программных модулем |
|  | | |  | (наименование дисциплины) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Обучающийся** |  | Ходоров Дмитрий Михайлович |  |  |
|  |  | (ФИО) |  | (подпись) |
| **Группа** |  | VДКИП-111прог |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Преподаватель** |  | Сибирев И. В. |  |  |
|  |  | (ФИО) |  | (подпись) |

**Москва 2025 г.**

# Аннотация

По согласованию во время вебинара работы выполнена на языке Kotlin т.к. он используется в работе и лучше мне знаком

# Задание №1(лабораторная работа 4, вариант 5)

Исходный код программы приведен в Листинге 1, скриншот работы программы на Рисунке 1

import kotlin.math.abs

//Вариант 5, лабораторная работа 4

class Complex(private val real: Double, private val imaginary: Double) {

operator fun plus(other: Complex): Complex {

return Complex(

real + other.real,

imaginary + other.imaginary

)

}

operator fun minus(other: Complex): Complex {

return Complex(

real - other.real,

imaginary - other.imaginary

)

}

operator fun times(other: Complex): Complex {

return Complex(

real \* other.real - imaginary \* other.imaginary,

real \* other.imaginary + imaginary \* other.real

)

}

override fun toString(): String {

return when {

imaginary == 0.0 -> "$real"

real == 0.0 -> "${imaginary}i"

else -> "$real ${if (imaginary < 0) "-" else "+"} ${abs(imaginary)}i"

}

}

}

val a = Complex(2.0, 3.0)

val b = Complex(4.0, 5.0)

val c = Complex(-1.0, 2.0)

val d = Complex(3.0, -4.0)

println("Сложение: $a + $b = ${a + b}")

println("Вычитание: $b - $a = ${b - a}")

println("Умножение: $c \* $d = ${c \* d}")

