```java

//Name -
//Date -
//Class -
//Lab  -

import java.util.Scanner;
import static java.lang.System.*;

public class Forest
{
    private Thing[][] grid;

    public Forest(int rows, int cols)
    {
        final String[] typeList = "cat dog tree rock".split(" ");
        final String[] nameList = "a b c d e f g h i j k l m n o p q r t s u v w x y z".split(" ");

        grid = new Thing[rows][cols];

        for(int o = 0; o < rows; o++)
            for(int i = 0; i < cols; i++)
                grid[o][i] = new Thing( typeList[(int)(Math.random() * 4)], nameList[(int)(Math.random() * 26)], (int)(Math.random() * 35 ));

        //populate the grid

    }

        //count the number of trapped animals
        //remember to use the isTrapped method
    public int numTrapped( )
    {
        int ret = 0;

        for( int o = 0; o < grid.length; o++ ) {
            for( int i = 0; i < grid[0].length; i++ )
                if( isTrapped( o, i ) )
                    ret++;
        }

        return ret;
    }

      //if location ! a rock &&
      //is surrounded by > 5 trees or rocks larger than 10
    public boolean isTrapped( int r, int c)
    {
        int sur = 0;
        if( !( c == grid[0].length - 1 || r == grid.length - 1 || r == 0 || c == 0 ) ) {
            sur += isRorT(grid[r - 1][c - 1])    +
                   isRorT(grid[r - 1][c])        +
                   isRorT(grid[r - 1][c + 1])    +

                   isRorT(grid[r][c + 1])        +
                   isRorT(grid[r][c - 1])        +

                   isRorT(grid[r + 1][c - 1])    +
                   isRorT(grid[r + 1][c])        +
                   isRorT(grid[r + 1][c + 1])    ;
        } else {
            if( c < grid[0].length - 1 )
                sur += isRorT(grid[r][c + 1]) ;
            if( c > 0 )
                sur += isRorT(grid[r][c - 1]) ;
```

```
65                  if( r > 0 )
66                      sur += isRorT(grid[r - 1][c]) ;
67                  if( r < grid.length - 1 )
68                      sur += isRorT(grid[r + 1][c]) ;
69
70                  if( r != 0 && c != 0 )
71                      sur += isRorT(grid[r - 1][c - 1]) ;
72                  if( r != grid.length - 1 && c != grid[0].length - 1 )
73                      sur += isRorT(grid[r + 1][c + 1]) ;
74                  if( r != 0 && c != grid[0].length - 1 )
75                      sur += isRorT(grid[r - 1][c + 1]) ;
76                  if( r != grid.length - 1 && c != 0 )
77                      sur += isRorT(grid[r + 1][c - 1]) ;
78              }
79
80              return sur >= 5;
81          }
82            //make sure the chosen row and column location is in the matrix
83          private boolean inBounds( int r, int c)
84          {
85              return (r < grid.length) && (r>=0) && (c <grid[0].length) && (c>=0);
86          }
87
88
89          public String toString()
90          {
91              String output="";
92              for( Thing [] row : grid )
93              {
94                  for( Thing val : row)
95                  {
96                      if(val==null)
97                          output = output +  "  null \t";
98                      else if(val.getSize() < 10)
99                          output += val + " \t";
100                     else
101                         output = output +  val + "\t";
102                 }
103                 output += "\n";
104             }
105             return output;
106         }
107
108         public void replaceTrapped() {
109             for( int o = 0; o < grid.length; o++ ) {
110                 for( int i = 0; i < grid[0].length; i++ )
111                     if( isTrapped( o, i ) )
112                         grid[o][i] = null;
113             }
114         }
115
116         private static int isRorT( Thing o ){
117             if(o == null)
118                 return 0;
119             if( o.getType().equals( "rock" ) )
120                 return 1;
121             if( o.getType().equals( "tree" ) && o.getSize() > 10 )
122                 return 1;
123             return 0;
124         }
125     }
```