Содержание

[Введение 3](#_Toc25572636)

[1. Анализ объекта 4](#_Toc25572637)

[1.1 Описание предметной области. 4](#_Toc25572638)

[1.2 Построение концептуальной модели предметной области. 7](#_Toc25572639)

[2 Постановка задачи 8](#_Toc25572640)

[2.1 Определение требований к программной системе. 8](#_Toc25572641)

[2.2 Описание аналогов системы. 9](#_Toc25572642)

[2.3 Обзор и обоснование выбора средств реализации 10](#_Toc25572643)

[3 Проектирование 11](#_Toc25572644)

[3.1 Разработка архитектуры программного продукта 11](#_Toc25572645)

[3.2 Проектирование структур хранения данных. 11](#_Toc25572646)

[4 Реализация 12](#_Toc25572647)

[4.1 Разработка архитектуры программного продукта 12](#_Toc25572648)

[4.2 Разработка интерфейса программного продукта 14](#_Toc25572649)

[Разработка алгоритмов реализации вариантов использования. 18](#_Toc25572650)

[5. Тестирование 20](#_Toc25572651)

[Литература 23](#_Toc25572652)

[Приложение А 24](#_Toc25572653)

# Введение

Линейное программирование - это раздел математики ориентируемый на нахождении экстремума в задачах, которые описываются линейными уравнениями.

Графический способ решения задач линейного программирования целесообразно использовать для решения задач с двумя переменными, когда ограничения выражены неравенствами.

Мой проект представляет собой программу для решение оптимизационных задач графическим методом. Данная курсовая работа выполнена на языке программирования высокого уровня Java с использованием IDE Eclipse.

# Анализ объекта

## Описание предметной области.

Рассмотрим ситуацию, когда исходная линия уровня и многоугольник допустимых решений не пересекаются, на многоугольник допустимых решений указывает вектор . При параллельном смещении исходной линии уровня в направлении вектора  первая общая точка с многоугольником решений определяет точку минимума целевой функции , а последняя общая точка – точку максимума целевой функции  (рис. 3).

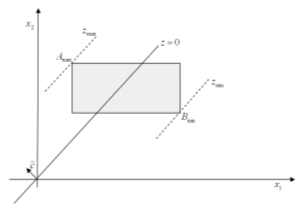


Рисунок 2 – Многоугольник допустимых решений и исходная линия уровня пересекаются, вектор  лежит во втором квадранте

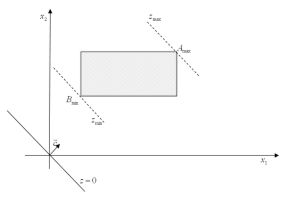


Рисунок 3 – Многоугольник допустимых решений и исходная линия уровня не пересекаются, вектор  лежит в первом квадранте

В процессе решения задачи линейного программирования возможны случаи, когда система ограничений задачи линейного программирования несовместна (рис. 4) и задача решений не имеет; целевая функция на множестве допустимых решений неограниченно возрастает или неограниченно убывает (рис. 5); целевая функция принимает экстремальное значение в любой точке некоторого отрезка (рис. 6).

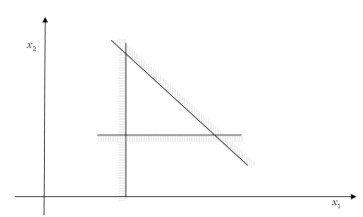


Рисунок 4 – Система ограничений задачи линейного программирования несовместна

Если система ограничений задачи линейного программирования несовместна, то множество допустимых решений не содержит ни одной точки и задача решений не имеет.