Листинг 1 - Исходный код программы

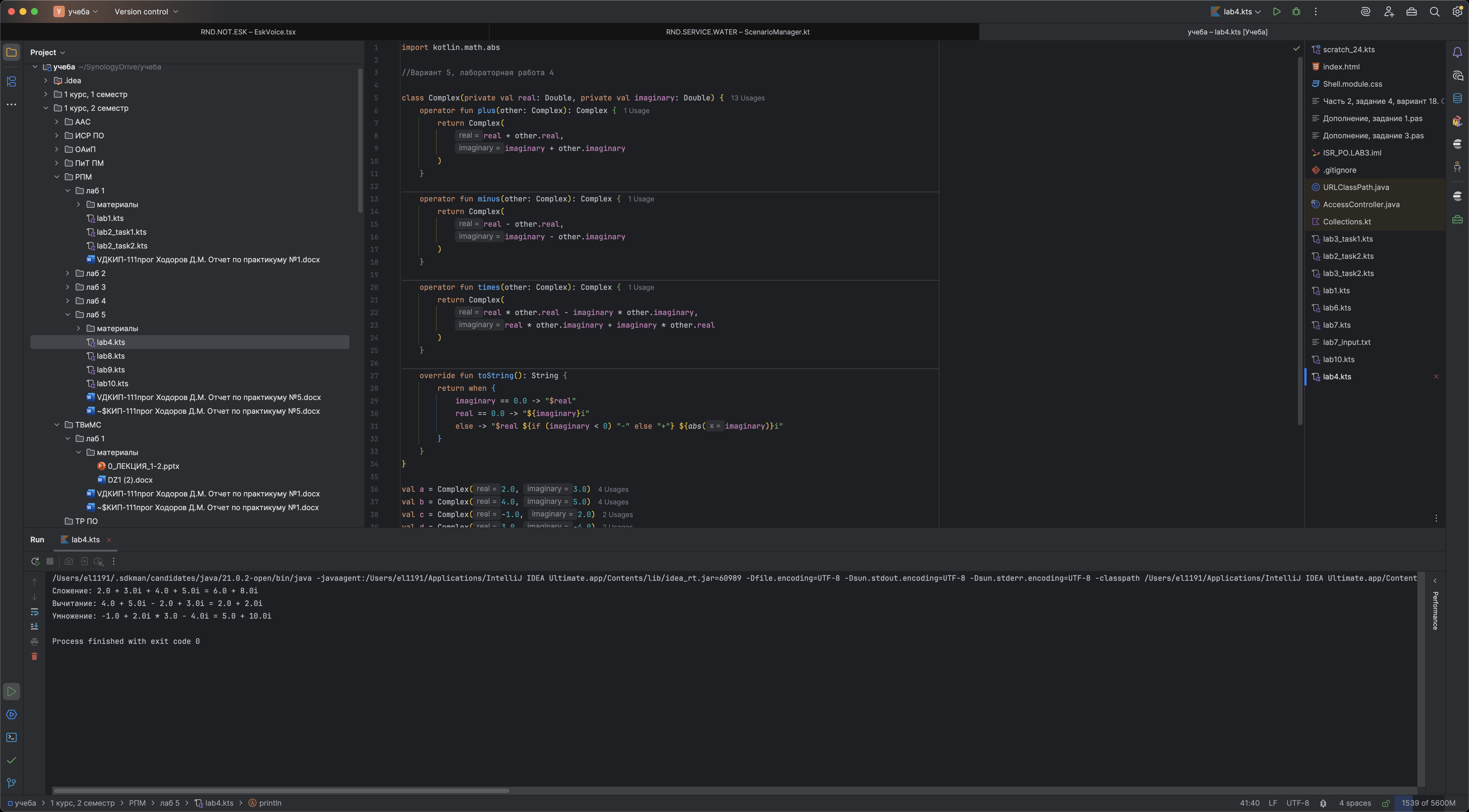


Рисунок 1 - Результат работы программы

# Задание №2(лабораторная работа 8, вариант 5)

Исходный код программы приведен в Листинг 2, скриншот работы программы на Рисунке 2

import kotlin.math.pow

//Вариант 5, лабораторная работа 8

class OctalNumber(val value: String) {

init {

require(value.matches(Regex("[0-7]+"))) { "Недопустимые символы в восьмеричном числе" }

}

constructor(decimal: Int) : this(decimal.toOctalString())

override fun equals(other: Any?): Boolean {

if (other !is OctalNumber) return false

return this.toDecimal() == other.toDecimal()

}

operator fun compareTo(other: OctalNumber): Int {

return this.toDecimal().compareTo(other.toDecimal())

}

fun toDecimal(): Int {

return value.foldIndexed(0) { index, acc, c ->

val power = value.length - index - 1

acc + c.toString().toInt() \* 8.pow(power)

}

}

override fun toString(): String {

return "0o$value"

}

operator fun get(index: Int): Char {

require(index in 0 until value.length) { "Неверный индекс" }

return value[index]

}

private fun Int.pow(exponent: Int): Int {

return this.toDouble().pow(exponent.toDouble()).toInt()

}

companion object {

fun Int.toOctalString(): String {

var n = this

return if (n == 0) "0" else {

buildString {

while (n > 0) {

append(n % 8)

n /= 8

}

}.reversed()

}

}

}

}

val num1 = OctalNumber("17")

val num2 = OctalNumber(15) // 15 в десятичной = 17 в восьмеричной

val num3 = OctalNumber("25")

println("num1 == num2: ${num1 == num2}") // true

println("num1 < num3: ${num1 < num3}") // true

println("num3 в десятичной: ${num3.toDecimal()}") // 2\*8 + 5 = 21

println("Форматированный вывод num3: $num3") // 0o25

println("Цифра с индексом 1 в num3: ${num3[1]}") // '5'

