**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**Отчет**

**по учебной практике**

**«Визуализация алгоритма на языке Java»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6381 |  | Вероха В. Н. |
| Студент гр. 6381 |  | Вергейчик Г. Л. |
| Студент гр. 6381 |  | Ширяев Я. А. |
| Руководитель |  | Фирсов М. А. |

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на учебную практику**

|  |  |  |
| --- | --- | --- |
| Студентка Вероха В. Н. группы 6381 | | |
| Студент Вергейчик Г. Л. группы 6381 | | |
| Студент Ширяев Я. А. группы 6381  Тема практики: Визуализация алгоритма на языке Java | | |
| Задание на практику:  Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.  Алгоритм: Backtracking (поиск с возвратом) на примере квадрирования квадрата. | | |
| Сроки прохождения практики: 27.06.2018 – 10.07.2018 | | |
| Дата сдачи отчета: 00.07.2018 | | |
| Дата защиты отчета: 00.07.2018 | | |
|  | | |
| Студентка |  | Вероха В. Н. |
| Студент |  | Вергейчик Г. Л. |
| Студент |  | Ширяев Я. А. |
| Руководитель |  | Фирсов М. А. |

**Аннотация**

Целью данной работы является разработка визуализатора алгоритма backtracking (поиск с возвратом) на примере квадрирования квадрата.

При разработке программы использовался язык Java и среда IntelliJ IDEA. Для создания графического интерфейса – библиотека Swing.

**Summary**

The goal of this work is developing a visualizer of the algorithm backtracking (search with a return) for the example of square squaring.

For the development of the program used the Java language and the IntelliJ IDEA environment. For graphical interface - the Swing library.

**Содержание**

[Введение 5](#_Toc518588183)

[1. Требования к программе 6](#_Toc518588184)

[3.1. Формулировка задания 6](#_Toc518588185)

[3.2. Формальная постановка задачи 6](#_Toc518588186)

[3.3. Объяснение алгоритма 6](#_Toc518588187)

[1.3.1. Теоретическая справка 6](#_Toc518588188)

[1.3.2. Описание алгоритма 7](#_Toc518588189)

[1.4. Исходные требования к программе 8](#_Toc518588190)

[1.4.1. Требования к вводу исходных данных 8](#_Toc518588191)

[1.4.2. Требования к визуализации 8](#_Toc518588192)

[3.5. Уточнение требований после сдачи первой версии 8](#_Toc518588193)

[3.6. Уточнение требований после сдачи второй версии 8](#_Toc518588194)

[2. План разработки и распределения ролей в бригаде 9](#_Toc518588195)

[3.1. План разработки 9](#_Toc518588196)

[3.2. Распределение ролей в бригаде 9](#_Toc518588197)

[3. Особенности реализации 9](#_Toc518588198)

[3.1. Спецификация 9](#_Toc518588199)

[3.1.1. Use case диаграмма 9](#_Toc518588200)

[3.1.2. Графический интерфейс 10](#_Toc518588201)

[3.2. Использованные структуры данных 17](#_Toc518588202)

[3.3. Основные методы 17](#_Toc518588203)

[3.4. Диаграмма классов 17](#_Toc518588204)

[4. Тестирование программы 17](#_Toc518588205)

[Заключение 17](#_Toc518588206)

[Приложение. Код програмы 17](#_Toc518588207)

# Введение

В данной работе разрабатывается визуализатор алгоритма backtracking (поиск с возвратом) на примере квадрирования квадрата.

Разработанная программа решает задачу о разбиении квадрата на конечное число меньших по размеру квадратов. Пользователь вводит размеры квадрата, а на выходе получает его разбиение на меньшие квадраты. При этом количество квадратов минимально.

Целью работы в проекте является получение навыков работы в команде, а также усовершенствование знаний о языке программирования Java и системы контроля версий GitHub.

# Требования к программе

## Формулировка задания

Разработать визуализатор алгоритма backtracking (поиск с возвратом) на примере квадрирования квадрата с помощью языка программирования Java.

## Формальная постановка задачи

Дан квадрат, размеры которого задает пользователь. Большой квадрат (поле) требуется разбить на меньшие по размеру квадраты.

Должны быть соблюдены следующие требования:

* количество квадратов разбиения минимально;
* внутри поля нет пустот;
* меньшие квадраты не перекрываются и не выходят за пределы поля.

Входные данные:

Размер поля в клетках - одно целое число **N** (2≤**N**≤40).

Выходные данные:

Разбиение поля на минимальное количество меньших квадратов. Визуализация работы алгоритма.

