

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'hamed Bouguerra - Boumerdes
Institute of Electrical and Electronics Engineering
Electronics Department



Option: Computer engineering
LAB REPORT n°3
OCTOBER 27, 2022

Title

EE423: Advanced Programming/ Packages, Arrays and Sorting

Authored by :
Agli Wafa
Zidane Aymen

Instructor :
Dr.A Zitouni

Session : 2022/2023

Contents

1	Exercise 01: <i>Understanding packages</i>	1
2	Exercise 02: <i>Arrays, Methods and Sorting</i>	3
2.1	3
2.2	4
2.3	4
2.4	5
2.5	5
2.6	6
2.7	6
2.8	7
2.9	7
2.10	7
2.11	8
3	Exercise 3 : <i>Creating a histogram</i>	8
3.1	8
3.2	9
3.3	10
4	Exercise 4 : <i>Matrices (2d arrays)</i>	11
4.1	11
4.2	11

Introduction Within this lab, we will, in the first part, learn to organize classes by putting them inside packages and, in the second part, we will learn to use arrays and matrices to solve different problems. we will also be introduced to sorting where you will be coding an algorithm called Sort by selection. **Tools and Software:**

1. A PC with ECLIPSE IDE V8.
2. Online LaTeX Editor for writing the report.

1 Exercise 01: *Understanding packages*

1. in order to use the class Car in the Helicopter class we can either

```
1  import pkg1_vehicle.Car;
2
```

or by importing all classes in package pkg1_vehicle :

```
1  import pkg1_vehicle.*;
2
```

2. the package statement that must be added in the class Helicopter is:

```
1  package pkg1_vehicle.pkg2_Air;
2
```

3. Adding the print() method inside the class Plane

```
1 package pkg1_vehicle.pkg2_Air;
2 public class Plane {
3     public void print(){
4         System.out.println("I am a plane");
5     }
6 }
```

4. creating an object p of type Plane and calling the function print()

```
1 package pkg1_vehicle;
2 import pkg1_vehicle.pkg2_Air.Plane;
3 public class Car{
4     public static void main(String[] args) {
5         Plane p = new Plane();
6         p.print();
7     }
8 }
```

5. Adding the print() method inside the class Motorcycle

```
1 package pkg1_vehicle;
2 public class Motorcycle {
3     public void print(){
4         System.out.println("I am a motorcycle");
5     }
6 }
```

6. creating an object motorcycle of type Motorcycle and calling the function print(); since Car and Helicopter are in the same package we may use the wildcard import pkg1_vehicle.*;

```
1 package pkg1_vehicle.pkg2_Air;
2 import pkg1_vehicle.Car;
3 import pkg1_vehicle.Motorcycle;
4 public class Helicopter {
5     Car c;
6     public static void main(String[] args) {
7         Motorcycle motorcycle = new Motorcycle();
8         motorcycle.print();
9     }
10 }
```

7. compiling and running the class Car:

```
src > javac C:\Users\agliwafa\eclipse - workspace\t\src\pkg1_vehicle\Car.java
```

```
src > java pkg1_vehicle.Car
```

8. compiling and running the class Helicopter:

```
javac C:\Users\agliwafa\eclipse - workspace\t\src\pkg1_vehicle\pkg2_Air\Helicopter.java
```

```
src > java pkg1_vehicle.pkg2_Air.Helicopter
```

2 Exercise 02: *Arrays, Methods and Sorting*

2.1

```
1 static String ping(String[] parms) {
2     if(parms==null || parms.length==0 ) return "No parameters passed.";
3     String conc = parms[0]; // conc takes first arrays element
4     /**
5      * Concatenate array's i value with the last resulting string
6      * in order to get the reverse order we put conc at last
7      */
8     for(int i= 1; i<parms.length;i++) conc = parms[i]+"," +conc ;
9     return conc;
10 }
```

Results :

- **Case 01:** if no string is yet allocated in the memory.

```
1     public static void main(String[] args) {
2         String [] arr = null;
3         System.out.println("Case 01: "+ping(arr));
4     }
5
```

- **Case 02:** if an empty array is passed as parameters.

```
1     public static void main(String[] args) {
2         String [] arr = {};
3         System.out.println("Case 02: "+ping(arr));
4     }
5
```

- **Case 03:** if array's length is equal to one.

```
1     public static void main(String[] args) {
2         String [] arr = {"first"};
3         System.out.println("Case 03: "+ping(arr));
4     }
5
```

- **Case 04:** if array's length is greater to one.

```
1     public static void main(String[] args) {
2         String [] arr = {"Our","first","arrays","exercice","."};
3         System.out.println("Case 04: "+ping(arr));
4     }
5
```

```
<terminated> Arrays [Java Application] C:\Users\agli wafa\p2\pool\plug
Case 01: No parameters passed.
Case 02: No parameters passed.
Case 03: first
Case 04: .,exercice,arrays,first,Our
```

2.2

```
1 static int readPositiveInteger() {
2     Scanner s = new Scanner(System.in);
3     int read;
4     System.out.print("Please enter a positive value: ");
5     do { //we prefer do-while since we want to read at least once
6         read = s.nextInt();
7         if(read<=0) System.out.print("not accepted! Enter again: "); //print an error
            message if read<=0
8         }while(read<=0);
9         return read;
10    }
11    public static void main(String[] args) {
12        System.out.println("read = "+readPositiveInteger());
13    }
```

Results :

```
<terminated> Arrays [Java Application] C:\Users\agli wafa\p2\pool\plu
please enter a positive integer: -2
you entred a negative number
please enter a positive integer: 0
you entred a negative number
please enter a positive integer: 1
Your input= 1
```

2.3

```
1 static int [] readIntegers(int n) {
2     int [] parms = new int[n];
3     for(int i= 0; i<parms.length;i++) {
4         System.out.print("Enter the element ["+i+"]: ");
5         parms[i]=readPositiveInteger();}
6     return parms;
7 }
```

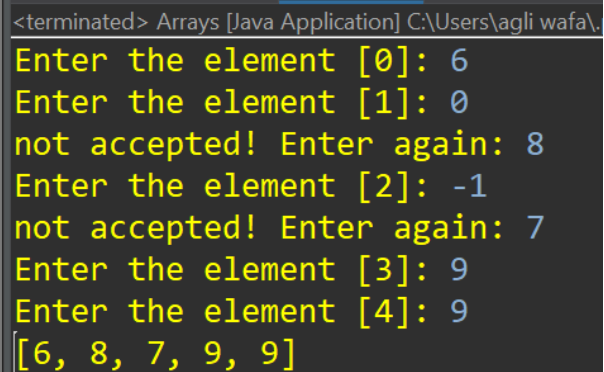
Results :

```
<terminated> Arrays [Java Application] C:\Users\agli wafa\
Enter the element [0]: 9
Enter the element [1]: 1
Enter the element [2]: 0
not accepted! Enter again: 6
Enter the element [3]: -5
not accepted! Enter again: 7
Enter the element [4]: -5
not accepted! Enter again: 6
[I@548b7f67
```

2.4

```
1 static String getArray(int[] parms) {
2     if(parms==null || parms.length==0) return "[]";
3     String conc = "["+parms[0];
4     for(int i= 1; i<parms.length;i++) conc = conc + ","+parms[i];
5     return conc+"]";
6 }
7 public static void main(String[] args) {
8     System.out.println(getArray(readIntegers(5)));}
```

Results :

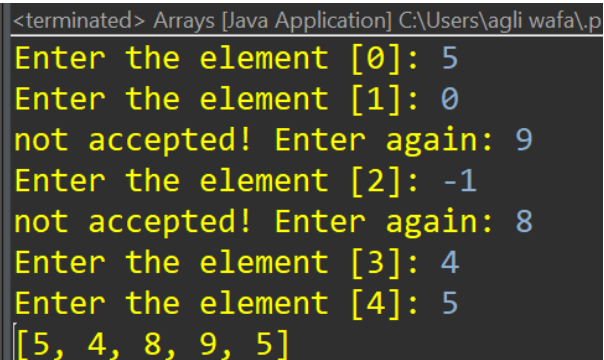


```
<terminated> Arrays [Java Application] C:\Users\agli wafa\...
Enter the element [0]: 6
Enter the element [1]: 0
not accepted! Enter again: 8
Enter the element [2]: -1
not accepted! Enter again: 7
Enter the element [3]: 9
Enter the element [4]: 9
[6, 8, 7, 9, 9]
```

2.5

```
1 static String getArrayReverse(int[] parms) {
2     if(parms==null || parms.length==0) return "[]";
3     String conc = parms[0]+"";
4     for(int i= 1; i<parms.length;i++) conc = parms[i]+","+conc;
5     return "["+conc;
6 }
7 public static void main(String[] args) {
8     System.out.println(getArrayReverse(readIntegers(5)));}
```

Results :

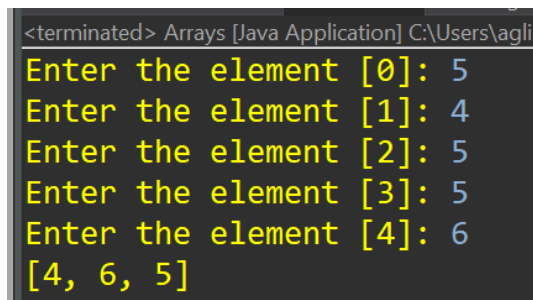


```
<terminated> Arrays [Java Application] C:\Users\agli wafa\p
Enter the element [0]: 5
Enter the element [1]: 0
not accepted! Enter again: 9
Enter the element [2]: -1
not accepted! Enter again: 8
Enter the element [3]: 4
Enter the element [4]: 5
[5, 4, 8, 9, 5]
```

2.6

```
1 static int[] findMinMaxAvg(int[] parms) {
2     int min =parms[0],max = parms[0],sum = parms[0];
3     for(int i= 1; i<parms.length;i++) {
4         if(parms[i]<min)min =parms[i];
5         if (parms[i]>max)max = parms[i];
6         sum = sum + parms[i];
7     }
8     int[] MinMaxAvg = {min,max,sum/parms.length};
9     return MinMaxAvg;
10 }
```

Results :



```
<terminated> Arrays [Java Application] C:\Users\agli
Enter the element [0]: 5
Enter the element [1]: 4
Enter the element [2]: 5
Enter the element [3]: 5
Enter the element [4]: 6
[4, 6, 5]
```

2.7

```
1 import java.util.Arrays;
2 public static boolean areTheSame(int arr1[], int arr2[]) {
3     if(arr1.length!=arr2.length) return false;
4     Arrays.sort(arr1); Arrays.sort(arr2); //Sort each arrays using Arrays.sort method
5     for (int i = 0; i < arr1.length; i++)
6         if (arr1[i] != arr2[i]) return false; //comparing element by element the arrays
7     return true;
8 }
```

Results :

- **Case 01:** if two arrays have identical values (order doesn't matter but repetitions does) and lengths

```
1     public static void main(String[] args) {
2         int [] arr1 = {4, 3, 1,4};
3         int [] arr2 ={1, 3, 4,4};
4         System.out.println(areTheSame(arr1,arr2));
5     }
6
```

- **Case 02:** if arrays are not identical by (finding a value only in one array) or not having the same occurrence number of a value

```
1     public static void main(String[] args) {
2         int [] arr1 ={1, 1, 2, 2, 2, 2, 1};
3         int [] arr2 = {2, 2, 2, 2, 1, 1, 2} ; System.out.println(areTheSame(arr1,arr2
4         ));}
5
```


2.8

```
1 static boolean areSorted(int[] parms) {
2     int len = array.length;
3     boolean inc = true, dec = true;
4     for (int i = 0; i < len - 1; i++) {
5         if(array[i] > array[i+1])//it's not incremently sorted
6             inc = false;
7         if(array[i] < array[i+1])//it's not decremently sorted
8             dec = false;
9         if(inc == false && dec == false)//if both are false no need to continue the loop
10            return false;
11    }
12    return true;
13 }
14
```

Results :

```
<terminated> Arrays [Java Application] C:\Users\agli wafa\.p2\pool\p
{1 , 1 , 2 , 2 , 2 , 2 , 1}: false
{2 , 2 , 2 , 2 , 1 , 1 , 1}: true
{2 , 2 , 2 , 2 , 3 , 3 , 3}: true
```

2.9

```
1 static int getIndexMinValue(int[] parms , int ind) {
2     int min =parms[ind]; int inx =ind;
3     for(int i= ind; i<parms.length;i++) {
4         if(parms[i]<min) {
5             min =parms[i];
6             inx = i;
7         }
8     }
9     return inx;
10 }
```

Results :

```
<terminated> Arrays [Java Application] C:\Users\agli wafa\.p2\pool\plugins\o
Index of min value starting from 2
in {1 , 10 , 5 , 2 , 9 , -2 , 1} is: 5
```

2.10

```
1 static int [] swapValues(int[] parms,int ind1,int ind2 ) {
2     int j;
3     j= parms[ind1];
4     parms[ind1] = parms[ind2];
5     parms[ind2]=j;
6     return parms;
7 }
```

Results :

```
<terminated> Arrays [Java Application] C:\Users\agli wafa\.p2\po
Before : [1, 10, 5, 2, 9, -2, 1]
After  : [1, 1, 5, 2, 9, -2, 10]
```

2.11

Algorithm 01 : Selection sort

Step 01 : find the index of the minimum value between i and n-1

Step 02 : swap the value at position i with the value at minIndex

Step 03 : Repeat the process until the position is at n-1

```
1 static int [] sortValues(int[] array) {
2     int[] copyOfArray = array.clone(); //we create a copy in order to preserve the
    original array
3     for (int j =0;j<array.length;j++) {
4         int min = getIndexMinValue(copyOfArray, j);
5         copyOfArray = swapValues(copyOfArray,j,min);
6     }
7     return copyOfArray;
8 }
9
```

Results :

```
<terminated> Arrays [Java Application] C:\Users\agli wafa\.p2\po
Before : [1, 10, 5, 2, 9, -2, 1]
After  : [-2, 1, 1, 2, 5, 9, 10]
```

3 Exercise 3 : *Creating a histogram*

3.1

```
1 static int[] getStats(int[] grades) {
2     int[] stats = new int[11];
3     int counter=0;
4     for(int g=0;g<stats.length;g++) {
5         for(int i=0;i<grades.length;i++) {
6             if(grades[i]==g) counter++;
7         }
8         stats[g]=counter;
9         counter=0;
10    }
11    return stats;
12 }
```

Results :

```
<terminated> Histogram [Java Application] C:\Users\agil\
grade[0] = 2
grade[1] = 2
grade[2] = 2
grade[3] = 3
grade[4] = 3
grade[5] = 3
grade[6] = 3
grade[7] = 4
grade[8] = 5
grade[9] = 5
grade[10] = 5
grade[11] = 6
grade[12] = 6
grade[13] = 6
grade[14] = 6
grade[15] = 6
grade[16] = 6
grade[17] = 6
grade[18] = 6
grade[19] = 6
grade[20] = 7
```

(a) Reading grades[0-20]

```
grade[21] = 7
grade[22] = 7
grade[23] = 7
grade[24] = 7
grade[25] = 7
grade[26] = 8
grade[27] = 8
grade[28] = 8
grade[29] = 9
grade[30] = 9
grade[31] = 9
grade[32] = 10
[0,0,3,4,1,3,9,6,3,3,1]
```

(b) Reading grades[21-32] + printing stats

Figure 1: results of executing code **3.1**

Nb: We used *readPositiveInteger* and *readIntegers* in order to input the grades

3.2

```
1 static String showDiag(int[] stats) {
2     String stars = "", newS="";
3     for(int i=0;i<stats.length;i++) {
4         for(int g=0;g<stats[i];g++) stars = "*" + stars; //get number of stars in the stat i
5         newS=newS+"\n"+i+"\t"+stars;
6         stars = "";
7     }
8     return newS;
9 }
```

Results :

```
0
1
2     ***
3     ****
4     *
5     ***
6     ****
7     ****
8     ***
9     ***
10    *
```

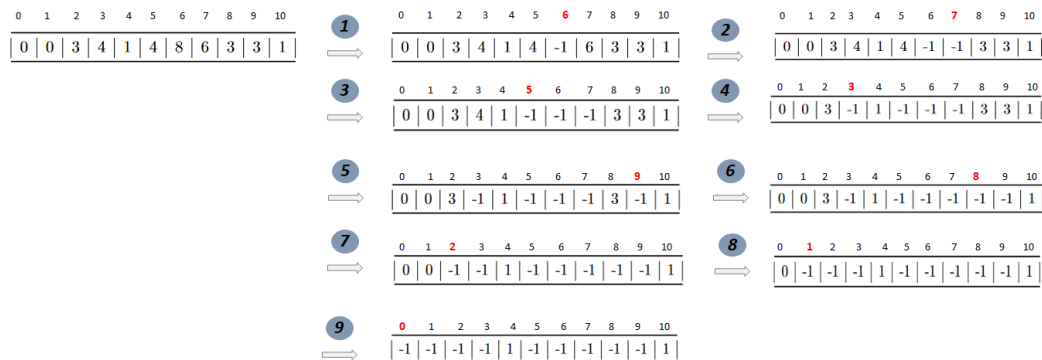
3.3

```

1 static int getIndexMaxValue(int[] parms) {
2     int max =parms[0]; int inx =0;
3     for(int i= 0; i<parms.length;i++) {
4         if(parms[i]>=max && parms[i]!= -1 ) {
5             max =parms[i];
6             inx = i;
7         }
8     }
9     return inx;
10 }
11
12 static String showDiagSorted(int[] stats) {
13     String stars = "",newS="";
14     for(int i=0;i<stats.length;i++) {
15         int max = getIndexMaxValue(stats);//find the highest stat
16         for(inghest t g=0;g<stats[max];g++)stars = "*" +stars; //get stats[max] star
17         newS=newS+"\n"+max+"\t"+stars;
18         stars = "";
19         stats[max] = -1; //stats[max] will be neglected in next itterations
20     }
21     return newS;
22 }

```

Results and illustration:



```

6 *****
7 *****
3 *****
9 *****
8 *****
5 *****
2 *****
10 *
4 *
1
0

```

Nb: The illustrative example is not the same as the execution one; We missed an iteration during illustration

4 Exercise 4 : *Matrices (2d arrays)*

4.1

```
1 public static int[][] readMatrix(int l, int c){
2     Scanner s = new Scanner(System.in);
3     int[][] mat= new int[l][c]; //allocate memory space for matrix with l row & c column
4     for(int i=0;i<l;i++){
5         for(int j=0;j<c;j++){
6             System.out.print("m["+i+"]["+j+"] : ");
7             mat[i][j]= s.nextInt();
8         }
9     }
10    return mat;
11 }
```

4.2

```
1 /**
2  * return lenght of max value in a matrix
3  */
4 public static int findMax(int[][] mtx) {
5     int max = mtx[0][0];
6     for(int i=0;i<mtx.length;i++){
7         for(int j=0;j<mtx[0].length;j++){
8             if(max<mtx[i][j]) max = mtx[i][j];
9         }
10    }
11    return String.valueOf(max).length();
12 }
13 public static void printMatrix(int[][] mtx, int max) {
14     int rowLength = mtx[0].length * max + mtx[0].length + 1; //number of - needed for
15     //each row
16     System.out.print(rowLength);
17     final char[] array = new char[rowLength];
18     Arrays.fill(array, '-');
19     String rowDivider = new String(array);
20     for(int i=0;i<mtx.length;i++){
21         System.out.println(rowDivider);
22         for(int j=0;j<mtx[0].length;j++){
23             // we used printf to create fixed space for each cell putting max value in
24             // consideration
25             System.out.printf("%-"+max+"d",mtx[i][j]);
26         }
27         System.out.println("|");
28     }
29     System.out.println(rowDivider);
30 }
```

```
<terminated> Matrices [Java Application] C:\Users\agli wafa
m[0][0] : 1000000
m[0][1] : 200
m[0][2] : 2
m[1][0] : 78
m[1][1] : 219
m[1][2] : 19999
m[2][0] : 209
m[2][1] : 7467
m[2][2] : 83999
25
-----
|1000000|200    |2      |
-----
|78      |219      |19999  |
-----
|209     |7467     |83999  |
-----
```

Figure 2: Reading and printing a matrix