Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «УрФУ имени первого Президента России Б.Н.Ельцина»

ИРИТ-РТФ

Центр ускоренного обучения

отчет ПО лабораторной работе N3

по дисциплине «Конструирование программного обеспечения»

Тема: МЕТРИКИ ХОЛСТЕДА

|  |  |
| --- | --- |
| Преподаватель, доц., к.т.н. | С.И. Тимошенко |
| Студент гр. РИЗ -490028у | А.А. Ахунов  В.Н. Жирнова  Д.А. Житников  Д.В. Федяков  А.А. Трифанов |

Екатеринбург 2022

содержание

[1 Постановка задачи 3](#_Toc115877596)

[2 Результаты проделанной работы 6](#_Toc115877597)

[2.1 Задание 1 6](#_Toc115877598)

[2.2 Задание 2 7](#_Toc115877599)

[2.3 Задание 3 10](#_Toc115877600)

[3 Анализ полученных результатов 15](#_Toc115877601)

# 1 Постановка задачи

Задание №1. Используя метрики Холстеда, оценить потенциальное число ошибок в программном обеспечении (ПО) для системы противоракетной обороны (ПРО) типа СОИ (стратегическая оборонная инициатива), энтузиастом которой был бывший президент США Рональд Рейган. Ограничиться расчетами только для одной боевой космической станции (БКС). Необходимые для расчета параметры взять из технического задания (ТЗ) на ПО для БКС:

1. число одновременно сопровождаемых целей равно 20;
2. для поражения каждой цели достаточно сделать 30 измерений каждого отслеживаемого параметра;
3. количество отслеживаемых параметров равно 10;
4. количество рассчитываемых параметров по каждой цели, передаваемых системе уничтожения, равно 3.

Для расчета понадобятся следующие метрики Холстеда:

1. потенциальный объем программы

,

где  – минимальное число различных операндов, в роли которого обычно выступает число независимых входных и выходных параметров;

1. уровень языка программирования  = 1,53 (соответствует языку АДА);
2. потенциальное число ошибок



Задание №2. Используя метрики Холстеда, выполнить для задачи, описанной в задании N 1, расчёты по формулам, описанным ниже:

1. Расчёт структурных параметров:
2. число модулей программного средства: ;
3. если k >> 8, то структура ПО будет многоуровневой (иерархической) с числом уровней в этом случае число модулей рассчитывается как .
4. Расчет длины программы:.
5. Расчет объема программного обеспечения: 
6. Расчет количества команд ассемблера: 
7. Расчет календарного времени программирования: 

где m – количество программистов в бригаде разработчиков; – производительность (число отлаженных команд в день), устанавливаемая директивно (обычно задается в пределах от 10 до 30); m и  задать самостоятельно.

1. Расчет потенциального количества ошибок: 
2. Расчет начальной надёжности ПО (времени наработки на отказ):  где *Тк* для соблюдения размерности следует перевести из количества дней в соответствующее количество часов (исходя из установленной длительности рабочего дня).

В отчете по лабораторной работе в анализе результатов необходимо пояснить причину разных значений для параметра В, полученных при выполнении задания №1 и №2.

Задание №3. Используя метрики Холстеда, оценить текущий рейтинг программиста и число ожидаемых потенциальных ошибок, которое он может допустить в разрабатываемом ПО. Для расчета использовать следующие рекуррентные соотношения:

1. рейтинг в i-й момент времени:

,

где n – количество написанных программистом программ за оцениваемый период;

m – количество программ, в которых найдены ошибки этого программиста;

Vj – объем j-й программы в Кбайт;

Bk – количество допущенных ошибок в к-й программе;

 – коэффициент считать обратно пропорциональным параметрам в скобках;

проверить следующие три варианта коэффициентов:

= 1/(λk+Rk),

= 1/(λk×Rk),

= 1/λk+1/Rk;

1. R0 считать равным 1000;
2. ожидаемое число ошибок равно;
3. считать уровень языка программирования = 1,53 (соответствует языку АДА);
4. за оцениваемый период программистом сделано 4 программы объемом 5, 7, 9 и 11 Кбайт соответственно; в трех предыдущих программах допущено соответственно 2, 5 и 4 ошибки; предполагается написать программу объемом 15 Кбайт.

