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 №1

по дисциплине: «Тестирование программного обеспечения»

по теме: «Исследование способов анализа областей эквивалентности и построения тестовых последовательностей»

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

(дата)

Руководитель практикума

ст. пр. Строганов В.А.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2018

1 ЦЕЛЬ РАБОТЫ

Исследовать способы анализа областей эквивалентности входных данных для тестирования программного обеспечения. Приобрести практические навыки составления построения тестовых последовательностей.

2 ПОСТАНОВКА ЗАДАЧИ

По варианту задаются требования к программам. Для каждой из них необходимо:

1. Написать программу, выполняющую заданные действия.
2. Определить области эквивалентности входных данных.
3. Составить примеры тестовых последовательностей.

Вариант 6

Задача 1. Дана целочисленная квадратная матрица. Определить позиции максимального и минимального элемента.

Задача 2. Дана строка. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Найти длину самого короткого слова.

Задача 3. Программа, переписывающую содержимое текстового файла t2в текстовый файл t1 (с сохранением деления на строки).

3 ХОД РАБОТЫ

Будем считать, что ввод данных для программ осуществляется идеальным пользователем через файл, имя которого передаётся первым параметром при запуске. При построении графа потоков проверку наличия первого параметра будем игнорировать.

1. Дана целочисленная квадратная матрица. Определить позиции максимального и минимального элемента.
   1. Исходный код программы

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** matrix {

**private** **static** InputStream *fin* = System.***in***;

**private** **static** OutputStream *fout* = System.***out***;

**private** **static** Scanner *scaner* = **new** Scanner(*fin*);

**private** **static** PrintStream *out* = **new** PrintStream(*fout*);

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

String inputFile = **null**;

**int** n = 50;

**int** matrix[][] = **new** **int**[n][n];

**int** maxI = 0;

**int** minI = 0;

**int** maxJ = 0;

**int** minJ = 0;

**try** {

System.***out***.print("Аргументы командной строки: ");

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

**if** (args[i].equalsIgnoreCase("-i")) {

inputFile = args[i + 1];

}

System.***out***.print(args[i] + " ");

}

System.***out***.println();

} **catch** (Exception e) {

e.printStackTrace();

}

**if** (inputFile != **null**) {

**try** {

*fin* = **new** FileInputStream(inputFile);

} **catch** (FileNotFoundException e) {

*fin* = System.***in***;

System.***out***.printf(inputFile + " - этот файл не найден \n");

System.*exit*(1);

e.printStackTrace();

} **finally** {

*scaner* = **new** Scanner(*fin*);

}

}

**if** ((inputFile != **null**) && (!inputFile.isEmpty())) {

**try** {

Scanner fin = **new** Scanner(**new** File(inputFile));

**while** (fin.hasNextInt()) {

n = fin.nextInt();

**if** (n == 0) {

System.***out***.printf("Пустая матрица");

System.*exit*(1);

}

**for** (**int** i = 0; i < n; i++)

**for** (**int** j = 0; j < n; j++)

matrix[i][j] = fin.nextInt();

}

fin.close();

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

}

System.***out***.println("Матрица");

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0; j < n; j++)

System.***out***.print(matrix[i][j] + " ");

System.***out***.print("\n");

}

**int** max = matrix[0][0];

**int** min = matrix[0][0];

**for** (**int** i = 0; i < n; i++)

**for** (**int** j = 0; j < n; j++) {

**if** (matrix[i][j] > max) {

max = matrix[i][j];

maxI = i;

maxJ = j;

}

**if** (matrix[i][j] < min) {

min = matrix[i][j];

minI = i;

minJ = j;

}

}

System.***out***.println("Матрица размерностью = " + n + "\*" + n);

System.***out***.print("Минимальный элемент: № строки = " + (minI + 1));

System.***out***.println(" № столбца = " + (minJ + 1));

System.***out***.print("Максимальный элемент: № строки = " + (maxI + 1));

System.***out***.print(" № столбца = " + (maxJ + 1));

