PROJECT REPORT

ON

**SUDOKU SOLVER VISUALIZER**

SCHOOL OF COMPUTER SCIENCE & ENGINEERING



**SUBMITTED BY:**

Name- Dharani Sree.M.C

Registration Number - 12208381

Roll. No – 12

Section-9SK01

**1. Introduction**

1.1 Background

Sudoku is a logic-based number placement puzzle that has become a favourite pastime for many puzzle enthusiasts worldwide. The objective is to fill a 9x9 grid with digits from 1 to 9 so that each row, each column, and each of the nine 3x3 sub grids contains all the digits from 1 to 9 exactly once.

1.2 Project Scope

The Sudoku Solver project aims to develop a software application with the following capabilities:

- An interactive graphical interface for solving Sudoku puzzles.

- A robust algorithm to solve Sudoku puzzles programmatically.

- Functionality for generating random Sudoku grids for users to solve.

**2. Features**

2.1 Graphical User Interface (GUI)

*2.1.1Grid Display*

The main component of the GUI is a 9x9 grid of text fields that users can interact with to input or view Sudoku puzzles. The grid is designed for easy readability and usability.

*2.1.2 Control Buttons*

The interface includes four main buttons for user interaction:

- Solve: Initiates the puzzle-solving process.

- Stop: Halts the solving process.

- Reset: Clears the grid for a new puzzle.

- Random Fill: Generates a random Sudoku puzzle.

2.2 Functionality

*2.2.1 Solver Algorithm*

The application uses a recursive backtracking algorithm to solve Sudoku puzzles. This method systematically fills the grid and backtracks when an invalid number placement is encountered.

*2.2.2 Random Grid Generator*

The application can generate random Sudoku puzzles by filling the grid completely and then removing a specified number of cells to create a puzzle with a unique solution.

2.3 Visual Feedback

The interface provides visual feedback using color coding:

- User Input: White text for numbers entered by the user.

- Random Fill: Green text for numbers generated by the random fill function.

- Solver Process: Yellow text for numbers placed by the solver algorithm.

**3. Implementation Details**

3.1 Technologies Used

- Java: The primary programming language used for developing the application.

- Swing: The GUI toolkit used for creating the graphical interface.

- Threading: Used to run the solving algorithm in a separate thread, keeping the interface responsive.

- Random: Used for generating random numbers and creating Sudoku puzzles.

3.2 Algorithms

*3.2.1 Backtracking Algorithm*

The backtracking algorithm is a recursive approach that fills each cell with numbers from 1 to 9, checking for validity at each step. If an invalid placement is detected, it backtracks and tries a different number.

*3.2.2 Cell Removal Algorithm*

After generating a complete Sudoku solution, the algorithm randomly removes a specified number of cells to create a puzzle with a unique solution.