# 2 Результаты проделанной работы

## 2.1 Задание 1

Была разработана программа, приведенная в листинге 1.

Листинг 1. Программа 1

public class exercise\_1 {

public static void main(String[] args) {

System.out.println("Задание №1");

Double a = 1.53;

Integer n2 = (20 \* 30 \* 10) + (20 \* 3);

System.out.println("Уровень языка программирования(a): " + a);

System.out.println("Минимальное число различных операндов(n): " + n2);

Double v = calcProgramScope(n2);

System.out.println("Потенциальный объем программы(V): " +v.intValue());

Double b = calcNumberErrors(v, a);

System.out.println("Потенциальное число ощибок(B): " + b.intValue());

}

//Потенциальный объем программы (V)

public static Double calcProgramScope(int n){

return (n + 2) \* (Math.log(n + 2) / Math.log(2));

}

//Потенциальное число ошибок (B)

public static Double calcNumberErrors(double v, double a){

return Math.pow(v, 2) / (3000 \* a);

}

}

Результат работы программы приведен на рисунке 1.

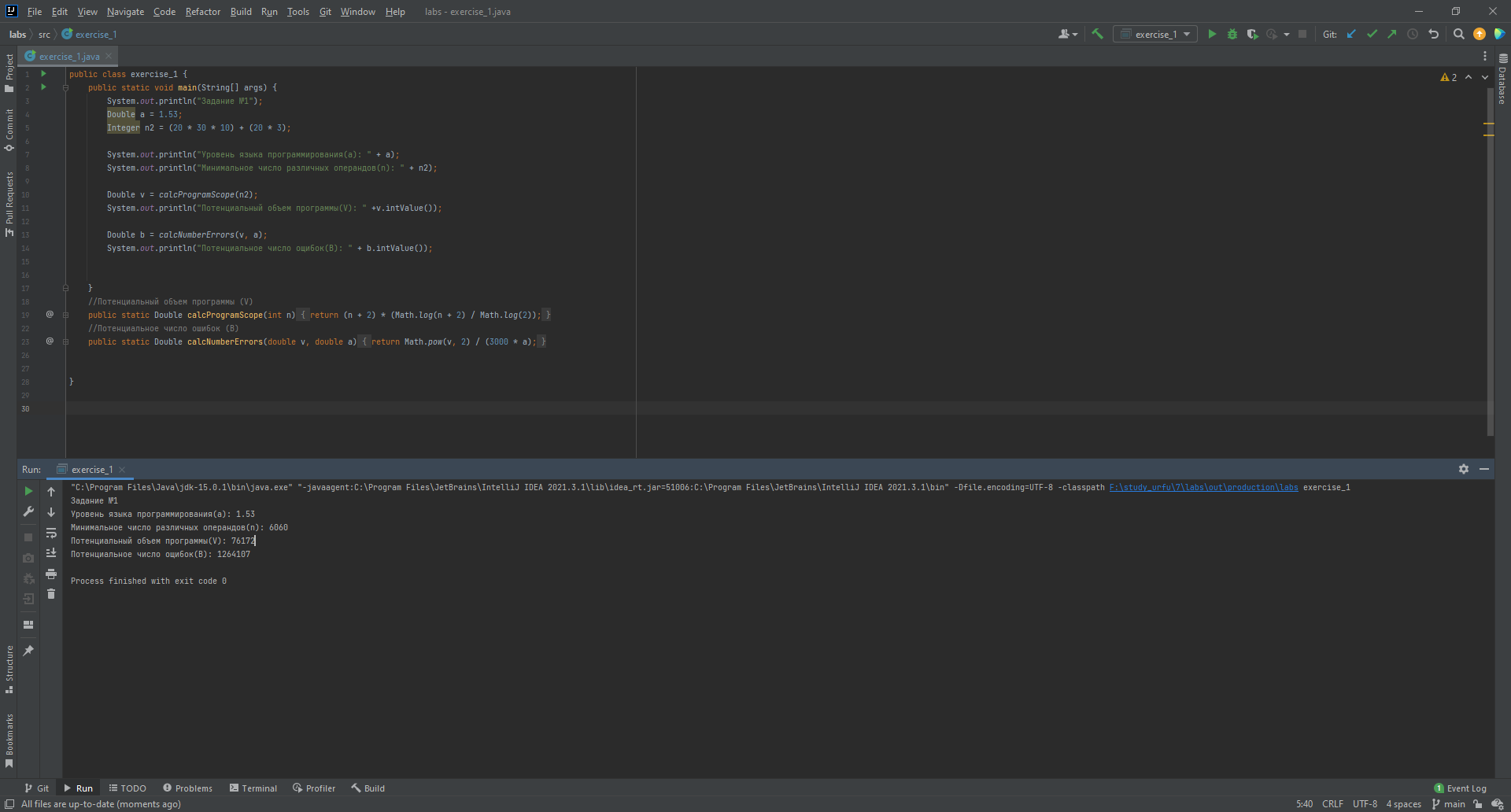


Рисунок 1 – Результат работы Программы 1

Данная программа вывела результат о том, что потенциальное число ошибок составляет 1 264 107 шт.

## 2.2 Задание 2

Была разработана программа из листинга 2.

Листинг 2. Программа 2

public class exercise\_2 {

public static void main(String[] args) {

System.out.println("Задание №2");

Integer n2 = (20 \* 30 \* 10) + (20 \* 3);

Double k = calcStructParam(n2.doubleValue());

System.out.println("Структурные параметры(K): " + k.intValue());

Double n = calcLengthProgram(k);

System.out.println("Длина программы(N): " + n.intValue());

Double v = calcProgramScope(k);

System.out.println("Объем программного обеспечения(V): " + v.intValue());

Double p = calcAssemblerInstruction(n);

