Java HW2: 数独生成器和求解器

姓名：黄文杰

学号：3210103379

1. 实验要求

编写数独程序(Sudoku Programming)

- **Sudoku生成器**

  —根据用户的提示数，生成Sudoku题目

  —1. 用户从命令行输入提示数（1~81）[在制作谜题时，提示数在22以下就非常困难，所以常见的数独题其提示数在 23~30之间]

  —2. 根据用户输入数字，自动生成数独题目，要求每行、每列、每格中空格能尽可能均匀分布

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

- **Sudoku求解器**

  —输入一个数独题目（可终端输入）

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

  —系统自动给出解答，并从终端输出

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

2. 实验思路

通过构造两个类`SudokuGenerator`和`SudokuSolver`来实现数独生成器和数独求解器的功能，这两个类放在`sudoku` package下。

- **SudokuGenerator**

  该类设计的主要方法有三个：`checkValid`、`printSudoku`和`generateSudoku`

  (1) `checkValid(int[][] data,int x,int y, int value)`：主要用于检查某个数能否放入当前的数独棋盘

```java
public static boolean checkValid(int[][] data,int x,int y, int value){
        // 检查当前的行和列是否已经包含相同的数
        for(int i=0;i<9;i++){
            if(data[x][i]==value){
                return false;
            }
            if(data[i][y]==value){
                return false;
            }
        }

        // 检查小的3*3九宫格是否已经包含相同的数
        int gridRow=3*(x/3);
        int gridCol=3*(y/3);
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                if((gridRow+i)==x && (gridCol+j)==y){
                    continue;
                }
                if(data[gridRow+i][gridCol+j]==value){
                    return false;
                }
            }
        }
        return true;
    }
```

接收参数：data:包含数独棋盘信息的二维数组；x、y:要填入的数对于二维数组的下标；value:要填入的数的值

代码解析：先判断当前的行和列是否已经包含相同的数，再判断3*3的九宫格是否已经包含相同的数，若是，则直接返回false，反之则返回true。

(2) `void printSudoku(int[][] data)`：主要用于打印数独棋盘

```
public static void printSudoku(int[][] data){
        for(int i=0;i<9;i++){
            for(int j=0;j<9;j++){
                if(data[i][j]==0){
                    System.out.print(". ");
                }else{
                    System.out.print(data[i][j]+" ");
                }
            }
            System.out.print('\n');
        }
    }
```

接收参数：data:包含数独棋盘信息的二维数组

代码解析：通过二重循环依次读出并打印data二维数组中的数，碰到空格（0即代表空格）则改为用字符'.'来代替输出，每隔九个打印一行。

(3) `void generateSudoku(int hints)`：主要用于生成数独的棋盘并返回对应的二维数组

```
private static void generateSudoku(int hints){
        int[][] data =new int[9][9];
        // 初始化
        for(int i=0;i<9;i++){
            for(int j=0;j<9;j++){
                data[i][j]=0;
            }
        }
        while(hints>0){
            // 生成随机二维数组的index和对应的值
            int x = (int) (Math.random() * 9);
            int y = (int) (Math.random() * 9);
            int value = (int) (Math.random() * 9) + 1;
            // 判断生成的数能否填入现有的棋盘
            if(data[x][y] == 0 && checkValid(data,x,y,value)){
                data[x][y]=value;
                hints--;
            }
        }
        printSudoku(data);
    }
```

接收参数：hints:用户输入的提示数

代码解析：构造一个int类型的二维数组(data)来存储数独棋盘的信息，然后构造循环来随机在9*9棋盘中的某个位置生成一个1~9的随机数，调用checkValid()方法来判断生成的数是否有效，如此往复直到生成了hints个有效的数字，然后调用printSudoku()方法来打印数独棋盘。

- **SudokuSolver**

  该类设计的主要方法有五个：`inputSudokuManually`、`inputSudokuAutomatically`、`getData`、`findEmptyCell`、`solveSudoku`

