Brandon Bardwell

Computer Science II

2/23/20

Module 3: Assignment 2

For this assignment I also used a website to help me out. For this project I was really confused for most of the time, a lot of the arithmetic side of java is still fairly confusing to me, but I’m getting a better understanding of the main resources used.

<http://soultionmanual.blogspot.com/2016/11/chapter-11-exercise-9-introduction-to.html>

My Code:

package Matrix;

import java.util.ArrayList;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

System.out.print("Enter the array size n: ");

int n = new Scanner(System.in).nextInt();

int[][] m = new int[n][n];

for (int i = 0; i < m.length; i++) {

for (int j = 0; j < m[i].length; j++) {

m[i][j] = (int) (Math.random() \* 2);

}

}

ArrayList<Integer> row = new ArrayList<>();

ArrayList<Integer> column = new ArrayList<>();

getHighestRow(m, row);

getHighestColumn(m, column);

Kit.displayGrid(m);

System.out.println("Highest row: " + row);

System.out.println("Highest column: " + column);

}

public static void getHighestRow(int[][] m, ArrayList<Integer> row) {

int highest = 0;

for (int i = 0; i < m.length; i++) {

int occurrence = 0;

for (int j = 0; j < m[i].length; j++) {

if (m[i][j] == 1) {

occurrence++;

}

}

if (highest < occurrence) {

highest = occurrence;

row.clear();

row.add(i);

} else if (highest == occurrence) {

row.add(i);

}

}

}

public static void getHighestColumn(int[][] m, ArrayList<Integer> column) {

int highest = 0;

for (int j = 0; j < m[0].length; j++) {

int occurrence = 0;

for (int i = 0; i < m.length; i++) {

if (m[i][j] == 1) {

occurrence++;

}

}

if (highest < occurrence) {

highest = occurrence;

column.clear();

column.add(j);

} else if (highest == occurrence) {

column.add(j);

}

}

}

}

**package** Matrix;

**import** javafx.scene.control.TextField;

**public** **class** Kit {

**public** **static** **final** **int** ***NOT\_VALID*** = -1;

**public** **static** String randomNumbers(**int** size) {

String s = "";

**for** (**int** i = 0; i < size; i++) {

s += " " + (**int**) (Math.*random*() \* 100);

}

**return** s;

}

**public** **static** Object[] getArray(Object... objects) {

Object[] temp = **new** TextField[objects.length];

**for** (**int** i = 0; i < objects.length; i++) {

temp[i] = objects[i];

}

**return** temp;

}

**public** **static** String binaryFormat(String binaryString) {

StringBuilder string = **new** StringBuilder(binaryString);

**int** extraZeros = string.length() % 4;

**if** (extraZeros != 0) {

**for** (**int** i = 0; i < extraZeros; i++) {

string.insert(0, "0");

}

}

**for** (**int** i = string.length() - 1; i >= 0; i--) {

**if** (i % 4 == 0 && i != 0) {

string.insert(i, " ");

}

}

**return** string.toString();

}

**public** **static** String hexToBinary(String hex) {

**return** *decimalToBinary*(*hexToDecimal*(hex));

}

**public** **static** String decimalToBinary(**long** n) {

StringBuilder stringBuilder = **new** StringBuilder();

**while** (n > 0) {

stringBuilder.insert(0, n & 1);

n >>= 1;

}

**return** stringBuilder.toString();

}

**public** **static** String decimalToHex(**long** n) {

StringBuilder stringBuilder = **new** StringBuilder();

**while** (n > 0) {

stringBuilder.insert(0, *decimalToHexChar*(n % 16));

n /= 16;

}

**return** stringBuilder.toString();

}

**public** **static** **long** hexToDecimal(String hex) {

**if** (!*isHexadecimal*(hex)) **return** ***NOT\_VALID***;

**long** n = 0;

**int** placeValue = hex.length() - 1;

**for** (**int** i = 0; i < hex.length(); i++) {

**char** ch = hex.charAt(i);

**if** (*isNumeric*(ch)) {

n += (*charToDecimal*(ch) \* Math.*pow*(16, placeValue--));

} **else** {

n += *hexLetterToInt*(ch) \* Math.*pow*(16, placeValue--);

}

}

**return** n;

}

**private** **static** **int** hexLetterToInt(**char** ch) {

ch = Character.*toUpperCase*(ch);

**return** (ch - 'A' + 10);

}

/\*\* returns Z is n is > 15 \*/

**private** **static** **char** decimalToHexChar(**long** n) {

**if** (n > 15) **return** 'Z';

**if** (n > 9) **return** (**char**)(n % 10 + 'A');

**return** (**char**)(n + '0');

}

**public** **static** **long** stringToDecimal(String s) {

**long** n = 0;

**int** placeValue = s.length() - 1;

**for** (**int** i = 0; i < s.length(); i++) {

**int** valid = *charToDecimal*(s.charAt(i));

**if** (valid == ***NOT\_VALID***) **return** ***NOT\_VALID***;

n += (Math.*pow*(10, placeValue--) \* valid);

}

**return** n;

}

**public** **static** String decimalToString(**long** n) {

StringBuilder s = **new** StringBuilder();

**while** (n != 0) {

s.insert(0, n % 10);

n /= 10;