System.out.println("Количество комманд ассемблера(P): " + p.intValue());

Double t = calcCalendarProgramming(n, 3, 15);

System.out.println("Календарное время программированя(T): " + t.intValue());

Double b = calcNumberErrors(v);

System.out.println("Расчет потенциального количества ошибок(B): " + b.intValue());

Double tn = calcSoftwareReliability(t, b);

System.out.println("Расчёт начальной надежности: " + tn.intValue());

}

//Потенциальный объем программы (V)

public static Double calcProgramScope(int n){

return (n + 2) \* (Math.log(n + 2) / Math.log(2));

}

//Потенциальное число ошибок (B)

public static Double calcNumberErrors(double v, double a){

return Math.pow(v, 2) / (3000 \* a);

}

//расчет структурных параметров (K)

public static Double calcStructParam(Double n){

if ( (n / 8) > 8)

return (n / 8) + (n / (Math.pow(8, 2)));

else

return n / 8;

}

//Расчет длины программы (N)

public static Double calcLengthProgram(Double k){

return (220 \* k) + k \* (Math.log(k) / Math.log(2));

}

//Расчет объема программного обеспечения (V)

public static Double calcProgramScope(Double k){

return k \* 220 \* (Math.log(48) / Math.log(2));

}

//Расчет количества команд ассемблера (P)

public static Double calcAssemblerInstruction(Double n){

return (3 \* n) / 8;

}

//Расчет календарного времени программирования(T)

public static Double calcCalendarProgramming(Double n, Integer m, Integer v){

return (3 \* n) / (8 \* m \* v);

}

//Расчет потенциального количества ошибок(B2)

public static Double calcNumberErrors(Double v){

return v / 3000;

}

//Расчет начальной надёжности ПО

public static Double calcSoftwareReliability(Double t, Double b){

return (t \* 8) / (2 \* Math.log(b));

}

}

Результат работы программы приведен на рисунке 2.

