NFA031 : Corrigé de l'examen – Session 2, avril 2015

Avril 2015

Exercice 1 — *QCM (2,5 points)*

— **Question 1** : réponse **1a**
— **Question 2** : réponse **2c** (si l'on trouve l'élément à une place donnée, le booléen `appartient` est bien mis à vrai, mais les itérations suivantes de la boucle sont susceptibles d'écraser cette affectation).
— **Question 3** : réponse **3c**
— **Question 4** : réponse **4b**
— **Question 5** : réponse **5a**

Exercice 2 — *Exécution de programme (3 points)*

ligne	`x`	`st`	`b`	`i`	**Test**	**Écran**
3	0	NEP	NEP	NEP		
4	0	"x"	NEP	NEP		
5	0	"x"	true	NEP		
6.init	0	"x"	true	1		
6.test	0	"x"	true	1	true	
7	0	"x 0"	true	1		
8	0	"x 0"	true	1	true	
9	10	"x 0"	true	1		
11	10	"x 0"	false	1		
6.incr	10	"x 0"	false	2		
6.test	10	"x 0"	false	2	true	
7	10	"x 0 10"	false	2		
8	10	"x 0 10"	false	2	false	
11	10	"x 0 10"	true	2		
6.incr	10	"x 0 10"	true	3		
6.test	10	"x 0 10"	true	3	true	
7	10	"x 0 10 10"	true	3		
8	10	"x 0 10 10"	true	3	true	
9	20	"x 0 10 10"	true	3		
11	20	"x 0 10 10"	false	3		
6.incr	20	"x 0 10 10"	false	4		
6.test	20	"x 0 10 10"	false	4	false	
13	20	"x 0 10 10"	false	NEP		valeur de x : 20 ↩
14	20	"x 0 10 10"	false	NEP		valeur de st : x 0 10 10 ↩
15	20	"x 0 10 10"	false	NEP		valeur de b : false ↩

Exercice 3 — *Programmes et boucles (4,5 points)*

Question 1

```
public class Exam15Exo3 {
  public static void main(String[] args) {
    // Question 1
    Terminal.ecrireString("Entrez le nombre de lignes à afficher : ");
    int n = Terminal.lireInt();
    for (int i=1; i <= n; i++) {
      if (i % 2 == 0) Terminal.ecrireStringln("**");
      else Terminal.ecrireStringln("****");
    }
  }
}
```

Question 2

```
// Question 2
int k = 1;
for (int i = 1; i <= 10; i++) {
  for (int j = 1; j <= 10; j++) {
    Terminal.ecrireString(k + " ");
    k++;
  }
  Terminal.sautDeLigne();
}
```

Question 3

```
// Question 3
int k = 1;
for (int i = 1; i <= 10; i++) {
  for (int j = 1; j <= 10; j++) {
    if (k < 10) Terminal.ecrireString("  ");
    else {
      if (k < 100) Terminal.ecrireString(" ");
    }
    Terminal.ecrireString(k + " ");
    k++;
  }
  Terminal.sautDeLigne();
}
```

Exercice 4 — *Programme et tableau (5 points)*

Question 1

L'énoncé de la question comportait semble-t-il une erreur. La question correcte était : "*Donnez le code pour créer ce tableau, lire au clavier toutes les* ***températures*** *et ensuite, afficher les* ***31 températures***.". Le code est alors :

```
double[] tab = new double[31];
// Lecture des valeurs
for (int i=0; i < tab.length; i++) {
  Terminal.ecrireString("Entrez la température du jour "+(i+1)+" : ");
  tab[i] = Terminal.lireDouble();
}
// Affichage des valeurs
Terminal.ecrireStringln("Les températures saisies sont les suivantes :");
for (int i=0; i < tab.length; i++) {
  Terminal.ecrireStringln("Jour "+(i+1)+" ==> "+ tab[i]);
}
```

Question 2

```
double[][] temp = new double[12][31];
for (int i=0; i < temp.length; i++) {
  Terminal.ecrireStringln("Mois "+(i+1));
  for (int j=0; j < temp[i].length; j++) {
    Terminal.ecrireString("    Entrez la température du jour "+(j+1)+" : ");
    temp[i][j] = Terminal.lireDouble();
  }
}
```