## Объяснение алгоритма

### 1.3.1. Теоретическая справка

Поиск с возвратом (англ. backtracking) - общий метод нахождения решений задачи, в которой требуется полный перебор всех возможных вариантов в некотором множестве. Как правило позволяет решать задачи, в которых ставятся вопросы типа: «Перечислите все возможные варианты …», «Сколько существует способов …», «Существует ли объект…» и т. п.

Термин backtracking был введен в 1950 году американским математиком Дерриком Генри Лемером.

Незначительные модификации метода поиска с возвратом, связанные с представлением данных или особенностями реализации, имеют и иные названия: метод ветвей и границ, поиск в глубину, метод проб и ошибок и т. д. Поиск с возвратом практически одновременно и независимо был изобретен многими исследователями ещё до его формального описания.

### 1.3.2. Описание алгоритма

Любое целое число может быть представлено в виде произведения конечного числа простых чисел. Поэтому задача квадрирования квадрата сводится к нахождению наименьшего простого делителя размера поля (большего квадрата), нахождения его разбиения, а затем масштабирования.

Для всех квадратов, стороны которых являются простыми числами в пределах диапазона от 2 до 61 (в условиях нашей задачи диапазон – [2, 40]), уже существует разбиение на минимальное число квадратов меньших размеров (см. рис. 1).

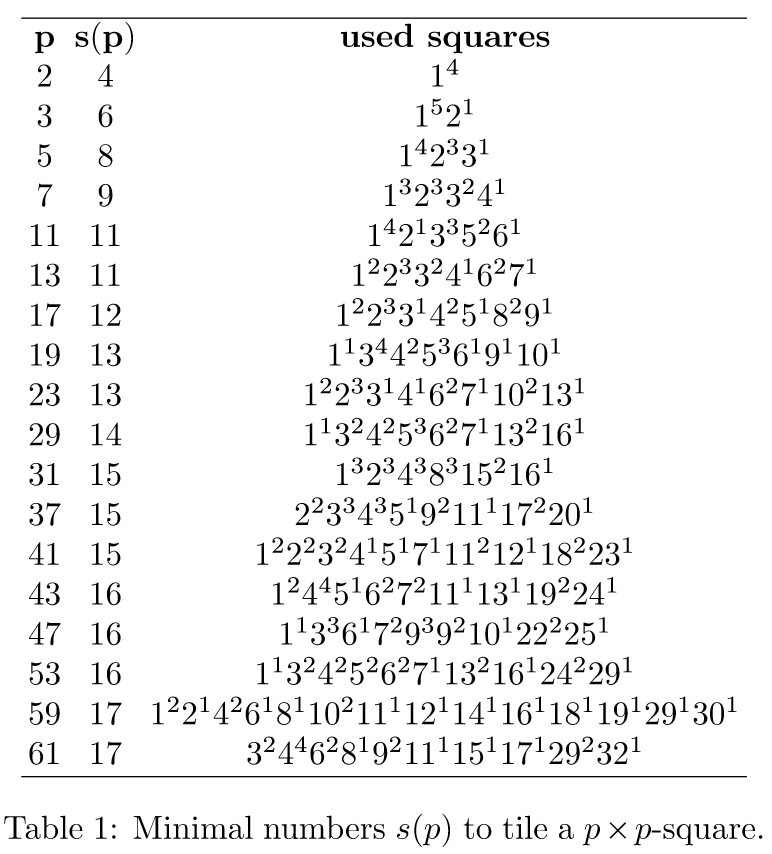


Рисунок 1. Разбиение квадратов, размер поля которого – простое число. Обозначения: p – размер поля (сторона квадрата, который нужно квадрировать) – простое число, s(p) – минимальное число квадратов, требуемое для разбиения большего квадрата.

В программе для размера поля, являющегося простым числом, задано количество меньших квадратов определенных размеров, с помощью которых может быть заполнен больший квадрат.

При квадрировании находится наименьший простой делитель введенного числа (размера поля). Далее применяется алгоритм поиска с возвратом на квадрате, сторона которого является простым числом, а затем масштабирует его, получая разбиение изначально заданного квадрата.

## Исходные требования к программе

### Требования к вводу исходных данных

Программа предоставляет пользователю графический интерфейс. Входные данные считываются при вводе размера поля (большего квадрата) пользователем. После запуска алгоритма, пользователь наблюдает результаты квадрирования большего квадрата по шагам, либо конечный результат.

### Требования к визуализации

Программа предоставляет интерфейс с пояснениями и схематичным изображением поля большего квадрата, а также квадратов вставки. Визуализация должна представлять собой окно с меню.

