The code defines a class **Solution** that contains a method **totalNQueens** which takes an integer **n** as an input and returns the total number of unique solutions to the problem of placing **n** queens on an **nxn** chessboard such that no two queens threaten each other.

The class has a private field **count** which is used to store the number of solutions. The **totalNQueens** method initializes the count to 0 and then calls the **backtrack** method, passing in the initial values of 0 for the row, left diagonal, column, and right diagonal, and **n** for the chessboard size.

The **backtrack** method is a recursive function that uses the bit manipulation technique to solve the problem. It takes 5 arguments:

* **row** represents the current row in the chessboard and it is represented using a bitmask, where the ith bit is set to 1 if the ith column of the current row is occupied by a queen.
* **ld** represents the occupied positions in the left diagonal and it is represented using a bitmask, where the ith bit is set to 1 if the ith left diagonal is occupied by a queen.
* **col** represents the occupied positions in the current column and it is represented using a bitmask, where the ith bit is set to 1 if the ith column is occupied by a queen.
* **rd** represents the occupied positions in the right diagonal and it is represented using a bitmask, where the ith bit is set to 1 if the ith right diagonal is occupied by a queen.
* **n** is the size of the chessboard.

The method first checks if the current row is equal to the total number of rows in the chessboard (i.e. (1 << n) - 1). If this is the case, it increments the count by 1 and returns, as this means a valid solution has been found.

If not, the method finds the possible positions where a queen can be placed in the current row using bitwise operations:

**pos = (~(row | ld | col | rd)) & ((1 << n) - 1)**

It then uses a while loop to iterate through all the possible positions represented by the 'pos' bitmask. In each iteration, it selects a possible position represented by the least significant bit (p = pos & -pos) and removes it from the 'pos' bitmask (pos -= p). It then calls the backtrack method again with the updated row, left diagonal, column and right diagonal bitmask for the next recursive call.

In short, the code uses backtracking and bit manipulation to solve the problem of counting the total number of unique solutions to the problem of placing n queens on an nxn chessboard such that no two queens threaten each other.