}

**return** s.toString();

}

**public** **static** **int** charToDecimal(**char** ch) {

**if** (!*isNumeric*(ch)) **return** ***NOT\_VALID***;

**return** ch - '0';

}

**public** **static** **boolean** isNumeric(String s) {

**for** (**int** i = 0; i < s.length(); i++) {

**if** (!*isNumeric*(s.charAt(i))) **return** **false**;

}

**return** **true**;

}

**public** **static** **boolean** isNumeric(**char** ch) {

**return** (ch >= '0' && ch <= '9');

}

**public** **static** **boolean** isHexadecimal(String s) {

**for** (**int** i = 0; i < s.length(); i++) {

**if** (!*isHexValid*(s.charAt(i))) **return** **false**;

}

**return** **true**;

}

**public** **static** **boolean** isHexValid(**char** ch) {

ch = Character.*toUpperCase*(ch);

**return** *isCharAtRange*(ch, '0', '9') || *isCharAtRange*(ch, 'A', 'F');

}

**public** **static** **boolean** isCharAtRange(**char** ch, **char** start, **char** end) {

**return** !(ch < start || ch > end);

}

**public** **static** **int** count(String str, **char** a) {

**int** count = 0;

**for** (**int** i = 0; i < str.length(); i++) {

**if** (str.charAt(i) == a) count++;

}

**return** count;

}

**public** **static** **void** displayGrid(**int**[][] grid) {

**for** (**int**[] aGrid : grid) {

System.***out***.printf("%20s", "");

**for** (**int** k = 0; k < aGrid.length; k++) {

System.***out***.printf("|%-2d|", aGrid[k]);

}

System.***out***.println("");

}

}

**public** **static** **void** displayGrid(**int**[] grid) {

**for** (**int** i = 0; i < grid.length; i++) {

System.***out***.printf("%-4d ", grid[i]);

**if** ((i + 1) % 10 == 0)

System.***out***.println("");

}

}

**public** **static** **void** displayTimesTable(**int** x, **int** y) {

// display 8x8 numbered grid to help solve exercise

**for** (**int** i = 1; i <= y; i++) {

System.***out***.printf("%20s", "");

**for** (**int** k = 1; k <= x; k++) {

System.***out***.printf("|%-2d|", i \* k);

}

System.***out***.println("");

}

}

**public** **static** **int**[] makeUnsortedArray(**int** size, **int** range) {

**int**[] array = **new** **int**[size];

**for** (**int** i = 0; i < array.length; i++) {

array[i] = (**int**) (Math.*random*() \* range + 1);

}

**return** array;

}

**public** **static** **double**[][] makeUnsortedArray(**int** rows, **int** columns, **int** range) {

**double**[][] array = **new** **double**[rows][columns];

**for** (**int** i = 0; i < array.length; i++) {

**for** (**int** j = 0; j < array[i].length; j++) {

array[i][j] = (**int**) (Math.*random*() \* range + 1);

}

}

**return** array;

}

**public** **static** **int**[] merge(**int**[] list1, **int**[] list2) {

**int**[] merge = **new** **int**[list1.length + list2.length];

**int** m = 0, l1 = 0, l2 = 0;

**boolean** l1isFinished = **false**;

**boolean** l2isFinished = **false**;

**while** (!l1isFinished || !l2isFinished) {

**if** (l1 == list1.length) l1isFinished = **true**;

**if** (l2 == list2.length) l2isFinished = **true**;

**if** (l1isFinished && l2isFinished) **break**;

**if** (!l1isFinished && l2isFinished) merge[m++] = list1[l1++];

**else** **if** (!l2isFinished && l1isFinished) merge[m++] = list2[l2++];

**else** **if** (list1[l1] <= list2[l2]) merge[m++] = list1[l1++];

**else** **if** (list2[l2] <= list1[l1]) merge[m++] = list2[l2++];

}

**return** merge;

}

**public** **static** **boolean** isSorted(**int**[] numbers) {

**for** (**int** i = 0; i < numbers.length - 1; i++) {

**if** (numbers[i] > numbers[i + 1]) **return** **false**;

}

**return** **true**;

}

**public** **static** **void** displayMatrix(**double**[][] matrix) {

**for** (**double**[] aMatrix : matrix) {

**for** (**int** column = 0; column < aMatrix.length; column++) {

System.***out***.printf("%5.0f ", aMatrix[column]);

}

System.***out***.printf("\n");

}

}

**public** **static** **void** display(Object[] objects) {

**int** count = 1;

**for** (Object o : objects) {

System.***out***.printf("%4s\n", o.toString());

**if** (count % 10 == 0) {

System.***out***.println("");

}

count++;

}

}

}

My Code Output:

Enter the array size n: 8

|1 ||0 ||1 ||0 ||0 ||0 ||0 ||0 |

|1 ||1 ||0 ||0 ||0 ||1 ||1 ||1 |

|0 ||0 ||0 ||0 ||1 ||1 ||0 ||1 |

|0 ||1 ||0 ||1 ||0 ||1 ||0 ||0 |

|0 ||0 ||1 ||1 ||1 ||0 ||0 ||1 |

|0 ||0 ||0 ||0 ||1 ||0 ||0 ||1 |

|1 ||1 ||0 ||0 ||1 ||1 ||1 ||1 |

|0 ||0 ||0 ||1 ||0 ||0 ||1 ||0 |

Highest row: [6]

Highest column: [7]