**Древо Жизни Андрассил**

*Только в Изумрудном Сне существует Древо Жизни Андрассил. Племена фурболгов, живущие в гигантском пне Нордскола, из поколения в поколение передают легенду об этом дереве, которое когда-нибудь вернётся в реальность и очистит мирные земли от порчи Древних Богов.*

*Главный шаман фурболгов хранит манускрипт, описывающий схему развития Андрассила. Смоделируйте её на компьютере, чтобы всегда можно было узнать, на каком году в каком состоянии оно будет находиться.*

Дерево описывается массивом размером HxW. Каждый элемент массива принимает значение либо 0 (пусто), либо 1 (ветка дерева). При этом у каждого элемента-ветки имеется дополнительная характеристика, равная сроку её жизни в годах.

Исходные данные: размер массива, количество лет для моделирования, и начальная структура дерева в виде текстового изображения (списка текстовых строк).

Например, размер 3x4, моделируем 12 лет, каждый пустой элемент задаётся символом ".", каждая ветка -- символом "+" (её возраст считается равным 1 году):

3 4 12

.+..

..+.

.+..

Каждый год дерево перерождается по следующему алгоритму:  
- в "первый" (чётный) год все ветки стареют на один год, и все пустые клетки заполняются новыми корнями с возрастом 1 год (визуально всё заполнено символами "+");   
- во "второй" (нечётный) год все ветки стареют на один год; ветки, возраст которых три или более лет, погибают, уничтожая также четыре окружающие ветки (если они существуют) -- по горизонтали и вертикали.

Процесс гибели веток с соседями происходит одновременно, т.е. надо учитывать, что каждая ветка с возрастом 3+ лет обязательно уничтожает окружающие ветки (нельзя удалять ветки-соседи простым перебором, потому что соседи тоже в свою очередь могут удалять своих соседей).

В примере для наглядности заменим ветки на числа с возрастом веток.

Исходное дерево:

.1..

..1.

.1..

Прошёл 1-й "чётный" год:

1211

1121

1211

2-й "нечётный" год:

2322

2232

2322

Уничтожение:

...2

2...

...2

3-й "чётный" год (по чётным годам старые ветки не уничтожаются):

1113

3111

1113

4-й "нечётный" год:

2224

4222

2224

Уничтожение:

.2..

..2.

.2..

и т. д.

Напишите программу, которая моделирует N лет развития дерева, и выводит его результирующую форму -- список/массив строк (используются только символы "." и "+").

Функция

string [] TreeOfLife(int H, int W, int N, string [] tree)

получает высоту H (количество строк) и ширину W (длина строк) дерева, количество лет моделирования N и сам список строк, задающий начальное дерево с помощью "." и "+".

Например, исходный пример:

TreeOfLife(3,4, 12, [".+..","..+.",".+.."])

[как постить решение](https://skillsmart.ru/algo/lvl1/how.html)

для начала преобразовываем String[] в char[][] и подменяем ‘+’ на ‘1’ для легкости чтения

String[] testTree = {".+...", ".+...","..+..","..+..","..+.."};

Будет выглядеть как

.1...

.1...

..1..

..1..

..1..

Дальше пишем метод первого года (точки становятся 1, остальные цифры становятся +1)

12111

12111

11211

11211

11211

Промежуточное решение1:

import java.util.Arrays;  
  
public class TreeOfLife {  
  
 public static String [] TreeOfLife(int H, int W, int N, String [] tree){  
  
 int yearCount = 0;  
 int yearLimit = N;  
 char[][] test2Tree = new char[H][W];  
  
 //creating chararray  
 for (int i = 0; i < H; i++){  
 for (int j = 0; j < W; j++){  
 if (tree[i].charAt(j) == '.') {  
 test2Tree[i][j] = '.';  
 }  
 else {  
 test2Tree[i][j] = '1';  
 }  
 }  
 }  
  
 *printArray*(test2Tree);  
  
 *firstYear*(test2Tree);  
  
 *printArray*(test2Tree);  
  
 *firstYear*(test2Tree);  
  
 *printArray*(test2Tree);  
  
 String[] finalResult = {"uwot", "mate"};  
 return finalResult;  
 }  
  
 public static void printArray (char[][] arr2print){  
 for (int j = 0; j < arr2print.length; j++) {  
 for (int i = 0; i < arr2print[0].length; i++) {  
 System.*out*.print(arr2print[j][i]);  
 }  
 System.*out*.println();  
 }  
 System.*out*.println("----------");  
 }  
  
 //first year growth  
 public static char[][] firstYear(char[][] inputChar1){  
 int h1 = inputChar1.length;  
 int w1 = inputChar1[0].length;  
  
 for (int i = 0; i < h1; i++){  
 for (int j = 0; j < w1; j++){  
 if (inputChar1[i][j] == '.') {  
 inputChar1[i][j] = '1';  
 }  
 else {  
 inputChar1[i][j] += 1;  
 }  
 }  
 }  
 return inputChar1;  
 }  
  