  (1) `int[][] inputSudokuManually(Scanner scanner)`：手动读入数独题目

```
private static int[][] inputSudokuManually(Scanner scanner) {
        char[][] sudokuPuzzle = new char[9][9];
        int[][] data = new int[9][9];
        // 读取上一次输入Int留下的换行符
        scanner.nextLine();//将换行符读掉
        System.out.println("请输入输入数独题目(数独题目中的空格请用.来代替),每个数字和'.'之间请不要用空格间隔,每隔9个换一次行:");
        for (int i = 0; i < 9; i++) {
            String row = scanner.next();
            sudokuPuzzle[i] = row.toCharArray();
        }
        return getData(sudokuPuzzle, data);
    };
```

  接收参数：scanner:一个Scanner对象，用于读取用户的输入

  代码解析：构造一个9*9的char类型二维数组来接受用户输入的题目信息，然后调用`getData()`方法将char类型的字符数组转换成便于后续操作的int类型的二维数组(data)

  (2) `int[][] inputSudokuAutomatically(Scanner scanner)`：通过copy数独生成器生成的结果来生成题目

```
private static int[][] inputSudokuAutomatically(Scanner scanner) {
        char[][] sudokuPuzzle = new char[9][9];
        int[][] data = new int[9][9];
        // 读取上一次输入Int留下的换行符
        scanner.nextLine();//将换行符读掉
        System.out.println("请粘贴从数独生成器中生成的数独题目:");
        for (int i = 0; i < 9; i++) {
            String row = scanner.nextLine();
            String[] elements = row.split(" ");
            for (int j = 0; j < 9; j++) {
                sudokuPuzzle[i][j] = elements[j].charAt(0);
            }
        }
        return getData(sudokuPuzzle, data);
    };
```

  接收参数：scanner:一个Scanner对象，用于读取用户的输入

代码解析：构造一个9*9的char类型二维数组来接受用户输入的题目信息，然后调用`getData()`方法将char类型的字符数组转换成便于后续操作的int类型的二维数组(data)，和手动输入唯一的区别就是输入的格式不同（手动不需要空格间隔，而自动则需要空格间隔）

(3) `int[][] getData(char[][] sudokuPuzzle, int[][] data)`：将字符型的数独棋盘生成int型的数独棋盘

```
private static int[][] getData(char[][] sudokuPuzzle, int[][] data) {
        for(int i=0;i<9;i++){
            for(int j=0;j<9;j++){
                if(sudokuPuzzle[i][j]>='1' && sudokuPuzzle[i][j]<='9'){
                    data[i][j]=sudokuPuzzle[i][j]-'0';
                }else if(sudokuPuzzle[i][j]=='.'){
                    data[i][j]=0;
                }else{
                    // 非法字符，异常退出
                    System.out.println("有非法字符，请确保输入的题目只包含数字和'.'字符！");
                    System.exit(1);
                }
            }
        }
        return data;
    };
```

接收参数：sudokuPuzzle:char类型的数独题目二维数组；data:int类型的数独题目二维数组（保存返回的结果）

代码解析：通过二重循环来遍历char类型的二维数组，并根据其中的字符内容来相应地转换成0~9的整数或异常退出。（0代表数独棋盘中的空格）

(4) `int[] findEmptyCell(int[][] data)`：找到数独棋盘中的第一个空格（以数组的形式来返回x,y下标信息）

```
private static int[] findEmptyCell(int[][] data) {
        for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                if (data[i][j] == 0) {
                    return new int[]{i, j};
                }
            }
        }
        return null;
    }
```

