Consider the problem of trying to place 8 queens on a chess board such that no queen can attack another queen.

Method/Constructor	Description
public Board (int size)	constructs empty board
public boolean isSafe(int row, int column)	place queen here
public void place(int row, int column)	place queen here
public void remove(int row, int column)	remove queen from here
public String toString()	text display of board
solveQueens (Board b)	tries to place 8 queens safely

```
// Searches for a solution to the 8 queens problem
// with this board, reporting the first result found.

public static void solveQueens(Board board) {
    if (!explore(board, 1)) {
         System.out.println("No solution found.");
         } else {
             System.out.println("One solution is as follows:");
             System.out.println(board);
        }
}
```

```
// Recursively searches for a solution to 8 queens on this
// board, starting with the given column, returning true if a
// solution is found and storing that solution in the board.
// PRE: queens have been safely placed in columns 1 to (col-1)

public static boolean explore(Board board, int col) {
    if (col > board.size()) {
         return true;
         // base case: all columns are placed
          } else {
              // recursive case: place a queen in this column.
              for (int row = 1; row <= board.size(); row++) {
                  if (board.isSafe(row, col)) {
                              board.place(row, col);
                              if (explore(board, col + 1)) {
                                  return true;
                                     }
                              b.remove(row, col);
                        }
                     }
                     return false;
                  }
              }
```