OBJECT ORIENTED PROGRAMMING

Lab 5: Arrays

### Exercise 1:

What mistakes were made at the begining of the following program?

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

**{ int n=10 ;**

**final int p=5 ;**

**int t1[] = {1, 3, 5} ;**

**int t2[] = {n-1, n, n+1} ;**

**int t3[] = {p-1, p, p+1} ;**

**int t4[] ;**

**t4 = {1, 3, 5} ;**

**float x1[] = {1, 2, p, p+1} ;**

**float x2[] = {1.25, 2.5, 5} ;**

**double x3[] = {1, 2.5, 5.25, 2\*p} ;**

**.....**

Answers

**int t1 [] = {1, 3, 5}; // OKAY**

**int t2 [] = {n-1, n, n + 1}; // OKAY**

**int t3 [] = {p-1, p, p + 1}; // OKAY**

**Note that expressions used in an array initializer do not need to be constant expressions. It suffices that they are calculable at the time when the corresponding declaration is executed, which is the case here.**

**int t4 [];**

**t4 = {1, 3, 5}; // mistake**

**The {...} notation can only be used in the declaration of an array. Here, you must either declare:**

**int t4 = {1, 3, 5};**

**or assign values ​​to each of the elements of t4, after its declaration.**

**float x1 [] = {1, 2, p, p + 1}; // OKAY**

**The values ​​in an array initializer do not have to be of the type of array elements, but only of an assignment-compatible type, which is the case here.**

**float x2 [] = {1.25, 2.5, 5}; // mistake**

**Here, on the other hand, the constants 1.25 and 2.5 are of a double type, not compatible by assignment with the float type of the array.**

**double x3 [] = {1, 2.5, 5.25, 2 \* p}; // OKAY**

**Here, all values ​​of the initializer are compatible by assignment with the double type.**

### Exercise 2:

Write a program which creates an array containing the values of the squares of the first n odd numbers, the value of n being read from the keyboard and which displays the values in the following form:

**How many values : 5**

**1 hase a square value of 1**

**3 hase a square value of 9**

**5 hase a square value of 25**

**7 hase a square value of 49**

**9 hase a square value of 81**

Answers

**We can use the lireInt method of the Keyboard class provided with the labs**

**In Java, the size of an array is only defined when it is created, which allows us to read it on the keyboard here:**

**public class CarrImp**

**{**

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

**{**

**int car[] ;**

**int n ;**

**System.out.print ("How many values : ") ;**

**n = Clavier.lireInt() ;**

**car = new int[n] ;**

**for (int i=0 ; i<n ; i++)**

**car[i] = (2\*i+1)\*(2\*i+1) ;**

**for (int i=0 ; i<n ; i++)**

**System.out.println ((2\*i+1) + " hase a square value of " + car[i]) ;**

**}**

**}**

### Exercise 3:

Write a **UtilTab** utility class that has the following **static** methods:

* **somme** which provides the sum of the values of an array of reals (double) of any size,
* **incre** which increments by a given value all the values of an array of reals (double).
* **genere** which in return provides an array of the first n odd numbers, the value of n being provided as an argument
* **somme2** which receives as argument two integer vectors of the same size and which in return provides an array representing the sum of these two vectors.

Write a small test program. To make things easier, we can also provide the **UtilTab** class with a method **affiche** for displaying the values of a real array.

Answers

**To achieve the incre method, we exploit the fact that when an array is passed as an argument to a method, the latter receives a copy of the corresponding reference, through which it can modify the values of the array. Here we find the same mechanism as for objects. Writing the other methods does not pose a particular problem.**