 //second year growth + death  
 public static char[][] secondYear(char[][] inputChar2){  
  
  
 return inputChar2;  
 }  
  
 public static void main(String[] args) {  
 int testH = 5;  
 int testW = 5;  
 int testN = 6;  
 String[] testTree = {".+...", ".+...","..+..","..+..","..+.."};  
  
 System.*out*.println(*TreeOfLife*(testH, testW, testN, testTree));  
  
  
  
 }  
  
}

ниже, как мне кажется, рабочая версия с финальной конвертацией в строковый массив и заменой символов только на ‘+’ и ‘.’

import java.util.Arrays;  
  
public class TreeOfLife {  
  
  
  
 public static String [] TreeOfLife(int H, int W, int N, String [] tree){  
  
 int yearCount = 0;  
 int yearLimit = N;  
 char[][] test2Tree = new char[H][W];  
 char[][] marked4death = new char[H][W];  
  
  
 //creating chararray(s)  
 for (int i = 0; i < H; i++){  
 for (int j = 0; j < W; j++){  
 if (tree[i].charAt(j) == '.') {  
 test2Tree[i][j] = '.';  
 }  
 else {  
 test2Tree[i][j] = '1';  
 }  
 marked4death[i][j] = 'o';  
 }  
 }  
  
 //main loop (years of modelling)  
 for (int y = 0; y < yearLimit; y++){  
 if (yearCount % 2 == 0){  
 System.*out*.println("even year");  
 *firstYear*(test2Tree);  
 *printArray*(test2Tree);  
 }  
  
 else {  
 System.*out*.println("odd year");  
 *secondYear*(test2Tree);  
  
 for (int i = 0; i < H; i++){  
 for (int j = 0; j < W; j++){  
 if (test2Tree[i][j] >= '3'){  
 marked4death[i][j] = 'x';  
 }  
 else marked4death[i][j] = 'o';  
 }  
 }  
 *printArray*(test2Tree);  
 //printArray(marked4death);  
 *deleteFields*(test2Tree,marked4death);  
 *printArray*(test2Tree);  
  
 }  
  
 yearCount++;  
 }  
  
 for (int i = 0; i < H; i++){  
 for (int j = 0; j < W; j++){  
 if (test2Tree[i][j] != '.')  
 test2Tree[i][j] = '+';  
 }  
 }  
 String[] finalResult = new String[H];  
 for (int i = 0; i < H; i++){  
 finalResult[i] = new String(test2Tree[i]);  
 }  
  
  
 return finalResult;  
 }  
  
 public static void printArray (char[][] arr2print){  
 for (int j = 0; j < arr2print.length; j++) {  
 for (int i = 0; i < arr2print[0].length; i++) {  
 System.*out*.print(arr2print[j][i]);  
 }  
 System.*out*.println();  
 }  
 System.*out*.println("----------");  
 }  
  
 //first year growth  
 public static char[][] firstYear(char[][] inputChar1){  
 int h1 = inputChar1.length;  
 int w1 = inputChar1[0].length;  
  
 for (int i = 0; i < h1; i++){  
 for (int j = 0; j < w1; j++){  
 if (inputChar1[i][j] == '.') {  
 inputChar1[i][j] = '1';  
 }  
 else {  
 inputChar1[i][j] += 1;  
 }  
 }  
 }  
 return inputChar1;  
 }  
  
 //second year growth + death  
 public static char[][] secondYear(char[][] inputChar2){  
 int h2 = inputChar2.length;  
 int w2 = inputChar2[0].length;  
  
 for (int i = 0; i < h2; i++){  
 for (int j = 0; j < w2; j++){  
 inputChar2[i][j] += 1;  
 }  
 }  
  
 return inputChar2;  
 }  
  
 //marking fields for destruction  
 public static char[][] deleteFields(char[][] inputChar3, char[][] deathMap){  
 int h3 = inputChar3.length;  
 int w3 = inputChar3[0].length;  
  
 for (int i = 0; i < h3; i++){  
 for (int j = 0; j < w3; j++){  
 if (deathMap[i][j] == 'x'){  
 inputChar3[i][j] = '.';  
 if (i > 0) inputChar3[i - 1][j] = '.';  
 if (i < h3 - 1) inputChar3[i + 1][j] = '.';  
 if (j > 0) inputChar3[i][j - 1] = '.';  
 if (j < w3 - 1) inputChar3[i][j + 1] = '.';  
 }  
 }  
 }  
  
 return inputChar3;  
 }  
  
  
  
  
 public static void main(String[] args) {  
 int testH = 5;  
 int testW = 6;  
 int testN = 6;  
 String[] testTree = {".+....", ".+..+.","..++..","..+...","..+..."};  
  
 System.*out*.println(Arrays.*toString*(*TreeOfLife*(testH, testW, testN, testTree)));  
  
  
  
 }  
  
}

вывод в конце выглядит так:

even year

331111

133131

113311

113111

113111

----------

odd year

442222

244242

224422

224222

224222

----------

...2.2

......

2....2

2...22

2...22

----------

[...+.+, ......, +....+, +...++, +...++]