Exact-Cover in Java

Here is the listing of a rather minimal implementation as a variant of Donald Knuths Algorithm X for the Exact-Cover problem in Java, explications below:

```
 1 import java.util.*;
 2
 3 public class AlgX {
 4   Map<Integer,Set<Integer>> rs=new TreeMap<>(),cs=new TreeMap<>();
 5   Set<Integer> s=new TreeSet<>();
 6   AlgX(int[][] a) {
 7     for (int y=0;y<a.length;y++) for (int x=0;x<a[y].length;x++) if (a[y][x]!=0) {
 8       cs.computeIfAbsent(x,i->new HashSet<>()).add(y);
 9       rs.computeIfAbsent(y,i->new HashSet<>()).add(x);
10     }
11   }
12   AlgX(AlgX a,int y){
13     for (int i:a.rs.keySet()) rs.put(i,new HashSet<>(a.rs.get(i)));
14     for (int i:a.cs.keySet()) cs.put(i,new HashSet<>(a.cs.get(i)));
15     s.addAll(a.s); s.add(y); Set<Integer> r=new HashSet<>();
16     for (int c:rs.get(y)) {r.addAll(cs.get(c)); cs.remove(c);}
17     rs.keySet().removeAll(r); for (Set<Integer> c:cs.values()) c.removeAll(r);
18   }
19   int minc() {
20     int min=Integer.MAX_VALUE,c=-1,s;
21     for (int i:cs.keySet()) if ((s=cs.get(i).size())<min) {min=s; c=i;}
22     return c;
23   }
24   void solve() {
25     if (cs.isEmpty()) System.out.println("Solution: "+s);
26     else for (int y:cs.get(minc())) new AlgX(this,y).solve();
27   }
28 }
```

Explications:

In line 4 we instantiate the two maps *rs* and *cs* for the rows and columns of a matrix, where the columns represent the "universe" and the rows represent subsets of that universe. We are looking for a set of rows that covers the universe exactly once in each column.

The set *s* in line 5 collects the rows belonging to a solution.

The constructor in lines 6-10 initializes the maps *rs* and *cs* from an int[][] containing the initial problem as matrix, as for example:

$$\begin{matrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{matrix}$$

The method *solve* in lines 24-28 prints the solution *s*, if no more columns are found, otherwise calls recursively *solve* for a new AlgX constructed in lines 12-18 for each row *y* in the column with the minimal number of covered lines found by method *minc* in line 19-23.

The constructor AlgX in lines 12-18 first makes a copy of *rs*, *cs* and *s* from the calling AlgX, adds the given row *y* to *s,* then for each column in row *y* collects all rows containing this column in a Set *r* and deletes this column from map *cs*.

Finally all rows in *r* are removed from map *rs* and in all remaining columns in *cs* the rows from *r* are removed.

This means, we reduce the original problem to a smaller problem with all rows and columns removed, that are already covered by the rows of the selected column y and if no more colums rest to cover, a solution is found.