}

}

|  |  |
| --- | --- |
|  |  |

* 1. Определение областей эквивалентности

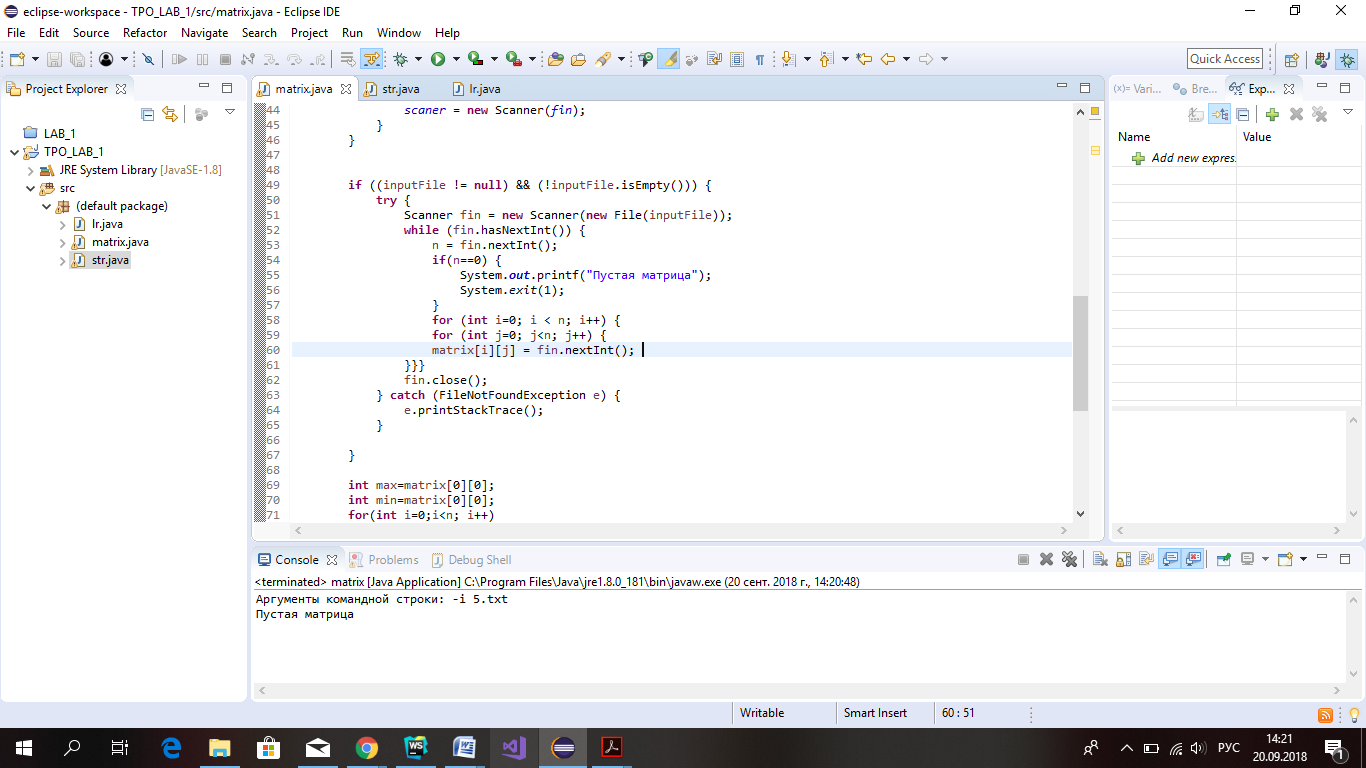
По размерам матрицы:

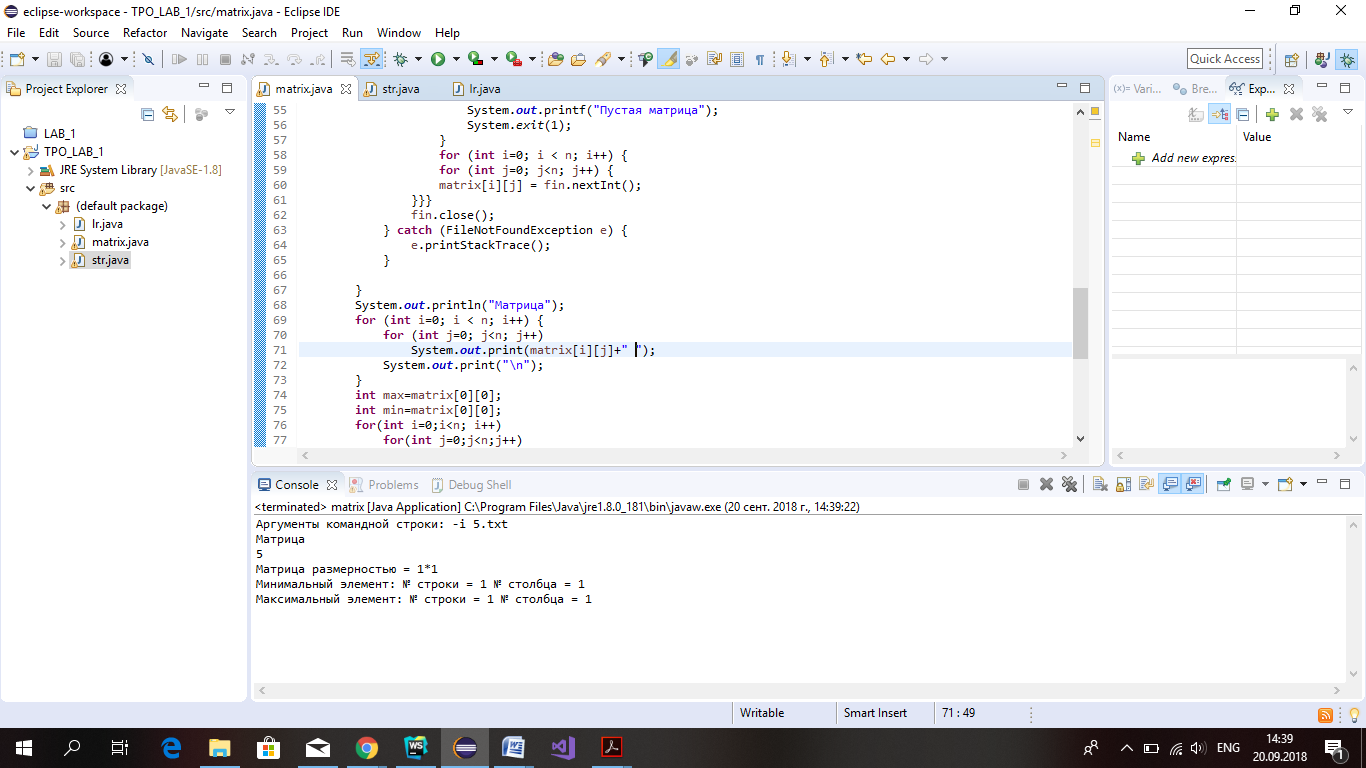
1. Матрица пуста
2. Матрица 1\*1
3. Матрица > 1\*1