Листинг 2 - Исходный код программы

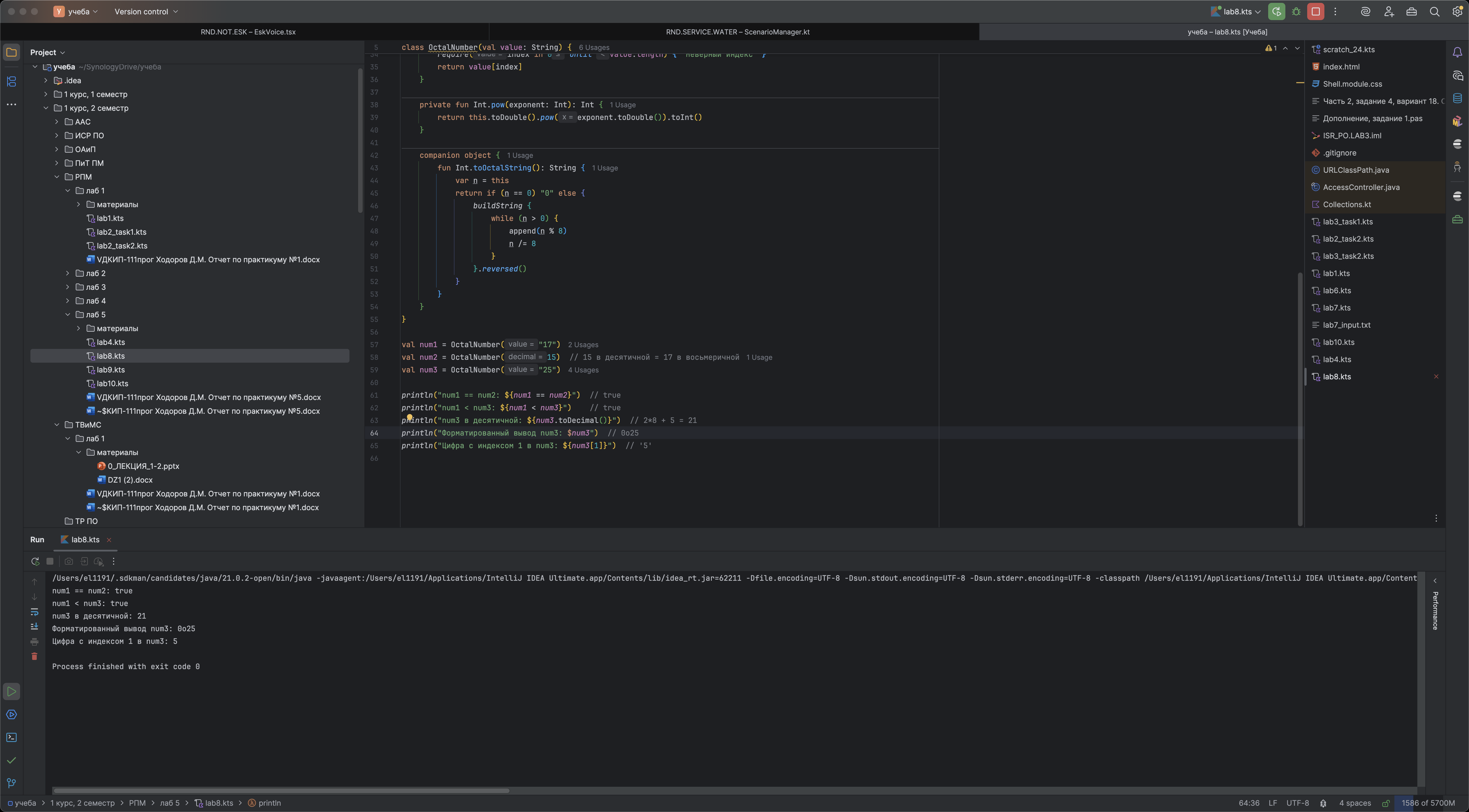


Рисунок 2 - Результат работы программы

# Задание №3(лабораторная работа 9, вариант 5)

Исходный код программы приведен в Листинге 3, скриншот работы программы на Рисунке 3. Класс Строка использовался из стандартной библиотеки языка