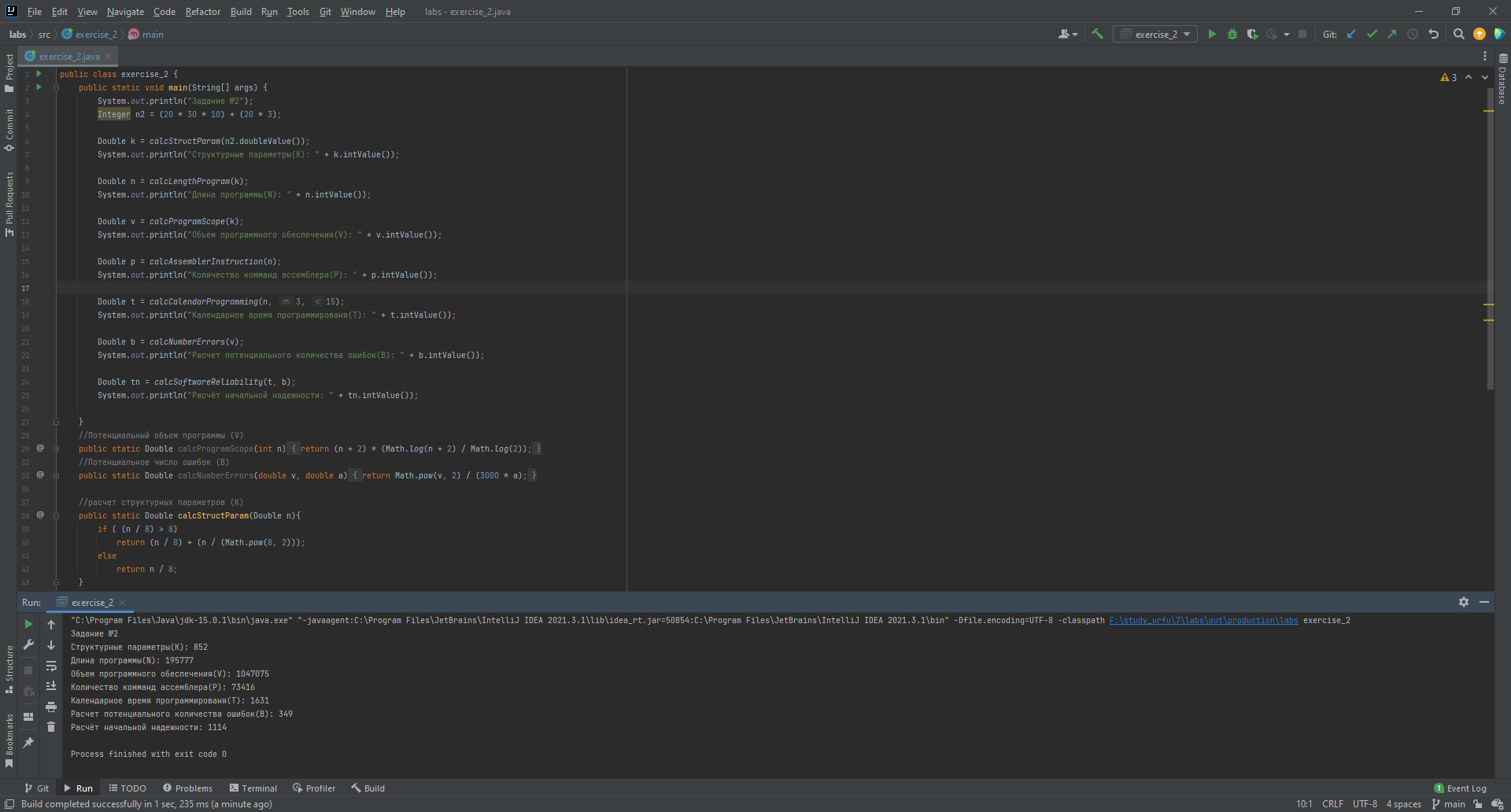


Рисунок 2 – Результат работы Программы 2

Данная программа в качестве результата вывела, что число модулей программного средства: 852 шт.; длина программы: 195 777 с.; объем программного обеспечения: 1 047 075 бит; количество команд ассемблера: 73 416 шт.; календарное время программирования: 1 631,48 д.; потенциальное количество ошибок: 349 шт. и начальная надёжность ПО составила 1 114,56 ч. наработки на отказ (система такое количество времени будет работать без сбоев).