**class UtilTab**

**{**

**static double** **somme (double[] t)**

**{**

**double s=0. ;**

**for (int i=0 ; i<t.length ; i++) s+= t[i] ; // for (int v : t)**

**s+= v ;**

**return s ;**

**}**

**static void incre (double[] t, double a)**

**{**

**for (int i=0 ; i<t.length ; i++) t[i] += a ;**

**}**

**static void** **affiche (double[] t)**

**{**

**for (int i=0 ; i<t.length ; i++) System.out.print (t[i] + " ") ;**

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

**}**

**public static int[] genere (int n)**

**{**

**int [] res = new int[n] ;**

**for (int i=0, j=1 ; i<n ; i++, j+=2) res[i] = j ;**

**return res ;**

**}**

**public static int[] somme2 (int t1[], int t2[])**

**{**

**int n = t1.length ;**

**if (n != t2.length) return null ;**

**int res[] = new int[n] ;**

**for (int i=0 ; i<n ; i++) res [i] = t1[i] + t2[i] ;**

**return res ;**

**}**

**}**

**public class TstUtil1**

**{**

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

**{**

**double t1[] = {1.25, 2.5, 3.5, 5.} ;**

**System.out.print ("t1 initial = ") ; UtilTab.affiche(t1) ;**

**System.out.println (" somme = " +UtilTab.somme(t1)) ;**

**UtilTab.incre (t1, 1.25) ;**

**System.out.print ("t1 incremente = ") ; UtilTab.affiche(t1) ;**

**System.out.println (" somme = " +UtilTab.somme(t1)) ;**

**int ta[] = {1, 5, 9} ;**

***Output illustration :***

t1 initial = 1.25 2.5 3.5 5.0

somme = 12.25

t1 incremente = 2.5 3.75 4.75 6.25

somme = 17.25

117

ta = 1 5 9

tb = 1 3 5

tc = 2 8 14

**System.out.print ("ta = ") ; UtilTab.affiche(ta) ;**

**int tb[] = UtilTab.genere (3) ;**

**System.out.print ("tb = ") ; UtilTab.affiche(tb) ;**

**int tc[] = UtilTab.somme2 (ta, tb) ;**

**System.out.print ("tc = ") ; UtilTab.affiche(tc) ;**

**}**

**}**

### Exercise 4:

Create a class called **Aleat** allowing to have sequences of random whole numbers. The following methods will be provided

* **constructor** **Aleat (int n, int lim),** **n** representing the number of desired values, belonging to the interval [0, **lim**],
* **getValeur (int n)** which provides the value of rank n of the sequence,
* **getValeurs ()** which provides an array containing all the values of the sequence,
* **histo ()** which provides a histogram of the values of the sequence, that means an array of **lim + 1** values in which an element of index **i** represents the number of times the value **i** is present in the sequence.

Write a small test program.

Answers

**Here the random values will be determined by the constructor and kept in a private array named val.**

**The Math.random method provides a real random number in the range [0, 1 [. We must therefore multiply it by lim + 1 and take the whole part to obtain an integer belonging to the interval [0, lim].**

**In getValues, we avoid returning the reference directly to the private array val because otherwise the calling method could modify its value. In fact, we return the reference to a copy of the table (a copy which, in turn, remains editable!). Finally, in histo, we have to create a new table to calculate the histogram.**

