|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 6**

**Вариант № 12**

**Название:** Коллекции

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Т.М. Курохтин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

**Цель:** изучить и освоить работу с коллекциями в Java.

**Задание 1:** определить множество на основе множества целых чисел. Создать методы для определения пересечения и объединения множеств.

Код программы:

class IntegerSet(private val elements: Set<Int>) {

// Конструктор для создания множества из списка целых чисел

constructor(vararg elements: Int) : this(elements.toSet())

// Метод для объединения множеств

fun union(other: IntegerSet): IntegerSet {

val unionSet = this.elements union other.elements

return IntegerSet(unionSet)

}

// Метод для пересечения множеств

fun intersection(other: IntegerSet): IntegerSet {

val intersectionSet = this.elements intersect other.elements

return IntegerSet(intersectionSet)

}

// Переопределяем метод toString для удобного отображения множества

override fun toString(): String {

return elements.joinToString(prefix = "{", postfix = "}")

}

}

fun main() {

// Примеры использования

val set1 = IntegerSet(1, 2, 3, 4, 5)

val set2 = IntegerSet(4, 5, 6, 7, 8)

println("Множество 1: $set1")

println("Множество 2: $set2")

val unionSet = set1.union(set2)

println("Объединение множеств: $unionSet")

val intersectionSet = set1.intersection(set2)

println("Пересечение множеств: $intersectionSet")

}

Результат работы программы представлен на рисунке 1.

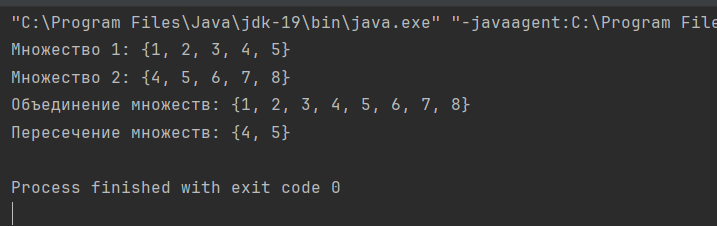


Рисунок 1 – Результат работы программы

**Задание 2:** Списки (стеки, очереди) I(1..n) и U(1..n) содержат результаты n измерений тока и напряжения на неизвестном сопротивлении R. Найти приближенное число R методом наименьших квадратов.

Код программы:

fun calculateR(iList: List<Double>, uList: List<Double>): Double {

require(iList.size == uList.size) { "Списки I и U должны быть одинакового размера" }

val n = iList.size

if (n == 0) {

throw IllegalArgumentException("Списки I и U не должны быть пустыми")

}

// Вычисляем суммы

val sumIU = iList.zip(uList).sumOf { it.first \* it.second }

val sumI2 = iList.sumOf { it \* it }

// Вычисляем R по формуле метода наименьших квадратов

return sumIU / sumI2

}

fun main() {

val iList = listOf(0.1, 0.2, 0.3, 0.4, 0.5)

val uList = listOf(1.1, 2.1, 3.1, 4.1, 5.1)

try {

val r = calculateR(iList, uList)

println("Приближенное значение сопротивления R: $r Ом")

} catch (e: IllegalArgumentException) {

println("Ошибка: ${e.message}")

} catch (e: Exception) {

println("Произошла ошибка: ${e.message}")

}

}

Результат работы программы показан на рисунке 2.

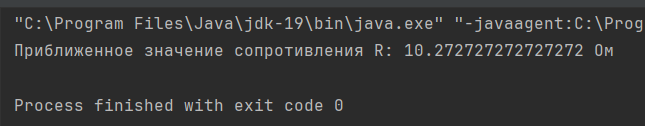


Рисунок 2 – Результат работы программы

**Задание 3:** На базе коллекций реализовать структуру хранения чисел с поддержкой следующих операций:

• добавление/удаление числа;

• поиск числа, наиболее близкого к заданному (т.е. модуль разницы минимален).

