Project 2: Eight Queens Problem

John Meyer • 05.04.2017

Overview

Problem

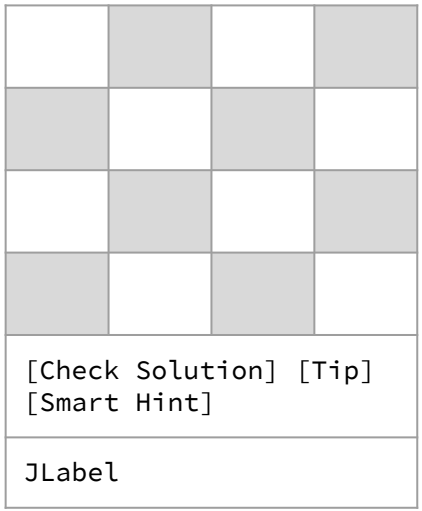
- Start with a blank chess board
- Fill the board with queen pieces such that no queen can attack another queen, if the two were of opposite colors

Task

Create a Java GUI where the user can solve this problem

Design - General Structure

Board extends `JFrame`



Each square is an object

```
public class ChessSquare extends
JComponent {
  // Define instance variables
  private Color backgroundColor;
  private Color highlightColor;
  private Color selectedColor;
  private Image queenImage;
  private boolean hasQueen;
}
```

Organization & Solution Checking

Multidimensional array of `ChessSquare` objects; `Enum` Solution Status

```
private ChessSquare[][] chessSquares;
private enum BacktrackingReturnType {
  Accepted,
  Possible,
  Abandoned
}
private Object[] checkSolution(ChessSquare[][] chessSquares) {
  // Nested for-loops
  return new Object[] { BacktrackingReturnType.Abandoned,
    row, col , rowOfFirstQueen, colOfFirstQueen };
}
```

Design - Blind Tip

```
private Point blindTip(ChessSquare[][] chessSquares) {  // Not recursive
  // Nested for loops
    ChessSquare square = chessSquares[row][col];
    if (!square.hasQueen()) {

      square.setQueen(true);
      BacktrackingReturnType returnVal = checkSolution(chessSquares)[0];
      square.setQueen(false);

      if (returnVal != BacktrackingReturnType.Abandoned) {
        return new Point(row, col);
      }
    }
  return null;
}
```

Design - Smart Tip

```
private ArrayList<Point> smartTip(ChessSquare[][] chessSquares) {  // Recursive
  ArrayList<Point> suggestedSequence = new ArrayList<Point>();
  if (solutionStatus != BacktrackingReturnType.Abandoned) {
    // Copy partial solution to `suggestedSequence`

    // Try to find a better solution
    // Nested for loops
      if (!square.hasQueen()) {
        square.setQueen(true);
        ArrayList<Point> newSuggestedSequence = smartTip(chessSquares);
        square.setQueen(false);

        // Replace the suggested sequence with the new one if it has more queens
        if (newSuggestedSequence.size() > suggestedSequence.size()) {
          suggestedSequence = newSuggestedSequence;
        }
  return suggestedSequence;
```

Interesting Features

- Chess Board is customizable
  - Colors
  - Queen image
  - Board side-length
    - Ex: 10 X 10 boards
    - Solution checking
    - Blind and smart tips
- Chess Board is scalable
  - Queens can be resized to many times their size