import java.io.File;

import java.io.FileNotFoundException;

import java.util.ArrayList;

import java.util.Scanner;

/\*\*

\*

\* @author Richelin Metellus

\* @version 04/11/2017

\*/

public class ArrayMethodsClient {

public static void main(String[] args) throws FileNotFoundException {

System.out.println("Entering main");

System.out.println(" Enter the absolute path of the file");

Scanner inputScanner = new Scanner(System.in);

String aboslutePath = inputScanner.nextLine();

File f = new File(aboslutePath);

// File f = new File("C:\\Users\\Rich\\Google Drive\\arrayData.txt");

if(f.exists())

{

int[] array1D = loadArray(f);

int[][] array2D = load2DArray(f);

prettyPrintArray(array1D, 'r');

System.out.println(" Column major form of one dimensional array");

prettyPrintArray(array1D, 'C');

int num = 1234453;

System.out.println(commasCounter(num));

System.out.println(" Printing 2 dimensionnal array in Row Major Form");

prettyPrintArray(array2D, 'R');

System.out.println(" Printing 2 dimensionnal array in Column Major Form");

prettyPrintArray(array2D, 'C');

}

else

System.out.println("Couldn't find the file. Make sure the file path"

+ " is in format C:\\---\\----\\filename.txt");

System.out.println("Exiting main");

}

public static int[] loadArray(File f) throws FileNotFoundException {

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

Scanner fileScanner = new Scanner(f);

String token;

fileScanner.useDelimiter(" ");

int[] fileArray = null;

while (fileScanner.hasNextLine()) {

Scanner lineScanner = new Scanner(fileScanner.nextLine());

while (lineScanner.hasNext()) {

token = lineScanner.next();

if (isInteger(token) == false) {

System.err.print("The file's token " + token + " is non-integer token");

return null;

} else {

bag.add(Integer.parseInt(token));

}

}

fileArray = new int[bag.size()];

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

fileArray[i] = bag.get(i);

}

}

bag.clear(); // to help garbage collection and save memory.

return fileArray;

}

public static int[][] load2DArray(File f) throws FileNotFoundException {

ArrayList<String> fileLineBag = new ArrayList<>(); // each line of the file will be save in a slot of the arrayList

Scanner fScanner = new Scanner(f);

int[][] file2DArray;

int[] rowArray = null;

String token;

while (fScanner.hasNextLine()) {

fileLineBag.add(fScanner.nextLine());

}

file2DArray = new int[fileLineBag.size()][]; // the 2 Dimensional array

int i = 0;

for (String S : fileLineBag) {

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

Scanner lineScanner = new Scanner(S);

lineScanner.useDelimiter(" ");

while (lineScanner.hasNext()) {

token = lineScanner.next();

if (isInteger(token) == false) {

System.err.print("The file's token " + token + " is non-integer token");

return null;

} else {

bag.add(Integer.parseInt(token));

}

}

file2DArray[i] = new int[bag.size()];

for (int j = 0; j < bag.size(); j++) {

file2DArray[i][j] = bag.get(j);

}

bag.clear();

i++;

}

return file2DArray;

}

public static void prettyPrintArray(int[] arr, char S) {

int maxNum;

String sNum;

int numWidth;

switch (S) {

case 'R':

case 'r':

maxNum = max(arr);

sNum = maxNum + "";

numWidth = sNum.length() + commasCounter(maxNum); // how many digits is in that number. which will determine how wide the cell should be.

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

printHorizontalBorder(numWidth);

}

System.out.println("+");

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

System.out.printf("| %," + numWidth + "d ", arr[i]);

}

System.out.println("|");

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

printHorizontalBorder(numWidth);

}

System.out.println("+");

break;

case 'C':

maxNum = max(arr);

sNum = maxNum + "";

numWidth = sNum.length() + commasCounter(maxNum); // how many digits is in that number. which will determine how wide the cell should be.

printHorizontalBorder(numWidth);

System.out.println("+");

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

System.out.printf("| %," + numWidth + "d ", arr[i]);

System.out.println("|");

printHorizontalBorder(numWidth);

System.out.println("+");

}

break;