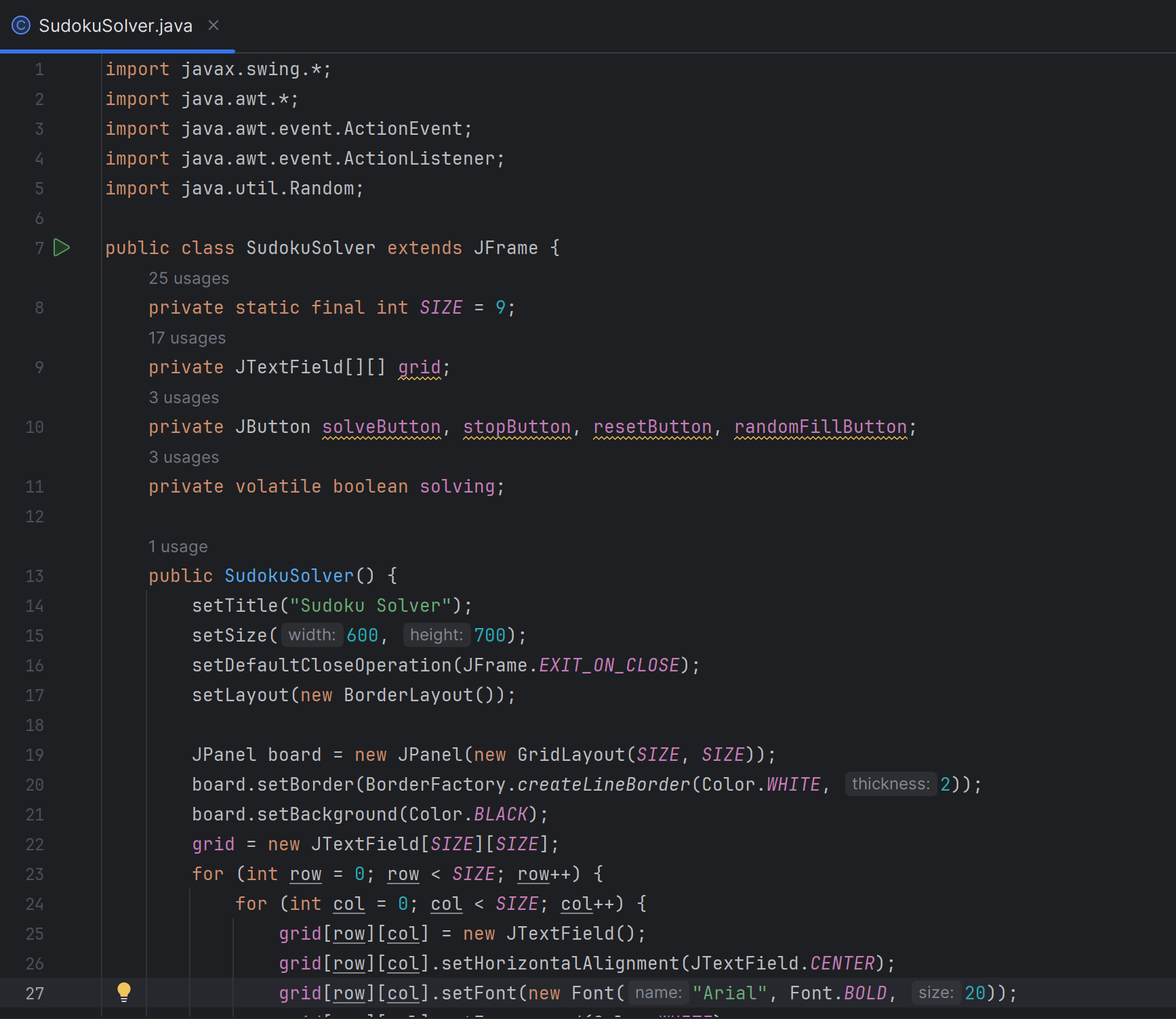
3.3 GUI Design

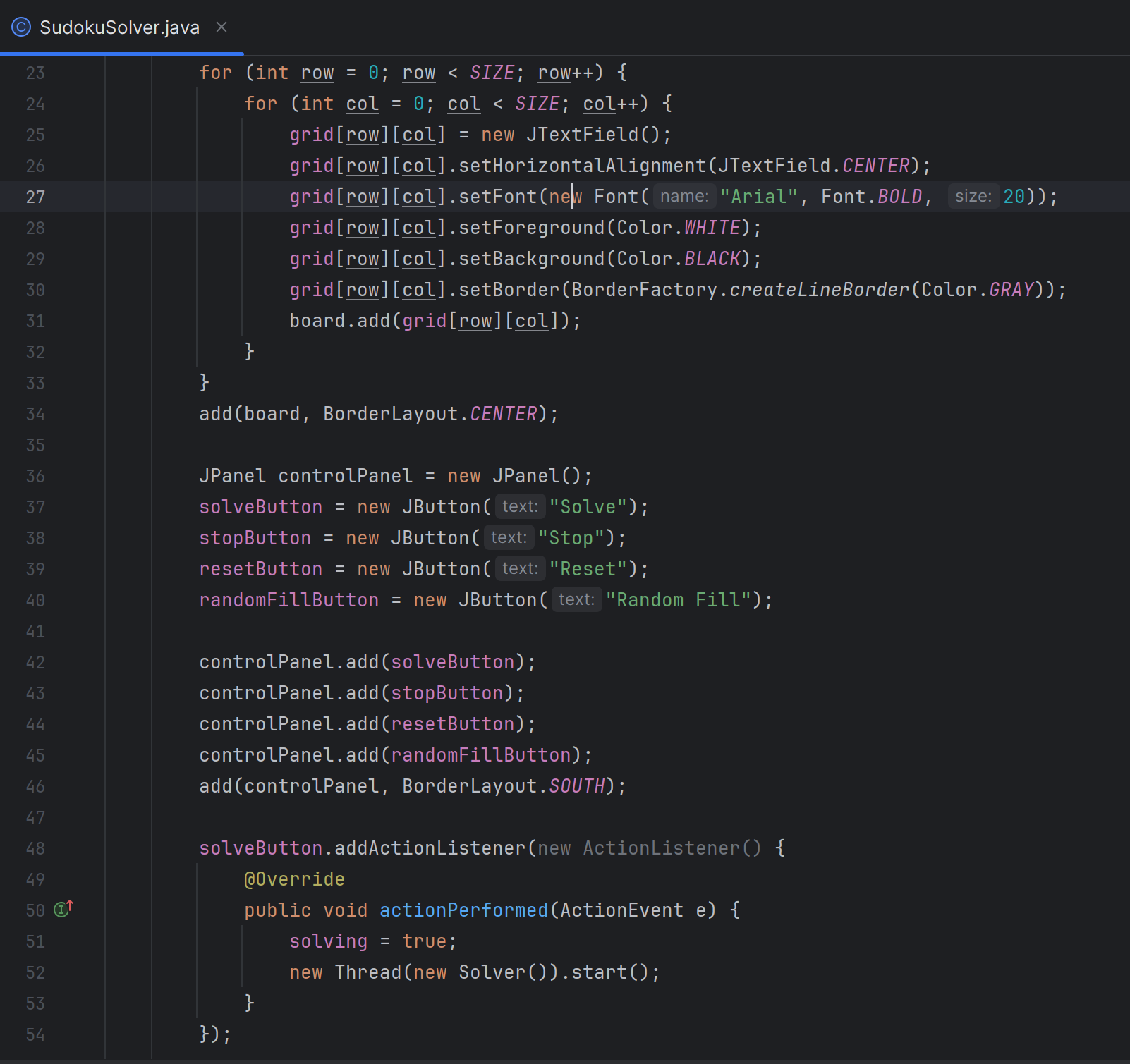
The GUI is designed using a ‘BorderLayout’ for the main components, with the Sudoku grid in the center and the control panel at the bottom. Custom fonts, colors, and borders are used to enhance the visual appeal and usability of the interface.