## Уточнение требований после сдачи первой версии

* Связать алгоритм с графическим интерфейсом.
* В разделе меню «About programm» отредактировать текст с информацией.
* Доработать раздел меню «Enter size».

## Уточнение требований после сдачи второй версии

* При переходе по шагам результата добавить кнопку «Result» и «Menu».
* Сдвинуть поле расстановки квадратов от края экрана.
* Сделать расставляемые квадраты разных цветов.
* Добавить status bar с номером шага. Например, 2/10.

# План разработки и распределения ролей в бригаде

## План разработки

28. 06. 2018 – разработка спецификации программы.

30. 06. 2018 – написание логики программы.

02. 07. 2018 – разработка алгоритма и визуализация меню.

04. 07. 2018 – связь алгоритма и графического интерфейса.

05. 07. 2018 – доработка графического интерфейса, визуализация хода работы алгоритма.

## План разработки

Распределение ролей представлено в таблице 1.

|  |  |
| --- | --- |
| Студент | Роль |
| Вероха В. Н. | Спецификация, отчет, тестирование |
| Вергейчик Г. Л. | Визуалицая |
| Ширяев Я. А. | Алгоритм |

Таблица 1. Распределение ролей.

# Особенности реализации

## Спецификация

### 3.1.1. Use case диаграмма

Ниже представлена диаграмма вариантов использования, показывающая взаимоотношения между пользователем и программой.

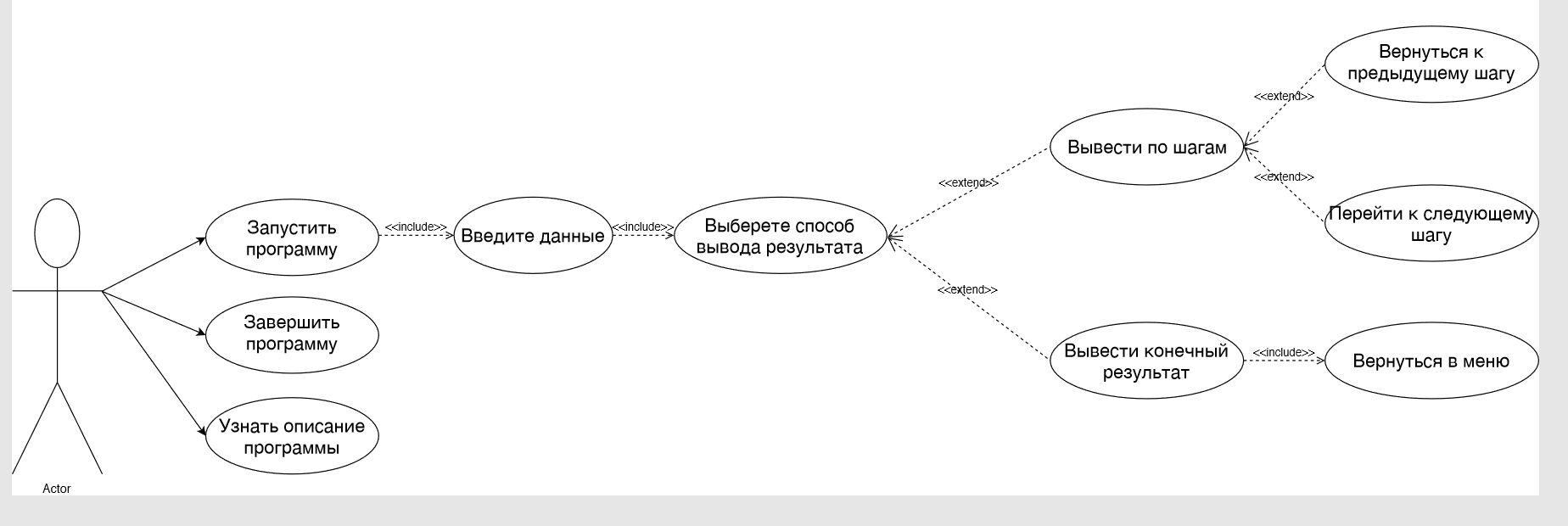


Рисунок 2. Use case диаграмма.

### 3.1.2. Графический интерфейс

Ниже представлены эскизы работы программы.

При запуске программы появляется меню (рис. 3) со следующими кнопками:

* «Enter the size» – запуск работы алгоритма (рис. 4);
* «About the program» – описание программы (рис. 12);
* «Exit» – завершение программы (рис. 13).