По значениям матрицы:

1. Все значения равны, следовательно максимальный элемент равен минимальному элементу
2. Все значения разные, следовательно максимальный элемент равен минимальному элементу
   1. Составление тестовых последовательностей

При составлении тестовых последовательностей по областям эквивалентности возможны следующие варианты: AA, BB, CA, CB.





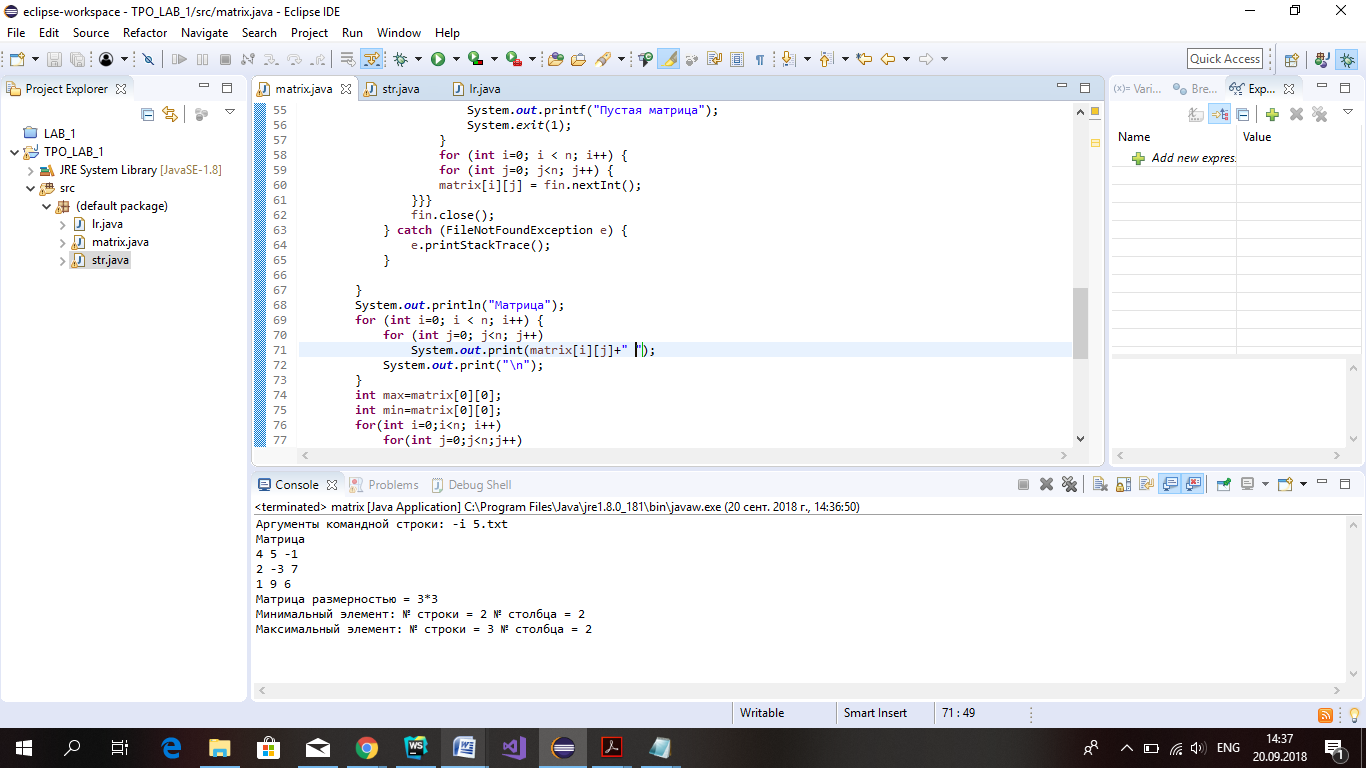
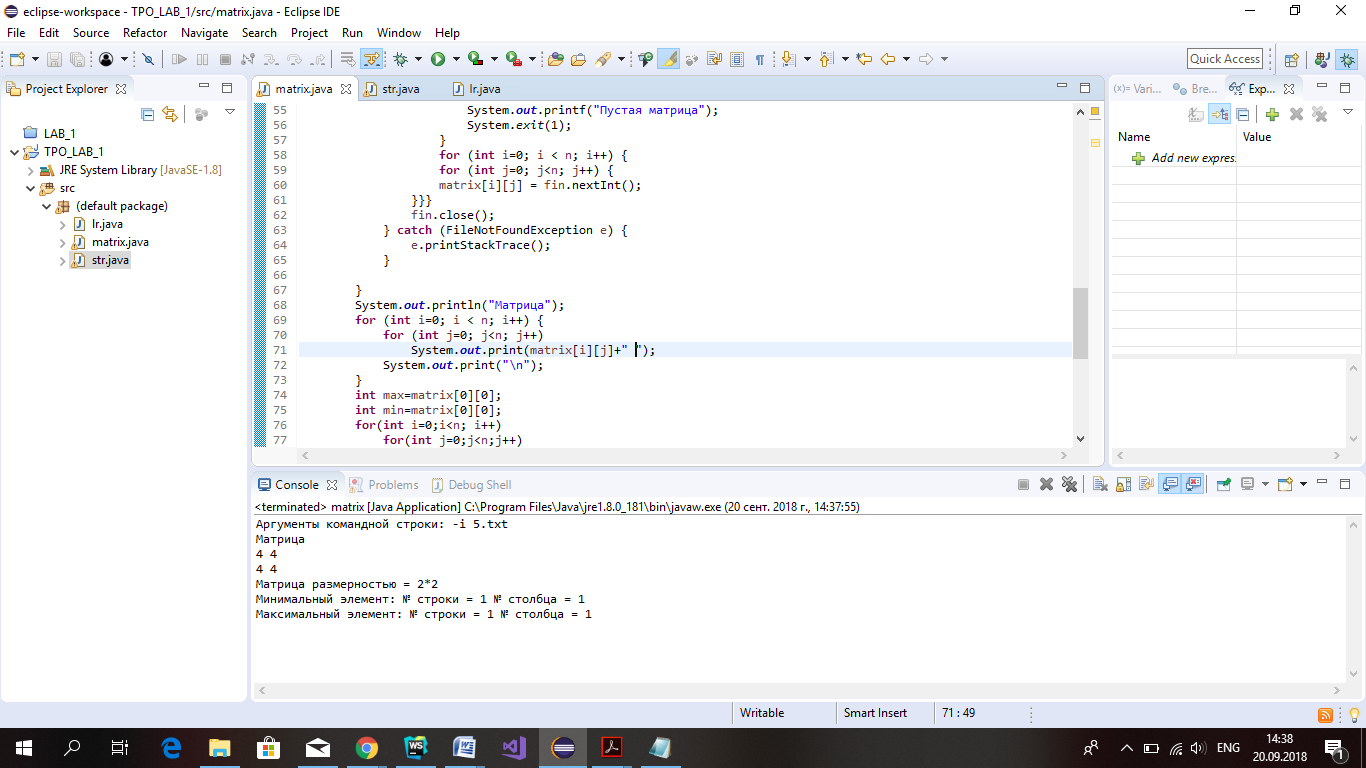


Рисунок 3.1 – Тестовые примеры для задачи №1

1. Дана строка. Группы символов, разделенные пробелами (одним или несколькими) и не содержащие пробелов внутри себя, будем называть словами. Найти длину самого короткого слова.
   1. Исходный код программы

**import** java.io.\*;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** str {

**private** **static** InputStream *fin* = System.***in***;

**private** **static** Scanner *scaner* = **new** Scanner(*fin*);

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

String inputFile = **null**;

String strLine = "";

**try** {

System.***out***.print("Аргументы командной строки: ");

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

**if** (args[i].equalsIgnoreCase("-i")) {

inputFile = args[i + 1];

}

System.***out***.print(args[i] + " ");

}

System.***out***.println();

} **catch** (Exception e) {

e.printStackTrace();

}

**if** (inputFile != **null**) {

**try** {

*fin* = **new** FileInputStream(inputFile);

} **catch** (FileNotFoundException e) {

*fin* = System.***in***;

System.***out***.printf(inputFile + " - этот файл не найден \n");

System.*exit*(1);

e.printStackTrace();

} **finally** {

*scaner* = **new** Scanner(*fin*);

}

}

**if** ((inputFile != **null**) && (!inputFile.isEmpty())) {

**try** {

Scanner fin = **new** Scanner(**new** File(inputFile));

**while** (fin.hasNext()) {

strLine = fin.nextLine();

}

fin.close();

} **catch** (IOException e) {

System.***out***.println("Ошибка");

}

}

System.***out***.println("Строка \n" + strLine);

String mytext = "";

StringTokenizer st = **new** StringTokenizer(strLine);

**int** minLengthOfWord = 0;

**while** (st.hasMoreTokens()) {

String word = st.nextToken();

**int** wordLength = word.length();

**if** (minLengthOfWord == 0) {

mytext = word;

minLengthOfWord = wordLength;

} **else** **if** (wordLength < minLengthOfWord) {

minLengthOfWord = wordLength;

mytext = word;

}

}

**if** (minLengthOfWord > 0)

System.***out***.println("Самое короткое слово = " + mytext + "\nДлина = " + minLengthOfWord);

**else** {

System.***out***.print("Пустая строка");

}

}

}

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |

* 1. Определение областей эквивалентности

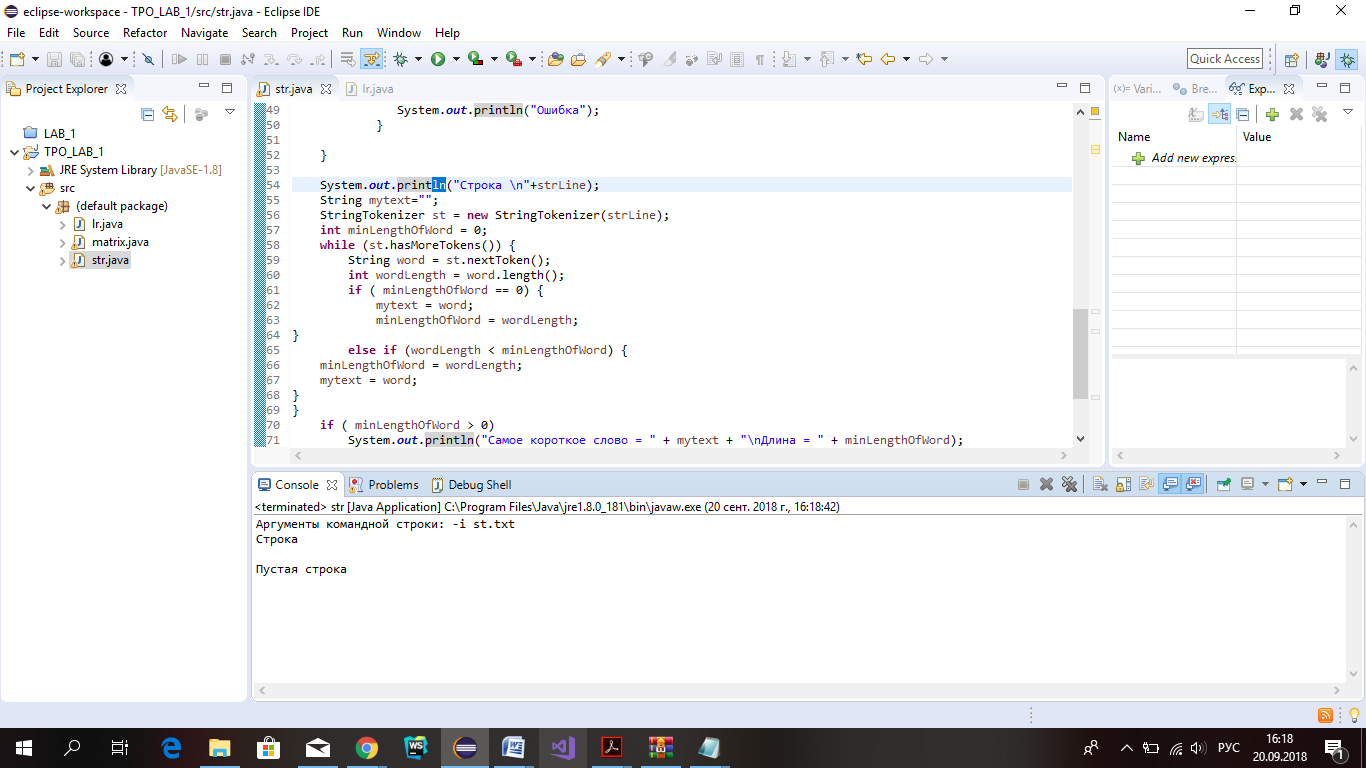
По размеру строки:

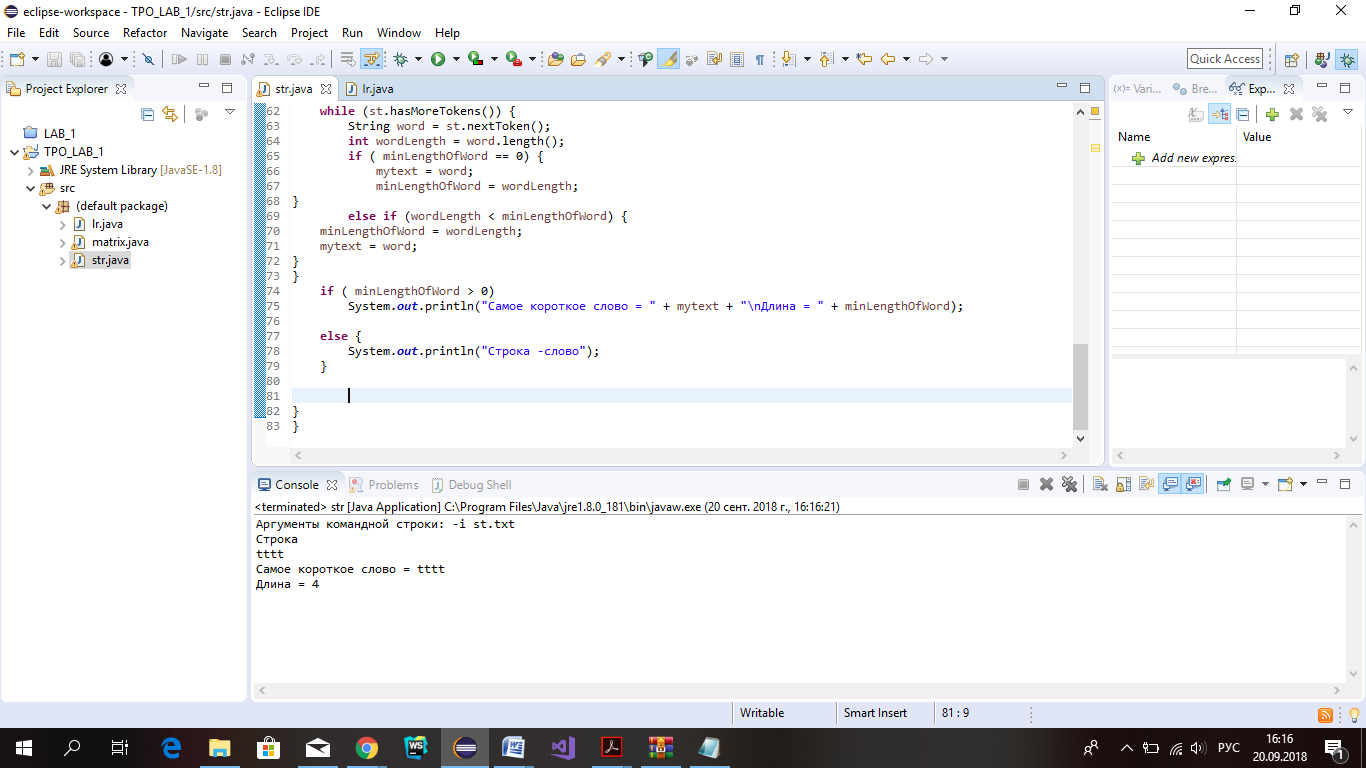
1. Пустая строка
2. Не пустая строка

По наличию необходимого слова:

1. Строка из одного слова, следовательно строка и есть самое короткое слово
2. Строка, состоящая из более чем одного слова
   1. Составление тестовых последовательностей

При составлении тестовых последовательностей по областям эквивалентности возможны следующие варианты: BA, BB.





