Status: - **COMPLETED**

The solution was fully tested for values of n = 4 and n=8. The output did produce the correct placement of Queens, so that they are not in any attacking position. The code is fully based on the Algorithm given in the class by the instructor.

**CODE**

package nqueens ;

/\*\*

\*

\* @author Ashutosh Agarwal

\*/

public class NQueensProblem

{

static int[] x;

public NQueensProblem(int n)

{

x = new int[n];

}

public boolean place(int k, int i)

{

for (int a= 0; a < k; a++)

{

if (x[a] == i || (a - k) == (x[a] - i) ||(a - k) == (i - x[a]))

{

return false;

}

}

return true;

}

public void write(int[] arr)

{

int N = arr.length;

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (arr[i] == j)

{

System.out.print("\* ");

}

else

{

System.out.print("- ");

}

}

System.out.println();

}

System.out.println();

}

public void Nqueens(int k, int n)

{

for (int i = 0; i < n; i++)

{

if (place(k, i))

{

x[k] = i;

if (k == n - 1)

{

write(x);

}

else

{

Nqueens(k + 1, n);

}

}

}

}

public static void main(String args[])

{

NQueensProblem q = new NQueensProblem(8);

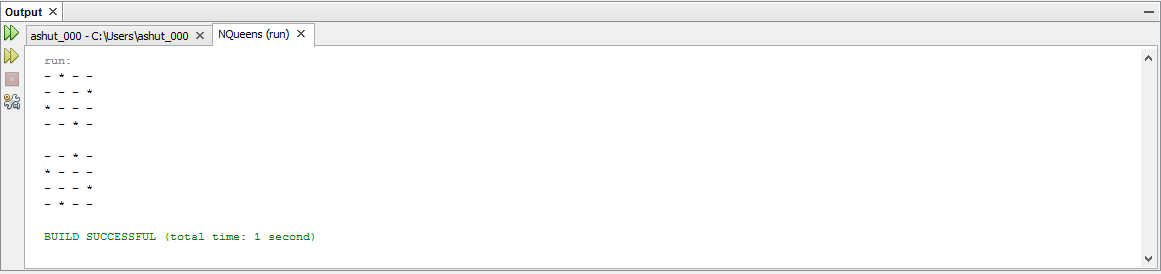
q.Nqueens(0, x.length);

}

}

**SCREENSHOTS OF THE OUTPUT**

For **n=4**



For **n=8**