Код программы:

import kotlin.math.abs

class NumberStorage {

private val numbers: MutableList<Int> = mutableListOf()

// Добавление числа

fun add(number: Int) {

numbers.add(number)

}

// Удаление числа

fun remove(number: Int) {

numbers.remove(number)

}

// Поиск числа, наиболее близкого к заданному

fun findClosest(number: Int): Int? {

if (numbers.isEmpty()) return null

var closest = numbers[0]

var minDifference = abs(number - closest)

for (num in numbers) {

val difference = abs(number - num)

if (difference < minDifference) {

closest = num

minDifference = difference

}

}

return closest

}

}

fun main() {

val storage = NumberStorage()

storage.add(10)

storage.add(20)

storage.add(30)

storage.add(40)

println("Наиболее близкое число к 24: ${storage.findClosest(24)}")

println("Наиболее близкое число к 32: ${storage.findClosest(32)}")

println("Наиболее близкое число к 3: ${storage.findClosest(3)}")

storage.remove(20)

println("Наиболее близкое число к 24 после удаления 20: ${storage.findClosest(24)}")

}

Результат работы программы представлен на рисунке 3.

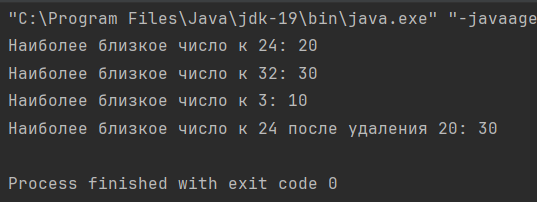
****

Рисунок 3 – Результат работы программы

**Задание 4:** 2. Реализовать класс, моделирующий работу N-местной автостоянки. Машина подъезжает к определенному месту и едет вправо, пока не встретится свободное место. Класс должен поддерживать методы, обслуживающие приезд и отъезд машины.

Код программы:

class ParkingLot(private val capacity: Int) {

private val slots: MutableList<Boolean> = MutableList(capacity) { false }

fun arrive(): Boolean {

for (i in 0..<capacity) {

if (!slots[i]) {

slots[i] = true

println("Машина припарковалась на месте $i")

return true

}

}

println("Нет свободных мест для парковки")

return false

}

fun depart(position: Int): Boolean {

if (position !in 0..<capacity) {

println("Неверная позиция: $position")

return false

}

if (slots[position]) {

slots[position] = false

println("Машина покинула место $position")

return true

}

println("Место $position уже свободно")

return false

}

fun printStatus() {

println("Состояние автостоянки: ${slots.joinToString(" ") { if (it) "X" else "\_" }}")

}

}

fun main() {

val parkingLot = ParkingLot(10)

parkingLot.printStatus()

parkingLot.arrive()

parkingLot.printStatus()

parkingLot.arrive()

parkingLot.printStatus()

parkingLot.depart(0)

parkingLot.printStatus()

parkingLot.depart(0)

parkingLot.printStatus()

parkingLot.arrive()

parkingLot.printStatus()

parkingLot.arrive()

parkingLot.printStatus()

parkingLot.arrive()

parkingLot.printStatus()

}

Результат работы программы показан на рисунке 4.

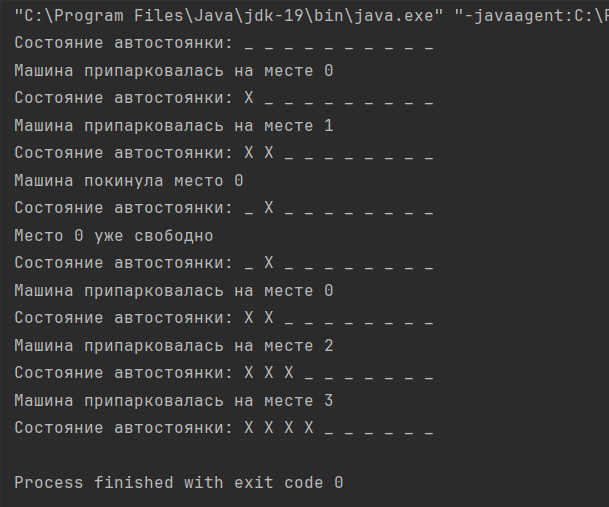


Рисунок 4 – Результат работы программы

**Вывод:** была освоена работа с коллекциями в Java.