**OS Project Documentation**

**Project Title:** N-Queens Solver

**Project Description:**

The N-Queens Solver project aims to implement a solution to the classical N-Queens problem using a graphical user interface (GUI) and multi-threading for enhanced performance. The N-Queens problem involves placing N chess queens on an N\*N chessboard in such a way that no two queens threaten each other.

**What We Have Actually Done:**

**NQUEENSSOLVERGUI Class:**

The NQueensSolverGUI class is part of a Java program for solving the N-Queens problem with a graphical user interface (GUI). The GUI is built using Java Swing and consists of a main window containing a chessboard and buttons for starting and stopping the solving process.

**Chessboard Class:**

The chessboard class is a JPanel in Java Swing that represents the chessboard for the N-Queens Solver GUI. It organizes and groups the graphical components together, providing a visual representation of the chessboard and the queens placed on it.

**NQUEENSSOLVER Class:**

The NQueensSolver class is designed to solve the N-Queens problem using a backtracking algorithm, and it implements the Runnable interface for concurrent execution.

**Team Members and Roles:**

**Mariam Medhat, Ahmed Saber:**

Role: GUI Design and Implementation

Contributions: Implemented the NQUEENSSOLVERGUI class to create an intuitive user interface for interacting with the N-Queens solver.

**Mariam Yasser, Noura Ali:**

Role: Chessboard Representation

Contributions: Developed the Chessboard class, which manages the chessboard state, validates queen placements, and updates the GUI.

**Mariam Mahmoud, Mariam Ezzat, Amer Medhat:**

Role: Multi-threaded Solver

Contributions: Implemented the NQUEENSSOLVER class, utilizing multi-threading to explore various solutions concurrently and efficiently solve the N-Queens problem.

**All the members:**

Role: Testing and Documentation

Contributions: Conducted thorough testing of the GUI, chessboard representation, and solver components. Documented the project, including code comments and user instructions.

**Code Documentation:**

**NQUEENSSOLVERGUI Class:**

- It utilizes Java Swing components to create a main window with a chessboard and control buttons.

- The GUI allows users to input the size of the chessboard, start and stop the solving process, and visualizes the solution.

- **Components:**

- JFrame: Represents the main window of the application.

- JButton (startButton, stopButton): Buttons to start and stop the solving process.

- chessboard: Custom JPanel representing the chessboard for displaying

queens' placements.

- NQueensSolver: Solver instance responsible for solving the N-Queens

problem using threads.

- Thread (solverThread): Thread for executing the solver concurrently with

the GUI.

**- Methods:**

- startSolver(): Initiates the solving process by creating a new solver instance

and starting the solver thread.

- stopSolver(): Interrupts the solver thread, allowing users to stop the

solving process.

**- Main Method:**

- The main method prompts the user to enter the size of the chessboard

through a JOptionPane input dialog.

- It creates an instance of the NQueensSolverGUI class with the specified

board size using SwingUtilities.invokeLater().

- Handles input validation and provides feedback in case of invalid input

or cancellation.

**Chessboard Class:**

- The `Chessboard` class represents a graphical chessboard for visualizing

the N-Queens problem. It extends the JPanel class to organize and group the

components together, it creates an instance of Chessboard with the desired board size.

**-Components:**

- private final int boardSize: Represents the size of the chessboard.

- private final boolean[][] queens: 2D array representing queen positions on the chessboard.

**-Constructor:**

- Chessboard(int boardSize): Initializes a new chessboard with the specified size.

**-Methods:**

- int getBoardSize(): Retrieves the size of the chessboard.

- void placeQueen(int row, int col): Places a queen on the chessboard at the specified position.

- boolean isQueenPlaced(int row, int col): Checks if a queen is placed at the specified position.

- void removeQueen(int row, int col): Removes a queen from the specified position on the chessboard.

- void removeAllQueens(): Removes all queens from the chessboard.

**- Overrides:**

- void paintComponent(Graphics g): Overrides the paintComponent method of JPanel to customize the painting of the chessboard. It draws cells and queens based on their positions.

**NQueensSolver Class:**

- The `NQueensSolver` class implements the Runnable interface and represents

a solver for the N-Queens problem. It attempts to find a solution by placing

queens on a chessboard without any two queens attacking each other.

**- Components:**

- private final Chessboard chessboard: The chessboard instance for visualizing

queens' placements.

- private final int boardSize: The size of the chessboard.

- private boolean stopFlag: Flag to indicate whether the solver should stop

its execution.

**- Constructor:**

- NQueensSolver(Chessboard chessboard, int boardSize): Initializes a new solver instance with the specified chessboard and board size.

**- Methods:**

- void stopSolver(): Stops the solver by setting the stopFlag to true.

- boolean isSafe(int row, int col): Checks if it is safe to place a queen at the given

row and column on the chessboard. It looks for queens in the same column or diagonals.

- boolean solveNQueens(int row): Recursively attempts to solve the N-Queens

problem by placing queens on the chessboard starting from the specified row.

It explores valid placements and backtracks when needed.

- void run(): The run method required by the Runnable interface. It starts the solving

process by calling solveNQueens with the initial row as 0.

**Conclusion:**

The N-Queens Solver project demonstrates an effective solution to the N-Queens problem with a user-friendly GUI and improved performance through multi-threading.