МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Сирота Марина Романовна

Институт информационных технологий и управления в технических системах

курс 3 группа ИС/б-32-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №6

по дисциплине: «Тестирование программного обеспечения»

по теме: «Исследование способов профилирования программного обеспечения»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. пр. Строганов В.А.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2018

1 ЦЕЛЬ РАБОТЫ

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

1. ПОСТАНОВКА ЗАДАЧИ
2. 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.
3. Выполнить профилирование разработанной программы, выявить функции, на выполнение которых тратится наибольшее время.
4. Модифицировать программу с целью оптимизации времени выполнения.
5. Выполнить повторное профилирование программы, сравнить новые результаты и полученные ранее, сделать выводы.

3 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Запустив профайлер JProfiler. Для наглядности профилирования в метод класса Area() искусственно внесли задержку выполнения. Построили граф вызовов (рисунок 3.1). Далее убрали искусственно созданную задержку выполнения и получили следующий граф вызовов – рисунок 3.2.

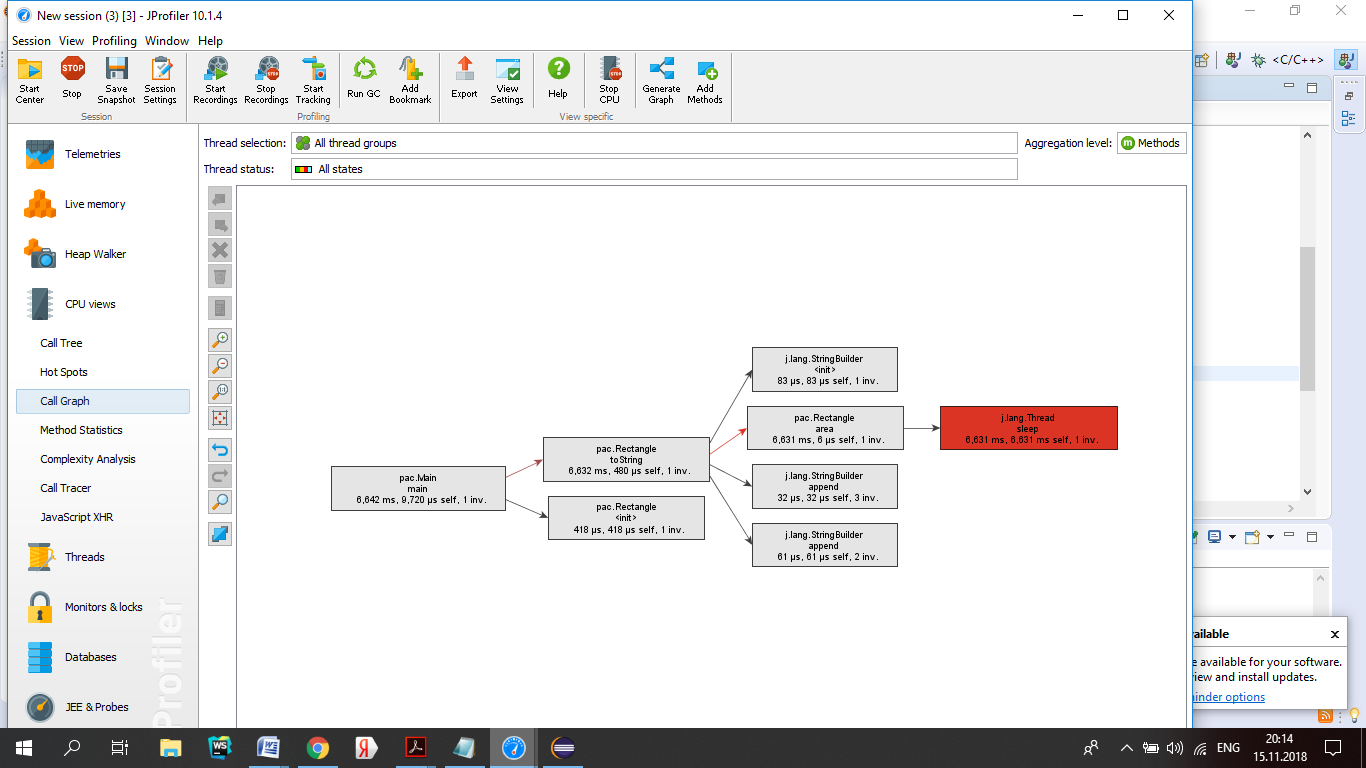


Рисунок 3.1 ‒ Граф вызовов 1

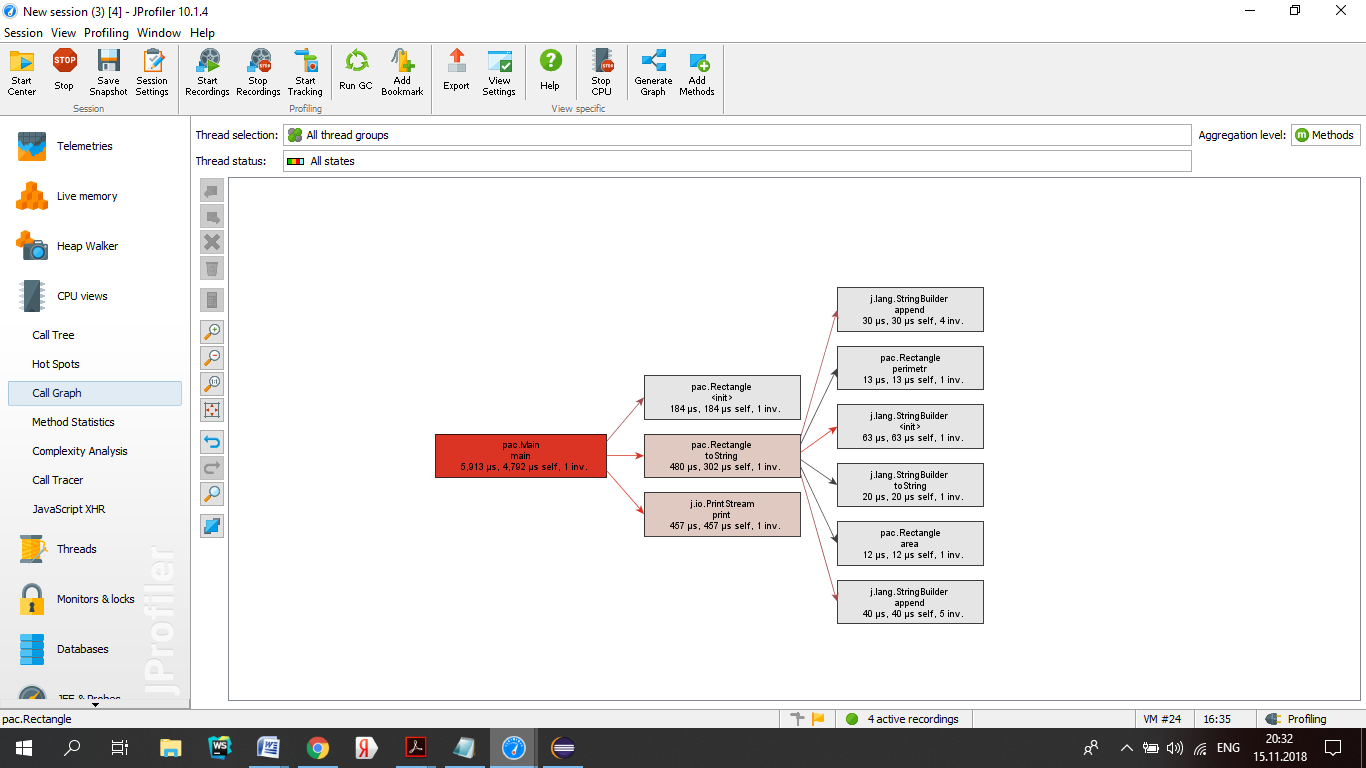


Рисунок 3.2 ‒ Граф вызовов 2

