Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Факультет информационных технологий и кибербезопасности

Кафедра информационных систем и программирования

**ОТЧЕТ**

по лабораторной работе №6.

Тема работы: «Тестирование методом серого ящика».

Наименование дисциплины: «Тестирование и отладка программного обеспечения».

Подготовил:

Студент группы 23-КБ-ПР1

Юданов М. Д.

Проверил:

Волик А. Г.

Краснодар

**2025**

**Цель работы**: изучить подход к тестированию методом серого ящика.

**Ход работы**:

1) Создать класс (в соответствии с вариантом задания из п.5), реализующий преобразование строки из одной системы счисления в другую.

2) Протестировать класс на основе метода серого ящика с использованием средств автоматизации.

3) Составить отчет о результатах проведенного тестирования.

Реализовать класс, преобразующий строку с числом из одной системы счисления в другую. Варианты реализуемых преобразований представлены ниже (в ячейках таблицы указан номер варианта).

****

**Код класса:**

object Convertator {

def convert2to16(binNumber: String): String = {

def convertNumberTo16(number: Int): String = {

if (number < 10) number.toString

else number match {

case 10 => "A"

case 11 => "B"

case 12 => "C"

case 13 => "D"

case 14 => "E"

case 15 => "F"

case \_ => "?" // на случай неожиданных значений

}

}

def convert2to10(binNumber: String): Option[Int] = {

val binNumberLen = binNumber.length

var decimalNumber = 0

for (i <- 0 until binNumberLen) {

val char = binNumber.charAt(i)

// проверяем, что символ 0 или 1

if (char != '0' && char != '1') {

return None

}

val digit = char.asDigit

val exponent = binNumberLen - 1 - i

val powerOfTwo = 1 << exponent

// проверка на переполнение (аналог subtractingReportingOverflow)

val temp = decimalNumber + digit \* powerOfTwo

if (temp < 0) return None

decimalNumber = temp

}

Some(decimalNumber)

}

// конвертация 2 → 10

val decimalNumberOpt = convert2to10(binNumber)

if (decimalNumberOpt.isEmpty) return "nil"

var decimalNumber = decimalNumberOpt.get

if (decimalNumber == 0) return "0"

// конвертация 10 → 16

var hexNumber = ""

while (decimalNumber > 0) {

val remainder = decimalNumber % 16

hexNumber = convertNumberTo16(remainder) + hexNumber

decimalNumber /= 16

}

hexNumber

}

}

2. Напишем тесты. Проверим сценарии на основании подходов чёрного и белого ящика.

**Код тестирований:**

import org.scalatest.funsuite.AnyFunSuite

class ConvertatorTests extends AnyFunSuite {

var testData: Seq[(String, String)] = Seq.empty

def beforeEach(): Unit = {

testData = Seq.empty

}

def afterEach(): Unit = {

testData = Seq.empty

}

test("simple numbers") {

beforeEach()

testData = Seq(

("1001", "9"),

("1010", "A"),

("1011", "B"),

("1100", "C"),

("1101", "D"),

("1110", "E"),

("1111", "F")

)

for ((bin, expectedHex) <- testData) {

assert(Convertator.convert2to16(bin) == expectedHex, s"Ожидалось $expectedHex для $bin")

}

afterEach()

}

test("single complex number") {

beforeEach()

val binNum = "110111101"

val hexNum = Convertator.convert2to16(binNum)

assert(hexNum == "1BD", s"Ожидалось 1BD для $binNum, но получено $hexNum")

afterEach()

}

test("non-existent number (contains invalid characters)") {

beforeEach()

val binNum = "101010i01010"

val hexNum = Convertator.convert2to16(binNum)

assert(hexNum == "nil", s"Ожидалось nil для $binNum, но получено $hexNum")

afterEach()

}

test("number overflow") {

beforeEach()

val binNum = "111111111111111111111111111111111111111111111111111111111111111111"

val hexNum = Convertator.convert2to16(binNum)

assert(hexNum == "nil", s"Ожидалось nil для $binNum (переполнение), но получено $hexNum")

afterEach()

}

}

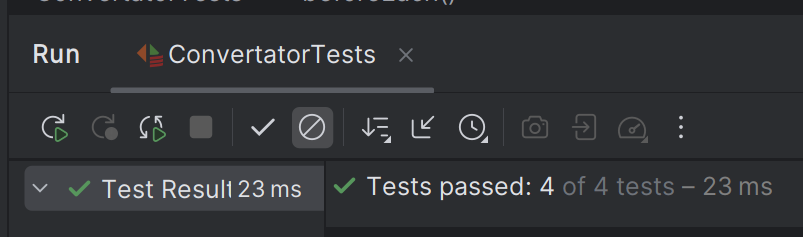


Рисунок 1 – Результаты тестирований.

4. Делаю вывод касательно проделанной работы.

**Вывод:** в ходе выполнения лабораторной работы я изучил подход к тестированию методом серого ящика.