## 2.3 Задание 3

Была разработана программа из листинга 3.

Листинг 3. Программа 3

public class Main {

public static void main(String[] args) {

List<Integer> sw = new ArrayList<Integer>() {{

add(5);

add(7);

add(9);

add(11);

}};

List<Integer> swWithErrors = new ArrayList<Integer>() {{

add(0);

add(2);

add(5);

add(4);

}};

double lambda = 1.53;

// Вычисляем для трех вариантов коэффициентов

// (адекватный результат получается только для третьего варианта)

for (int i = 1; i <= 3; ++i) {

double R = calcRating(sw, swWithErrors, lambda, i);

System.out.println("Рейтинг для варианта коэффициента " + i + ":");

System.out.println(String.format("%.4f", R));

// Вычисляем ожидаемое число ошибок

int nextB = (int)(getCoef(lambda, R, i) \* 15);

System.out.println("Ожидаемое число ошибок в следующей программе объемом 15 Кб: ");

System.out.println(String.format("%d шт.", nextB));

}

}

// Вычисление коэффициента c(lambda\_k, Rk), рассматриваем три варианта

public static double getCoef(double lambda, double Rk, int var) {

switch (var) {

case 1: {

return 1.0 / (lambda + Rk);

}

case 2: {

return 1.0 / (lambda \* Rk);

}

case 3: {

return 1.0 / lambda + 1.0 / Rk;

}

default: return 0;

}

}

// Рейтинг в i-й момент времени

// i = sw.size(), т.к. для n написанных программ можно вычислить рейтинг в n-ый момент времени

// размер swWithErrors предполагается таким же, т.е. по каждой программе есть данные о количестве ошибок в ней

public static Double calcRating(List<Integer> sw, List<Integer> swWithErrors, double lambda, int var) {

int i = sw == null ? 0 : sw.size();

// Базовый случай при i = 0: R0 = 1000

if (i == 0) return 1000.0;

// Рекурсивный вызов: вычисляем предыдущий рейтинг

// списки программ и ошибок в них сокращаются на 1 элемент

double prevR = calcRating(sw.subList(0, i - 1), swWithErrors.subList(0, i - 1), lambda, var);

// Сумма объемов программ

Integer vSum = sw.stream().reduce(0, Integer::sum);

// Сумма ошибок с коэффициентом c

double bSum = 0;

for (int k = 1; k <= i; k++) {

double bk = swWithErrors.get(k - 1); // количество ошибок

// Получаем коэффициент c, для его вычисления используем вычисленный по рекурсивной формуле рейтинг

// в момент времени k - 1

double c = getCoef(

lambda,

calcRating(sw.subList(0, k - 1), swWithErrors.subList(0, k - 1), lambda, var),

var

);

bSum += bk / c;

}

// Окончательно вычисляем рейтинг по формуле

return prevR \* (1.0 + 1E-3 \* (vSum - bSum));

}

Результат работы программы приведен на рисунке 3.

