Exam of Computer Science
SCAN 1st year - April 2021

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Total duration : *1h30*
Allowed materials : *None.*
Smartphones are not allowed

- The graduation can still change.
- The assignment is on 5 pages.
- Questions are independent. If it is necessary for A to be solved to solve B, then you can do as if you had solved A (and clearly indicate your assumption) to solve B.

Thought of the day:
« Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. » *Martin Golding*

Indicative marking scheme :

- Exercise 1 : 4 points
- Exercise 2 : 8 points
- Exercise 3 : 8 points

Before you start :

- Parts are independent. You can answer them in the order of your choice.
- The subject is long. Take some time to read all the questions before you start.
- The neatness of your exam script will be taken into account.

1 Program understanding (4pts)

We assume that the method `afficheTab` displays the content of a 2D array line by line.

(Q1.1) What is displayed by the following program?

```
1  public class Exo1 {
2     public static void main(String args[]){
3        int[][] t1 = { { 1 , 2 , 3 } , { 4 , 5 , 6 } , { 7, 8 , 9 } };
4        int[][] t2 = method1(t1);
5        afficheTab(t2);
6
7        t2 = t1;
8        displayArray(t2);
9        method2(t2);
10       displayArray(t2);
11       displayArray(t1);
12
13       int[][] t3 = method3();
14       displayArray(t3);
15    }
16
17    public static int[][] method1 (int[][] tab) {
18       int tab2[][] = new int[3][3];
19       for(int i = 0; i < tab.length; i++){
20          for(int j = 0; j < tab[i].length; j++){
21             if(i > j){
22                tab2[i][j] = 0;
23             } else {
24                tab2[i][j] = tab[i][j];
25             }
26          }
27       }
28       return tab2;
29    }
30
31    public static void method2 (int[][] tab) {
32       for(int i = 0; i < tab.length; i++){
33          tab[i][i] = 0;
34       }
35    }
36
37     public static int[][] method3(){
38       int t3[][] = new int[3][];
39       for(int i = 0; i < t3.length; i++){
40          t3[i] = new int[i+1];
41          for(int j = 0; j < i + 1; j++){
42             t3[i][j] = j + 1;
43          }
44       }
45       return t3;
46    }
47 }
```

2 Square arrays (8 points)

In Exo2 class, the square grid of a game is represented as an array with n lines and n columns that contains integer values between 1 and $n \times n$. In a first time, we will consider a 1D array of size $n \times n$ and we will write the methods required in the following steps.

(Q2.1) Complete the `creation1DArray` method that has the signature given below. This method creates an array of integers that contains integer values from 1 to `length` in successive order. For instance, if `length` is equal to 9, the array is $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

```
public static int[] creation1DArray (int length)
```

(Q2.2) Complete the `exchange` method that has the following signature. This method swaps the content of two cells at positions `pos1` and `pos2` in the array `tab` .

```
public static void exchange (int pos1 , int pos2 , int[] tab)
```

(Q2.3) Provide the signature and the implementation of the `shuffle` method that takes as parameter an array of integers `tab` and that uses the previous method to swap the contents of two randomly selected cells. This random exchange is repeated N times, where N is the length of the array `tab`.

Reminder : the method `Math.random()` generates a random number in $[0; 1[$

(Q2.4) Provide the signature and the implementation of the `creation2DArray` method that takes as parameter a 1D array and returns a 2D array. Without verification, we assume that the 1D array is of size $n \times n$ such that the returned 2D array has exactly n lines and n columns. The values of the 1D array will be stored line by line in the 2D array. For instance, if the 1D array is $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, the 2D array will be $\{\{1, 2, 3\}, \{4, 5, 6\}, \{7, 8, 9\}\}$.

