```java
package me.jamiepeterson.a110queens;

import java.util.LinkedList;
import java.util.Random;

public class Queens {

    //Board class (inner class)
    private class Board{

        private char[][] array;                 //2d Array is the board
        private int rows;                       //filled rows

        //Constructor
        private Board(int m, int n){
            array = new char[m][n];             //create array

            //initialize array to blanks
            for(int i = 0; i < m; i++)
                for(int j = 0; j < n; j++)
                    array[i][j] = ' ';

            rows = 0;
        }
    }

    private int m;
    private int n;

    //Queens Class Constructor
    public Queens(int m, int n){
        this.m = m;
        this.n = n;
    }

    //Solve Queens Problem
    public String solve(){
        LinkedList<Board> list = new LinkedList<Board>(); //List of boards
        LinkedList<Board> complete = new LinkedList<Board>(); //List of complete boards
        Random rand = new Random();            //RNG

        Board board = new Board(m,n);          //Create empty board
        list.addFirst(board);                  //Add to list

        while(!list.isEmpty()) {                //While list has boards
            board = list.removeFirst();         //Remove first board

            if(complete(board)) {               //If the board is a solution
                int choice = rand.nextInt(1);
                if(choice == 0)
                    complete.addFirst(board);
                else
                    complete.addLast(board);
            }else{
                LinkedList<Board> children = generate(board);

                for(int i = 0; i < children.size(); i++)
                    list.addFirst(children.get(i));
            }
        }
        if(complete.isEmpty())
            return "No Solution";      //Print if there is no solution
        else
            return display(complete.getFirst()); //Print one solution

    }

    //Method generates children of a board
    private LinkedList<Board> generate(Board board){
        LinkedList<Board> children = new LinkedList<Board>(); //Children list

        for(int i = 0; i < m; i++){             //Generate children
            Board child = copy(board);          //Create copy of parent
            child.array[child.rows][i] = 'Q';   //Put queen in the row

            if(check(child, child.rows, i))
                children.addLast(child);

            child.rows ++;                      //Increment Filled Rows
        }
        return children;                        //Return List of children
    }

    //Method checks whether queen at a given location causes conflict
    private boolean check(Board board, int x, int y){
        for(int i = 0; i < m; i ++)             //Go thru all locations
            for(int j = 0; j < n; j++){
                if(board.array[i][j] == ' ');   //If empty ignore
                else if(x == i && y == j);      //If same location ignore
                else if(x == i || y == j || x+y == i+j || x-y == i-j)
                    return false;               //Conflict if in same row, column
, or diagonal
            }
        return true;                            //No conflicts
    }

    //Method makes copy of board
    private Board copy(Board board){
        Board result = new Board(m,n);          //Empty board

        for(int i = 0; i < m; i++)              //Copy given board to empty board
            for(int j = 0; j < n; j++)
                result.array[i][j] = board.array[i][j];

        result.rows = board.rows;               //Copy filled rows

        return result;                          //Return copy
    }

    //Checks if board is complete
    private Boolean complete(Board board){
        return(board.rows == m);                //Check number filled rows equals board size
    }

    //Displays board
    private String display(Board board){
        String displayBoard = "-";

        for(int j = 0; j < n; j++)              //Top horizontal line
            displayBoard = displayBoard + "--";

        displayBoard = displayBoard + "\n";

        for(int i = 0; i < m; i++){             //Every row
            displayBoard = displayBoard + "|";  //First Line
            for(int j = 0; j < n; j++)          //Slots
                displayBoard = displayBoard +board.array[i][j]+ "|";

            displayBoard = displayBoard + "\n";  //Next Line

            displayBoard = displayBoard + "-";
            for(int j = 0; j < n; j++)          //Horizontal line
                displayBoard = displayBoard + "--";

            displayBoard = displayBoard + "\n";     //Next Line

        }
        return displayBoard; //return
    }
}
```