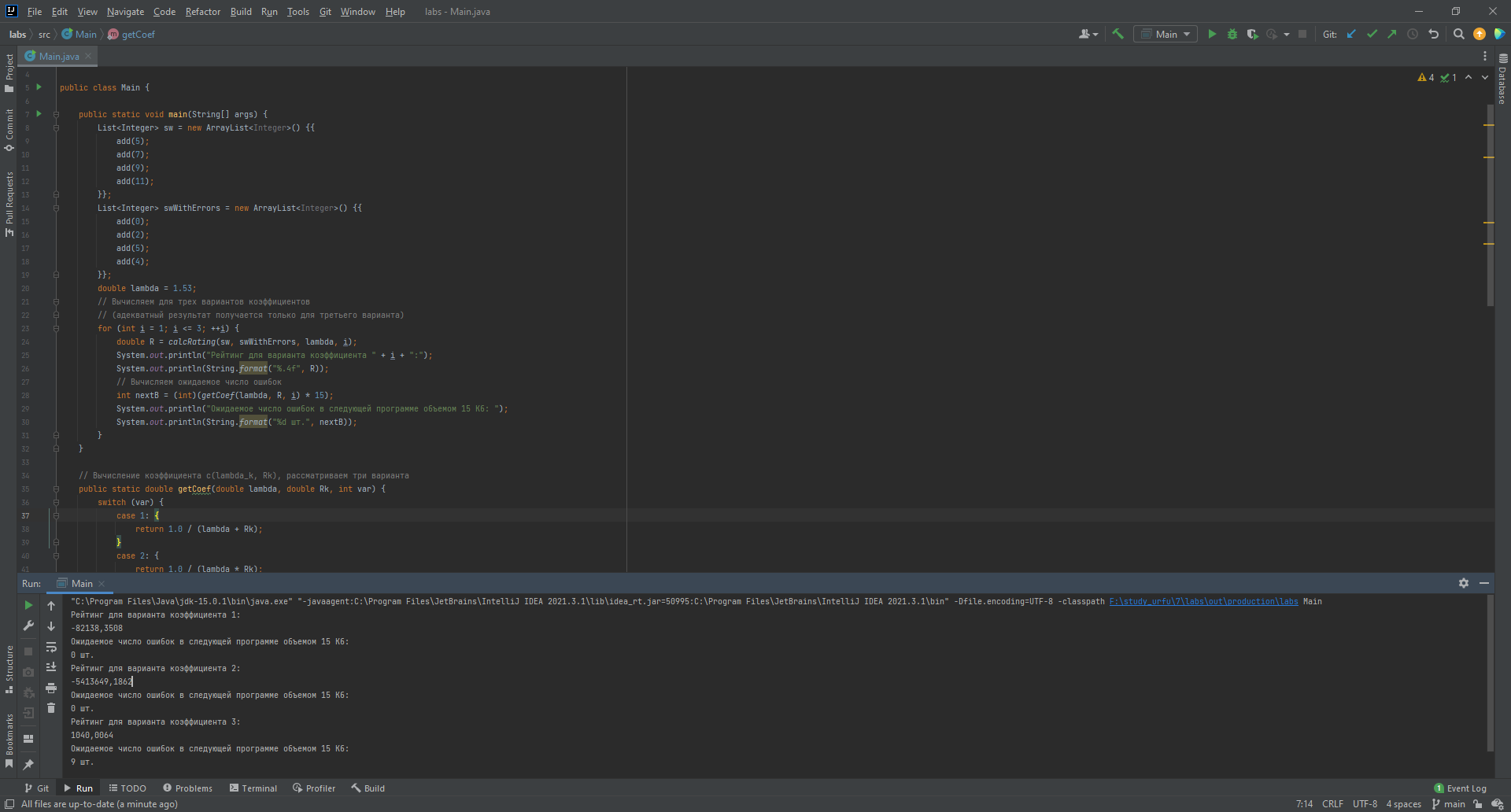


Рисунок 3 – Результат работы Программы 3

Программой было рассчитано, что рейтинг программиста при коэффициенте варианта 1 составил -82138,3508; а ожидаемое число ошибок в следующей программе объемом 15 Кб составило 0 шт. При варианте 2 коэффициента рейтинг составил - 5413649,1862; ожидаемое число ошибок в следующей программе объемом 15 Кб составило 0 шт. При варианте 3 коэффициента рейтинг составил 1040,0064; ожидаемое число ошибок в следующей программе объемом 15 Кб составило 9 шт.

# 3 Анализ полученных результатов

В результате работы были изучены материалы и разработано программное обеспечение, выполняющее расчеты при любых заданных входных параметрах для заданий №1 – №3, использующих метрики Холстеда.

По заданиями №1 и №2 следует отметить, что оптимальная структуризация (например, на модули, уровни) позволяет обеспечить значительное уменьшение потенциального количества ошибок. По этой причине значения параметра B, полученных при выполнении заданий №1 и №2, различаются фактически на два порядка.

В задании №3 адекватный результат получается только для третьего варианта коэффициента.