Федеральное государственное бюджетное образовательное учреждение высшего образования

«Уральский государственный горный университет»

Инженерно-экономический факультет

Кафедра информатики

Курсовой проект

По дисциплине «Программирование»

На тему «Разработка приложения «Судоку»»

Выполнила:
Студент гр.ИНФ-20-1
Мозговая Ю.В.

Проверила:
Ст. преп. каф. информатики
Волкова Е.А.

г. Екатеринбург, 2022 г.

1.Постановка задачи

1.1Формулировка задания

Формулировка задания

Реализовать логическую головоломку «Судоку». Игровое поле представляет собой квадрат размером 9×9, разделённый на меньшие квадраты со стороной в 3 клетки. Таким образом, всё игровое поле состоит из 81 клетки. В них уже в начале игры стоят некоторые числа (от 1 до 9), называемые подсказками. От игрока требуется заполнить свободные клетки цифрами от 1 до 9 так, чтобы в каждой строке, в каждом столбце и в каждом малом квадрате 3×3 каждая цифра встречалась бы только один раз.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Постановка задачи

Реализовать логическую головоломку «Судоку»

1.2 Алгоритмическое решение задачи

Для решения задачи используются разветвляющиеся и циклические алгоритмы. В качестве поля судоку используется двумерный массив, элементами которого являются числа от 0 до 9, где 0 соответствует пустой клетке.

Алгоритм генерации судоку:

I. Создается базовое поле – это полностью заполненное поле, подчиняющееся правилам судоку.

Заполняется первая строка числами от 1 до 9. Все последующие строки заполняются посредством сдвига влево на 3 или 4 элемента предыдущей строки.

1	2	3	4	5	6	7	8	9	
4	5	6	7	8	9	1	2	3	Сдвиг влево на 3
7	8	9	1	2	3	4	5	6	Сдвиг влево на 3
2	3	4	5	6	7	8	9	1	Сдвиг влево на 4
5	6	7	8	9	1	2	3	4	Сдвиг влево на 3
8	9	1	2	3	4	5	6	7	Сдвиг влево на 3
3	4	5	6	7	8	9	1	2	Сдвиг влево на 4
6	7	8	9	1	2	3	4	5	Сдвиг влево на 3
9	1	2	3	4	5	6	7	8	Сдвиг влево на 3

II. Базовое поле перемешивается так, что перемешанное поле также соответствует правилам судоку.

Для перемешивания используются следующие методы:

- Транспонирование матрицы
- Обмен двух строк внутри района (район – часть поля судоку размерностью 3 x 9)
- Обмен двух столбцов внутри района
- Обмен двух горизонтальных районов
- Обмен двух вертикальных районов

Эти методы вызываются случайным образом определенное количество раз. Таким образом получается перемешанное поле.

