Programmation avec Java: notions de base

NFA031

Deuxième session, avril 2014

Aucun document n'est autorisé. Le barème est donné à titre indicatif.

Exercice 1 : QCM

Attention : les réponses fausses sont pénalisées par le retrait de points. Il peut y avoir plusieurs bonnes réponses.

1. Considérez le code suivant :

```
String s;
int x;
s = Terminal.lireString();
x = s;
```

   (a) ce code est correct

   (b) ce code est incorrect

   (c) ce code peut être correct ou non, selon ce qui est tapé au clavier.

2. Considérez le code suivant :

```
int c =0;
while (c<=6) {
  c = c + 2;
  System.out.println(c+ "␣");
}
```

   Quels sont les affichages de ce code ?

   (a) 0 2 4 6

   (b) 2 4 6

   (c) 2 4 6 8

   (d) 0 2 4 6 8

3. Considérez le code suivant :

```
int c = 1;
while (c<5) {
  System.out.println(c+ "␣");
}
```

Quels sont les affichages de ce code ?

(a) 1 2 3 4

(b) 1 2 3 4 5

(c) 2 3 4

(d) une suite de 1 trop longue pour être imprimée ici.

4. Considérez le code suivant :

```
int c = 1;
while ( ___ ) {
  System.out.println(c+ "␣");
  c = c +1;
}
System.out.println(c+ "␣");
```

Quelle est la condition à employer dans la boucle pour que ce code affiche 1 2 3 4 5 6 7 8 ?

(a) `c < 9`

(b) `c < 8`

(c) `c != 8`

(d) `c <= 8`

5. On suppose que t est un tableau d'entiers qui contient au moins une case. On désire calculer la somme s des cases d'indice impair (t[1] + t[3] + ...). Choisissez le code correct à mettre à la place des blancs dans le listing suivant :

```
int s = 0;
for (int i = ___; i<t.length; __){
    s = s + t[i];
}
System.out.println(s);
```

(a) `int i=0` et `i++`

(b) `int i=1` et `i++`

(c) `int i=1` et `i=i+2`

(d) `int i=0` et `i=i+2`

6. On peut écrire une boucle

```
while(true){
   ...
}
```

si on souhaite que la boucle soit exécutée éternellement et que le programme ne s'arrête jamais.

(a) vrai

(b) faux

7. Considérez le code suivant :

```
public static void lamethode(int x){
   ...
}
```

   (a) il faut lire la valeur du paramètre `x` par un `Terminal.lireInt` au début de la méthode.

   (b) le paramètre `x` reçoit automatiquement la valeur de la variable `x` déclarée dans la méthode main.

   (c) la paramètre reçoit automatiquement le résultat d'un calcul spécifié dans l'appel de la méthode.

8. Considérez le code suivant :

```
int[] t1, t2;
t1 = new int[3];
t2 = t1;
```

   (a) il y a dans ce programme deux noms de tableaux et deux tableaux.

   (b) il y a dans ce programme deux noms de tableaux et un seul tableau.

   (c) il y a dans ce programme un nom de tableau et un tableau.

9. Considérez le code suivant :

```
public static void main(String[] args){
  int[] t={1,2,3};
  void meth(t);
  Terminal.ecrireIntln(t[0]);
}
public static void meth(int[] tab){
  tab[0]=10;
}
```

   Qu'affiche ce programme :

   (a) 1

   (b) 10

   (c) rien, car il n'est pas correct

10. Considérez le code suivant :

```
int x = 9;
if (x > 5){
   Terminal.ecrireString("un ");
}else if (x > 7){
   Terminal.ecrireString("deux ");
}
```

   Qu'affiche ce programme :

   (a) un

   (b) deux

   (c) un deux

11. Lorsqu'on écrit un programme Java en contexte professionnel, il est préférable de ne pas faire beaucoup de tests car on risquerait de trouver des bugs, ce qui ferait baisser la valeur commerciale du programme.
    (a) vrai
    (b) faux
    (c) cela dépend des termes du contrat avec le client.

Exercice 2 : exécution de programme (3 points)

Considérez le programe suivant :

```
1  public class Exo_1_2014_2{
2
3    public static int pg(int a, int b) {
4      Terminal.ecrireStringln("Debut pg: a= "+ a+" et b= "+ b);
5      while (a!=b) {
6        if (a>b) {
7          a=a-b;
8          Terminal.ecrireStringln(" a= "+ a);
9        } else {
10         b=b-a;
11         Terminal.ecrireStringln(" b= "+ b);
12       }
13     }
14     return(a);
15   }
16
17   public static void main(String[] args) {
18     int a = Terminal.lireInt();
19     int b = Terminal.lireInt();
20     Terminal.ecrireStringln("Resultat pour "+a+" et "+b+" ==> "+pg(a,b));
21   }
22 }
```

Retracez l'exécution du programme suivant en donnant le plus exactement possible les messages affichés lorsque :

1. a= 10 et b = 12 ;
2. a = 6 et b = 9.