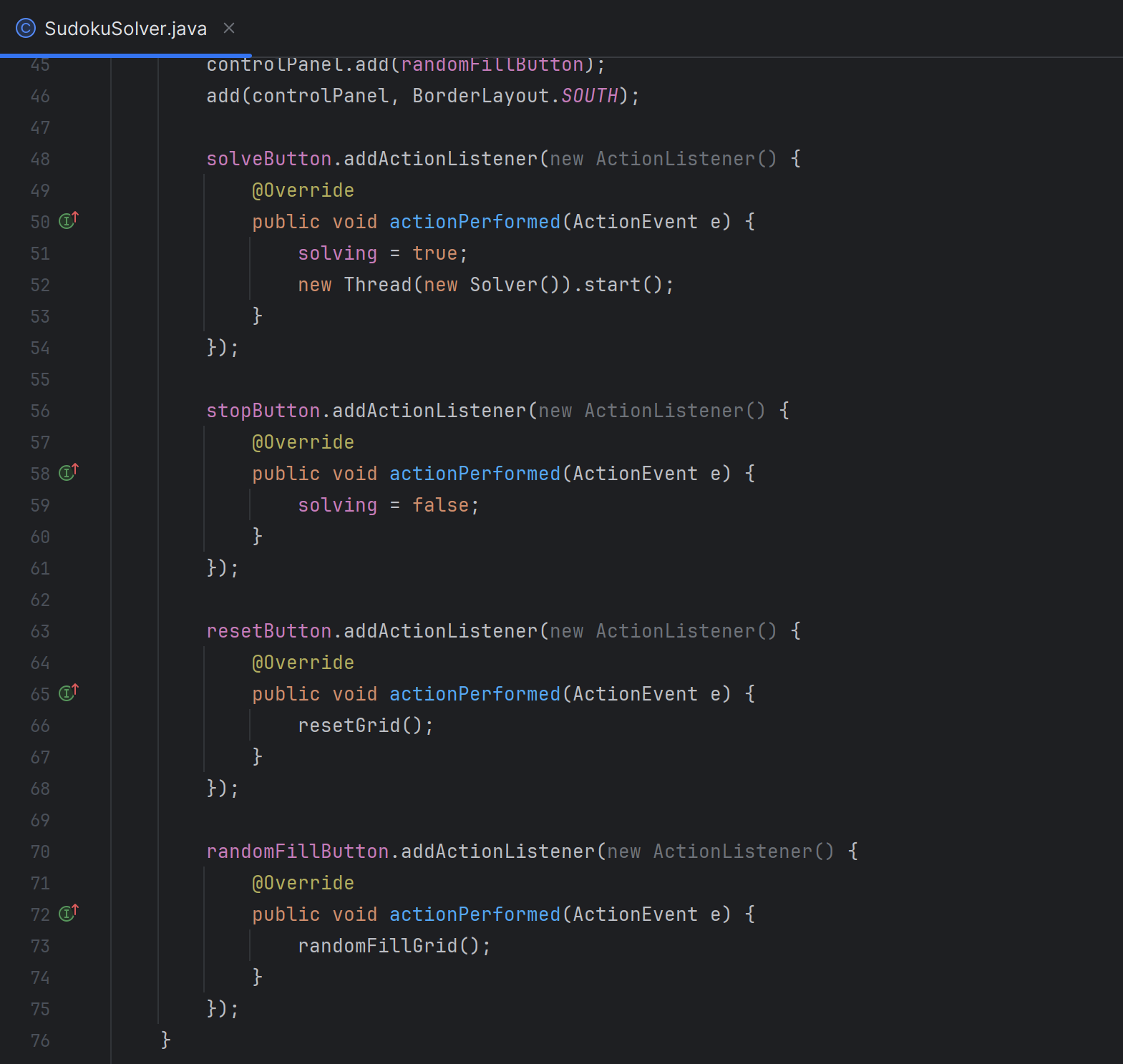
**4. Code Explanation**

4.1 Main Class and GUI Setup

The main class extends ‘JFrame’ and sets up the GUI components, including the grid and control buttons.