//Вариант 5, лабораторная работа 9

class BitString(val value: String) {

fun complement(): BitString {

val inverted = value.map { if (it == '0') '1' else '0' }.joinToString("")

val one = "1".padStart(value.length, '0')

val sum = addBinary(inverted, one)

return BitString(sum.takeLast(value.length))

}

operator fun plus(other: BitString): BitString {

val (a, b) = normalize(this, other)

return BitString(addBinary(a.value, b.value))

}

override fun equals(other: Any?): Boolean {

if (other !is BitString) return false

val (a, b) = normalize(this, other)

return a.value == b.value

}

override fun toString(): String = value

private fun addBinary(a: String, b: String): String {

val maxLen = maxOf(a.length, b.length)

// Дополняем нулями слева

val aPad = a.padStart(maxLen, '0')

val bPad = b.padStart(maxLen, '0')

val result = StringBuilder()

var carry = 0

for (i in maxLen - 1 downTo 0) {

val sum = (aPad[i] - '0') + (bPad[i] - '0') + carry

result.append(sum % 2)

carry = sum / 2

}

return result.reverse().toString()

}

private fun normalize(a: BitString, b: BitString): Pair<BitString, BitString> {

val maxLen = maxOf(a.value.length, b.value.length)

val aSign = if (a.value.isNotEmpty()) a.value[0] else '0'

val bSign = if (b.value.isNotEmpty()) b.value[0] else '0'

return Pair(

BitString(a.value.padStart(maxLen, aSign)),

BitString(b.value.padStart(maxLen, bSign))

)

}

}

val num1 = BitString("1010") // -6 (4-бит)

val num2 = BitString("0101") // +5

val num3 = BitString("101") // -3 (3-бит)

println("Дополнение 1010: ${num1.complement()}") // 0110

println("1010 + 0101 = ${num1 + num2}") // 1111 (переполнение)

val num5 = BitString("1101") // -3 (4-бит)