Exercice 3 : fonction (2 points)

Ecrire une fonction qui teste si tous les caractères d'un tableau de caractères sont des lettres. Vous pouvez utiliser la méthode `Character.isLetter` qui prend en paramètre un caractère et répond par une valeu booléenne.

Votre méthode ne doit faire aucune entrée/sortie (ni affichage, ni entrée au clavier). Elle doit prendre ses données en paramètres et renvoyer son résultat avec `return`. Elle doit fonctionner quelle que soit la taille du tableau.

Exercice 4 : tableau (2 points)

```
publc class Tabb{
    public static void main(String[] args){
        int[] tab, tabsansmax;
        tab = new int[10];
        for (int i=0; i< 10; i++){
            Terminal.ecrireString("Entrez un nombre: ");
            tab[i]=Terminal.lireInt();
        }
```

Donnez la suite du programme pour qu'il y ait dans `tabsansmax` les mêmes valeurs que dans tab à l'exception de la plus grande.

Par exemple, si tab vaut `{10, 12, 32, 17, 89, 57, 62, 23, 8, 11}`, alors tabsansmax doit prendre la valeur `{10, 12, 32, 17, 57, 62, 23, 8, 11}`.

S'il y a plusieurs occurrences de la valeur la plus grande, une seule de ces occurrences sera absente de `tabsansmax`.

Exercice 5 : affichage (1 point)

Ecrivez un programme qui lit un nombre n au clavier et dessine à l'écran une barre oblique composé de n caractères *. Par exemple, si le nombre vaut 4, le dessin affiché sera :

```
*
  *
    *
      *
```

Exercice 6 : tableau de trains (7 points)

On souhaite enregistrer et consulter les horaires de passage de trains sur les arrêts d'une ligne. Nous donnons un exemple correspondant à une ligne avec 4 arrêts, et où circulent 5 trains. Les données relatives aux horaires sont enregistrés au moyen de 3 tableaux. Dans le tableau `arrets` il y a une ligne par train (5), et une colonne par arrêt (4). La dernière colonne correspond à la station terminus de la ligne. Une case (i,j) contient true si le train numéro i s'arrête à la station numéro j. Ex : le 1er train ne dessert que l'avant dernière station et le terminus.

```
boolean [] [] arrets = {
                {false, false,true,true},
                {false, true,true, true},
                {true, false, false, true},
                {true, true, true, true},
                {false, false,true, true}};

int [] departs = {375, 450, 510, 690, 930};
int [] tempsTrajets = {11,20,7,17}
```

Par ailleurs, le tableau `departs` donne l'heure de départ (en minutes) pour chaque train. Ex : le 1er train part "à 375 minutes", autrement dit, à 6h15 du matin. Le tableau `tempsTrajets`, donne le temps de trajet (en minutes) entre deux arrêts consécutifs. Ex : le trajet gare de départ → 1er arrêt se fait en 11 minutes. Enfin, on considère que le temps passé dans une station où un train s'arrête est toujours de 3 minutes.

1. Ecrivez une méthode qui renvoie le premier numéro de train qui est un omnibus (s'arrête à toutes les stations). La méthode renvoie -1 si aucun train n'est omnibus.
2. Ecrivez une méthode qui affiche les numéros de train et heures de départ de tous les trains qui s'arrêtent à une station donnée.
3. Ecrivez une méthode `minutesToString` qui à partir d'une quantité en minutes calcule la chaîne qui lui correspond au format d'une heure intelligible. Ex : `minutesToString(375)` doit renvoyer la chaîne " 6h15" ;
4. Ecrivez une méthode qui pour un numéro de train donné, affiche son heure de départ, puis celle de passage dans chacun des stations où il s'arrête. Ex : pour le 1er train, qui ne dessert que la dernière station et le terminus on pourra afficher :

```
(train1)  depart: 6h15 - station3: 6h56 - terminus:7h09
```

5. Définissez dans une méthode main les variables données plus haut avec les valeurs de l'exemple. Ensuite vous utilisez impérativement les méthodes écrites afin d'afficher le premier numéro de train qui est un omnibus, puis les horaires de tous les trains, et enfin les horaires complets du premier train qui part après 8h et qui dessert le 2ème arrêt.

Nota bene : toutes vos méthodes prendront en paramètre les données nécessaires : tableaux et autres paramètres. Sauf pour les méthodes d'affichage, elles ne feront pas de lecture ni d'affichage.