Министерство науки и высшего образования РФ

Севастопольский государственный университет

Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №6

на тему «Исследование способов профилирования программного обеспечения»

по дисциплине «Тестирование программного обеспечения»

Выполнил:

Студент группы ИС/б 17-2-о

Черняев Н.Г.

Проверил:

Тлуховская Н.П.

г. Севастополь 2019

Цель работы

Исследовать критические по времени выполнения участки программного кода и возможности их устранения. Приобрести практические навыки анализа программ с помощью профайлера JProfiler.

Вариант №18

Разработать программу на основе библиотеки классов, реализованной и протестированной в предыдущей работе.

Программа должна как можно более полно использовать функциональность класса. При необходимости для наглядности профилирования в методы класса следует искусственно внести задержку выполнения.

Выполнить профилирование разработанной программы, выявить функции, на выполнение которых тратится наибольшее время.

Ход работы

В качестве тестируемого взаимодействия выберем лабораторную работу №5.

Класс LinesSumWithoutNegative.

**package** lab5;

**public** **class** LinesSumWithoutNegative {

**public** **int** Solve(**int**[][] matrix) {

**int** Sum = 0;

**for** (**int** i = 0; i < matrix.length; i++) {

**int** CurrentSum = 0;

**for** (**int** j = 0; j < matrix.length; j++) {

**if** (matrix[i][j] >=0 )

CurrentSum += matrix[i][j];

**else** {

CurrentSum = 0;

**break**;

}

}

Sum += CurrentSum;

}

**return** Sum;

}

}

Класс LinesSumWithoutNegativeTest.

**package lab5;**

**import static org.junit.jupiter.api.Assertions.\*;**

**import org.junit.jupiter.api.Test;**

**class LinesSumWithoutNegativeTest {**

**@Test**

**void test() {**

**LinesSumWithoutNegative test1 = new LinesSumWithoutNegative();**

**int output1 = test1.Solve(new int[][]{{2}});**

**LinesSumWithoutNegative test2 = new LinesSumWithoutNegative();**

**int output2 = test2.Solve(new int[][]{{2,2},{3,1}});**

**LinesSumWithoutNegative test3 = new LinesSumWithoutNegative();**

**int output3 = test3.Solve(new int[][]{{29,32,-2},{3,1,1},{1,2,3}});**

**LinesSumWithoutNegative test4 = new LinesSumWithoutNegative();**

**int output4 = test4.Solve(new int[][]{{20,92,-2},{-3,31,12},{6,-7,38}});**

**LinesSumWithoutNegative test5 = new LinesSumWithoutNegative();**

**int output5 = test5.Solve(new int[][]{{20,92,-2},{3,31,12},{6,-7,38}});**

**assertEquals(2, output1);**

**assertEquals(8, output2);**

**assertEquals(11, output3);**

**assertEquals(0, output4);**

**assertEquals(46, output5);**

**}**

**}**

Результаты

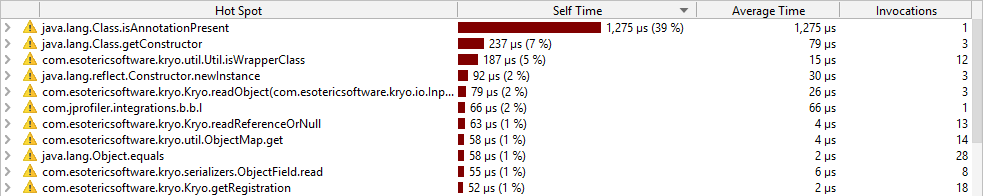


Рисунок 1 - Время выполнения основных методов программы

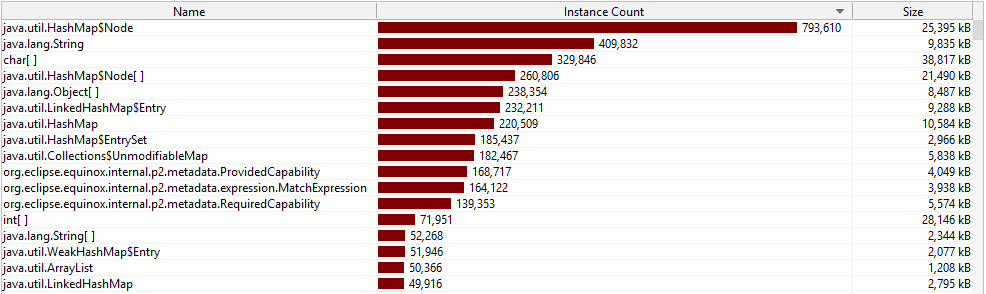


Рисунок 2 - Объем памяти, занимаемый основными объектами в программе

Вывод

Вданной лабораторной работе была исследованы критические по времени выполнения участки программного кода и возможности их устранения, а также приобретены практические навыки анализа программ с помощью профайлера JProfiler.