N-Queens Problem Solver

This report describes a Java program designed to solve the N-Queens problem and dynamically visualize the solutions using the Swing framework. The N-Queens problem is a classic chess puzzle with the objective of placing N chess queens on an NxN chessboard such that no two queens threaten each other.

A by Ashish Patel

Overview of the Java Program

1 **Problem Definition**

The N-Queens problem is a classic constraint satisfaction problem that has been studied extensively in computer science.

2 **Program Architecture**

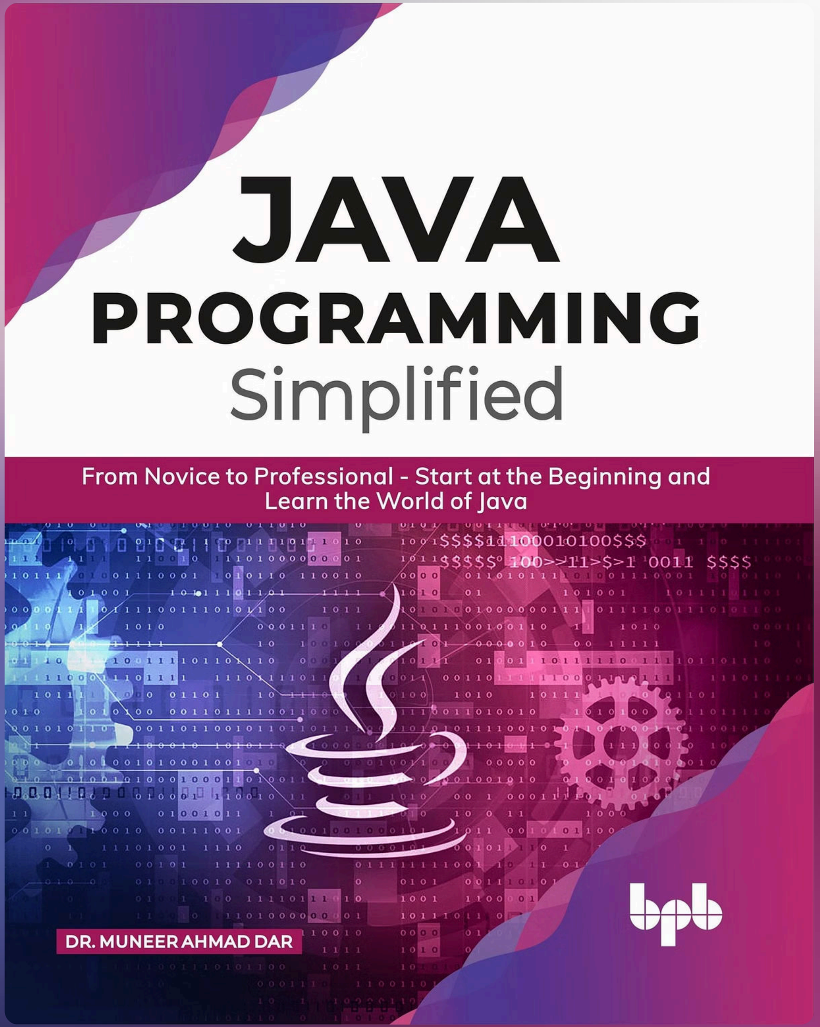
The program utilizes a backtracking algorithm to systematically explore possible queen placements on the chessboard.

3 **Swing Framework Integration**

The Swing framework is employed to provide a user-friendly graphical interface for inputting the value of N and visualizing the solutions dynamically.

4 **Dynamic Visualization**

The solutions to the N-Queens problem are displayed dynamically on the chessboard, highlighting the positions of the queens.


JAVA
PROGRAMMING
Simplified
From Novice to Professional - Start at the Beginning and Learn the World of Java
DR. MUNEER AHMAD DAR
bpb

Solving the N-Queens Problem

1

Initialization

The program initializes an NxN chessboard and sets up the necessary data structures to represent the queen placements.

2

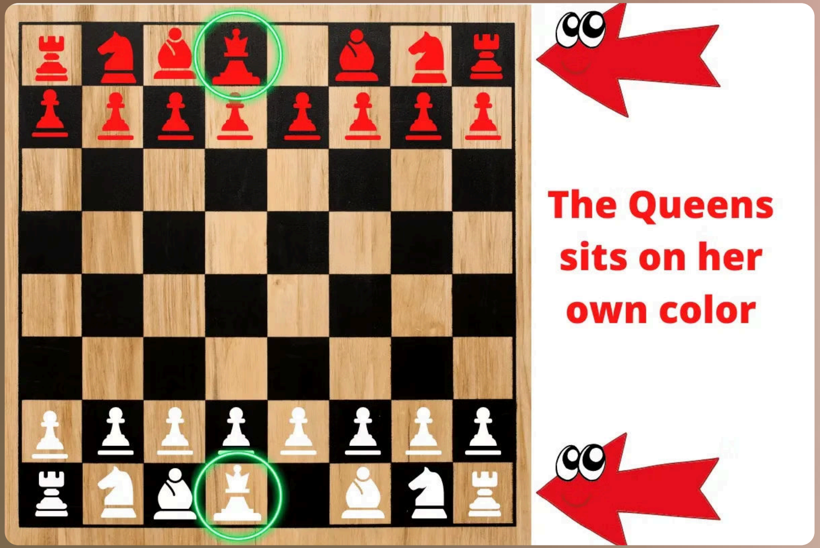
Backtracking Algorithm

The program utilizes a backtracking algorithm to systematically explore possible queen placements on the chessboard, ensuring that no two queens threaten each other.

