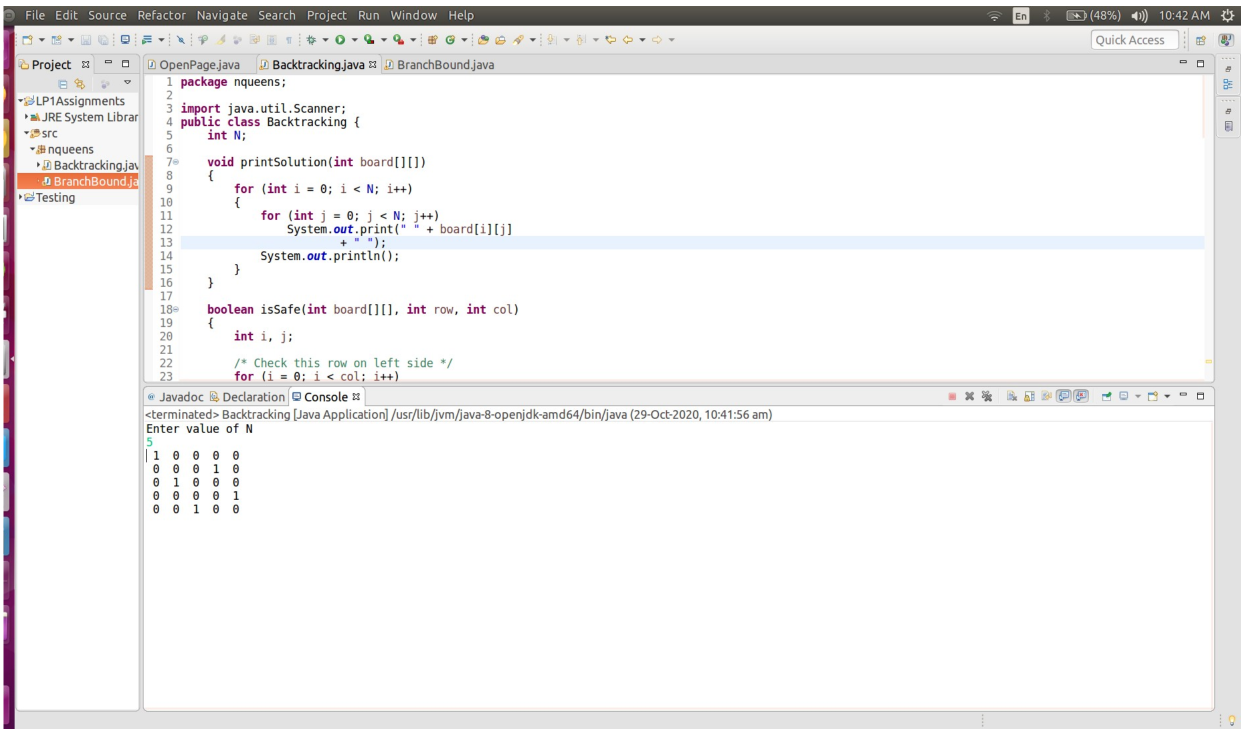
N Queens Backtracking

```
package nqueens;
import java.util.Scanner;
public class Backtracking {
    int N;
    void printSolution(int board[][])
    {
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
                System.out.print(" " + board[i][j]
                        + " ");
            System.out.println();
        }
    }
    boolean isSafe(int board[][], int row, int col)
    {
        int i, j;
        for (i = 0; i < col; i++)
            if (board[row][i] == 1)
                return false;
        for (i=row, j=col; i>=0 && j>=0; i--, j--)
            if (board[i][j] == 1)
                return false;
        for (i=row, j=col; j>=0 && i<N; i++, j--)
            if (board[i][j] == 1)
                return false;
        return true;
    }
    boolean solveNQUtil(int board[][], int col)
    {
        if (col >= N)
            return true;
        for (int i = 0; i < N; i++)
        {
            if (isSafe(board, i, col))
            {
                board[i][col] = 1;
                if (solveNQUtil(board, col + 1) == true)
                    return true;
                board[i][col] = 0; // BACKTRACK
            }
        }
        return false;
    }


    boolean solveNQ(int N)
    {
        int[][] board = new int[N][N];
        for(int i=0;i<N;i++)
        {
            for(int j=0;j<N;j++)
                board[i][j]=0;
        }
        if (solveNQUtil(board, 0) == false)
```

```java
        {
            System.out.print("Solution does not exist");
            return false;
        }
        printSolution(board);
        return true;
    }

    void setN(int n)
    {
        this.N=n;
    }
    public static void main(String[] args) {
      Backtracking q = new Backtracking();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter value of N");
        int N = sc.nextInt();
        q.setN(N);
        q.solveNQ(N);
    }
}
```



```java
package nqueens;

import java.util.Scanner;
public class Backtracking {
    int N;

    void printSolution(int board[][])
    {
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
                System.out.print(" " + board[i][j]
                        + " ");
            System.out.println();
        }
    }