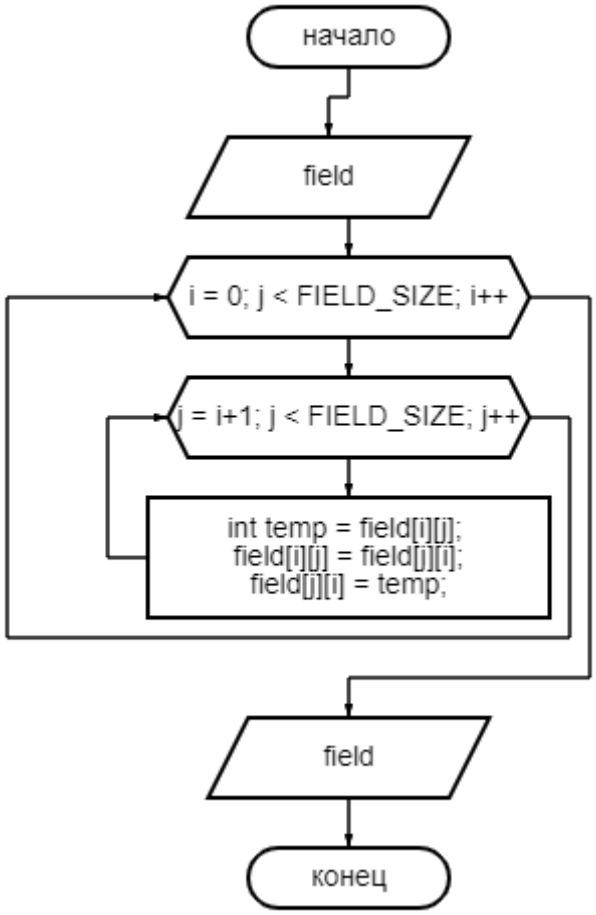

Рис.1 Транспонирование матрицы

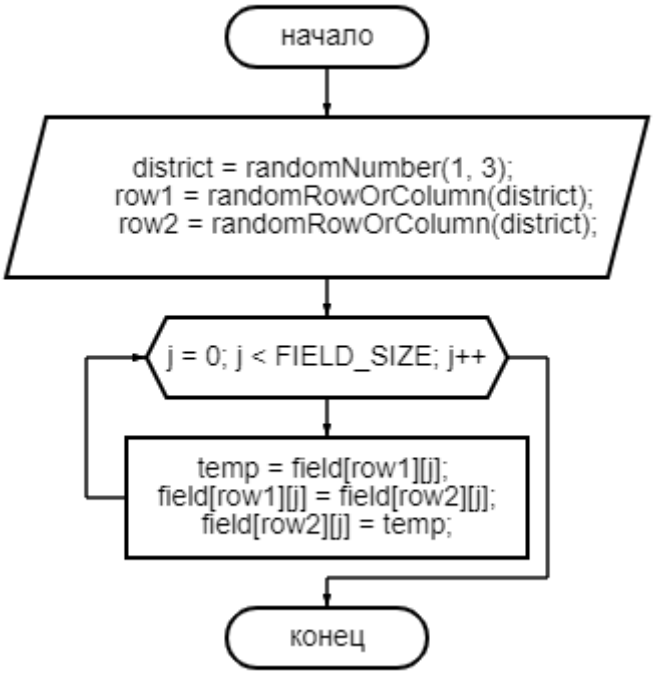

Рис.2 Обмен строк внутри района

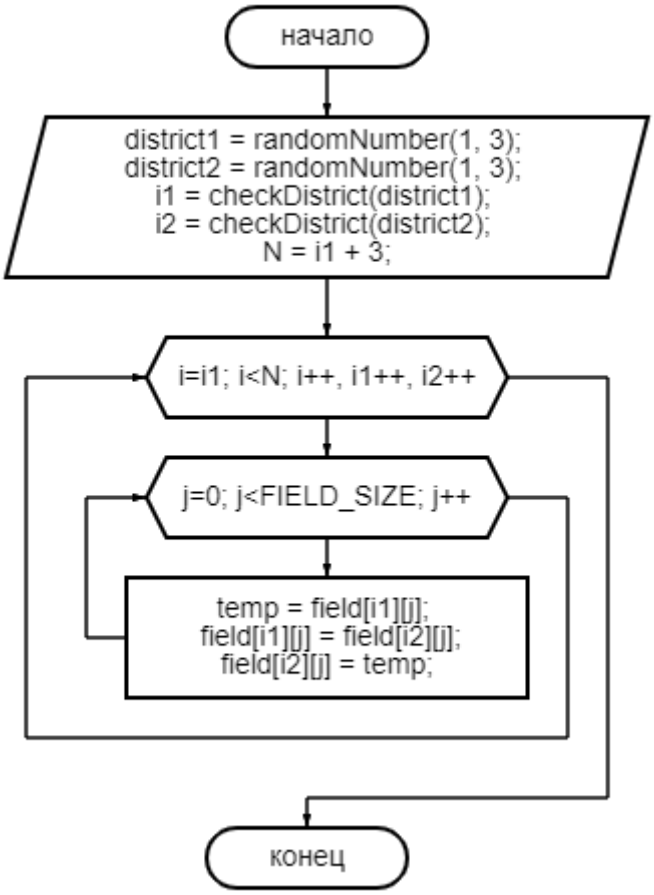

Рис.3 Обмен горизонтальных районов

III. Удаление клеток

Случайным образом задается количество пустых клеток и выбирается непустая клетка. Если при ее удалении судоку не имеет одного решения (не имеет решений вообще или имеет больше 1 решения), то этой клетке возвращается первоначальное значение. Клетки удаляются до тех пор, пока их общее количество не будет соответствовать заданному. Либо до тех пор, пока время выполнения задачи не превысит 1000 мс, что говорит о том, что алгоритм не может удалить заданное количество клеток так, чтобы было лишь одно решение. Тогда задается другое значение количества пустых клеток, и алгоритм начинает работу с начала.

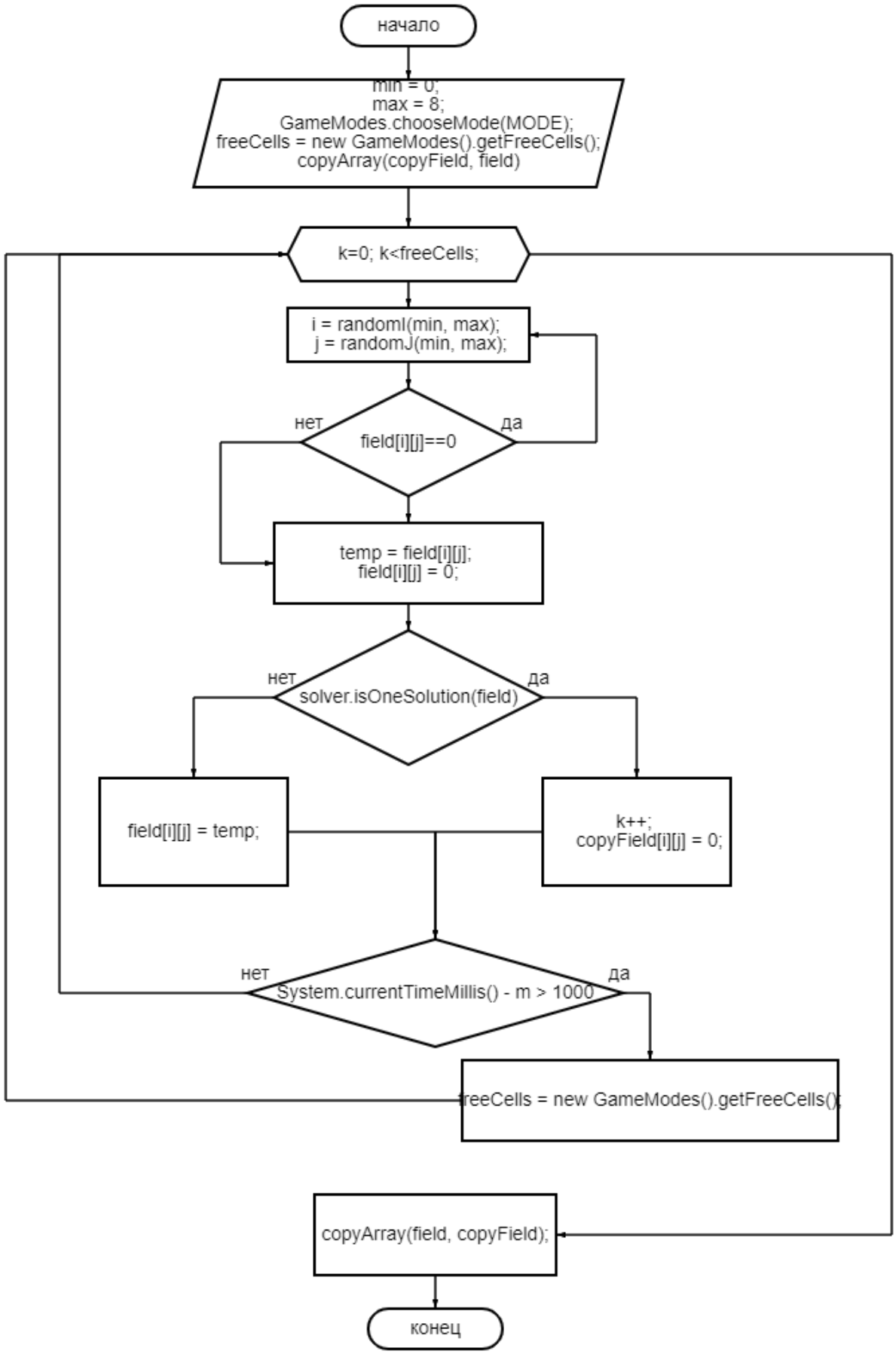

Рис.4 Обмен горизонтальных районов

Алгоритм решения судоку:

Судоку решается рекурсивным методом. Перебираются все возможные варианты до тех пор, пока все пустые клетки не будут заполнены, при этом происходит проверка: существует ли значение в строке, в столбце или блоке 3x3, чтобы избежать повторения чисел.

Алгоритм проверки судоку на единственное решение:

Проверка на единственность решения заключается в том, чтобы сначала заполнить судоку числами от 1 до 9, а затем числами от 9 до 1. Если после обоих заполнений решения оказались разными, то судоку имеет несколько решений. Если решения совпали, то судоку имеет только одно решение.