Рисунок 3. Меню программы.

При нажатии кнопки «Enter the size» программа запрашивает у пользователя размеры поля (рис. 4).

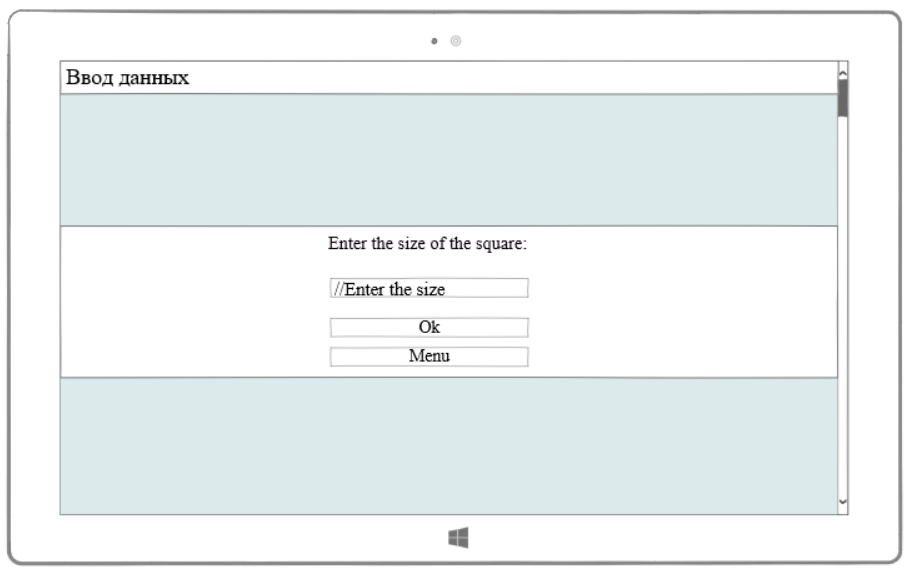


Рисунок 4. Ввод данных.

При неверном вводе размеров поля (большего квадрата) всплывает предупреждении (рис. 5). И у пользователя повторно запрашиваются размеры поля (рис. 6).

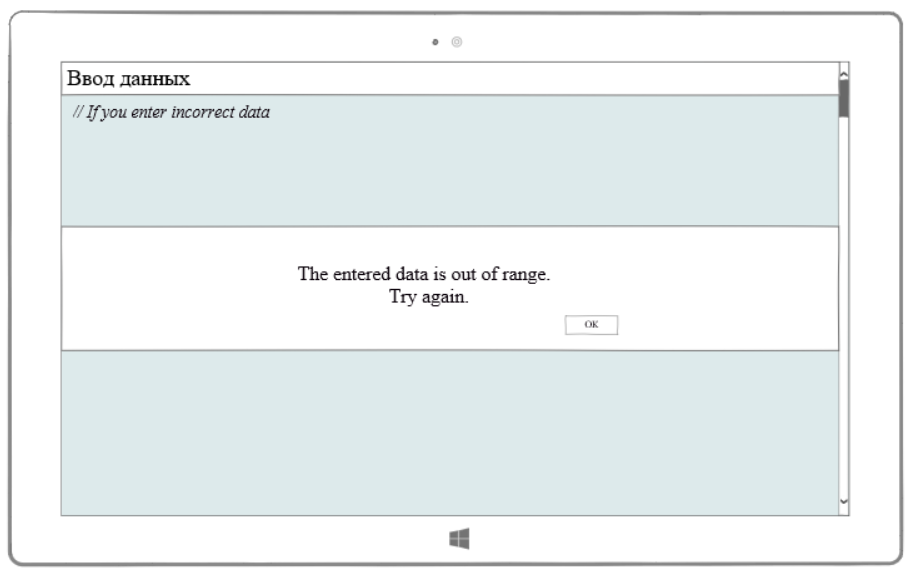


Рисунок 5. Предупреждении о неверно введенных данных.