    boolean isSafe(int board[][], int row, int col)
    {
        int i, j;

        /* Check this row on left side */
        for (i = 0; i < col; i++)
```

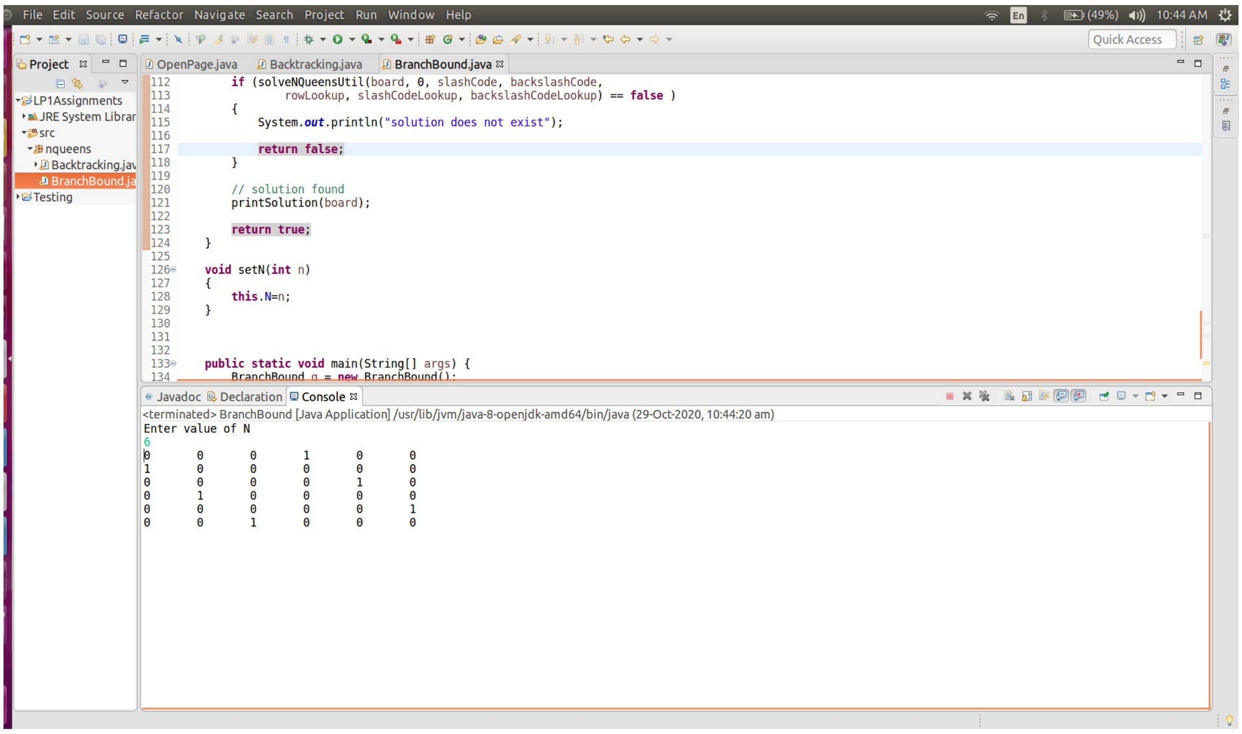
<terminated> Backtracking [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (29-Oct-2020, 10:41:56 am)

```
Enter value of N
5
 1  0  0  0  0
 0  0  0  1  0
 0  1  0  0  0
 0  0  0  0  1
 0  0  1  0  0
```

N Queens Branch and Bound

```
package nqueens;
import java.util.*;
public class BranchBound {
    int N;

    void printSolution(int board[][])
    {
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
                System.out.print(board[i][j]+"\t");
            System.out.println();
        }
    }


    boolean isSafe(int row, int col, int slashCode[][],
                int backslashCode[][], boolean rowLookup[],
                boolean slashCodeLookup[], boolean backslashCodeLookup[] )
    {
        if (slashCodeLookup[slashCode[row][col]] ||
                backslashCodeLookup[backslashCode[row][col]] ||
                rowLookup[row])
            return false;

        return true;
    }


    boolean solveNQueensUtil(int board[][], int col,
                          int slashCode[][], int backslashCode[][], boolean
rowLookup[],
                          boolean slashCodeLookup[], boolean
backslashCodeLookup[] )
    {
        if (col >= N)
            return true;

        for (int i = 0; i < N; i++)
        {
            if ( isSafe(i, col, slashCode, backslashCode, rowLookup,
                    slashCodeLookup, backslashCodeLookup) )
            {
                board[i][col] = 1;
                rowLookup[i] = true;
                slashCodeLookup[slashCode[i][col]] = true;
                backslashCodeLookup[backslashCode[i][col]] = true;

                if ( solveNQueensUtil(board, col + 1, slashCode, backslashCode,
                        rowLookup, slashCodeLookup, backslashCodeLookup) )
                    return true;
                board[i][col] = 0;
                rowLookup[i] = false;
                slashCodeLookup[slashCode[i][col]] = false;
```

```java
                backslashCodeLookup[backslashCode[i][col]] = false;
            }
        }
        return false;
    }


    boolean solveNQueens()
    {