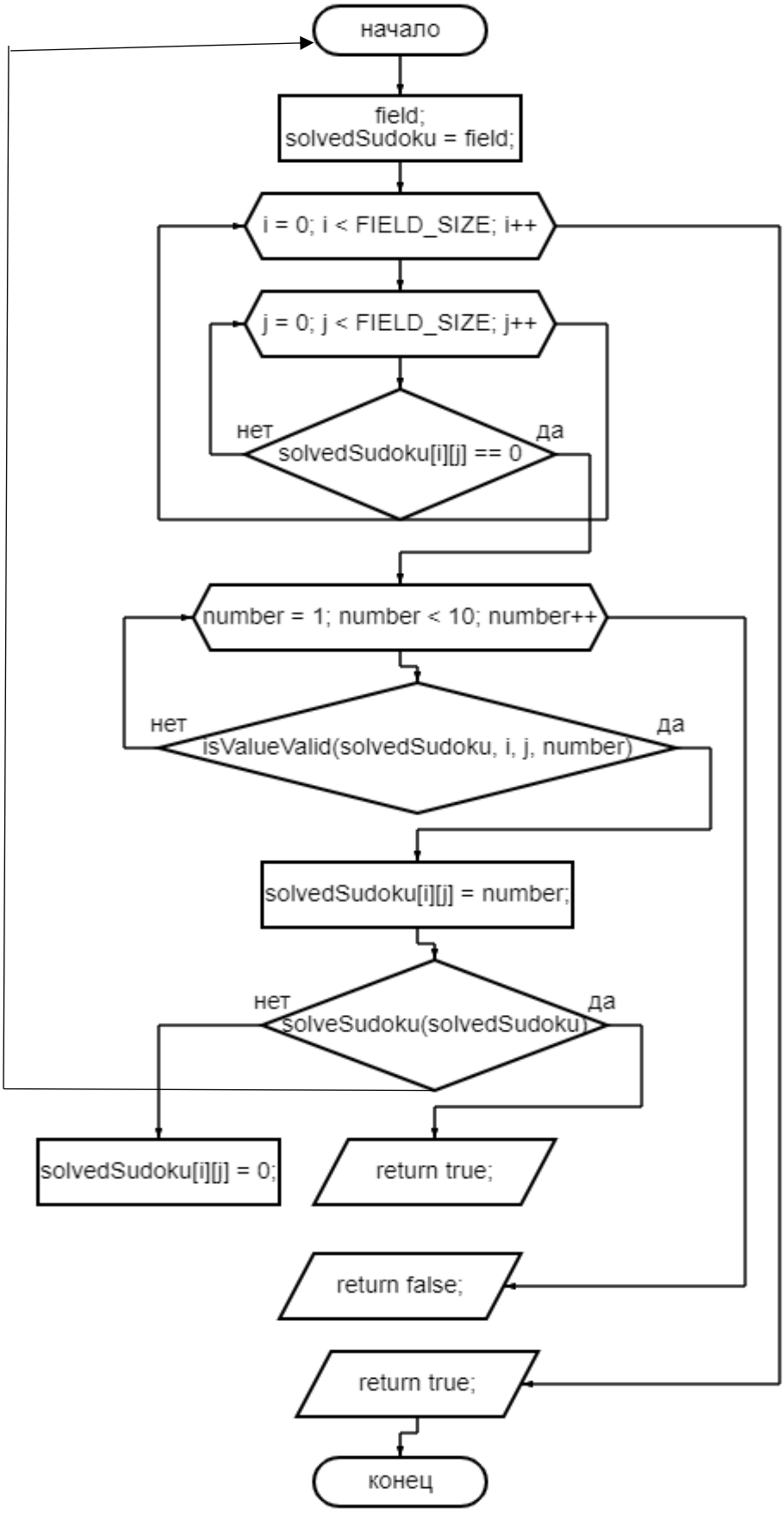

Рис.5 Алгоритм решения судоку

1.3 Контрольные примеры

В качестве контрольных примеров возьмем три матрицы, созданных путем перемешивания базового поля.

A:	B:	C:
714 825 936	174 369 258	714 825 936
825 936 147	396 582 471	825 936 147
936 147 258	285 471 369	936 147 258
471 582 693	417 693 582	147 258 369
693 714 825	528 714 693	258 369 471
582 693 714	639 825 714	369 471 582
369 471 582	741 936 825	471 582 693
258 369 471	852 147 936	582 693 714
147 258 369	963 258 147	693 714 825

A: транспонирование, обмен 1 и 3 горизонтальных районов, обмен 1 и 3 строк, обмен 5 и 6 строк

B: транспонирование, обмен 2 и 3 вертикальных районов, обмен 2 и 3 столбцов, обмен 2 и 3 строк

C: транспонирование, обмен 1 и 3 горизонтальных районов, обмен 1 и 2 горизонтальных районов

Удалим по три случайных клетки в каждой матрице:

A:	B:	C:
714 825 936	174 36*0* 258	714 825 936
8*0*5 936 147	396 582 471	825 9*0*6 147
936 147 258	285 471 369	936 147 258
471 582 693	417 693 *0*82	147 258 369
693 71*0* 825	528 714 693	258 369 471
582 693 714	639 825 714	369 471 582
369 471 582	7*0*1 936 825	47*0* 582 693
2*0*8 369 471	852 147 936	582 693 714
147 258 369	963 258 147	693 714 82*0*

Решение:

A:	B:	C:
714 825 936	174 36*0* 258	714 825 936
8*0*5 936 147	396 582 471	825 9*0*6 147
936 147 258	285 471 369	936 147 258
471 582 693	417 693 *0*82	147 258 369

693 71*0* 825	528 714 693	258 369 471
582 693 714	639 825 714	369 471 582
369 471 582	7*0*1 936 825	47*0* 582 693
2*0*8 369 471	852 147 936	582 693 714
147 258 369	963 258 147	693 714 82*0*

Начнем с элементов A(2, 2), B(1, 6), C(2, 5).

Для A число 1 не подходит, т.к. оно уже есть в строке. Число 2 подходит, т.к. не встречается в строке, столбце и блоке. Для B подходит число 9. Для C подходит 3.

A:	B:	C:
714 825 936	174 36***9*** 258	714 825 936
8***2***5 936 147	396 582 471	825 9***3***6 147
936 147 258	285 471 369	936 147 258
471 582 693	417 693 ***0***82	147 258 369
693 71***0*** 825	528 714 693	258 369 471
582 693 714	639 825 714	369 471 582
369 471 582	7***0***1 936 825	47***0*** 582 693
2***0***8 369 471	852 147 936	582 693 714
147 258 369	963 258 147	693 714 82***0***

Следующие элементы - A(5, 6), B(4, 7), C(7, 3).

Для A подходит число 4. Для B подходит число 5. Для C подходит 1. Эти числа не встречаются в соответствующих им строках, столбцах и блоках.

A:	B:	C:
714 825 936	174 36***9*** 258	714 825 936
8***2***5 936 147	396 582 471	825 9***3***6 147
936 147 258	285 471 369	936 147 258
471 582 693	417 693 ***5***82	147 258 369
693 71***4*** 825	528 714 693	258 369 471
582 693 714	639 825 714	369 471 582
369 471 582	7***0***1 936 825	47***1*** 582 693
2***0***8 369 471	852 147 936	582 693 714
147 258 369	963 258 147	693 714 82***0***

Следующие элементы - A(8, 2), B(7, 2), C(9, 9).

Для A подходит число 5. Для B подходит число 4. Для C подходит 5. Эти числа не встречаются в соответствующих им строках, столбцах и блоках.