Reminder : the method `Math.sqrt(x)` computes the square root of x

(Q2.5) Write the `main` method that uses the previous methods to create a square array with n lines and n columns that contains integer values between 1 and $n \times n$, each value appearing only once and its position being random. The value of n is directly set in the program with for instance : `int n = 3;`

We are now focusing on square grids that are **magic** : each number between 1 and $n \times n$ is included only once, and the sum of the numbers is the same in each line, column and diagonal. For instance, the following grid is magic because each sum is equal to 15, and all the numbers from 1 to 9 are present in the grid :

4	9	2
3	5	7
8	1	6

(Q2.6) Provide the signature and the implementation of the `verifSumLines` method. This method takes as parameter a 2D array of integers and returns the boolean value true if the sum is the same in each line of the array.

(Q2.7) BONUS : Provide the signature and the implementation of the `verifUnicity` method. This method takes as parameter a 2D array of integers and returns a boolean value that is true if values in the array are all different. We assume that all possible values in this array are in the interval 1 to $n \times n$ for a square array of size n (n lines and n columns).

3 Object Oriented Programming (8 points)

We want to develop a `Wallet` class to handle the digital wallet of a client that owns various currencies.

```
1  public class Wallet {
2     // Attributes declaration
3     double[] money;
4     String[] currencies;
5
6     public Wallet(double[] t, String[] d){
7        money = t;
8        currencies = d;
9     }
10
11
12    public void display(){
13       // A REMPLIR
14    }
15
16    public void exchange(double[][] rates, int soldCurrency, int boughtCurrency,
          double amountBought){
17       // A REMPLIR
18    }
19 }
```

The object `Wallet` is instantiated from two arrays representing the currencies, and the corresponding amount of money. Thus, to represent a wallet containing 3.5€, 10.4$ and 6¥, we create the following instance :

```
String[] currencies = { "EUR", "USD", "YEN" };
double[] money = { 3.5 , 10.4, 6 };
Wallet p = new Wallet(money, currencies);
```

(Q3.1) Complete the code of the `display` method of the `Wallet` class such that the instruction `p.display()` displays the following information :

```
The wallet contains 3.5 EUR, 10.4 USD, 6 YEN
```

(Q3.2) Provide the signature and the implementation of the `mergeWallet` method that, given two wallets, creates a new wallet whose content is the sums of the amounts of the two wallets. **We first assume that the currencies are identical.**

We represent the exchange rates for the currencies as a 2D array :

	EUR	USD	YEN
EUR	1	0.8	0.75
USD	1.2	1	0.9
YEN	1.34	1.05	1

For instance, 1 euro is exchanged against 0.8 Dollars with the current exchange rate.

(Q3.3) Write the code that defines the previous array in a variable named `rate`.

We are now interested in the exchange of money from one currency to another. For instance, with the current wallet, we could exchange 2 euros into 1.6 Dollars, and we will then have 1.5 Euros and 12 Dollars in the wallet.

(Q3.4) Complete the code of the `exchange` method so that we obtain the following execution on the example :

```
// 0 is for EUR, 1 for USD in the currency array
p.exchange(rate, 0, 1, 1.6);
p.display();
// the wallet contains 1.5 EUR, 12 USD, 6 YEN
```

One will verify that there is enough sold currency in the wallet before applying the exchange.

(Q3.5) Provide the signature and the code of the `equivalent` method that computes the equivalent of the wallet in a given currency.

For instance, after exchange, the wallet contains 1.5€ that are worth 1.2$, 6¥ that are worth 6.3$, and 12$, thus a total of 19.5$. The corresponding execution will be :

```
// 1 is the index of USD in the currency table
p.equivalent(rate, 1);
// -> the wallet contains the equivalent of 19.5 USD
```

With the exchange rates presented above, we can exchange 1¥ against 1.34€, then these 1.34€ into 0.75*1.34 = 1.005 ¥, thus allowing us to have more money than at the beginning.

(Q3.6) Provide the signature and the code of the `easyMoney` method of the `Wallet` class that finds if there is an exchange that can increase the amount of money. The method will display such an interesting exchange if it exists, by displaying "you must exchange EUR into YEN!" for example; and "no interesting exchange" if no such exchange exists. We assume that there is at most one advantageous exchange possible.

(Q3.7) BONUS : Provide a new version of the `mergeWallet` method that does not assume that currencies are in the same order.