        int board[][] = new int[N][N];
        for(int i=0;i<N;i++)
        {
            for(int j=0;j<N;j++)
                board[i][j]=0;
        }
        int slashCode[][] = new int[N][N];
        int backslashCode[][] = new int[N][N];
        boolean rowLookup[] = new boolean[N];
        for(int i=0;i<N;i++)
            rowLookup[i]=false;
        boolean slashCodeLookup[] = new boolean[2*N - 1];
        for(int i=0;i<2*N-1;i++)
            slashCodeLookup[i]=false;
        boolean backslashCodeLookup[] = new boolean[2*N - 1];
        for(int i=0;i<2*N-1;i++)
            backslashCodeLookup[i]=false;
        for (int r = 0; r < N; r++) {
            for (int c = 0; c < N; c++) {
                slashCode[r][c] = r + c;
                backslashCode[r][c] = r - c + N-1;
            }
        }
        if (solveNQueensUtil(board, 0, slashCode, backslashCode,
                rowLookup, slashCodeLookup, backslashCodeLookup) == false )
        {
            System.out.println("solution does not exist");
            return false;
        }
        printSolution(board);

        return true;
    }
    void setN(int n)
    {
        this.N=n;
    }
    public static void main(String[] args) {
      BranchBound q = new BranchBound();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter value of N");
        int N = sc.nextInt();
        q.setN(N);
        q.solveNQueens();

    }
}
```



```java
        if (solveNQueensUtil(board, 0, slashCode, backslashCode,
                rowLookup, slashCodeLookup, backslashCodeLookup) == false )
        {
            System.out.println("solution does not exist");

            return false;
        }

        // solution found
        printSolution(board);

        return true;
    }

    void setN(int n)
    {
        this.N=n;
    }



    public static void main(String[] args) {
        BranchBound q = new BranchBound();
```

```
<terminated> BranchBound [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (29-Oct-2020, 10:44:20 am)
Enter value of N
6
0       0       0       1       0       0
1       0       0       0       0       0
0       0       0       0       1       0
0       1       0       0       0       0
0       0       0       0       0       1
0       0       1       0       0       0
```