Проверим матрицы на единственность решения:

Заполнение в прямом порядке:

A:	B:	C:
714 825 936	174 36***9*** 258	714 825 936
8***2***5 936 147	396 582 471	825 9***3***6 147
936 147 258	285 471 369	936 147 258
471 582 693	417 693 ***5***82	147 258 369
693 71***4*** 825	528 714 693	258 369 471
582 693 714	639 825 714	369 471 582
369 471 582	7***4***1 936 825	47***1*** 582 693
2***5***8 369 471	852 147 936	582 693 714
147 258 369	963 258 147	693 714 82***5***

Заполнение в обратном порядке:

A:	B:	C:
714 825 936	174 36***9*** 258	714 825 936
8***2***5 936 147	396 582 471	825 9***3***6 147
936 147 258	285 471 369	936 147 258
471 582 693	417 693 ***5***82	147 258 369
693 71***4*** 825	528 714 693	258 369 471
582 693 714	639 825 714	369 471 582
369 471 582	7***4***1 936 825	47***1*** 582 693
2***5***8 369 471	852 147 936	582 693 714
147 258 369	963 258 147	693 714 82***5***

Таким образом, решение у всех трех судоку одно

2. Решение задачи

2.1 Выбор средств реализации

Для реализации приложения был выбран объектно-ориентированный язык программирования общего назначения - Java.

Преимущества Java:

Независимость от аппаратной части и ОС. Важно лишь наличие исполняющей среды и виртуальной Java машины - JVM. Компьютерная архитектура значения не имеет. Байт-код легко интерпретируется на любой машине.

Гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы, вызывают немедленное прерывание.

Надёжность. Программы на Java стабильно работают в любых условиях. Компилятор способен выявить ошибки ещё до выполнения кода, то есть на ранних стадиях. А контроль выполнения позволяет предотвратить сбои в памяти.

В качестве IDE была выбрана интегрированная среда разработки программного обеспечения IntelliJ IDEA. На данный момент это один из лучших и удобнейших инструментов для Java разработки. Преимущества: автодополнение кода, рефакторинг, профилирование, шаблоны кода, выделение конструкций цветом, интеграция и поддержка множества инструментов, необходимых разработчику.

2.2 Описание основных классов

Структура проекта:

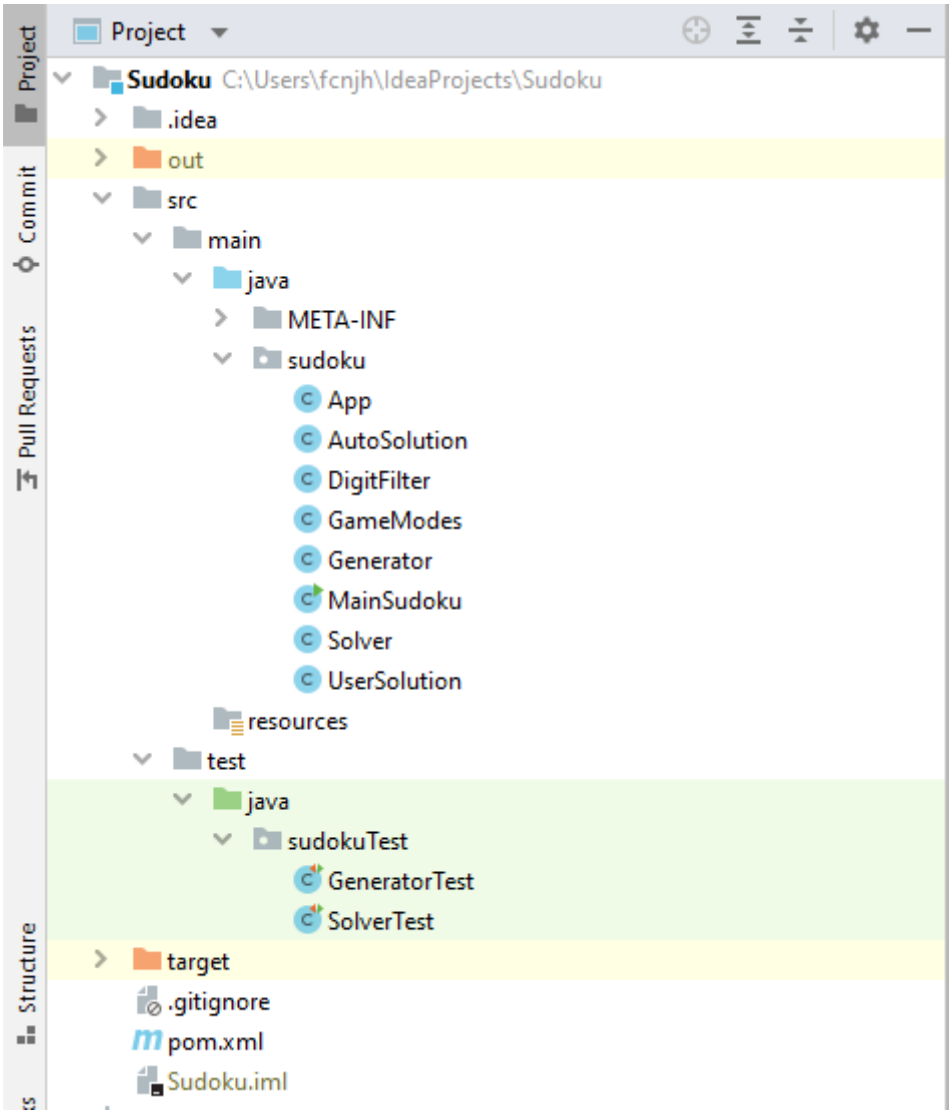

Рис. 6 Структура проекта

При решении задачи было разработано 9 классов, показанных на диаграмме (рисунок 7).

