This code defines a class `NQueens` to solve the N-Queens problem, which is to place N queens on an N×N chessboard such that no two queens attack each other. Here's an explanation of the class and its methods:

1. `__init__(self, n)`: The constructor method initializes the N-Queens solver with the board size `n`, initializes the board state as a list of `-1` (indicating no queen placed yet), and sets the `solution_count` to 0.

2. `is_safe(self, row, col)`: This method checks if it's safe to place a queen at position `(row, col)` on the board. It checks if there are no queens in the same column, diagonal, or row as the current position.

3. `solve(self, row)`: This is the main recursive backtracking function to find all solutions. It tries to place a queen in each column of the current row, then recursively tries to place queens in the next rows. If a solution is found (i.e., all rows are filled), it prints the solution and increments the `solution_count`.

4. `print_solution(self)`: This method prints a visual representation of the board with queens as 'Q' and empty squares as '.'.

5. `find_solutions(self)`: This method initiates the solving process by calling `solve(0)` to start from the first row.

6. `Example usage`: It creates an instance of `NQueens` with `n=8` to solve the 8-Queens problem and find all solutions.

When you run this code, it will print all possible solutions for placing 8 queens on an 8x8 chessboard such that no two queens attack each other. Each solution is numbered, and the positions of the queens are shown as 'Q' in the output.