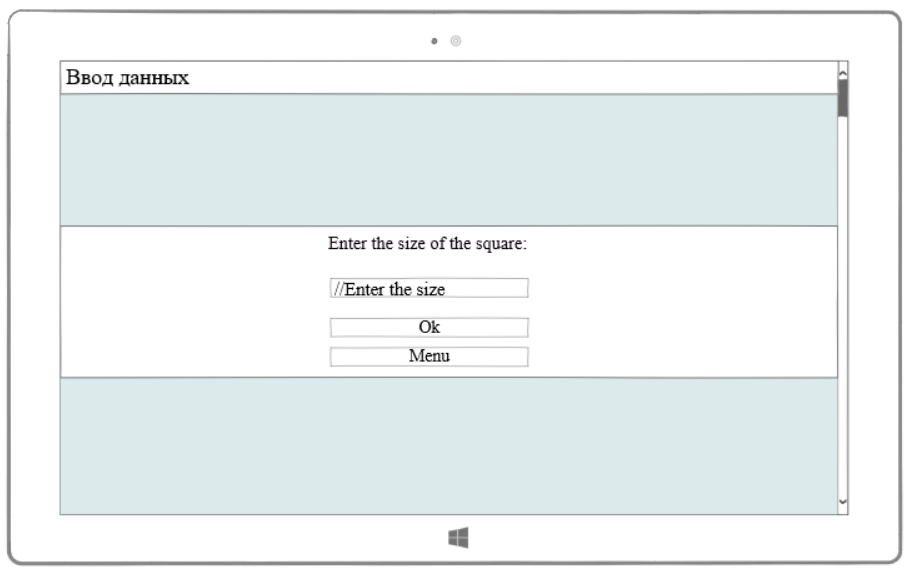


Рисунок 6. Повторный ввод данных.

Далее на примере поля, сторона которого равна 2, рассмотрим работу программы (рис. 7 – 11).

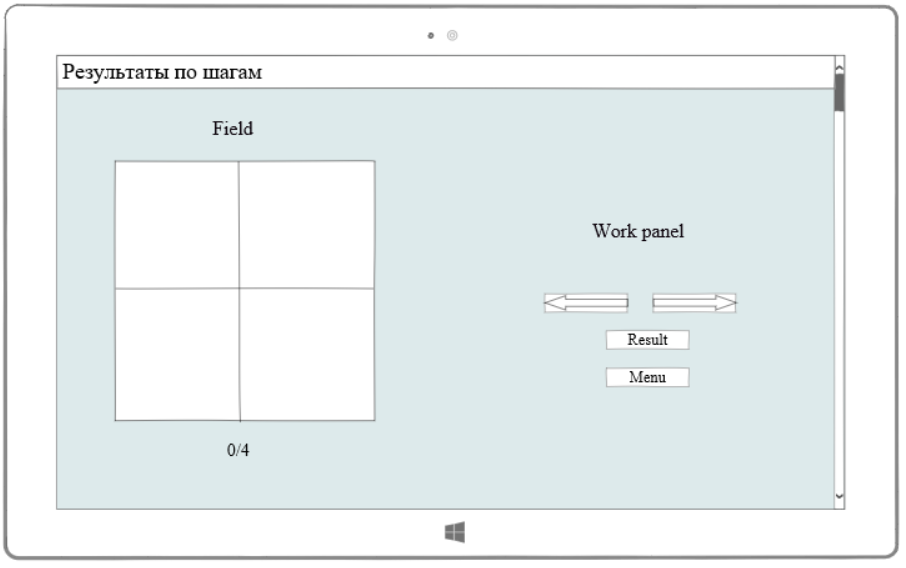


Рисунок 7. Результаты по шагам.

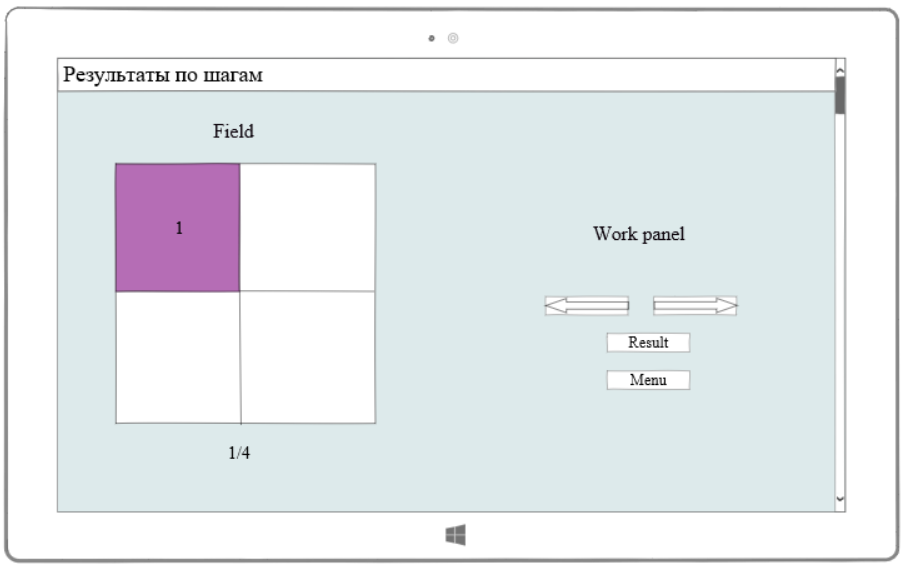


Рисунок 8. Результаты по шагам.