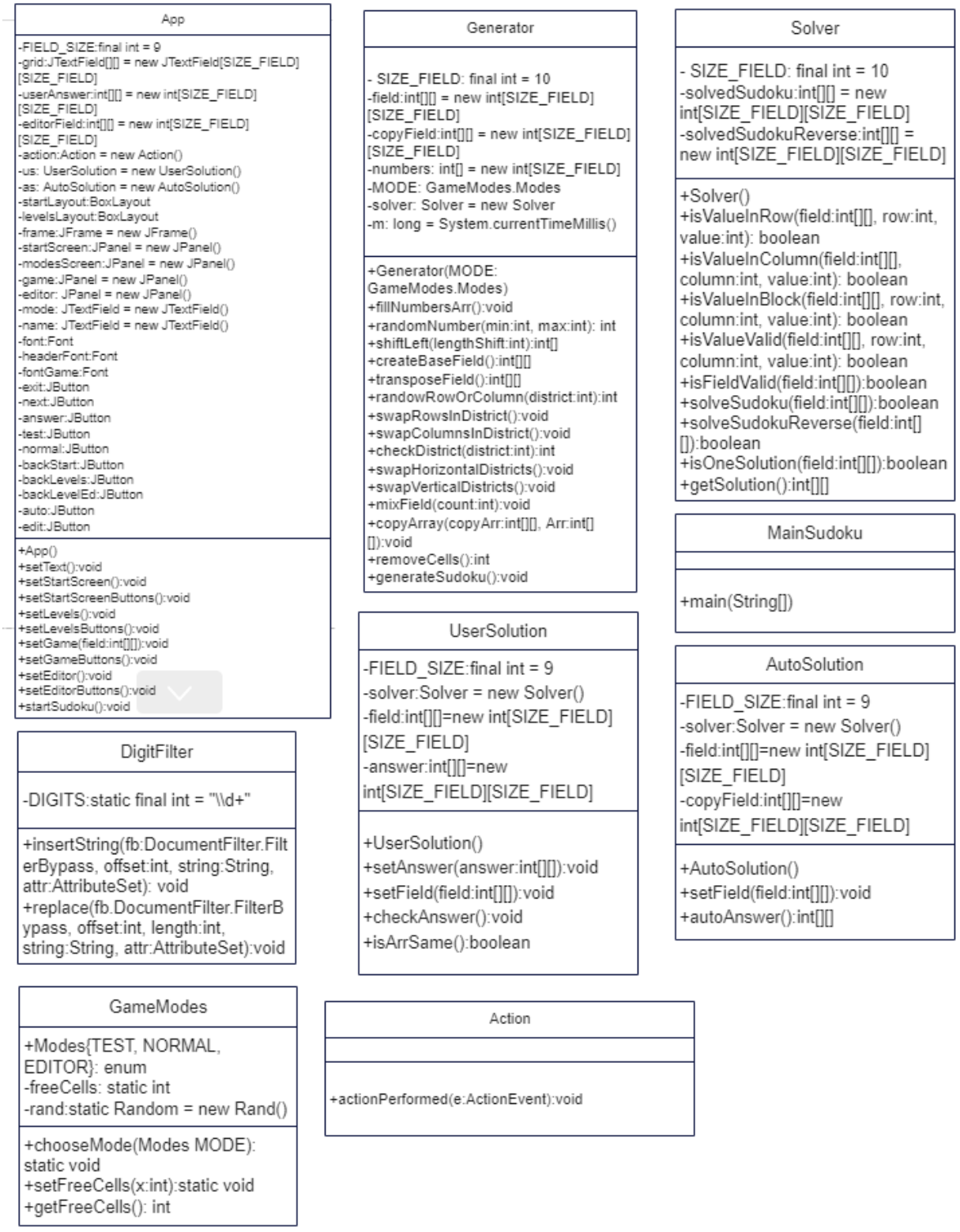

Рис.7 Диаграмма классов

Класс MainSudoku создает объект класса App и запускает приложение.

```java
public class MainSudoku {
    public static void main(String[] args) {
        App app = new App();
        app.startSudoku();
    }
}
```

Листинг 1. Класс MainSudoku

Класс Generator реализует алгоритм создания судоку: создается базовое поле, затем оно перемешивается с помощью различных методов и удаляется определенное количество клеток.

```java
/* создание базового поля:
* в соответствие с правилами судоку
* каждую строку сдвигаем на 3;
* при переходе в новый горизонатльный район,
* сдвигаем строку на 4*/
public int[][] createBaseField() {
    fillNumbersArr();
    int counter = 0;

    for(int i=0; i<FIELD_SIZE; i++) {
        for(int j=0; j<FIELD_SIZE; j++) {
            field[i][j] = numbers[j];
        }

        counter++;
        if(counter % 3 != 0) {
            shiftLeft( lengthShift: 3);
        }
        if(counter % 3 == 0) {
            shiftLeft( lengthShift: 4);
        }

    }
    return field;
}
```

Листинг 2. Создание базового поля

```java
/* 1ый метод для перемешивания поля:
* транспонирование матрицы */
public int[][] transposeField() {
    for(int i=0; i<FIELD_SIZE; i++) {
        for(int j=i+1; j<FIELD_SIZE; j++) {
            int temp = field[i][j];
            field[i][j] = field[j][i];
            field[j][i] = temp;
        }
    }
    return field;
}
```

Листинг 3. Один из методов перемешивания – транспонирование

```java
//удаление клеток
public int removeCells() {
    int min = 0;
    int max = 8;
    GameModes.chooseMode(MODE);
    int freeCells = new GameModes().getFreeCells();
    copyArray(copyField, field);
    for (int k = 0; k < freeCells; ) {
        int i = randomNumber(min, max); //выбор случайной клетки
        int j = randomNumber(min, max);
        while (field[i][j] == 0) {//если клетка уже удалена, выбираем другую
            i = randomNumber(min, max);
            j = randomNumber(min, max);
        }

        int temp = field[i][j];
        field[i][j] = 0;

        if (solver.isOneSolution(field)) { //удаляем клетку, если без нее
            k++; //судоку будет иметь одно решение
            copyField[i][j] = 0;
        }
        if (!solver.isOneSolution(field)) {
            field[i][j] = temp;
        }

        if (System.currentTimeMillis() - m > 1000) { //если выполнение
программы превышает 1000 мс,
            freeCells = new GameModes().getFreeCells(); //значит алгоритм не
может удалить заданное количество клеток
        }//выбирается новое значение количества пустых клеток
    }

    copyArray(field, copyField);
    return freeCells;
}
```

Листинг 4. Удаление клеток

Класс Solver реализует алгоритм рекурсивного решения судоку. Класс также содержит методы проверки на наличие значения в строке, в столбце или

блоке 3x3, чтобы избежать повторения чисел, и метод проверки на единственность решения.

```
public boolean isValueInBlock(int[][] field, int row, int column, int value) {
    int blockFirstRow = row - row % 3; //значения для нахождения расположения
    int blockFirstColumn = column - column % 3; //первого элемента блока
    for (int i = blockFirstRow; i < blockFirstRow + 3; i++) {
        for (int j = blockFirstColumn; j < blockFirstColumn + 3; j++) {
            if(field[i][j] == value) {
                return true; //такое значение уже есть в блоке
            }
        }
    }
    return false; //такого значения еще нет
}
```

Листинг 5. Проверка на наличие значения в блоке

```
public boolean solveSudoku(int[][] field) {
    solvedSudoku = field;
    for (int i = 0; i < FIELD_SIZE; i++) {
        for (int j = 0; j < FIELD_SIZE; j++) {
            if(solvedSudoku[i][j] == 0) {
                for (int number = 1; number < 10; number++) {//перебираются все возможные варианты
                    if(isValueValid(solvedSudoku, i, j, number)) {//подходит ли значение
                        solvedSudoku[i][j] = number;
                        if(solveSudoku(solvedSudoku)) {//рекурсия
                            return true; //решена
                        } else {
                            solvedSudoku[i][j] = 0;
                        }
                    }
                }
                return false; //нет решений
            }
        }
    }
    return true; //не нуждается в решении
}
```

Листинг 6. Решение судоку

```java
public boolean isOneSolution(int[][] field) {
    if (!solveSudoku(field)) {
        return false; //нельзя решить
    }
    solveSudokuReverse(field);
    return solvedSudoku == solvedSudokuReverse; //true одно решение
}
```

Листинг 7. Проверка решения на единственность

В классе GameModes содержится описание режимов игры, а также генерируется количество пустых клеток в соответствии с режимом. Класс DigitFilter содержит методы, позволяющие вводить в пустые клетки только числа. Класс UserSolution проверяет решение судоку пользователем. Класс AutoSolution решает судоку автоматически.

2.3 Интерфейс приложения

Класс App отвечает за создание интерфейса приложения (фрейм, панели, кнопки, поле игры). Внутри класса App вложен класс Action, который отвечает за привязку событий к кнопкам. Интерфейс приложения прост и понятен в использовании. Кнопка «Играть» в главном меню ведет в меню выбора режима.

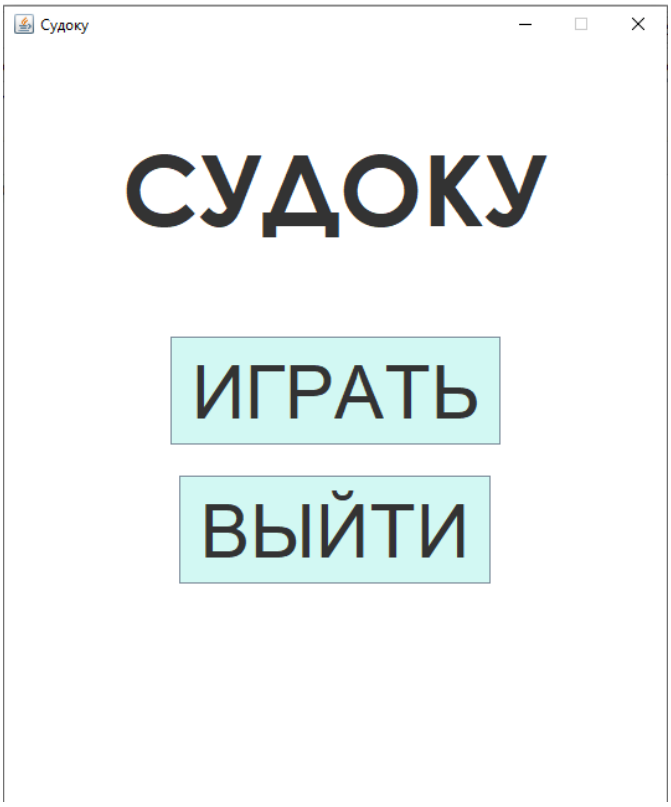

Рис. 8 Главное меню

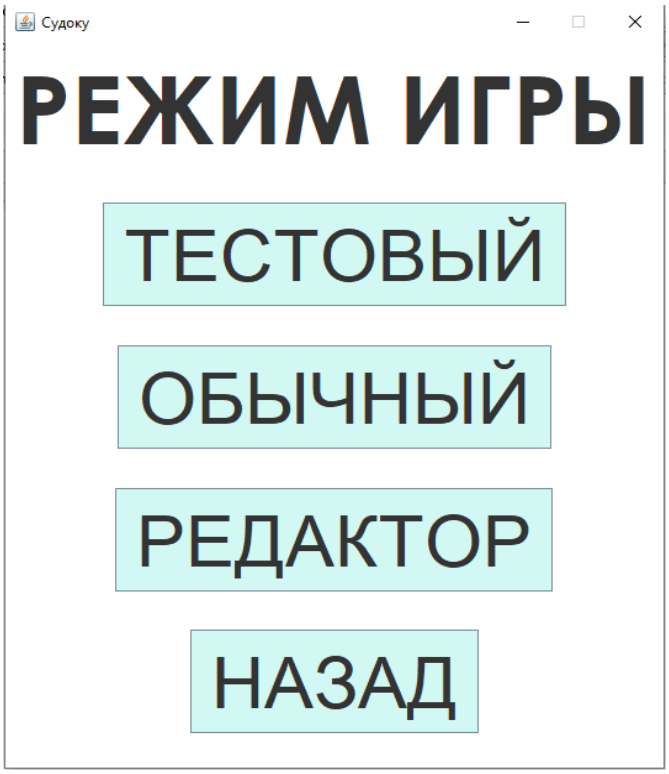

Рис. 9 Меню выбора режима

В меню выбора режима доступно три варианта: Тестовый (небольшое количество пустых клеток), Обычный (среднее количество пустых клеток), Редактор (возможность создать свое судоку).

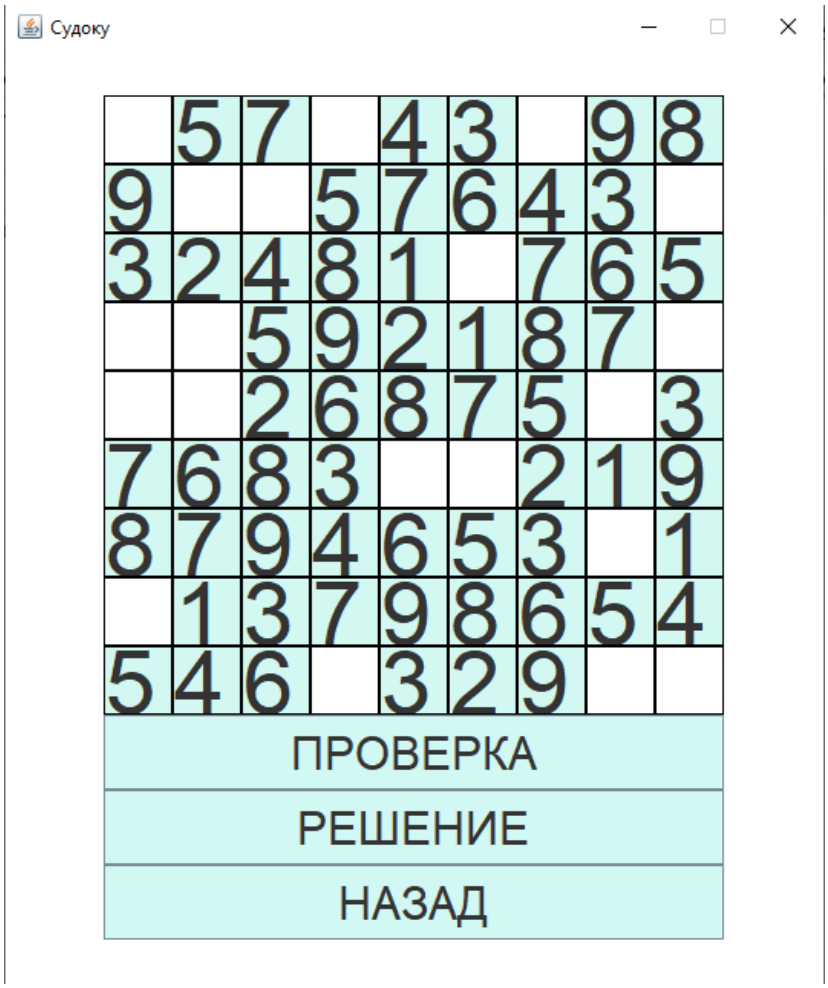

Рис.10 Игра

В режиме игры (Тестовый или Обычный) есть поле судоку, заполненное определенным количеством подсказок и пустых клеток. Так же есть две кнопки: Проверка (проверяет, решение пользователя) и Решение (заполняет пустые клетки автоматически).

В режиме редактора – пустая сетка, которую заполняет пользователь, создавая таким образом свой уровень судоку. Нажимая кнопку Далее, пользователь попадает в режим игры, но, вместо автоматически сформированной сетки, используется его собственная.

