le cnam
Nouvelle-Aquitaine

Sujet d'examen

Semestre 1 / session 1
Année 2018/2019

CODE UE : M-NFA031-1

INTITULE : Programmation avec Java : notions de base

DATE : vendredi 08 février 2019

HORAIRES : 18h00 à 20h00

DUREE : 2h00

CONSIGNES : Documents autorisés, matériel électronique interdit

NOMBRE DE PAGES : 4 celle-ci comprise

1. Cours (4 points)

1.1. Comment doit-on procéder pour ajouter une valeur à un tableau de base déjà rempli ? La question ne porte pas sur les ArrayList mais bien sur les tableaux de la forme :

```
xxx [] tab;
```

1.2. Dans le bout de code qui suit

```
t = t + tab[i] / 2;
```

a) De quoi s'agit-il ?
b) Quels sont les opérateurs ?
c) Qu'est-ce que tab ?
d) Quel doit être le type de i ?
e) Sachant que tab[i] est de type int, quels sont les types acceptables pour t ?

2. Erreurs (2 points)

2.1. Le programme suivant est une version simplifiée de la correction du premier exercice de l'atelier 4 qui affiche à l'écran les tables de multiplication de 2 à N, N étant donné par l'utilisateur. Dans ce programme des erreurs empêchant la compilation ont été introduites. Donnez-en 7 et proposez une correction pour chacune d'elle.

```
01.   public class TableMultiplication
02.         public static void main(String[] args) {
03.               int N;
04.               Terminal.ecrireString("Nombre maximum : ")
05.               N = Terminal.lireInt;
06.               if (n < 2 || n > 9) {
07.                     System.out.println(N + " : Valeur interdite");
08.                     return -1;
09.               }
10.
11.               for (table = 2; table <= N; table++) {
12.                     Terminal.ecrireIntln(
13.                                 "Table de multiplication par "
14.                                 + table);
15.                     for (int multi = 1; multi <= 9; multi++) {
16.                           Terminal.ecrireStringln("    " +
17.                                       table +  x  + multi
18.                                       + " = " + table * multi);
19.                     }
20.               }
21.         }
22.   }
```

3. Algorithmique (4 points)

3.1. Donner les entrées, les sorties et l'algorithme d'un programme lisant un nombre au clavier et affichant tous ses diviseurs à l'écran.

3.2. L'algorithme suivant permet d'afficher à l'écran le code en base 16 (code hexadécimal) d'un nombre lu au clavier. Le traduire sous forme de programme Java.

Entrées

n : un nombre entier positif.

Sortie

res : le code en base 16 de n sous forme de texte.

Variables

xp : un réel calculant les puissances successives de x initialisé à 1.

Algorithme

```
lire n
initialiser res à la chaîne vide
tant que n > 0
   digit reçoit le reste de la division entière de n
         par 16
   si digit <= 9
      res = (char) ('0' + digit) + res ;(1)
   sinon
      res = (char) ('a' + digit – 10) + res ;(1)
   fin si

   diviser n par 16
fin tant que

écrire res
```

fin de l'algorithme

4. Simulation (2 points)

4.1. La méthode suivante permet de chercher par dichotomie la valeur n prise en paramètre dans un tableau tab trié par ordre croissant, lui aussi, pris en paramètre. En simuler l'exécution pour tab = {2, 3, 5, 7, 9, 11, 13, 17, 19} et n=12. La simulation sera donnée sous forme de tableau comme dans les exercices de la séquence 2.

```
01.  public static boolean cherche (int n, int [] tab) {
02.     int binf = -1;
03.     int bsup = tab.length;
04.
05.     while (bsup > binf+1) {
06.        int milieu = (bsup+binf)/2;
07.        if (tab[milieu] == n)
08.           return true;
09.        if (tab[milieu] < n)
10.           binf = milieu;
11.        else
12.           bsup = milieu;
13.     }
14.     return false;
15.  }
```

[1] A recopier tel que

5. Exercices (8 points)

Dans cette partie, si vous n'arrivez pas à répondre à une question, vous pouvez utiliser la méthode dans les questions qui suivent comme si vous y aviez répondu.

On veut écrire un utilitaire pour un vendeur de bois à la découpe. Les questions qui suivent répondent à certains problèmes posés.

5.1. Ecrire une fonction prenant en paramètre les dimensions entières d'une planche (longueur et largeur) exprimées en cm et le prix au m² du bois. Elle retournera le prix de la planche. Si les dimensions ne sont pas correctes, une exception spécifique sera levée. Attention aussi aux unités : 1m = 100cm et 1m² = 10 000cm² !

5.2. Le prix au m² du bois varie en fonction du type de bois et de son épaisseur. On caractérisera les bois par leur appellation formée du type suivi de leur épaisseur, séparés par un espace. Par exemple, un contreplaqué de 10mm aura pour appellation « contreplaqué 10 ». Pour faire la correspondance entre l'appellation et le prix, on crée deux tableaux coordonnés à une dimension :

- nomBois de type String [ ] et
- prixBois de type double [ ].

Ecrire une fonction qui prend en paramètre le tableau nomBois et une appellation, cherche si l'appellation figure dans le tableau. Elle retournera -1 si elle ne la trouve pas, sinon, elle retournera l'indice où elle a trouvé l'appellation dans nomBois.

5.3. Ecrire une fonction prenant les mêmes paramètres que la précédente plus le tableau prixBois, mais retourne cette fois-ci le prix au m² du bois si l'appellation est correcte. Dans le cas contraire, une erreur spécifique sera levée.

5.4. La commande d'un client sera présentée sous forme de deux tableaux. Les deux tableaux contiendront autant de lignes que le nombre de planches que le client commande. Le premier contiendra l'appellation du bois que le client souhaite, le second formé de deux colonnes contiendra les dimensions de la planche.

Ecrire une fonction qui retourne sous forme de tableau de double le prix de chacune des planches commandées.

5.5. Prévoir l'affichage complet de la commande :

- Une ligne par planche avec : l'appellation du bois, les dimensions, le prix au m² et le prix de la planche
- Le montant total de la commande.