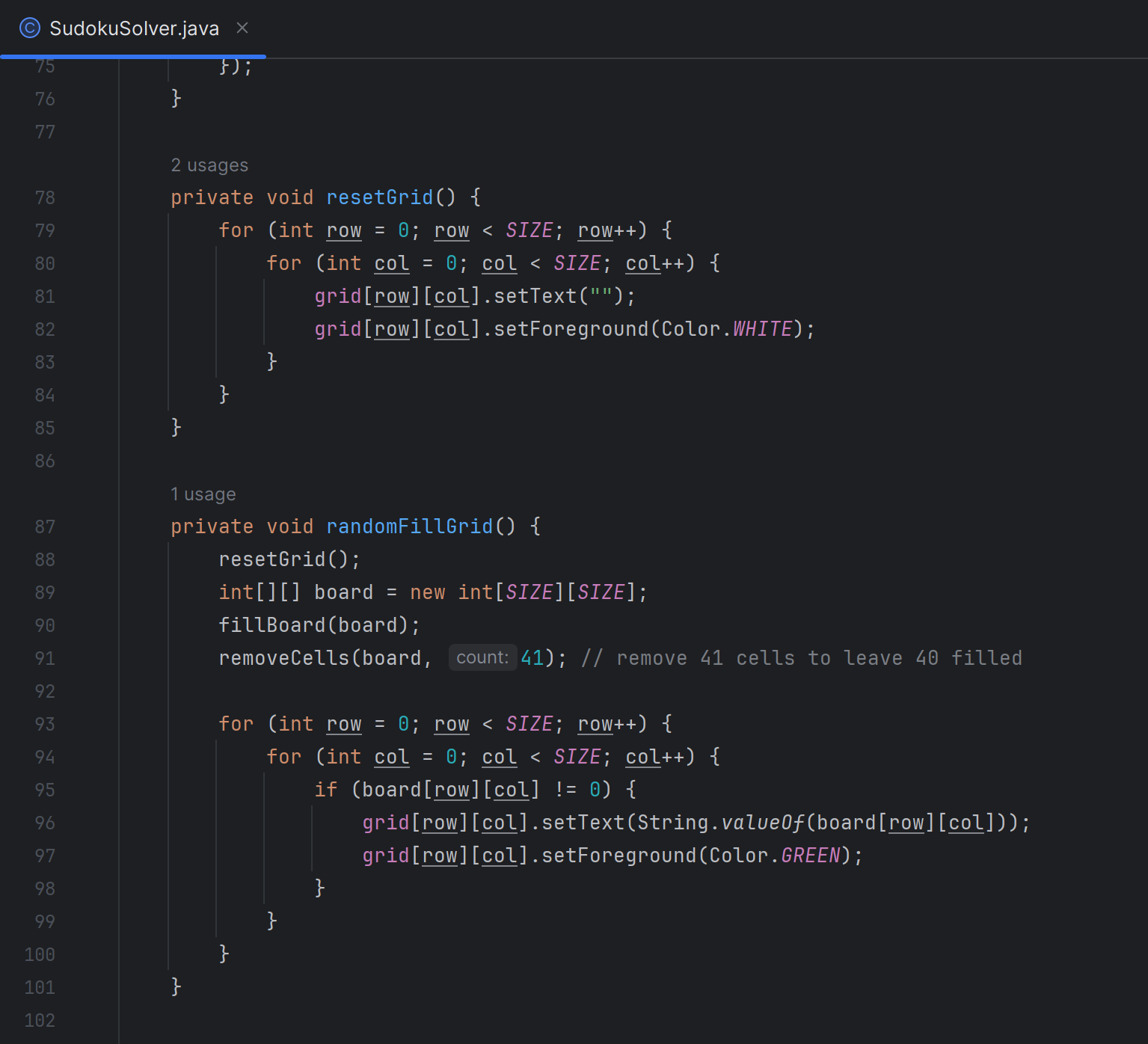






4.2 Grid Reset and Random Fill Methods

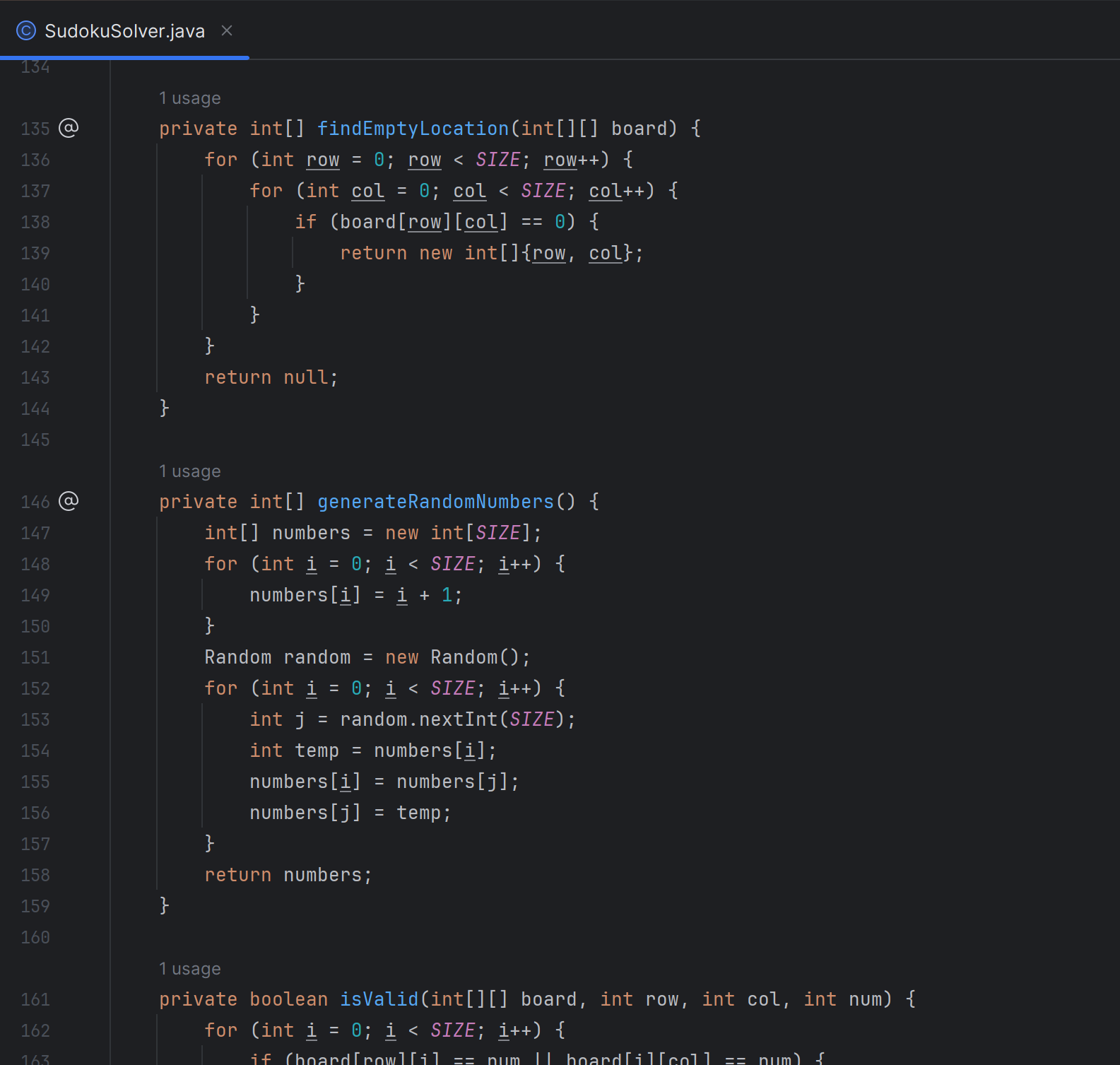
The methods for resetting the grid and filling it with a random puzzle are defined.

