package myfirstproject;

/\*The 8 Queens puzzle is used to determine the position of

8 queens on a chess board such that no one queen can attack

another. There are many possible solutions.\*/

import java.util.Arrays;

public class practice1{

public static void main(String [] args){

//initialize an array of size 64

boolean[] columns = new boolean[64];

/\*

\* i % 7 is the row. i / 8 is the column.

\* First we will randomly initialize the position of Q in column 1

\* by setting all other elements in the column, row, and diagonals to true.

\*/

int randInt= (int)Math.random()\*7;

int rowsBelow = 7 - randInt;

int rowsAbove = randInt;

int columnsRight = 7;

int columnsLeft = 0;

//set up elements true.

for (int j = 1; j <= rowsAbove; j++){

columns[randInt - j] = true;

}

//set down elements true.

for (int j = 1; j <= rowsBelow; j++){

columns[randInt + j] = true;

}

//set right elements true.

for (int j = 1; j <= columnsRight; j++){

columns[randInt + (j)\*8] = true;

}

//set right-down diagonal elements to true.

for (int j = 1; ; ){

int thisRowBelow = rowsBelow;

int thisColumnRight = columnsRight;

if((thisRowBelow != 0)||(thisColumnRight != 0)){

int temp = randInt;

columns[temp + 8 + j] = true;

temp = temp + 8 + j;

thisRowBelow = (temp % 7);

thisColumnRight = 7 - (temp / 8);

}

else break;

}

//set right-up diagonal elements to true.

for (int j = 1; ;){

int thisRowAbove = rowsAbove;

int thisColumnRight = columnsRight;

if ((thisRowAbove != 0)||(thisColumnRight != 0)){

int temp = randInt;

columns[temp + 8 - j] = true;

temp = temp + 8 - j;

thisRowAbove = 7 - (temp % 7);

thisColumnRight = 7 - (temp / 8);

}

else break;

}

//find a general solution for all other columns

for (int i = 8; i < 64; i++){

//run through and select the first false element in each column

if (columns[i] == false){

columnsRight = 7 - (i / 8);

columnsLeft = (i / 8);

rowsAbove = 7 - (i % 7);

rowsBelow = (i % 7);

//set right elements to true

for (int j = 1; j <= columnsRight; j++){

columns[i + (j)\*8] = true;

}

//set left elements to true

for (int j = 1; j <= columnsLeft; j++){

columns[i - (j)\*8] = true;

}

//set up elements to true

for (int j = 1; j <= rowsAbove; j++){

columns[i -j] = true;

}

//set down elements to true

for (int j = 1; j <= rowsBelow; j++){

columns[i + j] = true;

}

//set right-up elements true

for (int j = 1; ;){

int thisRowAbove = rowsAbove;

int thisColumnRight = columnsRight;

if ((thisRowAbove != 0)||(thisColumnRight != 0)){

int temp = i;

columns[temp + 8 - j] = true;

temp = temp + 8 - j;

thisRowAbove = 7 - (temp % 7);

thisColumnRight = 7 - (temp / 7);

}

else break;

}

//set right-down elements true

for (int j = 1; ; ){

int thisRowBelow = rowsBelow;

int thisColumnRight = columnsRight;

if((thisRowBelow != 0)||(thisColumnRight != 0)){

int temp = i;

columns[temp + 8 + j] = true;

temp = temp + 8 + j;

thisRowBelow = (temp % 7);

thisColumnRight = (7 - (temp / 8));

}

else break;

}

//set left-down elements true

for (int j = 1; ;){

int thisRowBelow = rowsBelow;

int thisColumnLeft = columnsLeft;

if ((thisRowBelow != 0)||(thisColumnLeft != 0)){

int temp = i;

columns[temp - 8 + j] = true;

temp = temp - 8 + j;

thisRowBelow = (temp % 7);

thisColumnLeft = (temp / 8);

}

else break;

}

//set left-up elements true

for (int j = 1; ;){

int thisRowAbove = rowsAbove;

int thisColumnLeft = columnsLeft;

if ((thisRowAbove !=0)||(thisColumnLeft != 0)){

int temp = i;

columns[temp - 8 - j] = true;

temp = temp - 8 - j;

thisRowAbove = 7 - (temp % 7);

thisColumnLeft = (temp / 8);

}

}

}

}

//print array elements that are false.

for (int i = 0; i < 64; i++){

if (i % 7 == 0){

if (columns[i] == false)

System.out.println("|Q");

else

System.out.println("| ");

}

else

if (columns[i] == false)

System.out.print("|Q");

else

System.out.print("| ");

}

}

}