public static void main(String[] args) {

// write your code here

int n = 6;

System.out.println(NQueens.nSol(n));

GenSchemeList<Board> nuova = NQueens.listOfSol(n);

System.out.println(nuova.toString());

drawBoards(nuova);

}

package com.company;

public class NQueens {

// DATA UNA SCACCHIERA nxn INSERIRE IL NUMERO MASSIMO DI REGINE SENZA CHE ESSE SI MINACCINO TRA LORO

public static final GenSchemeList<Board> NULL\_LIST = new GenSchemeList<Board>();

public static int nSol(int n){

return nCompl(new Board(n));

}

public static int nCompl(Board B){

int size = B.getSize();

int queens = B.getQueens();

if (queens == size) return 1;

else {

int r = queens+1;

int count = 0;

for (int c=1;c<=size;c++){

if (!B.sottoScacco(r,c)){

count += nCompl(B.addQueen(r,c));

} }

return count; } }

public static GenSchemeList<Board> listOfSol(int n){

return listOfCompl(new Board(n)); }

public static GenSchemeList<Board> listOfCompl(Board B){

int size = B.getSize();

int queens = B.getQueens();

if (queens == size) return NULL\_LIST.cons(B);

else {

int r = queens+1;

GenSchemeList<Board> sol = NULL\_LIST;

for (int c=1;c<=size;c++){

if (!B.sottoScacco(r,c)){

sol = sol.append(listOfCompl(B.addQueen(r,c))); } }

return sol; } }

} // FINE CLASSE NQUEENS

package com.company;

import java.util.function.BiPredicate;

public class Board {

private static final String ROWS = " 123456789ABCDEF";

private static final String COLS = " abcdefghijklmno";

private final int size; // dimensione board

private final int queens; //numero di regine piazzate

//coppie (i,j) = (r,c) delle coordinate di regine nella scacchiera

private final GenSchemeList<GenSchemeList<Integer>> queensCoords;

// rappresenta un valore procedurale che verrà invocato quando vengono passati i parametri

private final BiPredicate<Integer,Integer> minaccia;

private final String config; // RAPPRESENTAZIONE TESTUALE DELLA CONFIGURAZIONE DELLE REGINE

public Board(int n){

size = n;

queens = 0;

minaccia = (x,y) -> false; // NESSUNA POSIZIONE MINACCIATA

config = " ";

queensCoords = new GenSchemeList<>();

}

\* @param B scacchiera a cui aggiungere la regina

\* @param i colonna in cui aggiungere (abcd...)

\* @param j riga in cui aggiungere (1234...)

\* @return Board in cui è aggiunta la regina

private Board(Board B, int i, int j){

size = B.getSize();

queens = B.getQueens()+1;

minaccia = ( u, v ) -> ( (u == i) || // minaccia la riga i

(v == j) || // minaccia la colonna j

(u-v == i-j) || // minaccia le diagonali i-j e i+j

(u+v == i+j) ||

B.sottoScacco(u,v) // minaccia delle regine in board

);

config = B.getConfig() + COLS.charAt(j) + ROWS.charAt(i) + " ";

GenSchemeList<Integer> l = new GenSchemeList<>();

l = l.cons(j).cons(i); // list( i , j )

queensCoords = B.queensCoords.reverse().cons(l).reverse();

}

// @return dimensione della scacchiera nxn

public int getSize(){return size;}

// @return numero di regine sulla scacchiera

public int getQueens(){return queens;}

// \* @return configurazione stringa delle regine in Board

private String getConfig(){return config;}

// @param i colonna

//@param j riga

// @return true/false se la casella inserita (i,j) è minacciata

public boolean sottoScacco(int i, int j) {

if (queensCoords.isNull()) return false;

GenSchemeList<Integer> primo = queensCoords.car();

GenSchemeList<GenSchemeList<Integer>> resto = queensCoords.cdr();

// CONTROLLO MIACCIA

int u = primo.car();

int v = primo.reverse().car();

while (!((i == u) || (j == v) || (i - j == u - v) || (i + j == u + v))){

// CASELLA I;J NON MINACCIATA; CONTROLLO DELLA PROSSIMA REGINA

if (resto.isNull())return false;

primo = resto.car();

u = primo.car();

v=primo.cdr().car();

resto = resto.cdr();

} return true;

}

/\*\*

\* @param i colonna

\* @param j riga

\* @return board con la regina aggiunta in (i,j)

\*/

public Board addQueen(int i, int j){

return new Board(this,i,j);

}

/\*\*

\* @return conofigurazione finale della board

\*/

public String toString(){

return " " + config + " ";

}

public String getQueensCoords(){

return queensCoords.toString();

}

}