}

}

public static void prettyPrintArray(int[][] array2d, char S) {

int maxNum; // maximum number in the array

String sNum; // the maximum number as a string

int numWidth; // the width, lenght of the max number

int maxCol; // maximum column of either array

int nonEmptyCellsCount;

String emptySpace;

switch (S) {

case 'R':

case 'r':

maxNum = max(array2d);

sNum = maxNum + ""; //size of the max num;

maxCol = maxColumn(array2d);

numWidth = sNum.length() + commasCounter(maxNum); // how many digits is in that number. which will determine how wide the cell should be.

emptySpace = " ";

for (int i = 0; i < maxCol; i++) {

printHorizontalBorder(numWidth);

}

System.out.println("+");

for (int[] row : array2d) {

nonEmptyCellsCount = row.length;

for (int j = 0; j < maxCol; j++) {

if (nonEmptyCellsCount <= maxCol && j < nonEmptyCellsCount) {

System.out.printf("| %," + numWidth + "d ", row[j]);

} else {

System.out.printf("| %" + numWidth + "s ", emptySpace);

}

}

System.out.println("|");

for (int i = 0; i < maxCol; i++) {

printHorizontalBorder(numWidth);

}

System.out.println("+");

}

break;

case 'C':

case 'c':

maxNum = max(array2d);

sNum = maxNum + ""; //size of the max num;

maxCol = maxColumn(array2d); // max row since you'll print each array cell vertically instead. think of array2d transpose

System.out.println("max row: " +maxCol);

numWidth = sNum.length() + commasCounter(maxNum); // how many digits is in that number. which will determine how wide the cell should be.

emptySpace = " ";

for (int col = 0; col < maxCol; col++) { // maxCol of your file is 14.

System.out.println("|"); // after printing the cells horizontally/i.e transpose the column to be row, need a new linne

//row header, row seprator

for (int i = 0; i < array2d.length; ++i) {

printHorizontalBorder(numWidth);

}

System.out.println("+");

for(int row = 0; row < array2d.length; row++)

{

if(isOutOfBound(array2d,row, col)){

System.out.printf("| %" + numWidth + "s ", emptySpace);

}

else

{

System.out.printf("| %," + numWidth + "d ", array2d[row][col]);

}

}

}

System.out.println("|");

for (int i = 0; i < array2d.length; ++i) {

printHorizontalBorder(numWidth);

}

System.out.println("+");

break;

} // end of switch

}

// -------------------------- Public utility ------------------------------

public static boolean isInteger(String S) {

try {

Integer.parseInt(S);

} catch (NumberFormatException e) {

return false;

}

return true;

}

public static int max(int[] arr) {

int max = arr[0];

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

if (max < arr[i]) {

max = arr[i];

}

}

return max;

}

/\*\*

\*

\* @param array2d

\* @return the maximum integer in a 2D array.

\*/

public static int max(int[][] array2d) {

int max = 0;

int maxNum1d = 0;

for (int i = 0; i < array2d.length; i++) {

if(array2d[i].length != 0)

maxNum1d = max(array2d[i]); // the maximun in the 2nd level array(which a 1 one dismesial arrayusing the previous max fuction

if (maxNum1d > max) {

max = maxNum1d;

}

}

return max;

}

public static int commasCounter(int num) {

int commasCount = 0;

while (num >= 1000) {

num /= 1000;

commasCount++;

}

return commasCount;

}

/\*\*

\*

\* @param array2d two dimensional array

\* @return the maximun number column a 2 dim array have.

\*/

public static int maxColumn(int[][] array2d) {

int max = 0;

for (int[] row : array2d) {

if (row != null && row.length > max) {

max = row.length;

}

}

return max;

}

/\*\* will print +----- base on the specific width past as an argument \*/

public static void printHorizontalBorder(int n) {

System.out.print("+");

for (int i = 0; i < n + 4; i++) {

System.out.print("-");

}

}

/\*\*

\* This method take a two dim array, and handle it if array out of bounds

\* exception occurs.

\* @param array2d

\* @param rowIndex

\* @param columnIndex

\* @return true if an array out of bound occur.

\*/

public static boolean isOutOfBound(int[][] array2d, int rowIndex, int columnIndex) {

try {

int num = array2d[rowIndex][columnIndex];

return false;

}

catch (Exception e) {

return true;

}

}

}

**OUTPUT**