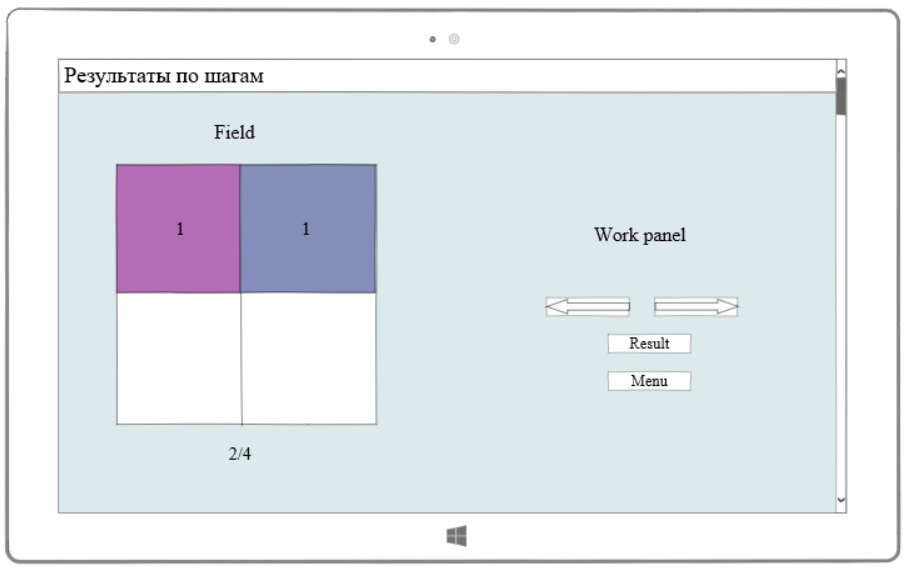


Рисунок 9. Результаты по шагам.

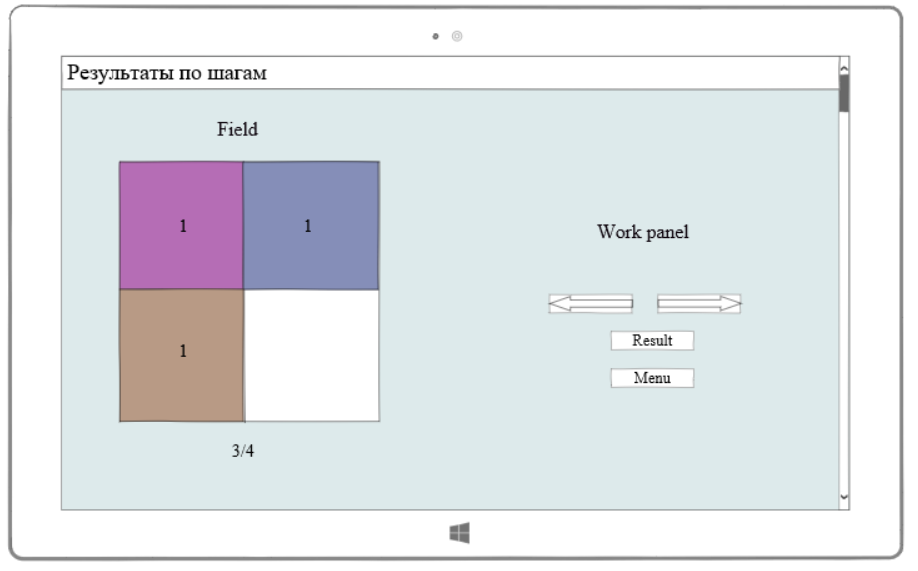


Рисунок 10. Результаты по шагам.

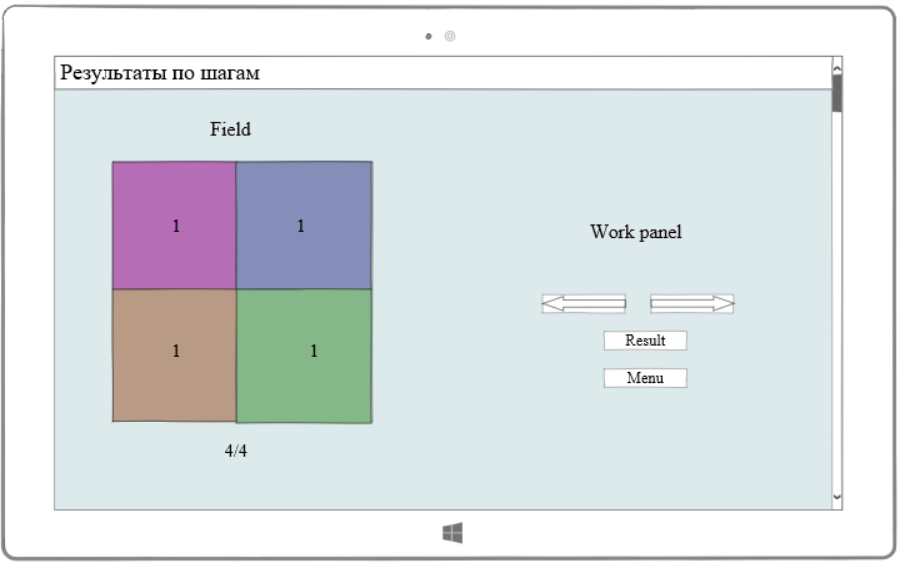


Рисунок 11. Конечный результат.

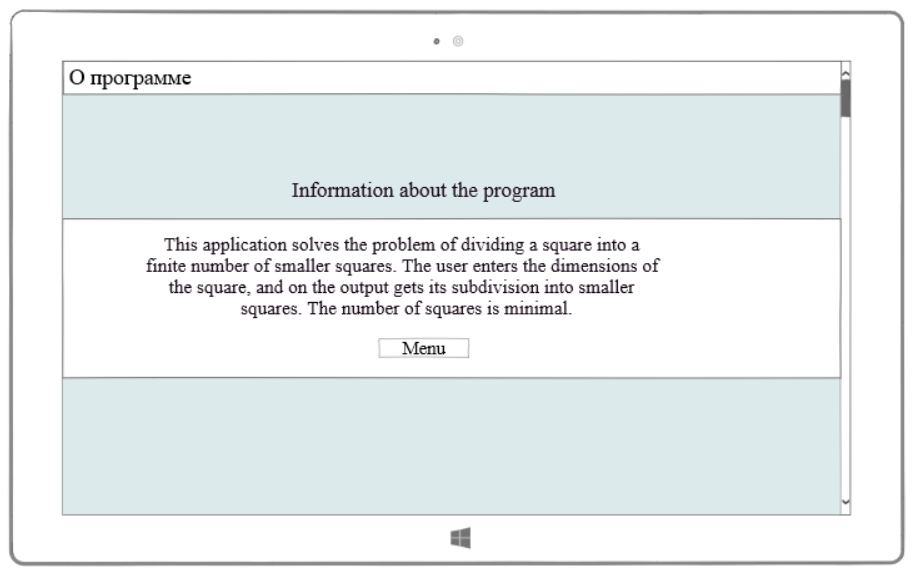


Рисунок 12. Описание программы.

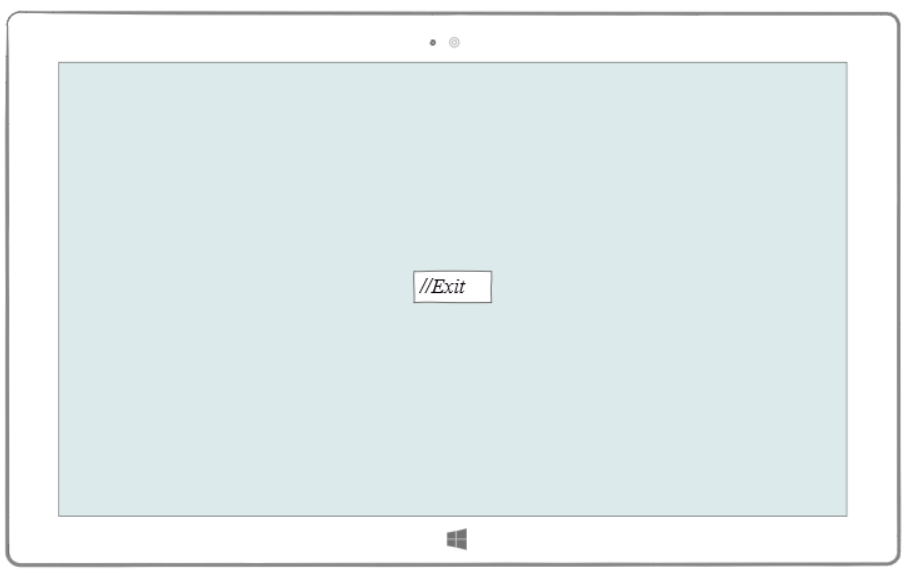


Рисунок 13. Завершение программы.