Question 3

```
public static double temperatureMoyenne(double[][] temp, int mois) {
  double somme = 0.0;
  for (int j=0; j<temp[mois-1].length; j++) {
    somme = somme + temp[mois-1][j];
  }
  return somme / temp[mois-1].length;
}
```

Question 4

Plusieurs solutions sont envisageables. Dans toutes les solutions, on va considérer que le caractère bissextile de l'année ne pose pas de problème, c'est-à-dire que le mois de février a un nombre de jours connu (28 ou 29). Sinon, il faudrait connaître l'année, déterminer si elle est ou non bissextile et calculer le nombre de jours du mois de février. Ceci étant précisé, voici différentes solutions :

Solution 1 : On garde le tableau des températures à deux dimensions, `temp`, surdimensionné (de taille 12×31) et on crée un tableau à une dimension, `nbJours`, qui contiendra le nombre de jours de chaque mois : `nbJours[0] == 31` (pour janvier), `nbJours[1] == 28` (pour février), ... Lorsqu'on parcourt le tableau `temp`, on sait alors que pour parcourir la ligne `i` qui correspond au mois `i`, il convient de parcourir les cases allant de l'indice 0 à l'indice `nbJours[i]`.

Solution 2 : On garde le tableau des températures à deux dimensions, `temp`, surdimensionné (de taille 12×31) et pour un mois `i` de moins de 31 jours, on met dans les dernières cases de la ligne i, correspondant aux jours du mois qui n'existent pas, une valeur ne pouvant pas correspondre à une température (-1000 par exemple). Ainsi, lorsqu'on parcourt le tableau `temp`, si on rencontre une température égale à -1000, on sait que le jour correspondant (`j`) n'existe pas et qu'il ne faut pas prendre en compte la case `temp[i][j]`.

Solution 3 : On utilise un tableau à deux dimensions, `t`, dont toutes les lignes n'ont pas le même nombre de cases. Ceci est possible en java :

```
double[][] t = new double[12][];
    t[0] = new double[31];
    t[1] = new double[28];
    t[2] = new double[30];
    ...
```

On peut alors parcourir le tableau `t` sans plus se soucier du nombre de jours de chaque mois. Plus exactement le nombre de mois est alors `t.length`, et le nombre de jours du mois `i` est `t[i].length` :

```
 for (int i = 0; i < t.length; i++)
   for(int j = 0; j < t[i].length; j++)
        // traitement de la case t[i][j]
```

Et là vous me dites que la réponse fait plus de 10 lignes. À ceci je réponds, gardez uniquement l'une des 3 solutions et écrivez petit.

Exercice 5 — *Méthodes pour les notes d'une classe (5 points)*

1.
```
public static int numCase(String[] tabNoms, String nom) {
  boolean trouve = false;
  int place = -1;
  int i = 0;
  while ((i < tabNoms.length) && !trouve) {
    if (tabNoms[i].equals(nom)) {
      trouve = true;
      place = i;
    }
    else i++;
  }
  return place;
}
```

2.
```
public static double noteEleve(double[] tabNotes, String[] tabNoms, String nom) {
  double note = -1.0;
  int place = numCase(tabNoms, nom);
  if (place != -1) note = tabNotes[place];
  return note;
}
```

3.
```
public static int nbAuDessusDeDix(double[] tabNotes) {
  int nb = 0;
  for (int i = 0; i < tabNotes.length; i++) {
    if (tabNotes[i] >= 10) nb++;
  }
  return nb;
}
```

4.
```
public static String unDesMajors(double[] tabNotes, String[] tabNoms) {
  // On suppose ici qu'il existe au moins un élève, sinon il
  // conviendrait de lever une exception ou de renvoyer une chaîne vide
  double noteMax = tabNotes[0];
  String nomMajor = tabNoms[0];
  for (int i = 1; i < tabNotes.length; i++) {
    if (tabNotes[i] > noteMax) {
      noteMax = tabNotes[i];
      nomMajor = tabNoms[i];
    }
  }
  return nomMajor;
}
```