**class Aleat**

**{**

**public Aleat (int n, int l)**

**{**

**nVal = n ; limite = l ;**

**val = new int[n] ;**

**for (int i=0 ; i<nVal ; i++)**

**val[i] = (int)((l+1)\*Math.random()) ;**

**}**

**public int getValeur (int num)**

**{**

**return val[num] ;**

**}**

**public int[] getValeurs ()**

**{**

**int[] res = new int[nVal] ;**

**for (int i=0 ; i<nVal ; i++)**

**res[i] = val[i] ;**

**return res ;**

**}**

**public int[] histo ()**

**{**

**int[] res = new int[limite+1] ; // pour aller de 0 à limite**

**for (int i=0 ; i<nVal ; i++) res[val[i]]++ ;**

**return res ;**

**}**

**private int[] val ;**

**private int nVal, limite ;**

**}**

**public class TstAleat**

**{**

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

**{**

**final int NS1=8, MAX1=5, NS2=10000, MAX2=9 ;**

**Aleat suite1 = new Aleat (NS1, 10) ;**

**System.out.print ("suite1, valeur par valeur = ") ;**

**for (int i=0 ; i<NS1 ; i++)**

**System.out.print (suite1.getValeur(i) + " ") ;**

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

**System.out.print ("suite1, globale = ") ;**

**int[] valeurs = suite1.getValeurs() ;**

**for (int i=0 ; i<NS1 ; i++)**

**System.out.print (valeurs[i] + " ") ;**

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

**int [] hist = suite1.histo() ;**

**System.out.print ("histogramme de suite1 = " ) ;**

**for (int i=0 ; i<=MAX1 ; i++) System.out.print (hist[i] + " ") ;**

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

**Aleat suite2 = new Aleat (NS2, MAX2) ;**

**hist = suite2.histo() ;**

**System.out.print ("histogramme de suite2 = " ) ;**

**for (int i=0 ; i<=MAX2 ; i++) System.out.print (hist[i] + " ") ;**

**}**

**}**

***Output illustration :***

suite1, valeur par valeur = 3 7 9 4 10 7 10 1

suite1, globale = 3 7 9 4 10 7 10 1

histogramme de suite1 = 0 1 0 1 1 0

histogramme de suite2 = 1057 1008 1010 1012 1050 940 976 963 963 1021

### Exercise 5:

Achieve an utility class concerning arrays of arrays of values of type double and containing the following static methods:

* **affiche (double t [ ] [ ]):** displays the values of t, on a one screen line for a table row,
* **boolean regulier (double t [ ] [ ])** : tests whether the array t is regular, that means if all its rows have the same size,
* **double [ ] sommeLignes (double t [ ] [ ])** : provides an array of double corresponding to the sums of the different lines of t,
* **double [ ] [ ] somme (double [ ] [ ] t1, double [ ] [ ] t2)** : ensures that the arrays t1 and t2 are regular and of the same dimensions and in this case provide their sum as a result; otherwise, it provides a null reference.

Write a small test program.

Answers

**Remember that the notion of array with several indices does not exist in Java, which in fact only has the composition of arrays: the elements of an array can in turn be arrays. In this case, the "element arrays" do not need to be the same size. If they are, the array is said to be "regular"; it then makes it possible to simulate the array with several indices of most other languages.**

**class Util2D**

**{**

**public static boolean regulier (double[][] t)**

**{**

**int n = t[0].length ;**

**for (int i=1 ; i<t.length ; i++)**

**if (t[i].length != n) return false ;**

**return true ;**

**}**

**public static double[] sommeLignes (double[][] t)**

**{**

**int nLignes = t.length ;**

**double[] res = new double[nLignes] ;**

**for (int i=0 ; i<nLignes ; i++)**

**{**

**res[i] = 0. ;**

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

**}**

**return res ;**

**}**

**public static double[][] somme (double[][] t1, double[][] t2)**

**{**

**if (!regulier(t1) || !regulier(t2)) return null ;**

**if (t1.length != t2.length) return null ;**

**if (t1[0].length != t2[0].length) return null ;**

**int nLig = t1.length ; int nCol=t1[0].length ;**

**double[][] som = new double[nLig][nCol] ;**

**for (int i=0 ; i<nLig ; i++)**

**for (int j=0 ; j<nCol ; j++)**

**som[i][j] = t1[i][j] + t2[i][j] ;**

**return som ;**

**}**

**public static void affiche (double[][] t)**

**{**

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

**{**

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

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

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

**}**

**}**

**}**

**public class TUtil2D**

**{**

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

**{**

**double[][] a = { {1, 2, 3}, {4, 5, 6}} ;**

**double[][] b = { {6, 5, 4}, {3, 2, 1}} ;**

**double[][] c = Util2D.somme (a, b) ;**

**System.out.println ("a = ") ; Util2D.affiche(a) ;**

**System.out.println ("b = ") ; Util2D.affiche(b) ;**

**System.out.println ("c = ") ; Util2D.affiche(c) ;**

**double[][] d = { { 1, 2}, {1, 2, 3}, {1}, {1, 2, 3, 4, 5}} ;**

**double [] sLig = Util2D.sommeLignes(d) ;**

**System.out.println ("d = ") ; Util2D.affiche(d) ;**

**System.out.print ("somme lignes de d = ") ;**

**for (int i=0 ; i<sLig.length ; i++) System.out.print (sLig[i] + " ") ;**

**}**

**}**

***Output illustration :***

a =

1.0 2.0 3.0

4.0 5.0 6.0

b =

6.0 5.0 4.0

3.0 2.0 1.0

c =

7.0 7.0 7.0

7.0 7.0 7.0

d =

1.0 2.0

1.0 2.0 3.0

1.0

1.0 2.0 3.0 4.0 5.0

somme lignes de d = 3.0 6.0 1.0 15.0