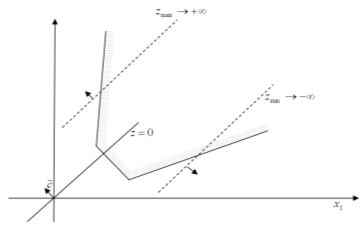


Рисунок 5 – Целевая функция на множестве допустимых решений задачи линейного программирования неограниченно возрастает или неограниченно убывает

## Построение концептуальной модели предметной области.

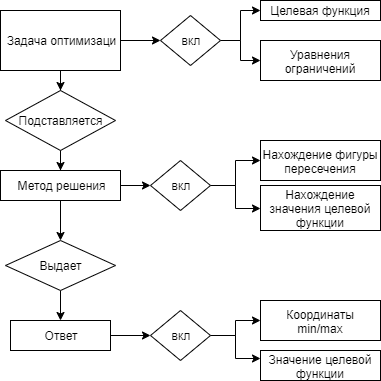


Рисунок 1.6 – модель предметной области

Концептуальная модель состоит задачи оптимизации, данные которой пользователю необходимо ввести (целевая функция, ограничения). После ввода пользователь нажимает кнопку «Решить», и данные задачи подставляются в метод решения. Сперва на графике отрисовываются прямые ограничений, затем находится минимальная (или максимальная) точка и подставляется в целевую функцию. Как только задача решена, метод выдает пользователю ответ.

# Постановка задачи

## Определение требований к программной системе.

1. Приложение, позволяющее решить уравнение методом Симпсона
2. Поддержка ввода данных как с экрана, так и с файла
3. Поддержка вывода данных как на экран, так и в файл
4. Дружелюбный интерфейс
5. Поддержка Java версии 9 и более

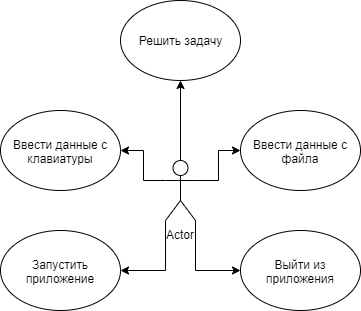


Рисунок 2.1 - диаграмма вариантов использования

## Описание аналогов системы.

В сети Интернет можно найти большое количество работ, которые реализуют решение данных задач. Практически все они представляют собой сайты, так что для решения уравнений необходимо подключение к интернету. Примером таким сайтов являются «math.semestr.ru»

Мой курсовой проект представляет собой приложение для ПК. Программа не требует подключения к интернету и имеет функции загрузки данных в файл.

### 2.3 Обзор и обоснование выбора средств реализации

Eclipse —интегрированная среда разработки ПО. Написана на языке Java в 2003 году. Также является кроссплатформенной. За счёт присоединяемых к этой среде дополнений — имеется возможность создавать программные продукты более чем на пяти языках программного кода.

Достоинства: Постоянное обновление версий среды разработки, поддержка многих языков (в том числе и русского), является бесплатной, поддержка многих языков программирования, среда имеет промышленный уровень, является гибкой — то есть легко настраивается как под любую платформу, так и под любого пользователя.

Для создания интерфейса было применено:

JFrame – класс, реализующий отображение окон приложения.

JLabel – элементы отображения надписей

JButton – элементы отображения кнопок

JTextField – элементы отображение полей ввода

JTable – элемент отображения таблиц

# Проектирование

## Разработка архитектуры программного продукта

Поскольку приложение не сетевое, для лучшей скорости сделано однопоточным.

Главное окно будет содержать следующие элементы:

* Поле «Количество ограничений»
* Таблицу, созданную по заданным размерам для хранения уравнений ограничений
* Кнопку «Создать таблицу»
* Кнопку «Рассчитать» - решение задачи
* Кнопку «Открыть файл» - для ввода данных из файла
* Кнопку «Очистить» - для очистки формы

## Проектирование структур хранения данных.

Хранение данных извне не используется. Для хранения данных внутри приложения используются следующие переменные:

**private** JTable table; - для хранения уравнений ограничений

**private** JTextField textField; - для хранения размерности таблицы

**private** JTextField textField\_1, textField\_2; - для хранения целевой функции

# Реализация

## Разработка архитектуры программного продукта

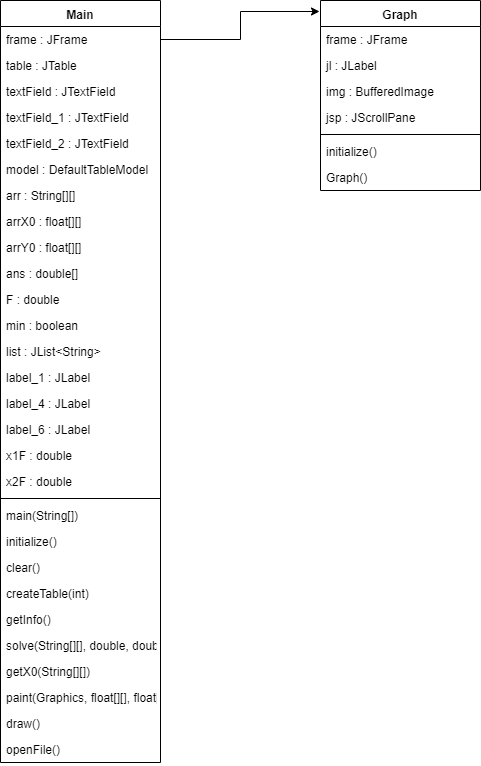


Рисунок 4.1 – Диаграмма классов

Главным классом является класс Main, у которого есть переменные для ввода данных таблицы, уравнений ограничений и целевой функции. Также, для простоты решений, было принято сохранить методы решения в данном классе.

При нажатии кнопки «Рассчитать», данные таблицы передаются в метод Solve(). Если пользователь ввел не все данные, некорректные или же не ввел их, ему выдастся соответствующее уведомление. В методе решения рисуются функции ограничений, находятся их точки пересечения и определяется, имеет ли уравнение решение. Как только с помощью метода будет получено решение, данные решения будут записаны на главном окне, а так-же откроется новое окно Graph с графиком.

## Разработка интерфейса программного продукта

Главное окно

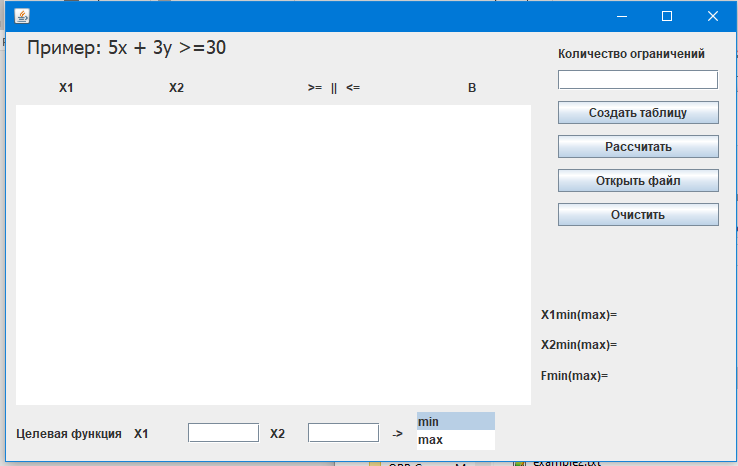


Рисунок 4.2.1 – Вид главного окна

Введем данные и создадим таблицу

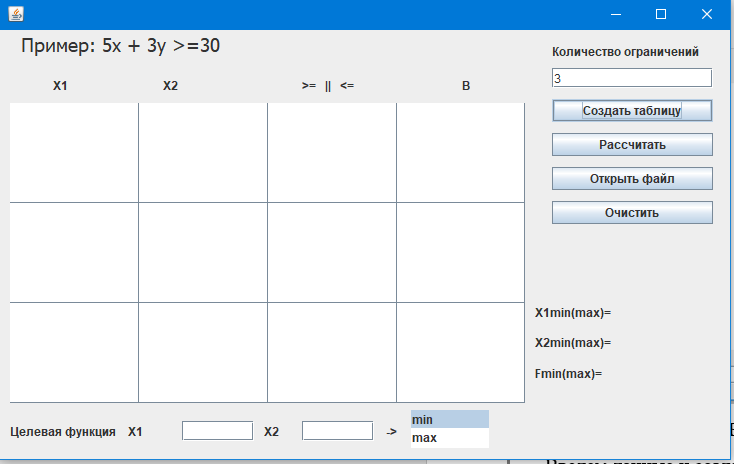


Рисунок 4.2.2 – Создание таблицы

Попробуем загрузить данные из файла

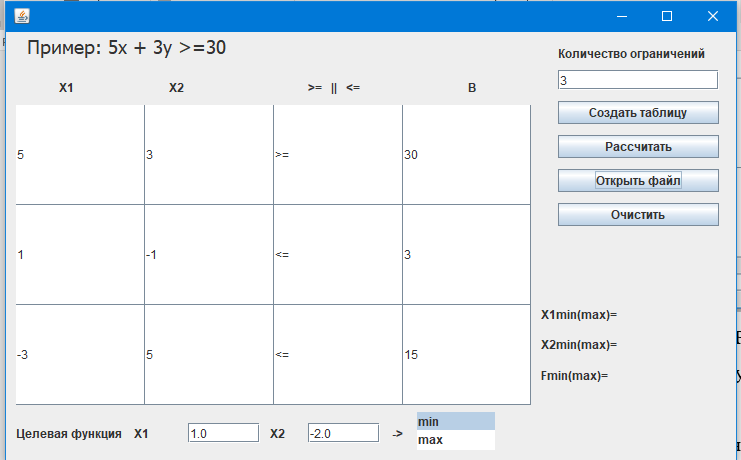


Рисунок 4.2.3 – Заполнение данных из файла

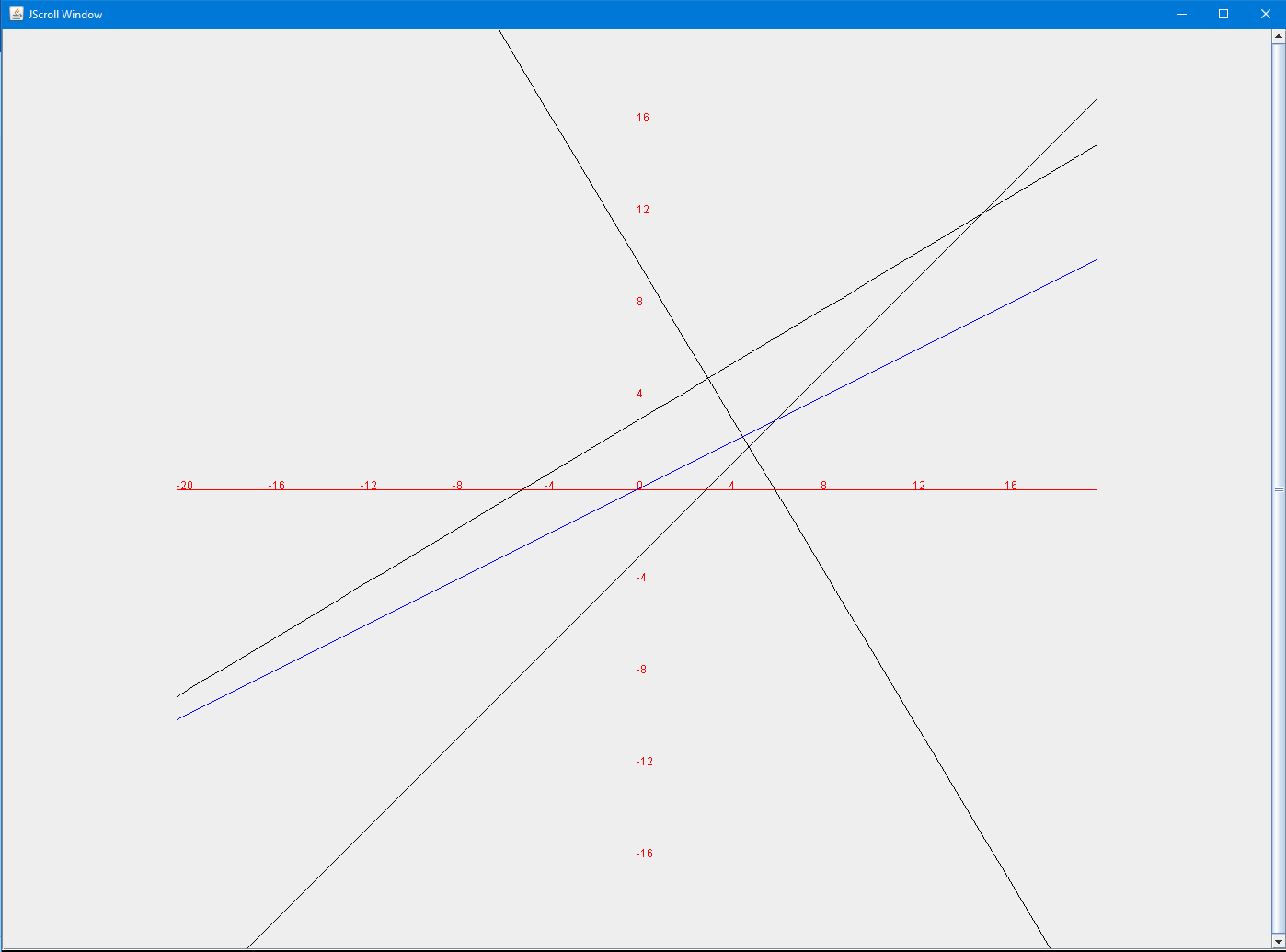
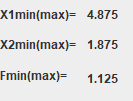
 

Рисунок 4.2.4 – Окно ответа

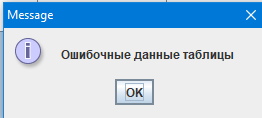


Рисунок 4.2.5 – Попытка решить без введенных данных

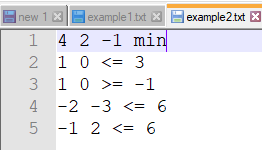


Рисунок 4.2.6 – Вид загружаемого файла

## Разработка алгоритмов реализации вариантов использования.

Общий алгоритм работы приложения:

1. Открывается главное меню, переход к 2 или 8
2. Вводит количество ограничений, если еще не введена, переход к 3
3. Кнопка создать таблицу создает таблицу, переход к 4
4. Ввод ограничений, если еще не введены, переход к 5
5. Ввод целевой функции, переход к 6
6. Если нажата кнопка решение, решается задача и переход к 7
7. Открывается окно ответа, переход к 2
8. Завершение работы приложения
9. Создание таблицы

**void** createTable(**int** row) {

table.removeAll();

table.setModel(model);

model.setRowCount(row);

model.setColumnCount(4);

table.setAutoResizeMode(JTable.***AUTO\_RESIZE\_ALL\_COLUMNS***);

table.setRowHeight(300 / row);

1. Метод получения значений целевой функции и вывода на экран

**void** solve(String[][] arr, **double** xMin, **double** yMin) {

**double** x1, x2, c;

String snak;

LinearProgram lp = **new** LinearProgram(**new** **double**[] { xMin, yMin });

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

x1 = Double.*parseDouble*(arr[i][0]);

x2 = Double.*parseDouble*(arr[i][1]);

c = Double.*parseDouble*(arr[i][3]);

snak = arr[i][2];

**if** (snak.contains("<="))

lp.addConstraint(**new** LinearSmallerThanEqualsConstraint(**new** **double**[] { x1, x2 }, c, "c" + i));

**else**

lp.addConstraint(**new** LinearBiggerThanEqualsConstraint(**new** **double**[] { x1, x2 }, c, "c" + i));

}

lp.setMinProblem(!min);

LinearProgramSolver solver = SolverFactory.*newDefault*();

ans = solver.solve(lp);

**for** (**double** d : ans)

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

F = xMin \* ans[0] + yMin \* ans[1];

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

label\_1.setText(ans[0] + "");

label\_4.setText(ans[1] + "");

label\_6.setText(F + "");

}

1. Метод рисования осевых линий, ограничений и целевой функции

**void** paint(Graphics gr, **float**[][] arrX02, **float**[][] arrY02) {

gr.setColor(Color.***red***);

gr.drawLine(0, 500, 1000, 500);

gr.drawLine(500, 0, 500, 1000);

**for**(**int** i=0; i<1000;i=i+100) {

String str=(i-500)/25+"";

gr.drawChars(str.toCharArray(), 0, str.toCharArray().length, i, 500);

}

**for**(**int** i=0; i<1000;i=i+100) {

String str=(-1\*(i-500)/25)+"";

gr.drawChars(str.toCharArray(), 0, str.toCharArray().length, 500, i);

}

gr.setColor(Color.***BLACK***);

**int** x1, y1, x2, y2;

**for** (**int** i = 0; i < arrX02.length; i++) {

**for** (**int** j = 0; j < 199; j++) {

x1 = j - 100;

y1 = (**int**) arrX02[i][j];

x2 = j - 99;

y2 = (**int**) arrX02[i][j + 1];

gr.drawLine(500 + x1\*25, 500 - y1, 500 + x2\*25, 500 - y2);

}

}

gr.setColor(Color.***BLUE***);

**int** x1FF=(**int**) (-x1F\*-500/x2F);

**int** x2FF=(**int**) (-x1F\*500/x2F);

gr.drawLine(0, 500 - x1FF, 1000, 500 - x2FF);

}

# 5. Тестирование

Проведем тестирование программы, пройдя все варианты взаимодействия. Запустим само приложение. Запуск происходит без проблем.

Попробуем найти решение не вводя значений.

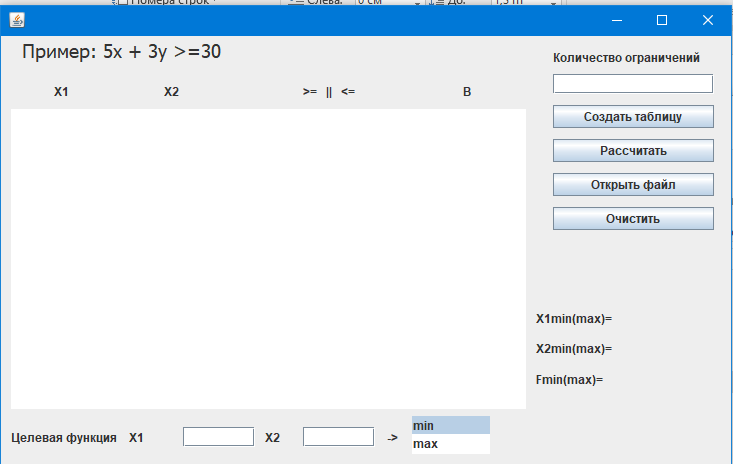


Рисунок 5.1 – Запуск главного окна

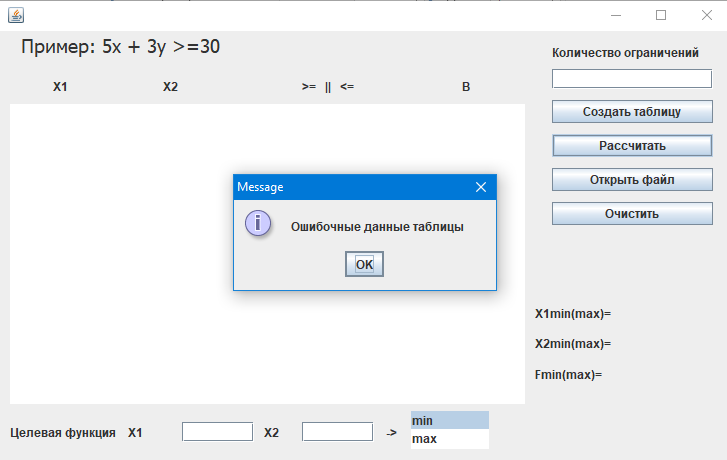


Рисунок 5.2 – Попытка решения задачи без введенных значений

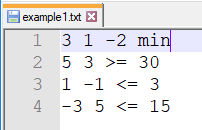
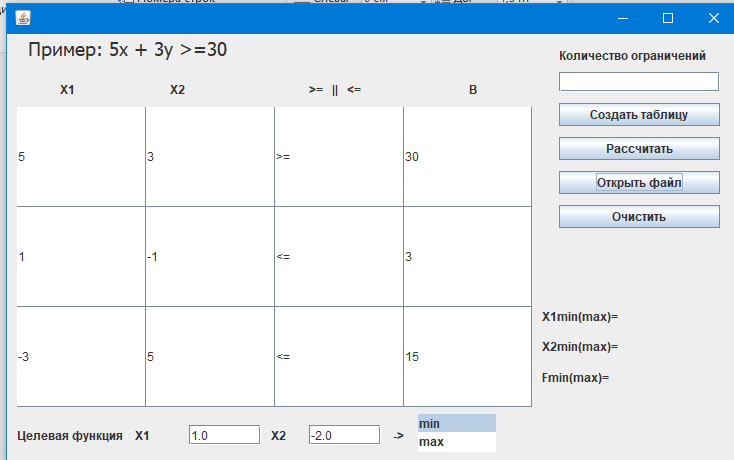
 

Рисунок 5.2 – Загрузка данных из файла

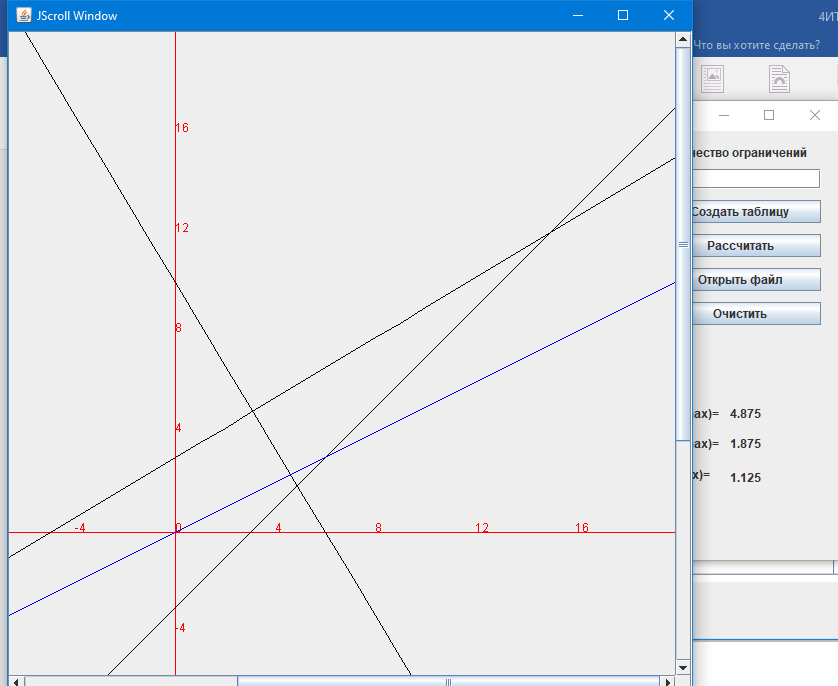


Рисунок 5.4 – Вывод ответа

Работа приложения происходит без ошибок

## Литература

1. Хорстманн, К.С. Java 2. Библиотека профессионала, том1. Основы. 8-е издание, : Пер. с англ. / К. С. Хорстманн, Г. Корнелл - М.: ООО «Вильямс», 2012. - 816 с.
2. Дирк, Л. Самоучитель Java 7: Пер. с нем. / Л. Дирк, П. Мюллер. – СПб: БХВ-Петербург, 2013. – 464 с.: илл.
3. Шилдт, Г. Java. Полное руководство. 8-е издание, : Пер. с англ. / Г. Шилдт. – М.:ООО «Вильямс», 2012. – 1104 с.
4. Блинов, И.Н. Java. Промышленное программирование./ И.Н. Блинов, В.С. Романчик –Минск: «Четыре четверти», 2013. – 896 с.
5. Онлайн калькулятор [Электронный ресурс]. – Режим доступа: math.semestr.ru. – Дата доступа: 20.11.2019.
6. Делюсь знаними [Электронный ресурс]. – Режим доступа: http://matecos.ru/. – Дата доступа: 20.11.2019.
7. Сайт преподователя экономики [Электронный ресурс]. – Режим доступа: http://galyautdinov.ru/. – Дата доступа: 20.11.2019.
8. Мир математики [Электронный ресурс]. – Режим доступа: https://matworld.ru/. – Дата доступа: 20.11.2019.

## Приложение А

Исходный код, откомпилированная программа и записка находятся на диске.