3

Solution Validation

Once a potential solution is found, the program verifies that it meets the problem's constraints and adds it to the list of solutions.



Implementing the Backtracking Algorithm

Recursive Function

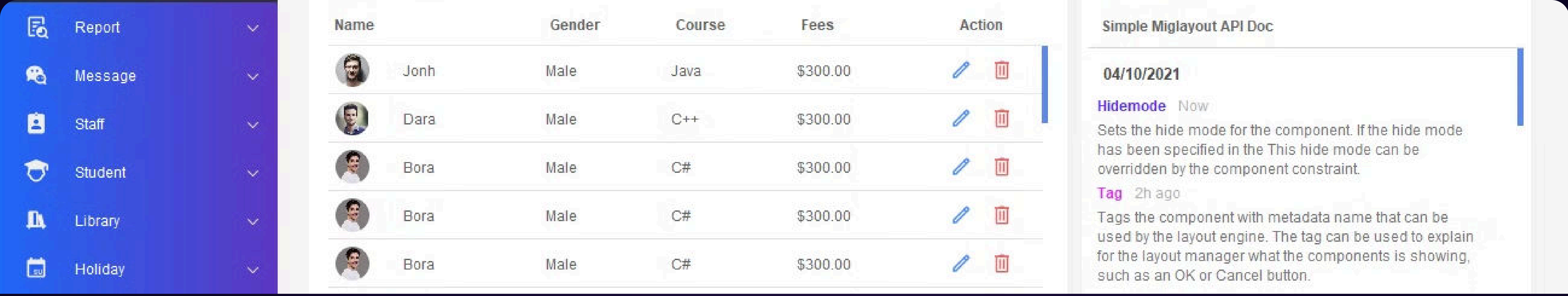
The backtracking algorithm is implemented as a recursive function that explores possible queen placements in each column of the chessboard.

Constraints

The recursive function checks for conflicts between the current queen placement and previously placed queens on the same row, diagonal, and anti-diagonal.

Solution Construction

If a valid placement is found for a queen in a column, the function recursively explores the next column. If all columns are explored, a solution is found.



Name	Gender	Course	Fees	Action
Jonh	Male	Java	$300.00	
Dara	Male	C++	$300.00	
Bora	Male	C#	$300.00	
Bora	Male	C#	$300.00	
Bora	Male	C#	$300.00	

Simple Miglayout API Doc

04/10/2021

Hidemode Now

Sets the hide mode for the component. If the hide mode has been specified in the This hide mode can be overridden by the component constraint.

Tag 2h ago

Tags the component with metadata name that can be used by the layout engine. The tag can be used to explain for the layout manager what the components is showing, such as an OK or Cancel button.

Integrating the Swing Framework

1

Input Panel

The program creates an input panel that allows the user to enter the value of N, representing the size of the chessboard.

2

Chessboard Visualization

The program displays the chessboard dynamically, using a 2D array of JLabels to represent each square on the chessboard.

3

Solution Display

When a solution is found, the program updates the chessboard visualization, highlighting the positions of the queens.

4

Dynamic Update

The program continuously updates the chessboard visualization as the backtracking algorithm explores possible queen placements.

Designing the Graphical User Interface

Input Field

The GUI includes an input field for the user to enter the value of N, which represents the size of the chessboard.

Solve Button

The GUI includes a "Solve" button that triggers the execution of the backtracking algorithm to find the solutions.

Chessboard Panel

The GUI includes a panel that displays the chessboard dynamically, updating as the backtracking algorithm explores possible queen placements.

Solution Count

The GUI displays the number of solutions found for the given value of N.

Visualizing the Solutions Dynamically

Placement	Visualization
Queen placement	Highlights the selected square with a different color.
Conflict Detection	Displays red squares for cells where a queen would be under attack.
Solution Found	Highlights the positions of the queens in the solution with a distinct color.

Conclusion and Future Enhancements

Efficiency

The program provides a visually engaging and efficient solution to the N-Queens problem.

Future Enhancements

Future enhancements could include adding support for different chessboard sizes, visual feedback for the backtracking process, and the ability to save and load solutions.

Applications

The program can be used as a teaching tool for understanding backtracking algorithms and constraint satisfaction problems.

Conclusion

The program successfully solves the N-Queens problem and provides a dynamic visualization of the solutions, making it a valuable tool for learning and exploration.