Module 11 Assignment Results Document

## Program listing:

/\* *Edgar Rosales*

*16 May 2024*

*CSD320-J318 Programming with Java (2245-DD)*

*Write methods using the following headers that returns the location of the largest element in the array passed*

*to the method returning a one-dimensional array that contains two location elements.*

*public static int [] locateLargest (double [][] arrayParam)*

*public static int [] locateLargest (int [][] arrayParam)*

*Then, write methods using the following headers that returns the location of the smallest element in the array passed*

*to the method returning a one-dimensional array that contains two location elements.*

*public static int [] locateSmallest (double [][] arrayParam)*

*public static int [] locateSmallest (int [][] arrayParam)*

\*/

import *java*.*util*.*Random*;

import *java*.*text*.*DecimalFormat*;

*public* *class* SizeOp {

    // *Overloaded Method (double) to find the largest element in a double array.*

*public* *static* int [] locateLargest(double [][] arrayParam){

        int [] located = new int[2];

        double max = 0;

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

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

                if(arrayParam[i][j] > max){

                    max = arrayParam[i][j];

                    located[0] = i;

                    located[1] = j;

                }

            }

        }

        return located;

    }

    // *Oveloaded Method (int) to find the largest element in a double array.*

*public* *static*  int [] locateLargest(int [][] arrayParam){

        int [] located = new int[2];

        int max = arrayParam[0][0];

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

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

                if(arrayParam[i][j] > max){

                    max = arrayParam[i][j];

                    located[0] = i;

                    located[1] = j;

                }

            }

        }

        return located;

    }

    // *Overloaded Method (double) to find the smallest element in a double array.*

*public* *static*  int [] locateSmallest(double [][] arrayParam){

        int [] located = new int[2];

        double min = Double.*MAX\_VALUE*;

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

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

                if(arrayParam[i][j] < min){

                    min = arrayParam[i][j];

                    located[0] = i;

                    located[1] = j;

                }

            }

        }

        return located;

    }

    // *Overloaded Method (int) to find the smallest element in a double array.*

*public* *static*  int [] locateSmallest(int [][] arrayParam){

        int [] located = new int[2];

        int min = Integer.*MAX\_VALUE*;

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

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

                if(arrayParam[i][j] < min){

                    min = arrayParam[i][j];

                    located[0] = i;

                    located[1] = j;

                }

            }

        }

        return located;

    }

    // *Overloaded Method to genrate a double array and fill it with random elements formated to 2 decimal places.*

*public* *static* double [][] createDDRandomDoubleArray() {

        Random rand = new Random();

        DecimalFormat df = new DecimalFormat("#.##");

        int rows = rand.nextInt(10) + 1;

        int cols = rand.nextInt(10) + 1;

        double [][] arrayParam = new double[rows][cols];

        for(int i=0; i<rows; i++){

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

                double randomValue = rand.nextDouble() \* 1000;

                String formattedValue = df.format(randomValue);

                arrayParam[i][j] = Double.parseDouble(formattedValue);

            }

        }

        return arrayParam;

    }

    // *Overloaded Method to genrate a double array and fill it with random elements.*

*public* *static* int [][] createIntRandomDoubleArray() {

        Random rand = new Random();

        int rows = rand.nextInt(10) + 1;

        int cols = rand.nextInt(10) + 1;

        int [][] arrayParam = new int[rows][cols];

        for(int i=0; i<rows; i++){

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

                arrayParam[i][j] = rand.nextInt(1000) + 1;

            }

        }

        return arrayParam;

    }

    // *Main Method, calls Methods to create arrays then methods to find smallest and largest elements*

*public* *static* void main(String[] args){

        double [][] ddArray = createDDRandomDoubleArray();

        int [][] intArray = createIntRandomDoubleArray();

        int [] smallIntArray = locateSmallest(intArray);

        int [] largeIntArray = locateLargest(intArray);

        int [] smallDDArray = locateSmallest(ddArray);

        int [] largeDDArray = locateLargest(ddArray);

        //*Print the Integer array, thn its smallest and larger values*

        System.*out*.println("Generated double array (Integer):");

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

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

                System.*out*.print(intArray[i][j] + "\t");

            }

            System.*out*.println();

        }

        System.*out*.println();

        System.*out*.println("Smallest element in integer array:\t" + intArray[smallIntArray[0]][smallIntArray[1]]);

        System.*out*.println("Largest element in integer array:\t" + intArray[largeIntArray[0]][largeIntArray[1]]);

        //*Print the generated double array od type double, then its lowest and largest values*

        System.*out*.println("\n\nGenerated double array (Double):");

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

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

                System.*out*.print(ddArray[i][j] + "\t");

            }

            System.*out*.println();

        }

        System.*out*.println();

        System.*out*.println("Smallest element in double array:\t" + ddArray[smallDDArray[0]][smallDDArray[1]]);

        System.*out*.println("Largest element in double array:\t" + ddArray[largeDDArray[0]][largeDDArray[1]]);

    }

}

GitHub Link:  
<https://github.com/erosales48/csd/blob/master/csd-320/Module-11/SizeOp.java>

Example 1:  
A screenshot of a computer

Description automatically generated

Example 2:  
A screenshot of a computer

Description automatically generated

Example 3:  
A screenshot of a computer

Description automatically generated