Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Отчёт

по лабораторной работе №3

на тему:

**ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ И ЛЯМБДА-ВЫРАЖЕНИЯ**

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ф.В. Усенко

(подпись)

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. Ю. Позняк

(подпись) гр. 214302

Минск, 2024

**ЗАДАНИЕ**

Создание калькулятора с динамическими операциями: Напишите программу, которая позволяет пользователю добавлять новые арифметические операции в виде лямбда-выражений и применять их к выражениям. Программа должна поддерживать сохранение и загрузку этих операций.

**РЕШЕНИЕ**

import java.io.File

import java.io.ObjectInputStream

import java.io.ObjectOutputStream

import java.util.Scanner

typealias Operation = (Double, Double) -> Double

class OperationNotFoundException(message: String): Exception(message)

class DynamicCalculator {

private val operations = mutableMapOf<String, Operation>()

fun addOperation(name: String, operation: Operation) {

operations[name] = operation

}

@Throws(OperationNotFoundException::class)

fun applyOperation(name: String, a: Double, b: Double): Double? {

val operation = operations[name] ?: throw OperationNotFoundException("Операция '$name' не найдена.")

return operation.invoke(a, b)

}

fun saveOperations(filename: String) {

ObjectOutputStream(File(filename).outputStream()).use { it.writeObject(operations) }

}

fun loadOperations(filename: String) {

ObjectInputStream(File(filename).inputStream()).use {

@Suppress("UNCHECKED\_CAST")

operations.putAll(it.readObject() as Map<String, Operation>)

}

}

fun listOperations() {

operations.keys.forEach { println(it) }

}

}

fun main() {

val scanner = Scanner(System.`in`)

val calculator = DynamicCalculator()

while (true) {

println("Выберите опцию: [1] Добавить операцию, [2] Применить операцию, [3] Сохранить операции, [4] Загрузить операции, [5] Выйти")

when (scanner.nextInt()) {

1 -> {

println("Введите имя операции:")

val name = scanner.next()

println("Введите операцию (пример: + для сложения):")

val symbol = scanner.next()

when (symbol) {

"+" -> calculator.addOperation(name) { a, b -> a + b }

"-" -> calculator.addOperation(name) { a, b -> a - b }

"\*" -> calculator.addOperation(name) { a, b -> a \* b }

"/" -> calculator.addOperation(name) { a, b -> if (b != 0.0) a / b else Double.NaN }

else -> println("Неизвестная операция")

}

println("Операция добавлена")

}

2 -> {

println("Доступные операции:")

calculator.listOperations()

println("Введите имя операции:")

val name = scanner.next()

println("Введите два числа:")

val a = scanner.nextDouble()

val b = scanner.nextDouble()

try {

val result = calculator.applyOperation(name, a, b)

println("Результат: $result")

} catch (e: OperationNotFoundException) {

println(e.message)

}

}

3 -> {

println("Введите имя файла для сохранения:")

val filename = scanner.next()

calculator.saveOperations(filename)

println("Операции сохранены")

}

4 -> {

println("Введите имя файла для загрузки:")

val filename = scanner.next()

calculator.loadOperations(filename)

println("Операции загружены")

}

5 -> return

else -> println("Неверная опция")

}

}

}

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Как объявить функцию в *kotlin*? В чем разница между обычной функцией и однострочной функцией?

Функции в *kotlin* объявляются с использованием ключевого слова *fun*, после которого следует имя функции, параметры и возвращаемый тип. Если функция состоит из одного выражения, её можно объявить как однострочную, без использования *return. Fun multiply(a: int, b: int) = a \* b*

1. Что такое функция высшего порядка, и как её использовать? Приведите пример.

Функции, принимающие другие функции: функции высшего порядка принимают другие функции как параметры. Это позволяет делать код более гибким и расширяемым.

Функции, возвращающие функции: функции высшего порядка могут также возвращать другие функции, что позволяет создавать динамически изменяемые функции.

1. Как передать функцию в качестве параметра другой функции? Приведите пример.

Fun calculate(x: int, y: int, operation: (int, int) -> int): int {

return operation(x, y)

}

Fun main() {

val result = calculate(4, 2) { a, b -> a + b }

println(result) // выводит 6

}

1. Чем отличается лямбда-выражение от обычной функции?

Лямбда-выражение – это функция без имени, которая может быть объявлена непосредственно в месте использования. Лямбды могут принимать параметры и возвращать значения.