Lab08 – Arrays and methods

Enrique Saracho Felix

100406980

CPSC 1150

08/07/2023

# Exercise 1

## Program Arrays

**File name:** Arrays.java

**Purpose:** To allow the user to enter 5 double values and display the average of those values.

**Packages:** java.util.Scanner

**Input:** 5 double values, which are then stored in an array (*myArray*).

**Output:** A double value representing the average of the 5 input values.

**Pseudocode:**

Algorithm Arrays

START

(**main**)

Set double[ 5 ] *myArray*

Print “Enter 5 double values”

For *i* from 0 to 4 {

Read *myArray*[ *i* ]

}

Print “Average = ” + **average**(*myArray*)

(**average**, parameter(s): integer[ ] *array*)

Set integer *sum* = 0

For *element* in *array* {

*sum* += *element*

}

Return *sum* / *array*.length

(**average**, parameter(s): double[ ] *array*)

Set double *sum* = 0

For *element* in *array* {

*sum* += *element*

}

Return *sum* / *array*.length

END Arrays

**Test run(s):**

A screen shot of a computer

Description automatically generated A black screen with white text

Description automatically generated A black screen with white text

Description automatically generated

# Exercise 2

## Program Matrix

**File name:** Matrix.java

**Purpose:** To allow the user to create a randomly generated matrix, display it, multiply it by a number, and check whether is symmetric.

**Packages:** javax.swing.JOptionPane

**Limitations:** The program will create error messages if the values entered are invalid (not between 1 and 6 for the menu).

**Bugs:** The numbers of the matrix in the messages displayed don’t follow the format specified.

**Input:** Integer values, depending on the part of the program, to choose an option, or to enter values that the program needs for its methods.

**Output:** Various messages with results of the program menu options, errors, input fields, and the menu itself.

**Pseudocode:**

Algorithm Matrix

START

(**main**)

Set integer userInput = 0

Set double[0][0] matrix

While ( userInput != 6 ) {

Read userInput

Switch ( userInput ) {

Case 1:

Read m

matrix = **genMatrix**( m )

Break

Case 2:

If ( matrix.length = 0 ) {

Print error message

} else {

**printMatrix**( matrix )

}

Break

Case 3:

Print error message

Break

Case 4:

If ( matrix.length = 0 ) {

Print error message

} else {

Read c

Set matrix1 = **Multiply**(c, matrix)

**printMatrix**( matrix1 )

}

Break

Case 5:

If ( matrix.length = 0 ) {

Print error message

} else {

Print **isSymetric**( matrix )

}

Break

Case 6:

Break

Default:

Print error message

}

}

(**getMatrix**, parameters: m)

Set double[m][m] matrix

For ( i from 0 to m - 1 ) {

For ( j from 0 to m - 1 ) {

matrix[ i ][ j ] = random double between 0 and 100

}

}

Return matrix

(**printMatrix**, parameters: matrix)

Set integer m = matrix.length

Set string elements = “”

For ( i from 0 to m - 1 ) {

For ( element in matrix[ i ] ) {

elements += element + space

}

Print new line

}

Print elements

(**isSymmetric**, parameters: matrix)

Set integer m = matrix.length

For ( i from 0 to m - 1 ) {

For ( j from 0 to m - 1 ) {

If ( matrix[ i ][ j ] != matrix[ j ][ i ] ) {

Return false

}

}

}

Return true

(**Multiply**, parameters: c, matrix1)

Set integer m = matrix1.length

Set double[m][m] matrix

For ( i from 0 to m - 1 ) {

For ( j from 0 to m -1 ) {

matrix[ i ][ j ] = c \* matrix1[ i ][ j ]

}

}

Return matrix

END Matrix

**Test run(s):**

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated