Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**КУРСОВОЙ ПРОЕКТ**

**Дисциплина:** Алгоритмы и структуры данных

**Тема:** Решатель судоку.

Выполнил

студент гр. 3530901/90003 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кзыргалина Л.И.

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ахин М.Х.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Санкт-Петербург   
2020

**Оглавление**

[**ТЕХНИЧЕСКОЕ ЗАДАНИЕ** 3](#_Toc59036287)

[**МЕТОД РЕШЕНИЯ** 4](#_Toc59036288)

[**СКРИНШОТЫ ПРОГРАММЫ** 7](#_Toc59036289)

[**ЛИТЕРАТУРА** 9](#_Toc59036290)

# **ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Графический интерфейс представляет собой поле клеток 9 х 9, которое можно заполнить исходными данными с помощью боковой вертикальной панели кнопок с цифрами от 1 до 9. Квадраты 3 х 3 отделяются разными цветами.

Пользователь выбирает клетку поля и нажимает на кнопку с цифрой, которую хочет добавить в поле. Таким образом происходит заполнение игрового поля.

Внизу представлены 2 кнопки: «Очистить» и «Решить».

«Очистить» - очищает поле и запускает Решатель снова. Поле готово для ввода новых данных и решения задачи.

«Решить» - решает судоку и заполняет пустые клетки игрового поля цифрами.

Пример графического интерфейса:

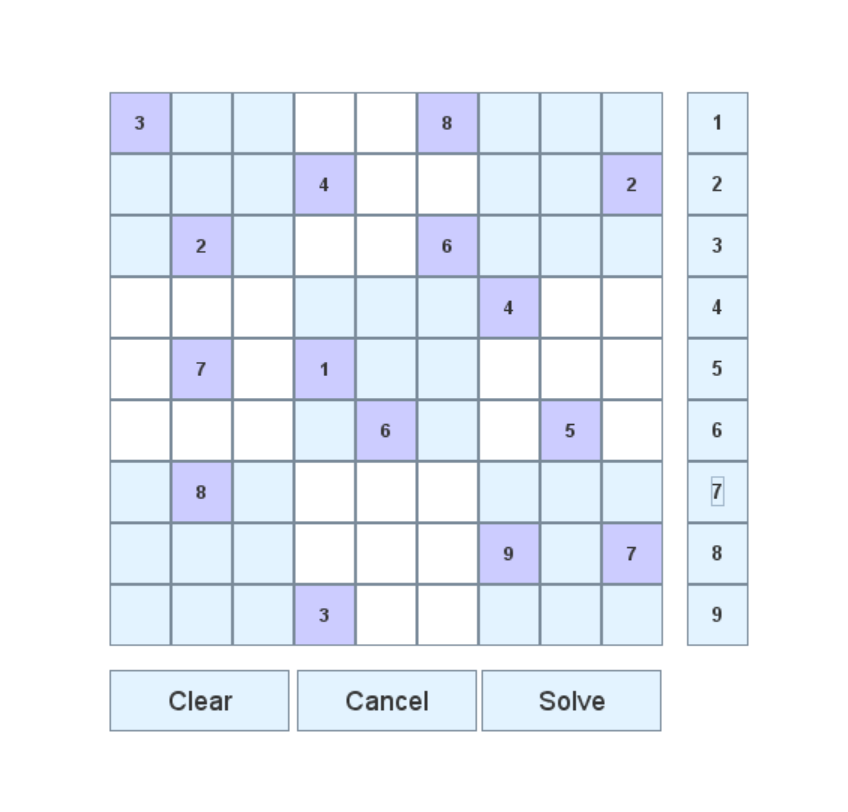
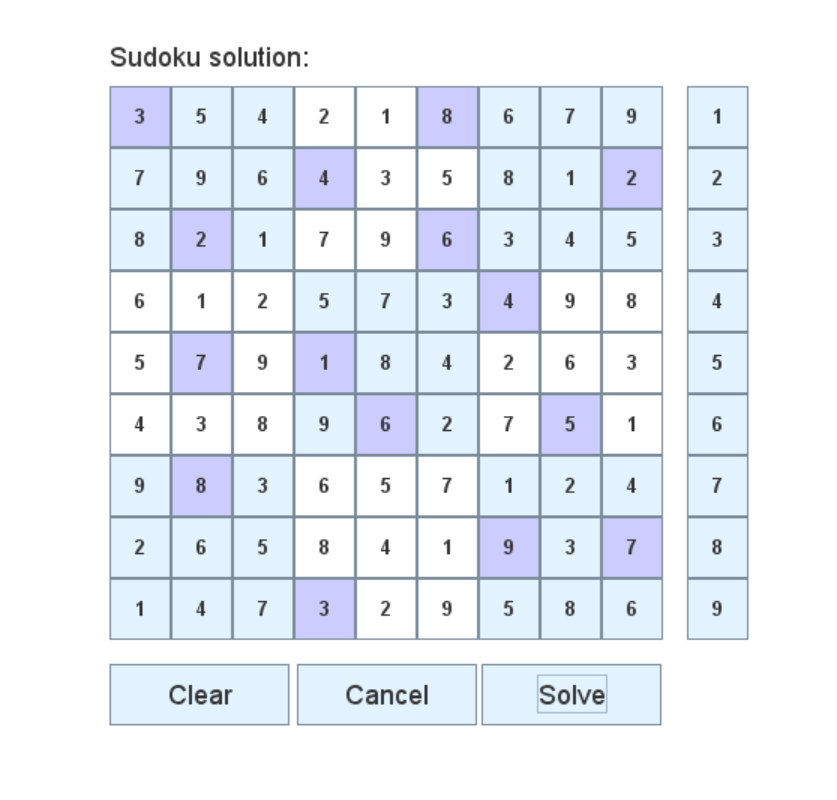
 

Рис. 1 Ввод данных Рис.2 Ответ к заданным входным данным

GitHub репозиторий проекта: <https://github.com/Kzyrgalina/Sudoku>

# **МЕТОД РЕШЕНИЯ**

В проекте использована концепция MVC (Model-View-Controller) для отделения бизнес-логики от визуализации. Код разделен на четыре части части, классы Main.java, Controller.java, View.java и пакет model.

Класс View.java – Создает окно приложения с помощью класса JFrame(), cтроит поле 9 х 9 для ввода и вывода данных. Там же реализуется специальная клавиатура и кнопки «Solve», «Clear» и «Cancel». Ко всем кнопкам добавляется Action listener. Все размеры и положение объектов задаются вручную. Клетки в зависимости от своего положения в поле окрашиваются в голубой или белый цвет для удобства пользователя. В течение ввода данных View.java добавляет новые цифры в поле или выводит сообщение.

Класс **Controller.java** – принимает View.java и Model.java и является связующим звеном между ними. Реализует действия при нажатии на кнопку и сообщает данные в Model.java. После получения ответа от Model.java передает данные в класс View.java для отображения в GUI.

Класс **Main.java** – создает объекты класса View.java, Model.java и Controller.java, принимающий первые два. Запускает программу.

Содержимое пакета model:

Класс **Cell.java** – наследуется от класса JButton(). Принимает два целых числа, обозначающих положение кнопки в матрице. Таким образом объект класса имеет представление о своем положении в двумерном массиве. У каждой клетки есть состояние: EMPTY – пустая, SELECTED – клетка выбрана для установки числа, BUSY – в клетке есть число, она занята.

Класс **CellState.java** – класс-перечисление, содержит состояния клетки: EMPTY, SELECTED, BUSY.

Класс **Model.java** – класс реализует математическую модель, которая решает задачу точного покрытия множества для матрицы значений.

Класс **SolveState.java** – класс-перечисление, представляет собой варианты состояния решателя, такие как: ввод данных, получен ответ, нет решения (ENTER\_THE\_DATA, HAVE\_ANSWER, NO\_SOLUTION, INCORRECT\_ANSWER).

Класс **Triplet.java** – Реализует такую структуру данных как кортеж для трех элементов, используемых для решения задачи.

Более подробная работа программы:

Пользователь выбирает клетку поля и нажимает на кнопку с цифрой, которую хочет добавить в поле. Таким образом происходит заполнение игрового поля.

После нажатия кнопки «Solve» программа решает судоку и выводит ответ, заполняя пустые клетки цифрами.

Решение происходит с помощью Алгоритма Х, предложенного Дональдом кнутом для решения задачи точного покрытия множества, к которым можно отнести задачу решения судоку. В решении используются два ассоциативных массива, содержащие все возможные варианты цифр для каждой клетки.

Математическая модель использует следующие методы:

**sudokuSolver(int matrix[][])** – заполняет ассоциативные массивы rows (строки) и cols (столбцы) кортежами (Triplet), в которых содержатся все возможные варианты покрытия множества. Создает стэк answer для хранения решения задачи. Вызывает рекурсивный метод **algorithmX(LinkedHashmap rows, LinkedHashmap columns, Stack cover)** для заполненных столбцов, колонок и первичного, на тот момент еще пустого, покрытия. Возвращает уже заполненную матрицу с готовым ответом.

**algorithmX(LinkedHashmap rows, LinkedHashmap columns, Stack cover)** – ищет точное покрытие для данной задачи. Выбирается кортеж с минимальным числом элементов в соответствующем столбце. Далее он добавляется в стэк cover возможного покрытия и производится удаление пересекающихся подмножеств в rows и cols с помощью **extractIntersects**(**LinkedHashmap rows, LinkedHashmap columns, Triplet baseRow)**. Рекуррентно вызывает сам себя после успешного удаления. Если решение заходит в тупик, удаленные подмножества возвращаются на места, используя метод **resoreIntresects**(**LinkedHashmap rows, LinkedHashmap columns, Triplet baseRow, Stack buf)**, последний добавленный кортеж удаляется из стэка cover и метод снова вызывается уже от следующего возможного значения. В результате работы возвращается решение в виде стэка cover, элементы которого удовлетворяют всем условиям.

**extractIntersects**(**LinkedHashmap rows, LinkedHashmap columns, Triplet baseRow**) – удаляет пересекающиеся с baseRow столбцы и строки в rows и cols. Возвращает стэк-буфер, в который предварительно был добавлен последний удаленный столбец из cols.

**resoreIntresects**(**LinkedHashmap rows, LinkedHashmap columns, Triplet baseRow, Stack buf)** – возвращает на место столбец из буфера в ассоциативный массив cols. Удаляет из стэка верхний элемент.

Если пользователь при вводе данных совершил ошибку или решил изменить входные данные, он может снова выбрать клетку, цифру в которой хочет изменить, а потом нажать кнопку «Cancel». Тогда значение в поле пропадет и можно будет ввести новое или оставить его пустым. Также есть альтернативный вариант. Можно выделить клетку на поле, а после выбрать другую цифру на специальной клавиатуре, тогда значение в клетке заменится на выбранное.

Если решение пользователю больше не нужно или он решил ввести другие входные данные, можно воспользоваться кнопкой «Clear». Поле будет очищено от всех значений. ы