println("101 == 1101: ${num3 == num5}") // true

Листинг 3 - Исходный код программы

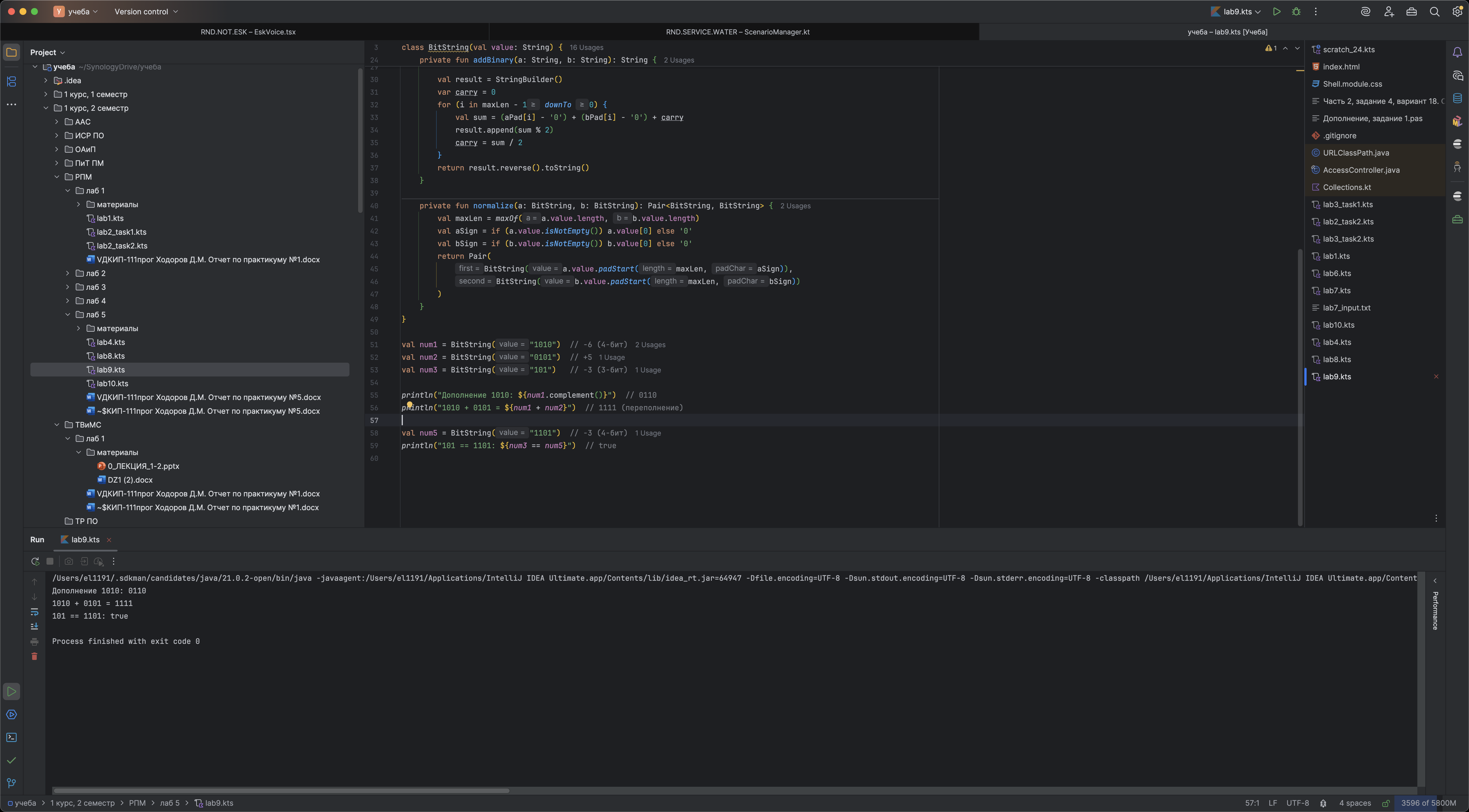


Рисунок 3 - Результат работы программы

# Задание №3(лабораторная работа 10, вариант 5)

Исходный код программы приведен в Листинге 4, скриншот работы программы на Рисунке 4

import java.util.\*

//Вариант 5, лабораторная работа 10

data class AEROFLOT(

val destination: String,

val flightNumber: Int,

val aircraftType: String

)

val scanner = Scanner(System.`in`)

val flights = Array(3) { AEROFLOT("", 0, "") }

// Ввод данных

println("Введите данные о 3 рейсах:")

for (i in 0 until 3) {

println("\nРейс ${i + 1}:")

print("Пункт назначения: ")

val destination = scanner.nextLine()

print("Номер рейса: ")

val flightNumber = scanner.nextInt()

scanner.nextLine() // Очистка буфера

print("Тип самолета: ")

val aircraftType = scanner.nextLine()

flights[i] = AEROFLOT(destination, flightNumber, aircraftType)

}

// Сортировка по алфавитному порядку пунктов назначения

flights.sortBy { it.destination }

// Запрос типа самолета для поиска

print("\nВведите тип самолета для поиска рейсов: ")

val searchType = scanner.nextLine()

// Поиск и вывод рейсов для указанного типа самолета

val matchingFlights = flights.filter { it.aircraftType.equals(searchType, ignoreCase = true) }

println("\nРезультаты поиска:")

if (matchingFlights.isNotEmpty()) {

println("Рейсы, обслуживаемые самолетом типа '$searchType':")

matchingFlights.forEach {

println("Пункт назначения: ${it.destination}, Номер рейса: ${it.flightNumber}")

}

} else {

println("Рейсов, обслуживаемых самолетом типа '$searchType', не найдено.")