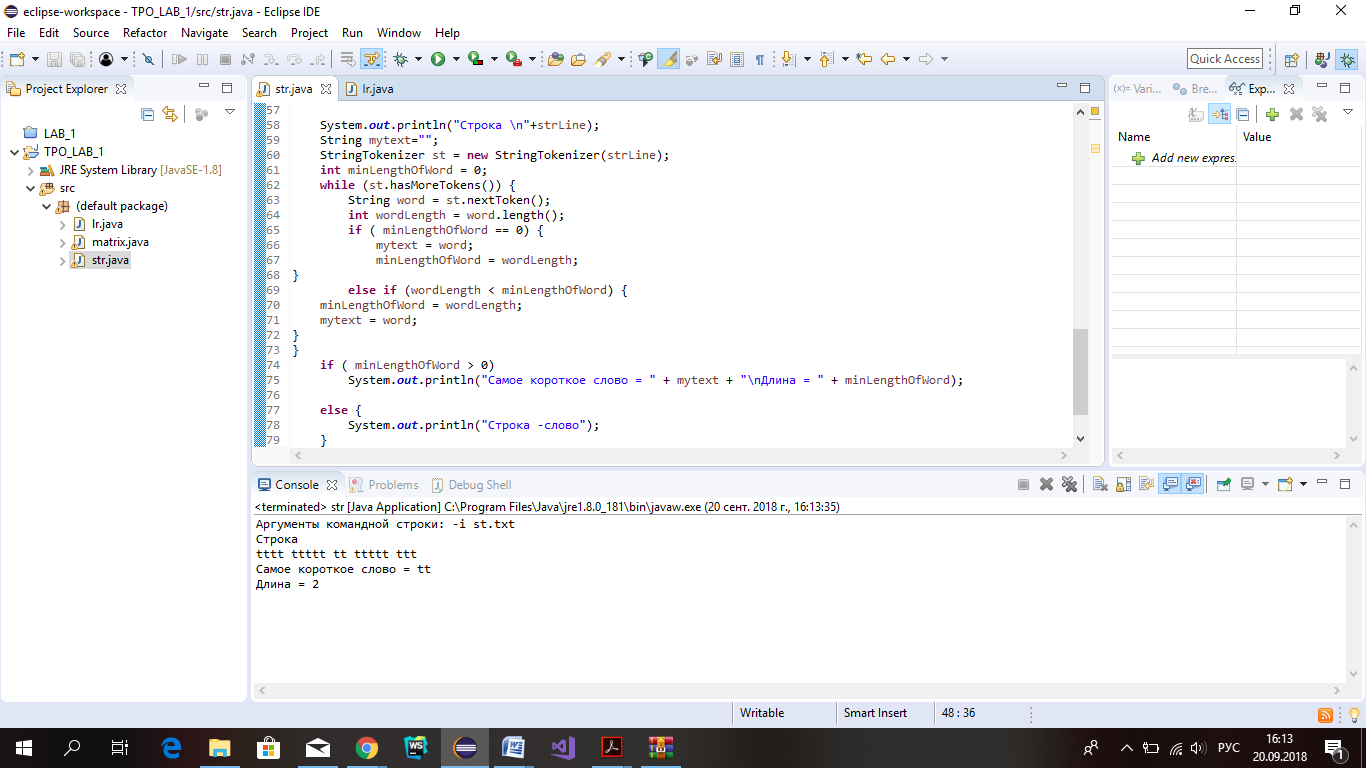


Рисунок 3.2 – Тестовые примеры для задачи №2

1. Программа, переписывающую содержимое текстового файла t2в текстовый файл t1 (с сохранением деления на строки).
   1. Исходный код программы

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** FileDemo {

**private** **static** String *inputFile* = **null**;

**private** **static** String *outputFile* = **null**;

**private** **static** BufferedReader *in* ;

**private** **static** PrintWriter *out*;

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

// FileReader fr = null;

// FileWriter fw = null;

**try** {

System.***out***.print("Аргументы командной строки: ");

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

**if** (args[i].equalsIgnoreCase("-i")) {

*inputFile* = args[i + 1];

} **else** **if** (args[i].equalsIgnoreCase("-o")) {

*outputFile* = args[i + 1];

}

System.***out***.print(args[i] + " ");

}

System.***out***.println();

} **catch** (Exception e) {

e.printStackTrace();

}

**try** {

**try** {

*in* = **new** BufferedReader(**new** FileReader(*inputFile*));

*out* = **new** PrintWriter(**new** FileWriter(*outputFile*));

String l;

**if** ((l = *in*.readLine()) == **null**){

System.***out***.println(*inputFile* + " - файл пуст ");

System.*exit*(1);

}

**while** ((l = *in*.readLine()) != **null**) {

*out*.println(l);

}

} **catch** (IOException e) {

e.printStackTrace();

}

} **finally** {

*closed*(*in*);

*closed*(*out*);

}

System.***out***.println("Файл скопирован ");

}

**public** **static** **void** closed(Closeable stream) {

**try** {

**if** (stream != **null**) {

stream.close();

}

} **catch** (IOException e) {

}

}

}

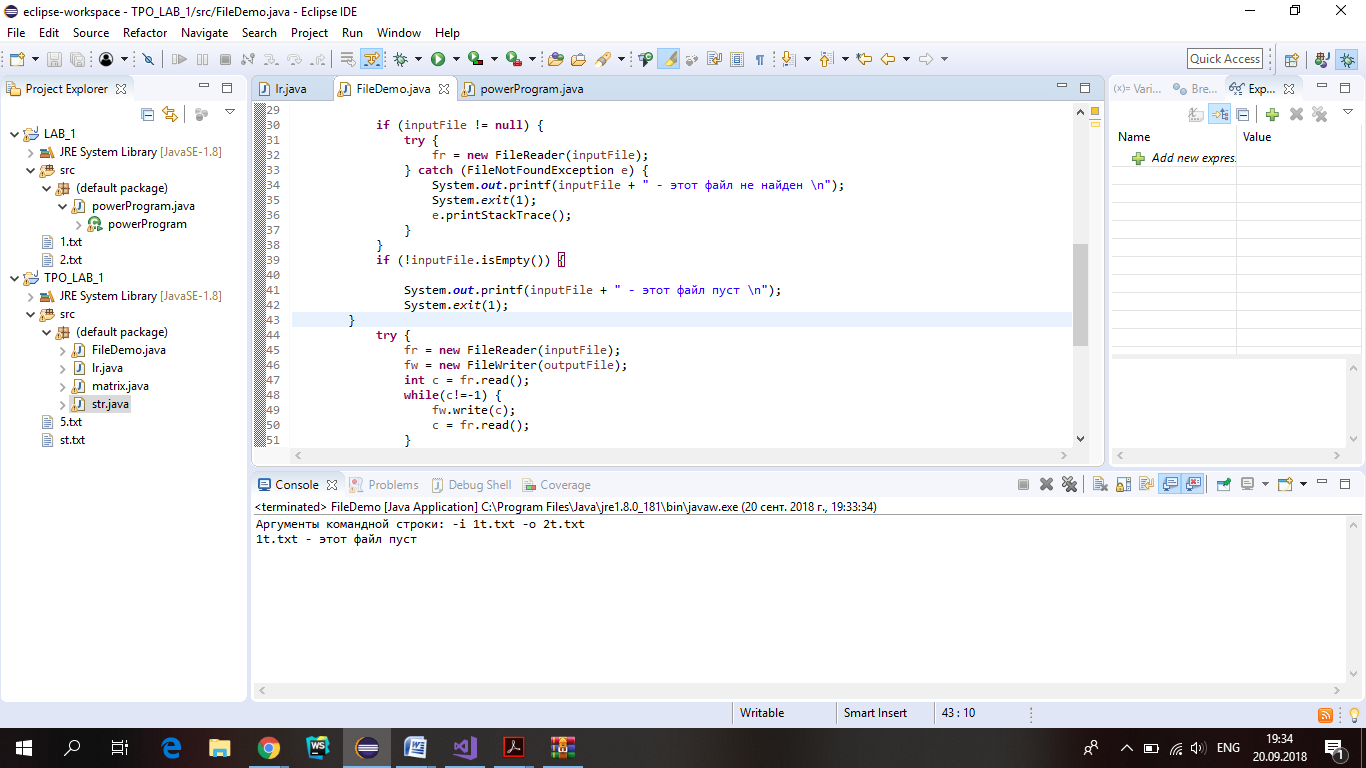
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  |  | |  | |  |

