```
1
2   public class BeamSearch {
3
4       /**
5        * @param args the command line arguments
6        */
7       public static boolean isSafe(int[][] board,int row, int col,int num)
8       {
9           // Row has the unique (row-clash)
10          for (int d = 0; d < board.length; d++)
11          {
12
13              // Check if the number we are trying to
14              // place is already present in
15              // that row, return false;
16              if (board[row][d] == num) {
17                  return false;
18              }
19          }
20
21          // Column has the unique numbers (column-clash)
22          for (int r = 0; r < board.length; r++)
23          {
24
25              // Check if the number
26              // we are trying to
27              // place is already present in
28              // that column, return false;
29              if (board[r][col] == num)
30              {
31                  return false;
32              }
33          }
34
35          // Corresponding square has
36          // unique number (box-clash)
37          int sqrt = (int)Math.sqrt(board.length);
38          int boxRowStart = row - row % sqrt;
39          int boxColStart = col - col % sqrt;
40
41          for (int r = boxRowStart;
42               r < boxRowStart + sqrt; r++)
43          {
44              for (int d = boxColStart;
45                   d < boxColStart + sqrt; d++)
46              {
47                  if (board[r][d] == num)
48                  {
49                      return false;
50                  }
51              }
52          }
53
54          // if there is no clash, it's safe
55          return true;
56      }
57
58      public static boolean solveSudoku(
59          int[][] board, int n)
60      {
61          int row = -1;
62          int col = -1;
63          boolean isEmpty = true;
64          for (int i = 0; i < n; i++)
```

```
        {
            for (int j = 0; j < n; j++)
            {
                if (board[i][j] == 0)
                {
                    row = i;
                    col = j;

                    // We still have some remaining
                    // missing values in Sudoku
                    isEmpty = false;
                    break;
                }
            }
            if (!isEmpty) {
                break;
            }
        }

        // No empty space left
        if (isEmpty)
        {
            return true;
        }

        // Else for each-row backtrack
        for (int num = 1; num <= n; num++)
        {
            if (isSafe(board, row, col, num))
            {
                board[row][col] = num;
                if (solveSudoku(board, n))
                {
                    // print(board, n);
                    return true;
                }
                else
                {
                    // replace it
                    board[row][col] = 0;
                }
            }
        }
        return false;
    }

    public static void print(
        int[][] board, int N)
    {

        // We got the answer, just print it
        for (int r = 0; r < N; r++)
        {
            for (int d = 0; d < N; d++)
            {
                System.out.print(board[r][d]);
                System.out.print(" ");
            }
            System.out.print("\n");

            if ((r + 1) % (int)Math.sqrt(N) == 0)
            {
                System.out.print("");
            }
        }
    }
    public static void main(String args[]) {
        // TODO code application logic here
         int[][] board = new int[][] {
            { 3, 0, 6, 5, 0, 8, 4, 0, 0 },
            { 5, 2, 0, 0, 0, 0, 0, 0, 0 },
            { 0, 8, 7, 0, 0, 0, 0, 3, 1 },
            { 0, 0, 3, 0, 1, 0, 0, 8, 0 },
            { 9, 0, 0, 8, 6, 3, 0, 0, 5 },
            { 0, 5, 0, 0, 9, 0, 6, 0, 0 },
```

```
            { 1, 3, 0, 0, 0, 0, 2, 5, 0 },
            { 0, 0, 0, 0, 0, 0, 0, 7, 4 },
            { 0, 0, 5, 2, 0, 6, 3, 0, 0 }
        };
        int N = board.length;

        if (solveSudoku(board, N))
        {
            // print solution
            print(board, N);
        }
        else {
            System.out.println("No solution");
        }
    }
}
```

Execute Mode, Version, Inputs & Arguments

CommandLine Arguments

Stdin Inputs

Result

CPU Time: 0.09 sec(s), Memory: 32040 kilobyte(s) **compiled and executed in 0.576 sec(s)**

```
3 1 6 5 7 8 4 9 2
5 2 9 1 3 4 7 6 8
4 8 7 6 2 9 5 3 1
2 6 3 4 1 5 9 8 7
9 7 4 8 6 3 1 2 5
8 5 1 7 9 2 6 4 3
1 3 8 9 4 7 2 5 6
6 9 2 3 5 1 8 7 4
7 4 5 2 8 6 3 1 9
```

Know Your JDoodle

- JDoodle Supports 76+ Languages with Multiple Versions and 2 DBs. **Click here** to see all.
- Fullscreen - side-by-side code and output is available. click the "⛶" icon near execute button to switch.
- Dark Theme available. Click on "•••" icon near execute button and select dark theme.
- You can embed code from JDoodle directly into your website/blog. **Click here** to know more.
- JDoodle offers an API service. You can execute programs just by calling our API. **Click here** to know more.
- If you like JDoodle, Please share us in Social Media. **Click here** to share.
- Check our **Documentation Page** for more info.

JDoodle is serving the programming community since 2013

JDoodle For Your Organisation

- Do you have any specific compiler requirements?
- Do you want to integrate compilers with your website, webapp, mobile app, courses?
- Do you need more than our **Embed** and **API** features?
- Looking for Multiple Files, Connecting to DB, Debugging, etc.?
- Are you building any innovative solution for your students or recruitment?
- Want to run JDoodle in-house?
- Custom Domain, White labelled pages for your institute?

Contact us - We are happy to help!