## Основные классы программы

Ниже представлены основные классы программы:

*class Controller* – инициализирует Model и View;

*class View* – окно, вызывающее эскизы;

*class Model* – описывает алгоритм поиска с возвратом;

*class Field* – задает поле для расстановки квадратов;

*class Square* – описывает меньшие квадраты, которые нужно расставить;

*class PrimeNumbers* – задает разбиение большего квадрата, сторона которого простое число, на меньшие квадраты (см. рис. 1);

*class Number* – производит поиск простого числа в заданном размере поля для упрощения вычислений;

*class StartApplication* – вывод начального меню;

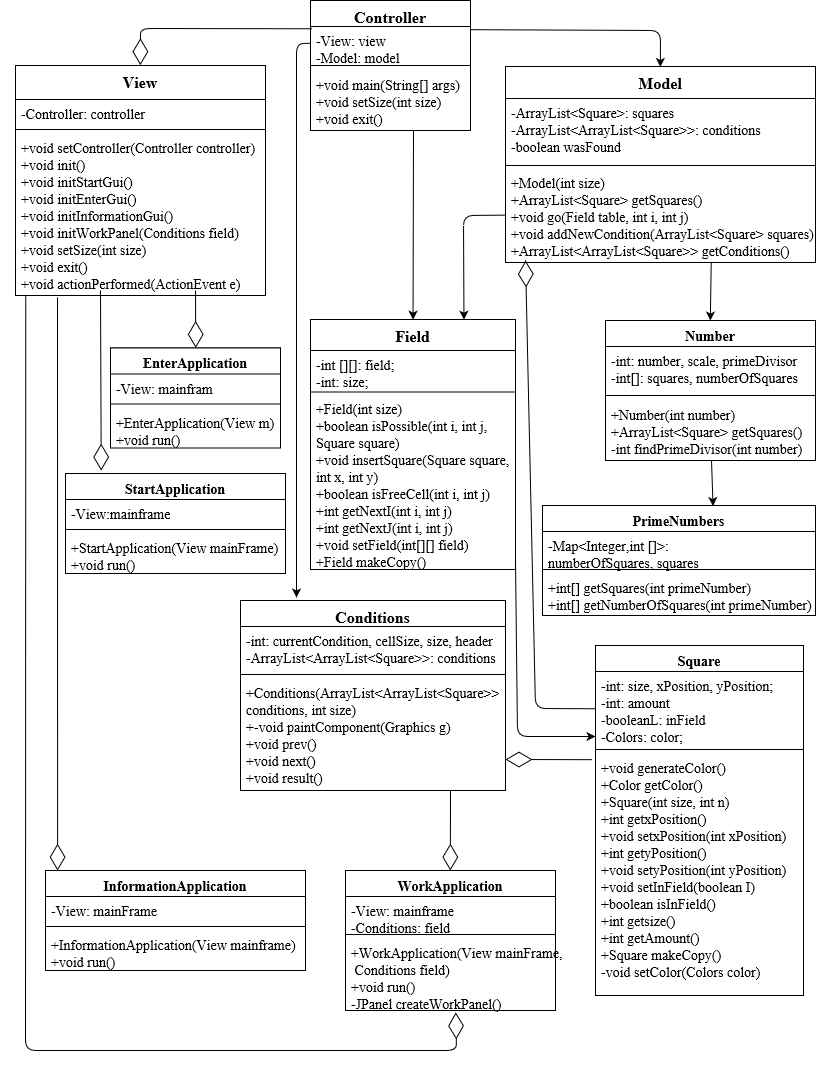
*class EnterApplication* – обрабатывает кнопку ввода размера поля;

*class InformationApplication* – обрабатывает кнопку «About programm»;

*class WorkApplication* – обрабатывает кнопки рабочей области при просмотре результатов;

*class Conditions* – описывает все состояния квадратов по шагам.

## Диаграмма классов



# Тестирование программы

# Заключение

В ходе данной учебной практики была создана программа, которая визуализирует процесс выполнения алгоритма backtracking на примере задачи квадрирования квадрата, используя язык программирования Java.

Программа имеет понятный для пользователя интерфейс и пошагово показывает алгоритм backtracking с помощью простых методов визуализации, а также соответствует важным параметрам парадигмы ООП.

Были получены навыки работы с онлайн-репозиторием на GitHub, а также навыки работы в команде.

# Приложение. Код програмы