IE ALGORITHMIQUE & PROGRAMMATION
SCAN S2
April 2019

Total duration : 1.5 hours
Authorized documents : None

— The subject is on 4 pages
— The points are given as indication and the subject
— Exercises are independent.

Exercise 1 : Code understanding (5pts)

(1.1) What is displayed during the execution of the following program?

```
1  public class Exo1 {
2      public static void main(String[] args) {
3          int n=4;
4          int [][] t1;
5          int [][] t2;
6          int [][] t3;
7          t1 = tabNum1(n);
8          display(t1);
9          display2(t1);
10         t2 = modif(t1);
11         display(t2);
12         t3 = t2;
13         t1 = t2;
14         if (t1 == t3) {
15             System.out.println("the arrays are identical");
16             } else {
17                 System.out.println("the arrays are different");
18         }
19         if (t1 == t2) {
20             System.out.println("the arrays are identical");
21             } else {
22                 System.out.println("the arrays are different");
23         }
24     }
25
26     public static int [][] tabNum1(int n) {
27         int[][] tab = new int[n][n];
28         for (int i = 0; i < tab.length; i++){
29             for (int j = 0; j < tab[0].length; j++){
30                 tab[i][j] = 1 + n*i + j;
31             }
32         }
33         return tab;
34     }
```

```
35
36      public static void display(int [][] tab){
37          for(int i = 0; i < tab.length; i++){
38              for(int j = 0; j < tab[0].length; j++){
39                  System.out.print(tab[i][j]+"␣");
40              }
41              System.out.println();
42          }
43          System.out.println();
44      }
45
46      public static void display2(int [][] tab){
47          for(int i = 0; i < tab.length; i++){
48              for(int j = 0; j < tab[0].length; j++){
49                  if (j>=i) System.out.print(tab[i][j]+"␣");
50              }
51          }
52          System.out.println();
53          System.out.println();
54      }
55
56      public static int [][] modif(int [][] t) {
57          int [][] tab = new int[t.length][t[0].length];
58          for (int i = 0; i < tab.length; i++){
59              for (int j = 0; j < tab[0].length; j++){
60                  tab[i][j] = t[i][j]%10;
61              }
62          }
63          return tab;
64      }
65  }
```

Exercise 2 : 2D arrays (7pts)

We want to implement a car sharing system in a urban area. A driver going on a planned travel can offer places in his car to potential passengers. It is thus necessary to connect drivers with passengers so that they find travels corresponding to their needs.

We have a 1D array, called `cities`, that contains the initials of all the cities in the area. It is declared and initialized as follows :

```
char [] cities = {'V','E','L','S','A','C','D'};
```

The drivers each have an identifier and have registered their travel in a database. This database is represented by a 2D array, called `stop`, of type `int [] []` that lists for each driver, the time of passage in each of the cities. If the driver does not travel through a city, the corresponding value is `-1`.

- Line number `i` : driver identifier,
- Column number `j` : index of the city in the array `cities`,
- `stop[i][j]` $= h$: hour h at which the driver i will stop in the city j,
- If `stop[i][j]` $= -1$, then the driver i does not travel through the city of index j.

-1	2	3	-1	4	-1	7
1	3	-1	-1	4	5	-1
-1	1	4	5	-1	6	8

TABLE 1 – Array of stops. Ex. the driver 2 stops in city L at 4 :00.

Based on these two arrays, we are going to develop an application allowing passengers to find a driver for their travel.

(2.1) Write a method `indexCity` that returns the index of the city corresponding to the initial given as parameter, or -1 if this initial is not found.

```
/**
* Find the index of the city given the initial
* @param cities: array of cities
* @param initial: initial of the city
* @return index of the initial in the cities array
*/
public static int indexCity(char[] cities, char initial)
```

(2.2) Write the method `listStops` that displays the indexes of the cities in which the driver i will stop, as well as the time of stop.

```
/**
* Display the indexes of the cities and the time of stops of a driver
* @param stop: array of stops
* @param driver: driver's identifier
*
* Example of display:
* Index 0 hour 1
* Index 1 hour 3
* Index 4 hour 4
* Index 5 hour 5
*/
public static void listStops(int[][] stop, int driver)
```

(2.3) Write a method `searchTravel` that displays the identifiers of the drivers that stop in the cities with initial $C1$ and $C2$, and stops in $C1$ after hour h. We assume that $C1$ and $C2$ are in the array `cities`.

Remark : We need you to pay a special attention to the definition of the signature of this method

```
1 /* Example of display for C1=E h=2 and C2=A
2 * The driver 0 stops in E at 2:00
3 * The driver 1 stops in E at 3:00
4 */
```

(2.4) *Write another version of* `searchTravel` *that returns the identifier of the driver which travel is the shortest (timewise), and* -1 *if no driver fulfills the travel requirements.*

Exercise 3 : Object oriented programming (8pts)

Arrays allow to store a fixed number of values of the same type. In this exercise, we will focus on the *stack* structure, that allows to store an arbitrary number of values of the same type and to access them in a precise order. Lets consider the analogy with a stack of plates. One can put (or *push*) a new plate on the top of the stack, and one can only remove (or *pop*) a plate that is at the top of the stack.

We have a class Stack under the form of a compile file `Stack.class` - we do not have the source code `Stack.java`. We only know the signatures of the public methods. The rest, including the attributes, is private. The public methods are the following :

```
Stack (int maxSize)          // Constructor: create a stack of maximum size:
   maxSize
boolean isEmpty()            // Return true if the stack is empty
boolean isFull()             // Return  true if the stack is full
void push(int value)         // Push a value on the stack. If the stack is
   full, do nothing.
int pop()                    // Pop  and return the value at the top of the
   stack (the value is removed from the stack). If the stack is empty,
   return -1.
void empty()                 // Empty the stack
void display()               // Display the content of the stack from the top
   to the bottom.
void sort()                  // Sort the content of the stack. The smallest
   values will be placed at the top of the stack.
```

(3.1) *Create a class* `TestStack` *containing a main method, that creates a stack, then pushes in this stack the values from 1 to 10, and displays its content.*

(3.2) *In the same main method, create a second stack and transfer the content of the first stack into the second stack, in such a way that the display of the second stack produces* ***exactly the same result*** *as the display of the first stack.*

(3.3) *In a new method main, create two stacks. Generates random values taken in [0..9] and use them to fill the two stacks until one of them is full. The first stack will contain even values while the second stack will contain odd values. Suggest a solution as simple as possible.*

(3.4) *Create a method* `displaySort` *taking two stacks as parameters, and displaying all their values under the form of a unique sorted sequence (from the lowest to the greatest value).*
Warning : *This solution should not use a third stack, nor arrays, but you are of course allowed to use the public methods of the* `Stack` *class.*