接收参数：data:包含数独棋盘信息的二维数组

代码解析：通过二重循环遍历data数组，找到第一个等于0的项，就返回其下标信息（构造一个Int[]数组来返回）；若找不到，则返回null

(5) `boolean solveSudoku(int[][] data)`：递归解题

```java
private static boolean solveSudoku(int[][] data) {
        int[] emptyCell = findEmptyCell(data);

        if (emptyCell == null) {
            return true; // 无空格，说明已经是一个完成的数独
        }

        int row = emptyCell[0];
        int col = emptyCell[1];

        for (int value = 1; value <= 9; value++) {
            if (checkValid(data, row, col, value)) {
                data[row][col] = value;

                if (solveSudoku(data)) {
                    return true; // 判断是都已经完成
                }

                data[row][col] = 0; // 回溯
            }
        }

        return false;
    }
```

接收参数：data:包含数独棋盘信息的二维数组

代码解析：调用`findEmptyCell()`方法来获取当前棋盘第一个空格的下标索引信息，若无空格，则返回true（这是递归出口）；反之，则采用回溯递归的方法依次尝试填空，直至找到解法或者证明无解，并返回结果。

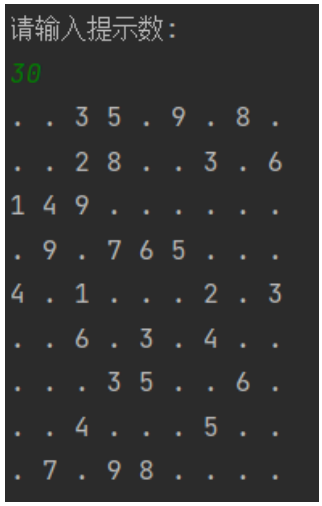
(6) `main`方法：主要提供选择菜单来接受手动输入的题目信息或者copy过来的题目信息

```java
public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        System.out.println("请输入您的选择:");
        System.out.println("1. 手动输入一个题目");
        System.out.println("2. copy数独生成器生成的题目");

        int option =s.nextInt();
        int[][] data = new int[9][9];

        if(option==1){
            data=inputSudokuManually(s);
        } else if (option==2) {
            data=inputSudokuAutomatically(s);
        }else{
            System.out.println("非法字符！");
            System.exit(1);
        }

        if (solveSudoku(data)) {
            System.out.println("\n数独解答如下:");
```

```
            printSudoku(data);
        } else {
            System.out.println("该数独无解！");
        }

    }
```

3. 实验结果

- **数独生成器结果展示**



- **数独求解器结果展示**

（1）手动输入题目

```
请输入您的选择:
1. 手动输入一个题目
2. copy数独生成器生成的题目
1
请输入输入数独题目(数独题目中的空格请用.来代替),每个数字和'.'之间请不要用空格间隔,每隔9个换一次行:
..35.9.8.
..28..3.6
149......
.9.765...
4.1...2.3
..6.3.4..
...35..6.
..4...5..
.7.98....
该数独无解!
```

（2）copy数独生成器生成的题目

```
请输入您的选择：
1. 手动输入一个题目
2. copy数独生成器生成的题目
2
请粘贴从数独生成器中生成的数独题目：
. . 4 . . . . 6 .
. . . 4 6 . . . .
. . . 1 . 5 9 7 .
. . . . . 3 6 . 2
5 9 . 8 . 4 . . .
. 3 . . . . . . .
. . . 7 4 9 . . .
. . . . . . . 8 3
. 1 . . . 8 7 . 9

数独解答如下：
1 5 4 3 9 7 2 6 8
8 7 9 4 6 2 3 5 1
6 2 3 1 8 5 9 7 4
7 4 8 5 1 3 6 9 2
5 9 6 8 2 4 1 3 7
2 3 1 9 7 6 8 4 5
3 8 2 7 4 9 5 1 6
9 6 7 2 5 1 4 8 3
4 1 5 6 3 8 7 2 9
```

4. 实验心得

通过本次实验，我了解了如何用Java写递归算法，也掌握了如何调用其他类的方法的技巧。总的来说，这次实验的难度能让人接受，通过这样的实验也能帮助我们更好地学习Java，了解和熟悉Java的语法。