# **СКРИНШОТЫ ПРОГРАММЫ**

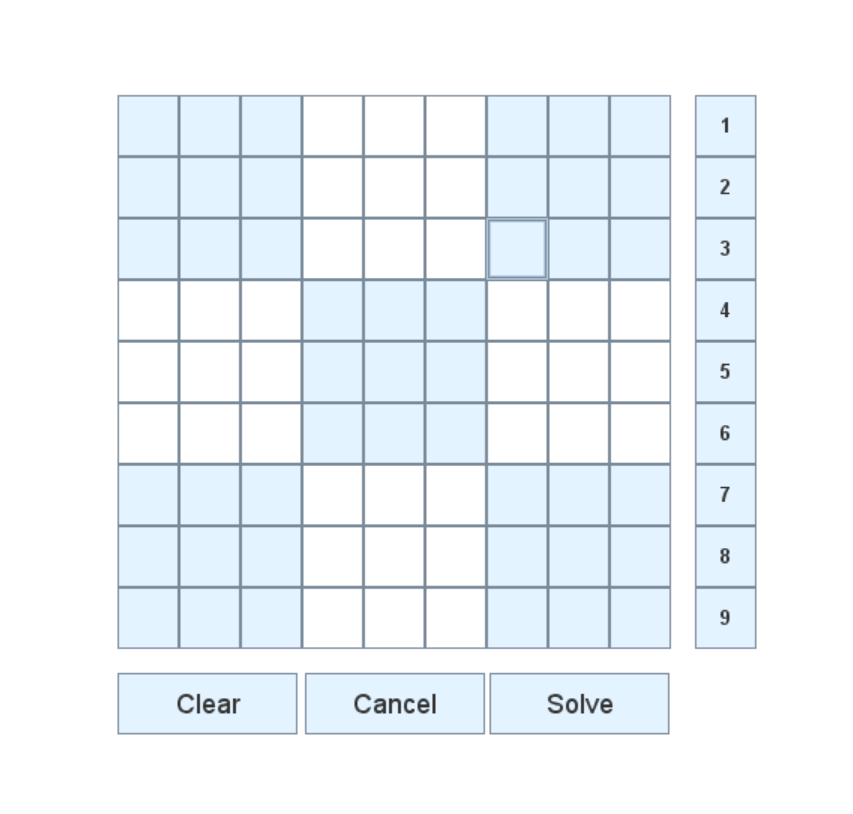


Рис. 3 Начало работы программы

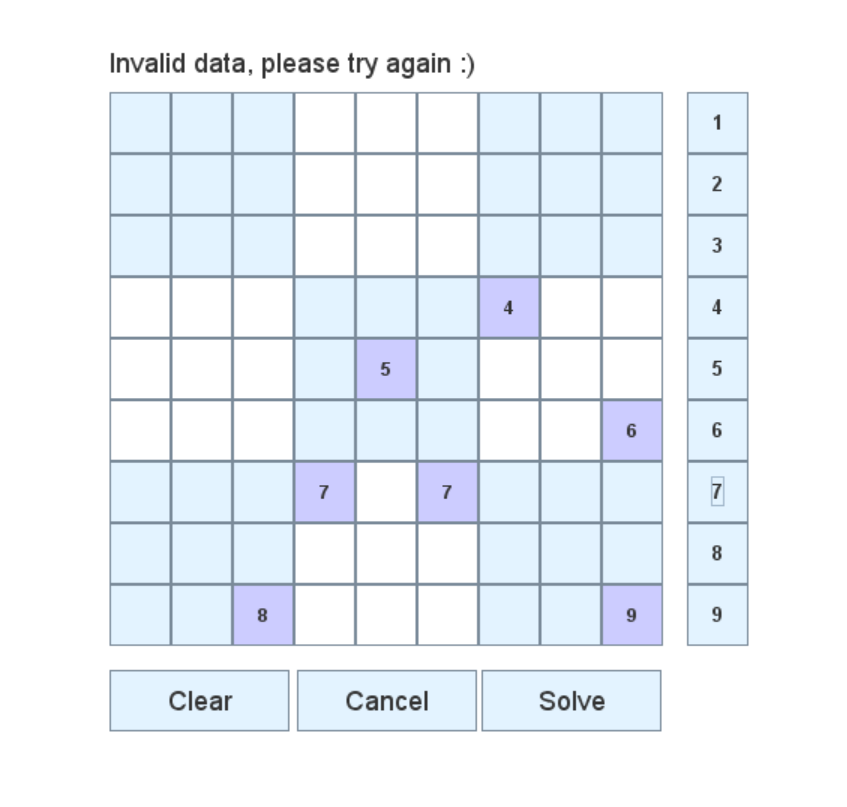


Рис. 4 Сообщение о некорректности входных данных

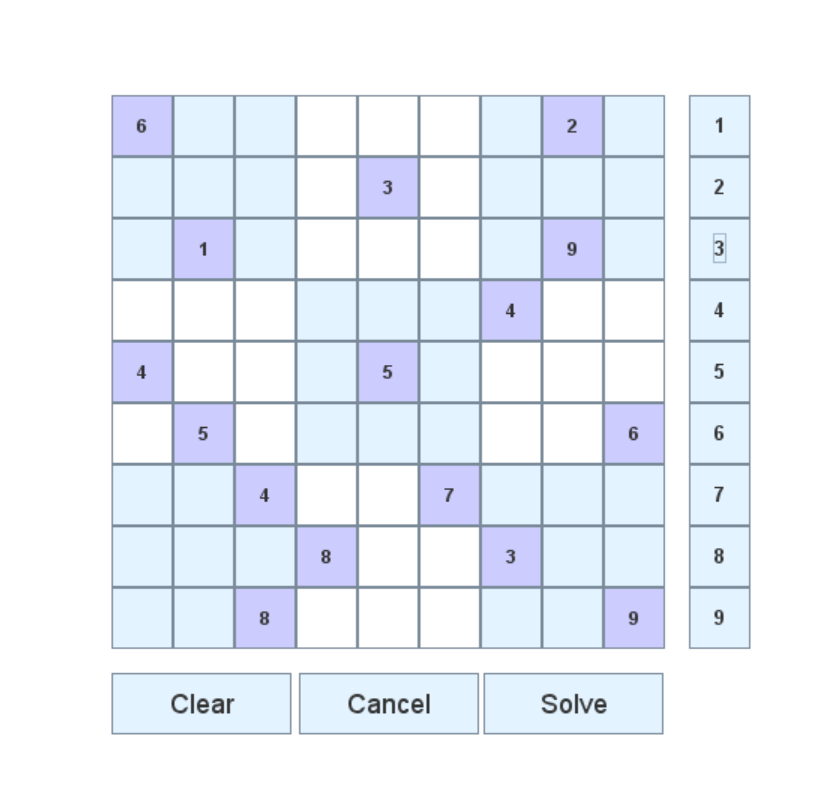


Рис. 5 Заполнение поля входными данными

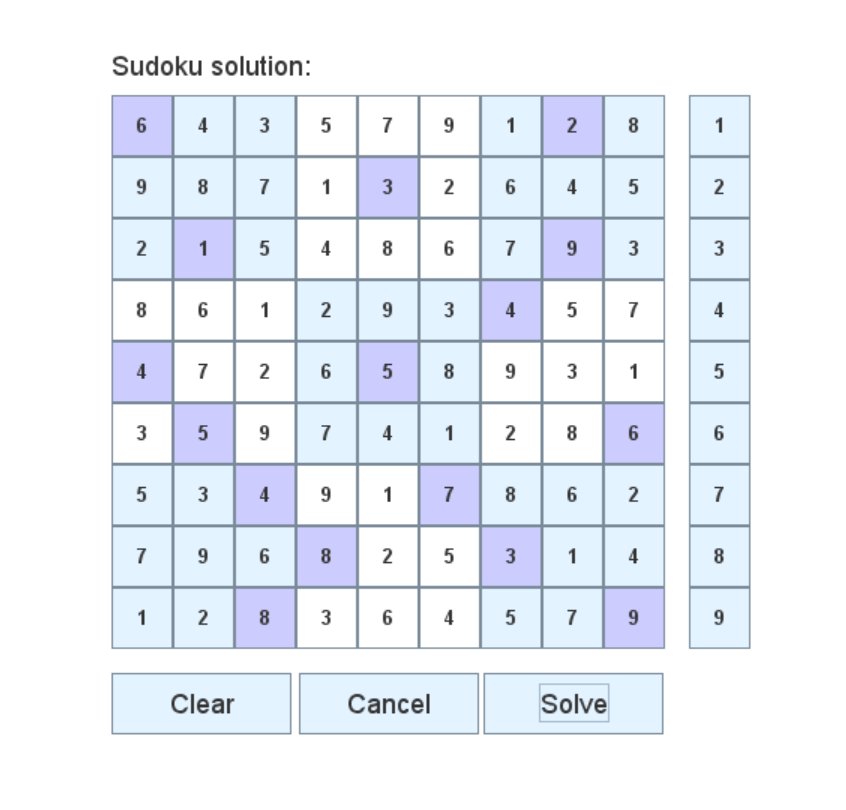


Рис.6 Решение судоку

# **ЛИТЕРАТУРА**

<https://en.wikipedia.org/wiki/Sudoku_solving_algorithms> - алгоритмы для решения судоку

<https://ru.qaz.wiki/wiki/Knuth%27s_Algorithm_X> – описание Алгоритма Х Дональда Кнута

<https://ru.qaz.wiki/wiki/Exact_cover#Sudoku> – судоку как задача точного покрытия множества

<https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html> - документация для Java Swing

<https://docs.oracle.com/en/java/> - документация для Java