**package** pac;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Rectangle m1 = **new** Rectangle(7, 8);

String s = m1.toString();

System.***out***.print(s);

}

}

**package** pac;

**import** java.util.Scanner;

**public** **class** Rectangle **implements** RInterface {

**private** **int** width;

**private** **int** height;

**public** Rectangle(**int** w, **int** h) {

width = w;

height = h;

}

**public** **static** Rectangle read(Scanner scn) {

**return** **new** Rectangle(scn.nextInt(), scn.nextInt());

}

@Override

**public** **int** getWidth() {

**return** width;

}

@Override

**public** **int** getHeight() {

**return** height;

}

@Override

**public** **int** area() {

**return** width \* height;

}

@Override

**public** **int** perimetr() {

**return** (width + height) \* 2;

}

@Override

**public** String toString() {

StringBuilder sb = **new** StringBuilder();

sb.append("width: ");

sb.append(width);

sb.append("; height: ");

sb.append(height);

sb.append("; area: ");

sb.append(area());

sb.append("; perimetr: ");

sb.append(perimetr());

sb.append(";");

**return** sb.toString();

}

}

**package** pac;

**public** **interface** RInterface {

**int** getWidth();

**int** getHeight();

**int** area();

**int** perimetr();

}

ВЫВОДЫ

В ходе шестой лабораторной работы по дисциплине «Тестирование программного обеспечения» были изучены подходы профилированию программного обеспечения.