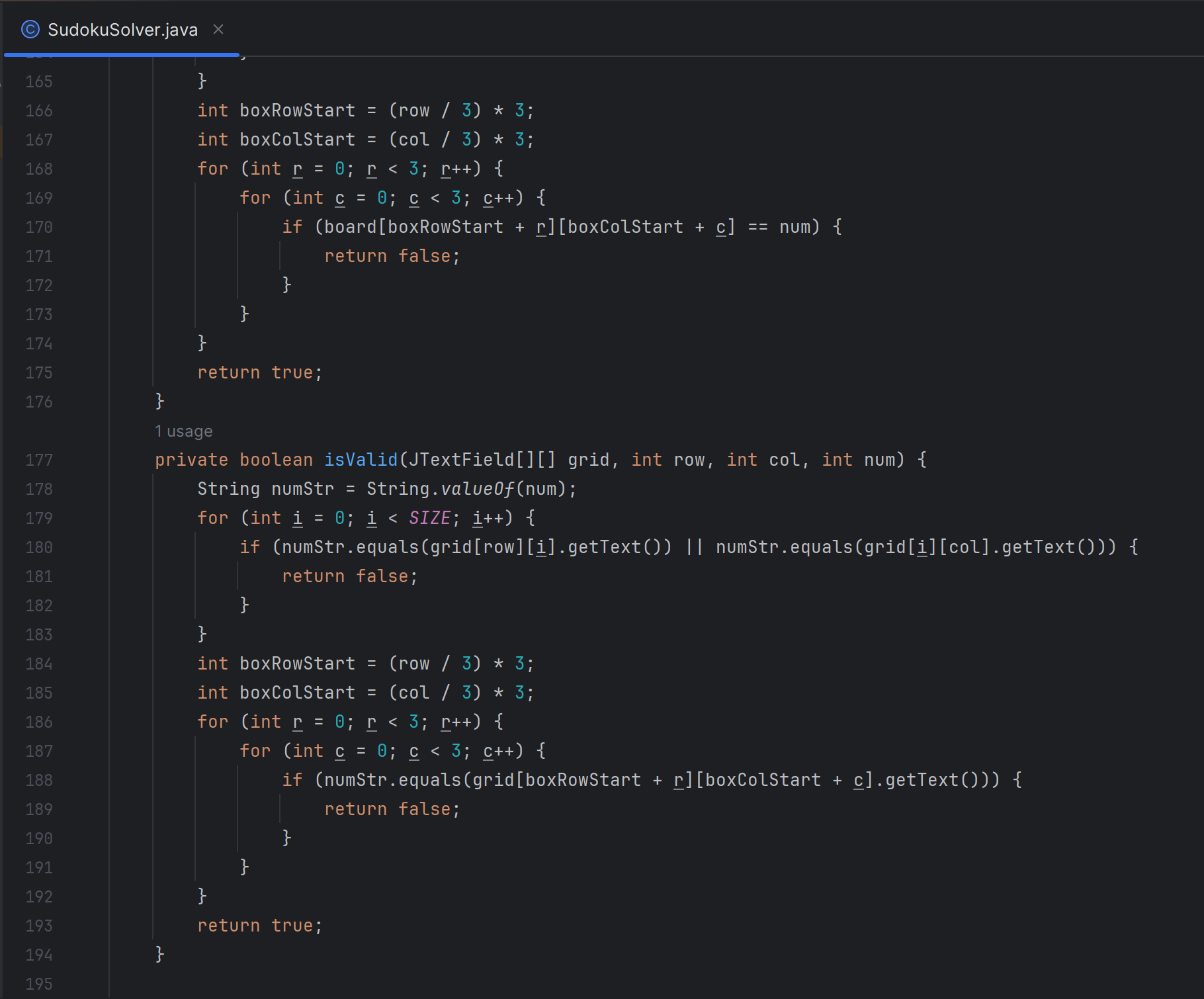


4.3 Sudoku Solver Algorithm

The solving algorithm uses backtracking to fill the grid.

