

```
1: package it.unisa.di.aa.sudokuforzabruta;
2:
3: public class Coordinates {
4:     int r;
5:     int c;
6:
7:     public Coordinates(int r, int c) {
8:         this.r = r;
9:         this.c = c;
10:    }
11:
12:    public int getR() {
13:        return r;
14:    }
15:
16:    public int getC() {
17:        return c;
18:    }
19:
20: }
```

```
1: package it.unisa.di.aa.sudokuforzabruta;
2:
3: class Solution implements Cloneable {
4:     int[][] numeri;
5:     int size;
6:     int sqrsize;
7:     int recursion_level;
8:
9:     public Solution(int[][] numeri, int size, int rl) {
10:         this.size = size;
11:         this.recursion_level = rl;
12:         sqrsize = (int) Math.sqrt(size);
13:         this.numeri = new int[size][size];
14:         //this.numeri = numeri ---> Copierebbe solo l'indirizzo dell'array
!!!
15:         for (int i=0; i<size; i++) {
16:             for (int j=0; j<size; j++) {
17:                 this.numeri[i][j]=numeri[i][j];
18:             }
19:         }
20:     }
21:
22:     public int[][] getNumeri() {
23:         return numeri;
24:     }
25:
26:     public int getRecursionLevel() {
27:         return recursion_level;
28:     }
29:
30:
31:     @Override
32:     protected Object clone() throws CloneNotSupportedException {
33:         return super.clone();
34:     }
35:
36:     public int getValue(int r, int c) {
37:         return numeri[r][c];
38:     }
39:
40:     public void setCell(int r, int c, int value) {
41:         System.out.println(String.format("Setting cell [%d][%d]=%d in solu
tion "+this,r,c,value));
42:         numeri[r][c] = value;
43:     }
44:
45:     private boolean existsDuplicate(int[] n) {
46:         //System.out.print("Controlllo:");
47:         //for (int i=0; i<n.length; i++) System.out.print(" "+n[i]);
48:         //System.out.println("");
49:         boolean flag = false;
50:         for (int i=0; i<n.length; i++) {
51:             if (n[i]==0) continue;
52:             for (int j=0; j<n.length; j++) {
53:                 if (n[j]==0) continue;
54:                 if ((n[i] == n[j]) && (i != j)) flag=true;
55:             }
56:         }
57:         return flag;
58:     }
59:
60:     public boolean isLegal() {
61:         boolean flag = true;
62:         for (int i=0; i<size; i++) {
63:             //Check column i
64:             if (existsDuplicate(numeri[i])) {
65:                 //System.out.println("Riga "+i);
66:                 return false;
67:             }
68:             //Check row i
```

```
69:         int[] row = new int[size];
70:         for (int j=0; j<size; j++) {
71:             row[j]=numeri[j][i];
72:         }
73:         if (existsDuplicate(row)) {
74:             //System.out.println("Colonna "+i);
75:             return false;
76:         }
77:     }
78:     //Check subsquares
79:     for (int s1=0; s1<sqrsize; s1++) {
80:         for (int s2=0; s2<sqrsize; s2++) {
81:             int[] subsquare = new int[size];
82:             int k=0;
83:             for (int i=0; i<sqrsize; i++) {
84:                 for (int j=0; j<sqrsize; j++) {
85:                     subsquare[k++]=numeri[s1*sqrsize+i
][s2*sqrsize+j];
86:                 }
87:             }
88:             if (existsDuplicate(subsquare)) {
89:                 //System.out.println("Sottomatrice "+s1+"
"+s2);
90:                 return false;
91:             }
92:         }
93:     }
94:     return true;
95: }
96:
97: public int howMany() {
98:     int c=0;
99:     for (int i=0; i<size; i++) {
100:         for (int j=0; j<size; j++) {
101:             if (numeri[i][j] != 0) c++;
102:         }
103:     }
104:     return c;
105: }
106:
107: public boolean isComplete() {
108:     return (this.howMany() == size*size);
109: }
110:
111: public Coordinates getEmptyCell() {
112:     for (int i=0; i<size; i++) {
113:         for (int j=0; j<size; j++) {
114:             if (numeri[i][j] == 0) {
115:                 Coordinates point = new Coordinates(i, j);
116:                 return point;
117:             }
118:         }
119:     }
120:     return null;
121: }
122:
123: public void print() {
124:     int c=0;
125:     System.out.println("solution: "+this);
126:     for (int i=0; i<size; i++) {
127:         for (int j=0; j<size; j++) {
128:             System.out.print(" "+numeri[i][j]);
129:         }
130:         System.out.println("");
131:     }
132:     System.out.println("-----");
133: }
134: }
```

```
1: package it.unisa.di.aa.sudokuforzabruta;
2:
3: public class Sudoku {
4:     static int[][] n4 = {
5:         {0,0,0,0},
6:         {0,0,0,0},
7:         {0,0,0,0},
8:         {0,0,0,0} };
9:
10:    static int[][] n9 = {
11:        {0,0,0,0,0,0,0,0,0},
12:        {0,0,0,0,0,0,0,0,0},
13:        {0,0,0,0,0,0,0,0,0},
14:        {0,0,0,0,0,0,0,0,0},
15:        {0,0,0,0,0,0,0,0,0},
16:        {0,0,0,0,0,0,0,0,0},
17:        {0,0,0,0,0,0,0,0,0},
18:        {0,0,0,0,0,0,0,0,0},
19:        {0,0,0,0,0,0,0,0,0} };
20:
21:
22:    static int size = 4;
23:
24:
25:    public static void main(String[] args) {
26:        // TODO Auto-generated method stub
27:
28:        Solution sol = new Solution(n4,size,0);
29:        Solution sol2 = recursion(sol);
30:
31:        if (sol2 != null) {
32:            sol2.print();
33:            System.out.println("Solution is: "+sol2.isLegal());
34:        }
35:        else {
36:            System.out.println("No solution");
37:        }
38:    }
39:
40:
41:    private static Solution recursion(Solution s) {
42:        int rl = s.getRecursionLevel();
43:        System.out.println("Recursive level="+rl);
44:        Coordinates point = s.getEmptyCell();
45:        if (point == null) {
46:            //Solution is complete
47:            if (s.isLegal() && s.isComplete()) {
48:                return s;
49:            }
50:            else {
51:                return null;
52:            }
53:        }
54:        //fill all possible values
55:        for (int v=1; v<size+1; v++) {
56:            //System.out.println("iteration v="+v);
57:            //Il meotodo clone di Java non funziona!!!
58:            Solution news= new Solution(s.getNumeri(),size,rl+1);
59:            news.setCell(point.getR(),point.getC(),v);
60:            Solution r = recursion(news);
61:            if (r != null) return r;
62:        }
63:        System.out.println("Tried all possibilities. Returning null");
64:        return null;
65:    }
66: }
67:
```