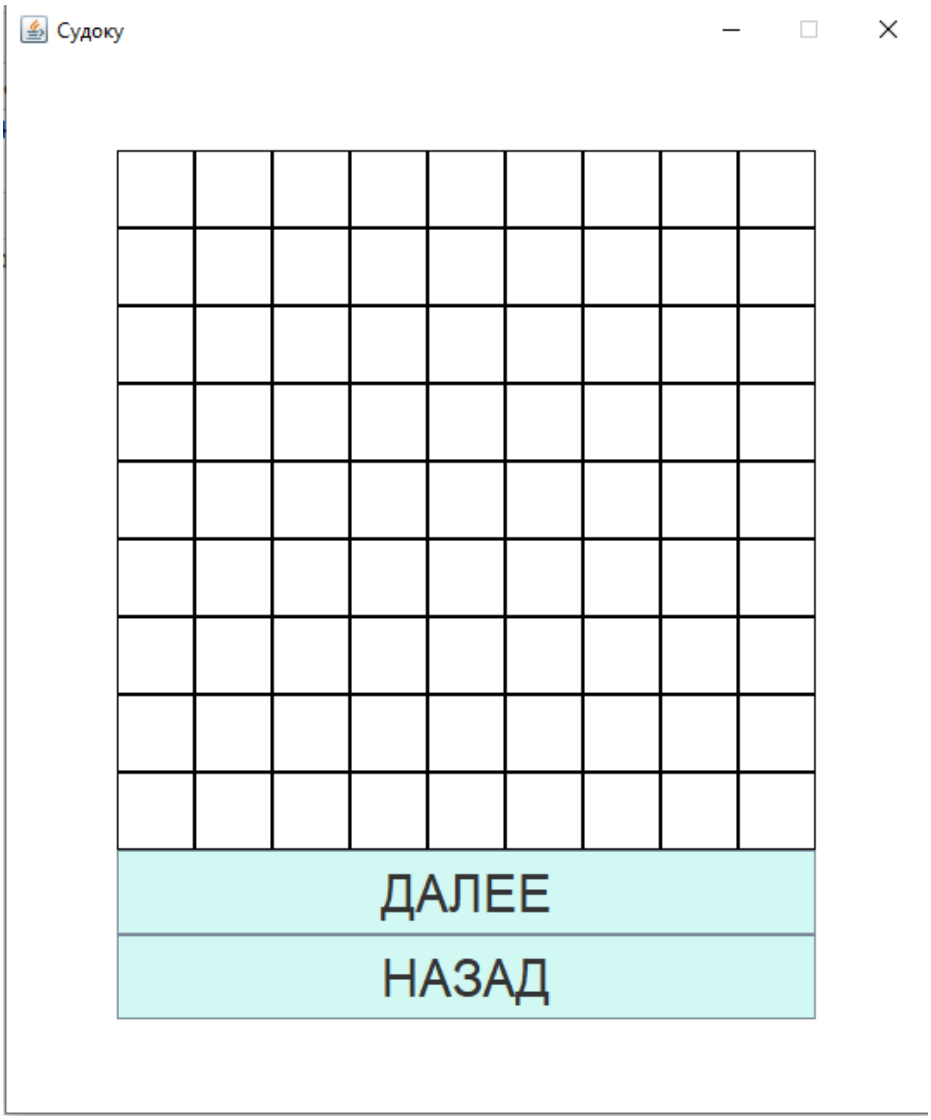

Рис.11 Редактор

```
//создание панели редактора
public void setEditor() {
    setEditorButtons();
    editor.setBackground(Color.WHITE);
    editor.setLayout(new GridBagLayout());
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 1;
    gbc.fill = GridBagConstraints.BOTH;

    for (int i = 0; i < FIELD_SIZE; i++) {
        for (int j = 0; j < FIELD_SIZE; j++) {
            grid[i][j] = new JTextField("");
            PlainDocument doc = (PlainDocument) grid[i][j].getDocument();
            doc.setDocumentFilter(new DigitFilter());//чтобы вводились только
цифры
            int finalI = i;
            int finalJ = j;
            grid[i][j].addKeyListener(new KeyAdapter() {
                @Override
                public void keyTyped(KeyEvent e) {
                    if (grid[finalI][finalJ].getText().length() == 1) {
                        e.consume();//чтобы нельзя было ввести больше одного
числа
```

```
                    }
                }
            });
            grid[i][j].setFont(font);
            grid[i][j].setPreferredSize(new Dimension(45, 45));
            grid[i][j].setBorder(BorderFactory.createLineBorder(Color.BLACK,
1));
            gbc.gridwidth = 1;
            gbc.gridheight = 1;
            editor.add(grid[i][j], gbc);
            gbc.gridx++;
        }
        gbc.gridx = 1;
        gbc.gridy++;
    }

    gbc.gridwidth = 12;
    gbc.gridy = 12;
    gbc.gridx = 0;
    editor.add(next, gbc);
    gbc.gridy = 13;
    editor.add(backLevEd, gbc);
}
```

Листинг 8. Создание панели редактора

```
private class Action implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
      …
      //проверка ответа пользователя
      if (e.getSource() == answer) {
          for (int i = 0; i < FIELD_SIZE; i++) {
              for (int j = 0; j < FIELD_SIZE; j++) {
                  if (grid[i][j].getText().equals("")) {
                      userAnswer[i][j] = 0;
                  } else {
                      userAnswer[i][j] =
      Integer.parseInt(grid[i][j].getText());
                  }
              }
          }
          us.setAnswer(userAnswer);
          us.checkAnswer();
      }
      …
}
```

Листинг 9. Часть класса Action. Проверка ответа пользователя при нажатии на кнопку Проверка

3. Тестирование приложения

3.1 Контрольный робот

Чтобы протестировать работу приложения воспользуемся unit-тестированием и фреймворком JUnit. Так как основные методы приложения находятся в классах Generator и Solver, то тестировать будем их.

Тесты класса Generator:

```java
@Test
void createBaseFieldTest() {
    int[][] baseField = new int[][] {
            {1,2,3,4,5,6,7,8,9},
            {4,5,6,7,8,9,1,2,3},
            {7,8,9,1,2,3,4,5,6},
            {2,3,4,5,6,7,8,9,1},
            {5,6,7,8,9,1,2,3,4},
            {8,9,1,2,3,4,5,6,7},
            {3,4,5,6,7,8,9,1,2},
            {6,7,8,9,1,2,3,4,5},
            {9,1,2,3,4,5,6,7,8}
    };
    assertArrayEquals(baseField, g.createBaseField());
}
```

Листинг 10. Тест создания базового поля

```java
@Test
void removeCellTest() {
    final int SIZE = 9;
    int[][]field;
    g.createBaseField();
    g.mixField( count: 20);
    g.removeCells();
    field = g.getField();
    int c = 0;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (field[i][j] == 0) {
                c++;
            }
        }
    }
    assertEquals(g.removeCells(), c);
}
```

Листинг 11. Тест удаления клеток в судоку

Тесты класса Solver:

Данные для тестов были взяты из Раздела 1.3 Контрольные примеры.

```java
@Test
void isFieldValidTestA() {
    assertTrue(s.isFieldValid(fieldA));
}

@Test
void isOneSolutionTestA() {
    assertTrue(s.isOneSolution(fieldA));
}

@Test
void isSolvedTestA() {
    assertTrue(s.solveSudoku(fieldA));
}

@Test
void isSolvedReversedTestA() {
    assertTrue(s.solveSudokuReverse(fieldA));
}
```

Листинг 12. Тесты для судоку А. Проверка на соответствие судоку правилам, единственность решение, решение прямым и обратным методом

```java
@Test
void solutionTestA() {
    s.isOneSolution(fieldA);
    int[][] answer = new int[][] {
            {7,1,4,8,2,5,9,3,6},
            {8,2,5,9,3,6,1,4,7},
            {9,3,6,1,4,7,2,5,8},
            {4,7,1,5,8,2,6,9,3},
            {6,9,3,7,1,4,8,2,5},
            {5,8,2,6,9,3,7,1,4},
            {3,6,9,4,7,1,5,8,2},
            {2,5,8,3,6,9,4,7,1},
            {1,4,7,2,5,8,3,6,9}
    };
    assertArrayEquals(answer, s.getSolution());
}
```

Листинг 13. Соответствует ли решение судоку ответу

3.2 Работа приложения на контрольных тестах

Все тесты, написанные для классов Solver и Generator проходят успешно, а их результаты сходятся с результатами, вычисленными вручную (Раздел 1.3 Контрольные примеры), поэтому можно сделать вывод, что все работает корректно.

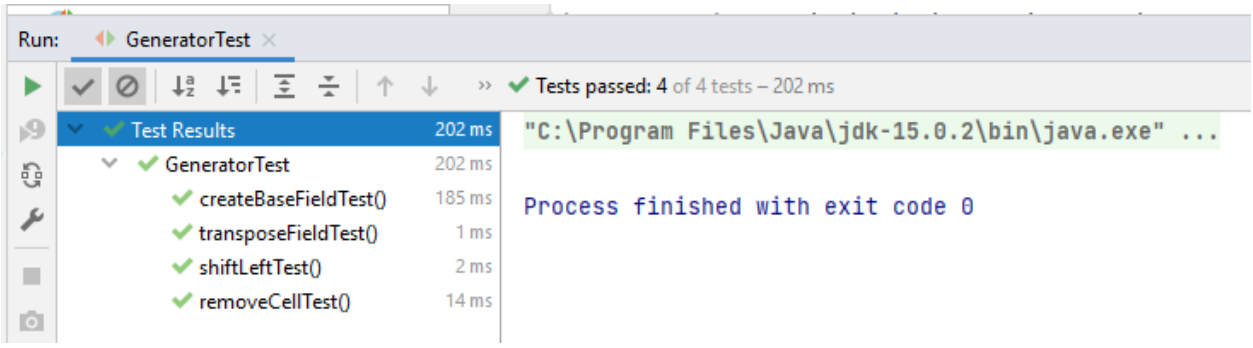

Рис.12 Результаты тестирования класса Generator

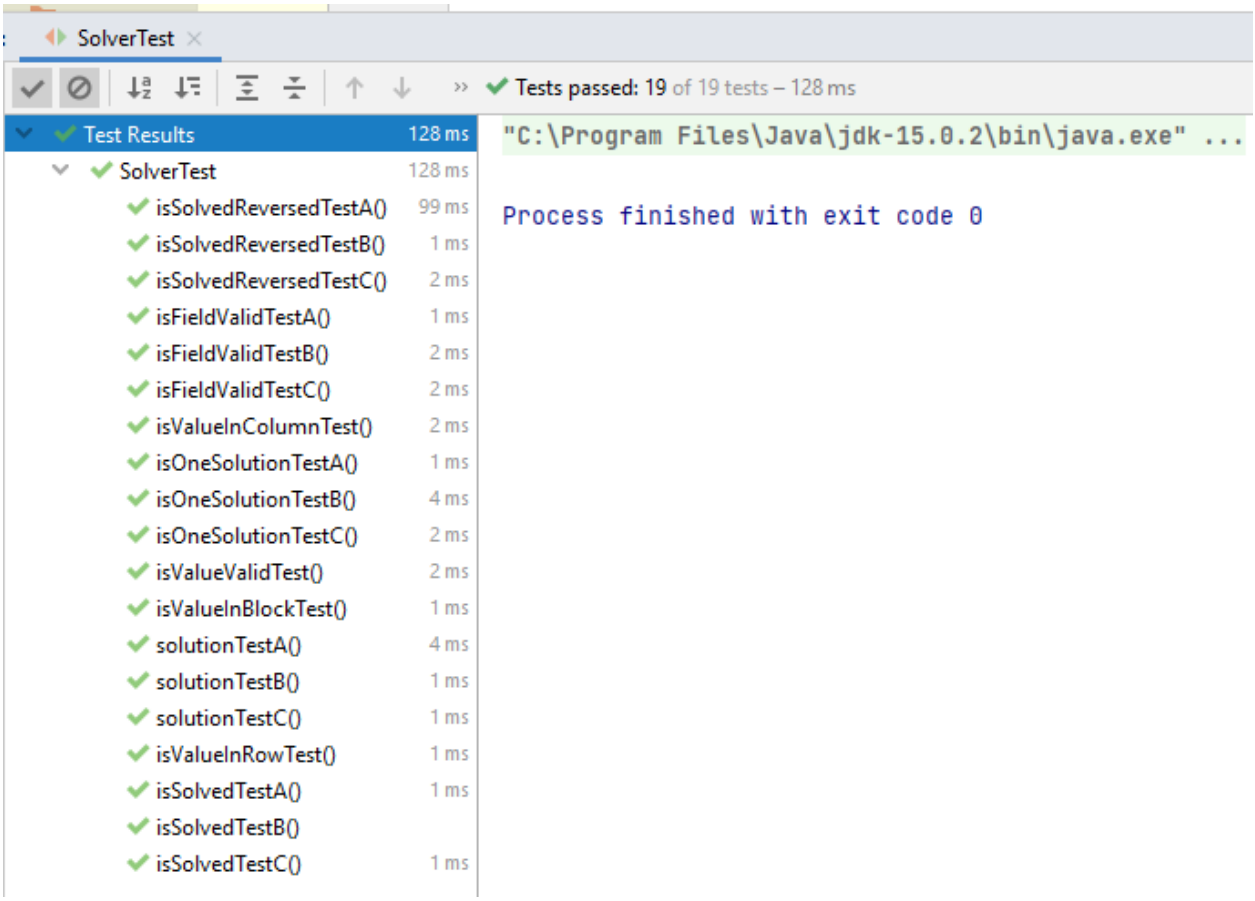

Рис. 13 Результаты тестирования класса Solver

3.3 Результаты работы приложения

В результате выполнения курсовой работы было разработано приложение, реализующее игру «Судоку». Производится генерация уровней, что гарантирует огромное количество уникальных судоку. Существует

возможность создания собственных уровней, при этом выполняется проверка введенного судоку на соответствие правилам. В приложении используется простой и понятный интерфейс, который будет удобен любому пользователю.