* 1. Определение областей эквивалентности

По количеству предложений:

1. Файл t1
2. Файл t1 не пуст
   1. Составление тестовых последовательностей

При составлении тестовых последовательностей по областям эквивалентности возможны следующие варианты: A, B.



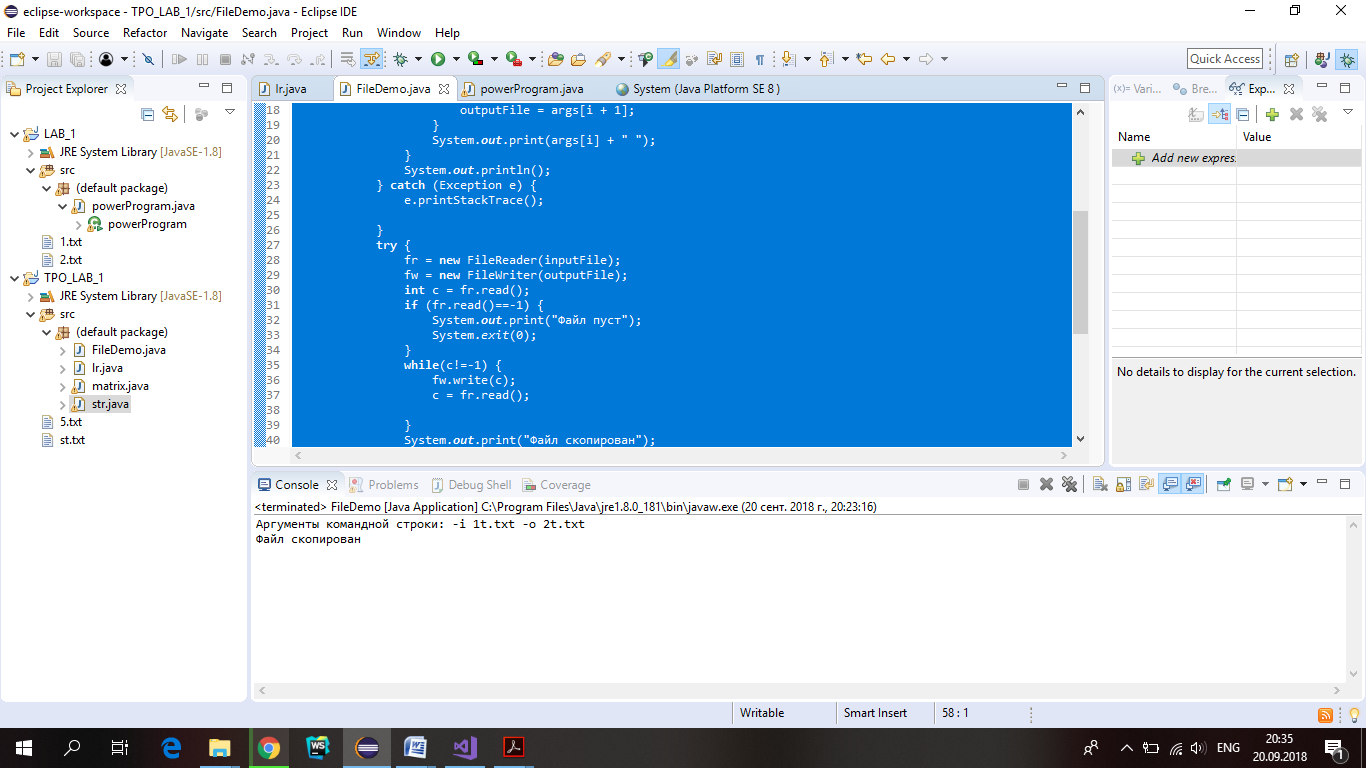


Рисунок 3.3 – Тестовые примеры для задачи №3

ВЫВОДЫ

В ходе первой лабораторной работы по дисциплине «Тестирование программного обеспечения» были исследованы способы анализа областей эквивалентности входных данных для тестирования программного обеспечения.

Были приобретены практические навыки определения областей эквивалентности составления тестовых последовательностей, построения графа потоков управления и определения независимых ветвей программы.