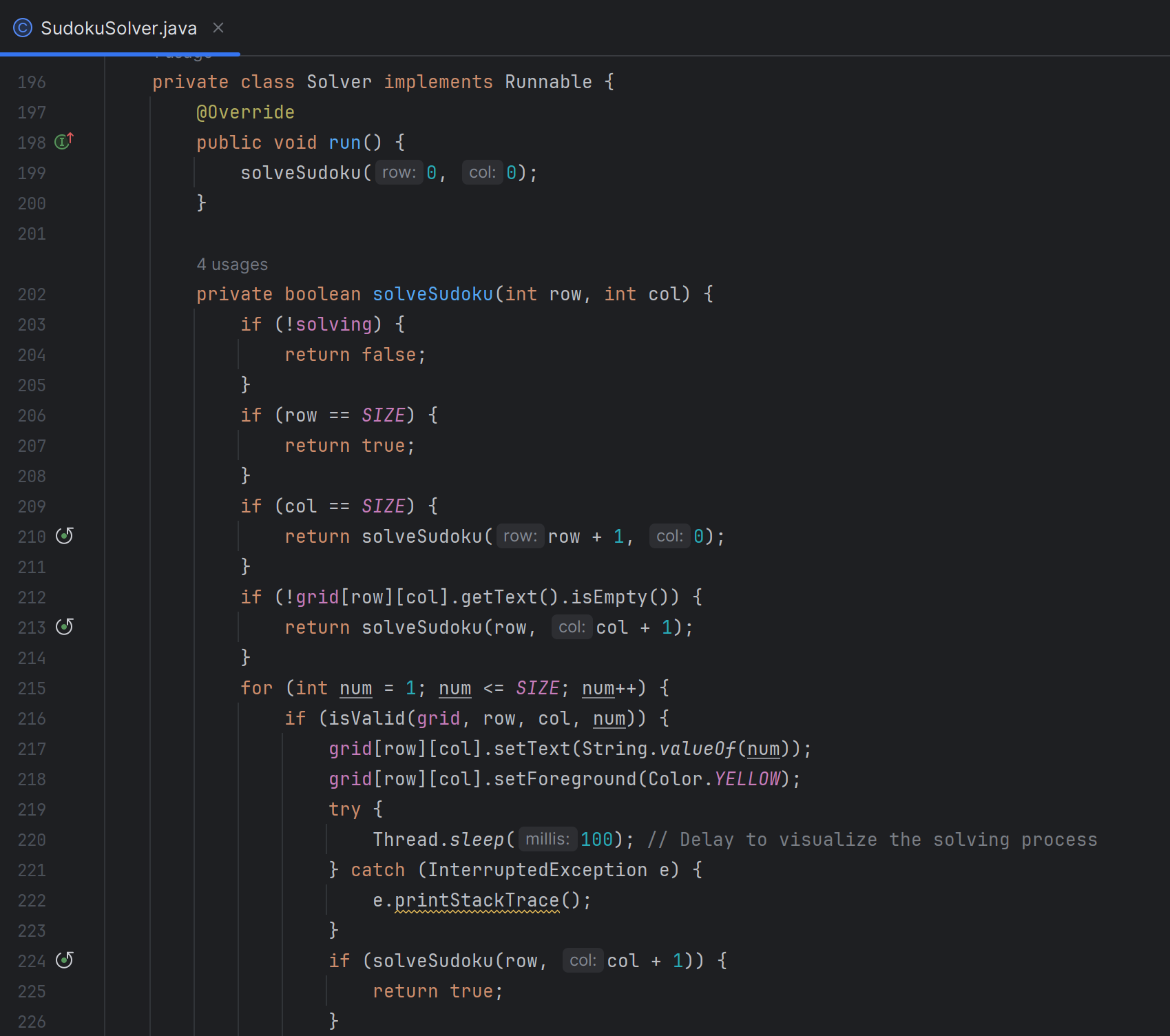


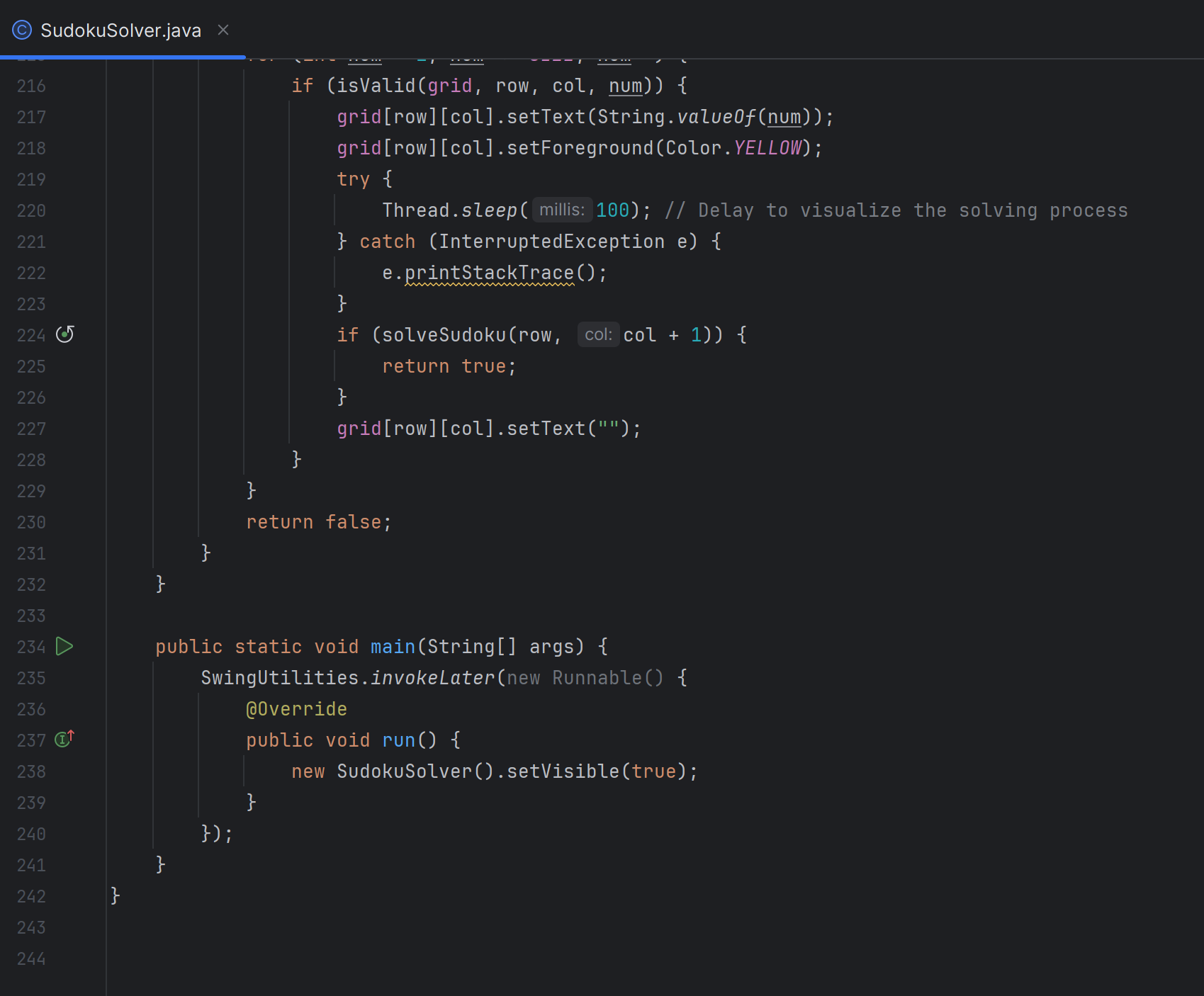




4.4 Solver Runnable Class

The ‘Solver’ class implements ‘Runnable’ to solve the Sudoku puzzle on a separate thread.





**5. Usage**

5.1 Solving Puzzles

*5.1.1 Manual Input*

Users can manually enter numbers into the grid using the text fields. The application checks for validity and highlights invalid entries.

*5.1.2 Automatic Solving*

Clicking the "Solve" button initiates the solving process. The algorithm proceeds step-by-step, filling in cells and providing visual feedback through color changes.

5.2 Interaction

*5.2.1 Stopping the Solver*

Users can click the "Stop" button to interrupt the solving process at any time.

*5.2.2 Resetting the Grid*

The "Reset" button clears all entries in the grid, allowing users to start afresh.

*5.2.3 Generating Random Puzzles*

The "Random Fill" button generates a new Sudoku puzzle with a predetermined number of pre-filled cells, offering a ready-to-solve challenge.

**6. Conclusion**

The Sudoku Solver project successfully integrates a user-friendly graphical interface with a robust solving algorithm, providing an interactive and educational experience for users. The application not only assists in solving Sudoku puzzles but also offers functionalities for puzzle generation, enhancing its utility for both casual users and puzzle enthusiasts. Future improvements could include advanced solving techniques, enhanced user feedback mechanisms, and support for different grid sizes and puzzle difficulties.

**SOME OUPUT SAMPLES**