}

Листинг 4 - Исходный код программы

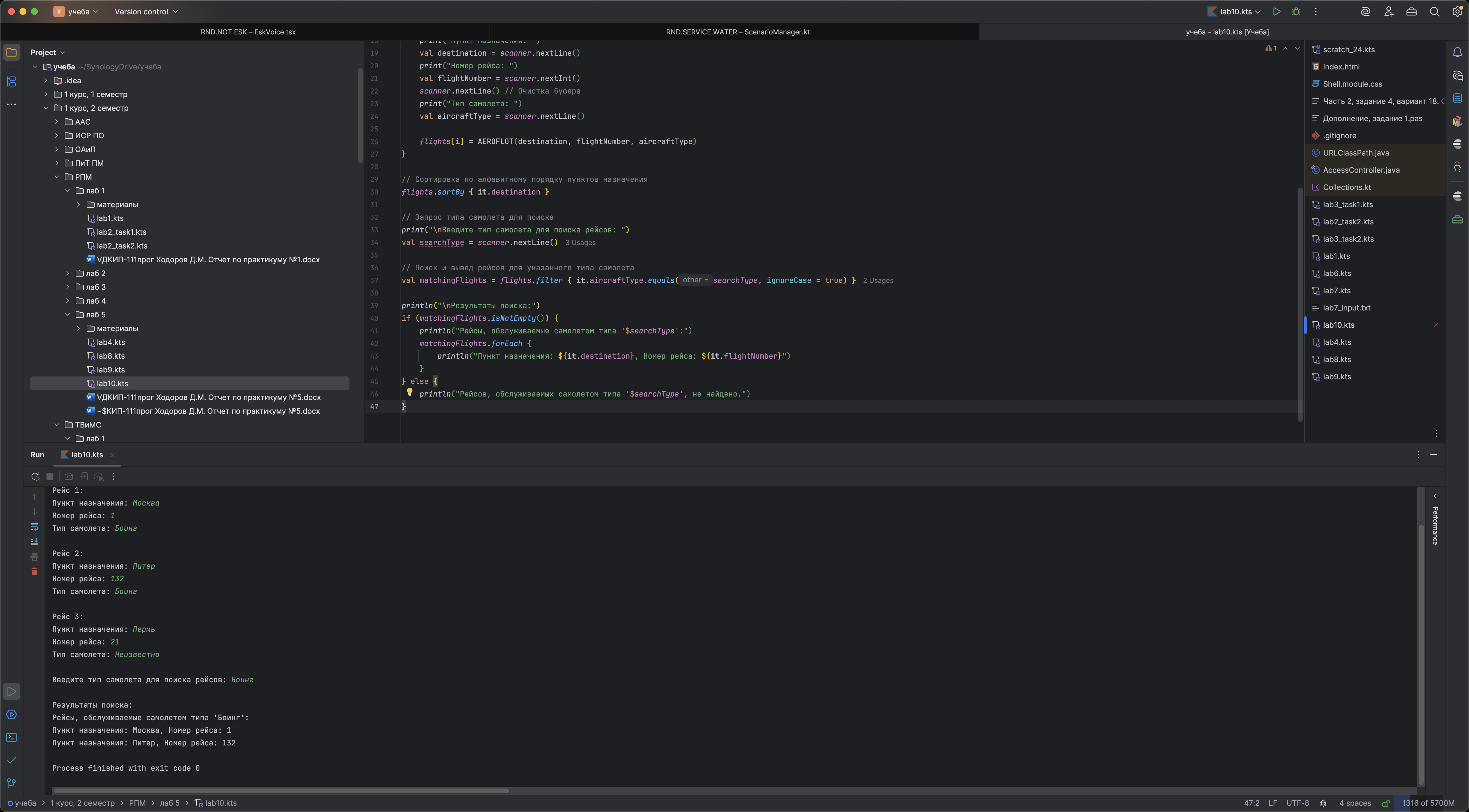


Рисунок 4 - Результат работы программы