```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class Main {
    public static void main(String[] args) throws IOException, NullPointerException{

        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));

        String[] mosaic_size = input.readLine().split(" ");
        int lines = Integer.parseInt(mosaic_size[0]);        // define o nÃºmero de
linhas do mosaico
        int cols = Integer.parseInt(mosaic_size[1]);       // define o nÃºmero de colunas
do mosaico

        Mosaic m = new Mosaic(cols);
        long result=1;                        // variÃ¡vel que guarda o nÃºmero total de
maneiras diferentes de contruir o tabuleiro dado com diferentes peÃ§as

        for(int l=0;l<lines;l++){

            String[] read_line = input.readLine().split("");    // string read_line vai
guardar o estado da linha introduzida no terminal
            result*=m.line_ways(read_line);
        }

        System.out.println(result);
    }
}

class Mosaic{

    private int c;
    private Integer[] values_bricks={1,2,3,4,6,8,10,12,16};

    public Mosaic(int cols){
        this.c=cols;
    }

    public long line_ways(String[] m){

        int count_equal_bricks=0;        // variÃ¡vel que guarda o nÃºmero de peÃ§as
consecutivas iguais
        long res=1;                      // variÃ¡vel que guarda o nÃºmero total de
maneiras diferentes de preencher uma dada linha do mosaico

        for(int col=0; col<this.c;col++){

            if(!m[col].equals(".")){

                if(col==0)
                    count_equal_bricks=1;

                else{

                    if(m[col].equals(m[col-1]))count_equal_bricks+=1;
                    else{
```

```
                        res*=number_of_ways(count_equal_bricks);
                        count_equal_bricks=1;
                    }
                }
            }
        }
        res*=number_of_ways(count_equal_bricks);

        return res;
    }


    public long number_of_ways(int bricks){

        long[] b= new long[bricks+1];         // array que irÃ¡ possuir todas as maneiras
diferentes de criar um dado tamanho atÃ© ao tamanho calculado(bricks)
        b[0]=1;

        for(int t=1; t<b.length;t++){
            long soma=0;                                          // variÃ¡vel que guarda
o somatÃ³rio de todas as maneiras diferentes para t
            for(int i=0;i<this.values_bricks.length;i++){
                if(this.values_bricks[i]<=t)
                    soma+=b[t-this.values_bricks[i]];
            }
            b[t]=soma;
        }

        return b[bricks];
    }
}
```