Algorithmique et Programmation

ED no. 7 : Exercices sur sous-programmes

V. Aponte

4 décembre 2018

1 Exercice 1 : erreurs

Dans ce bout de code, signalez les erreurs en expliquant precisément la cause (maximum 1 ligne).

```
1  public static int f(int x, int y){
2      int r = x+ y + 1;
3  }
4  public static void p(String x){
5      boolean y = false;
6      System.out.println(x+y);
7  }
8  public static void main(String [] args){
9      boolean x = true;
10     int y = 2;
11     int w = f(y,3);
12     p("cc");
13     w = p("aa");
14     int z = r;
15     int k = f(x,y);
16     y = f();
17 }
```

Exercice 2 : étude de code

Question 1

```
public static void afficheEntre(int a, int b){
  Terminal.ecrireInt(a);
  for (int i=a+1; i<=b; i++){
    Terminal.ecrireString("," +i);
  }
  System.out.println(".");
}
public static void main(String[] args){
   afficheEntre(2,2);
```

```
    afficheEntre(10,20);
  }
```

1. Qu'affiche ce bout de code ?
2. Le sous-programme est il une procédure ou une fonction ?

Question 2

```
  static boolean estMajuscule(char c){
    return ('A' <= c && c <= 'Z');
  }
  static boolean estMinuscule(char c){
    return ('a' <= c && c <= 'z');
  }
  static boolean estLettre(char c){
      return ( estMajuscule(c) || estMinuscule(c));
  }
  public static void main(String[] args) {
    System.out.print("Entrez␣un␣caractere:␣");
    char c1 = Terminal.lireChar();
    if (estLettre(c1)){
      if (estMajuscule(c1))
        System.out.println(c1+"␣est␣une␣lettre␣majuscule");
      else
        System.out.println(c1+"␣est␣une␣lettre␣minuscule");
    } else
        System.out.println(c1+"␣n'est␣est␣pas␣une␣lettre");
  }
```

1. Qu'affiche ce bout de code ?
2. Les sous-programmes sont ils des procédures ou des fonctions ?

Question 3

Modification en place, état au retour. Considérez ce code :

1. Qu'affiche-t-il ?
2. Expliquez ce qui change dans les valeurs des variables du main entre les moments avant et après pour les deux appels de méthodes `changeCase` et `changeRien`. Pourquoi leur différence de comportément ? Soyez précis.

```
public class ChangeCaseOuRien{

  public static void afficheTabInt(int [] t){
   for (int i=0; i<t.length; i++){
     System.out.print("␣"+ t[i] );
   }
   System.out.println();
```

```
}
 public static void changeCase(int indice, int v, int [] t){
   t[indice] = v;
 }
 public static void changeRien(int v){
   v = v+1;
 }
 public static void main(String[] args){
  int [] tab = {1,2,3,4};
  int a = tab[0];

  changeCase(1,a+10,tab);
  System.out.println("Contenu de tab apres appel:");
  afficheTabInt(tab);
  System.out.println("Contenu de a apres appel: "+a);

  System.out.println("  ***** Test 2 ****");
  changeRien(tab[3]);
  System.out.println("Contenu de tab apres appel:");
  afficheTabInt(tab);
}
```

Exercice 3 : variations sur l'appartenance

Sous-programmes simples.

Question 1

Voici le code d'un sous-programme qui teste si un entier appartient à un tableau d'entiers :

```
public static boolean appartient(int v, int [] tab){
  for (int i=0; i<tab.length; i++){
    if (v==tab[i])
      return true;
  }
  return false;
}
```

1. Expliquez son fonctionnement : comment se fait-il qu'il fait un `return` au milieu d'une boucle ?
2. Pourquoi retourne t-il systématiquement false une fois sorti de la boucle ?
3. Ecrivez un programme main qui initialise un tableau, lit un entier puis fait un appel à la fonction pour tester si l'entier lu est dans le tableau. Vois afficherez le message pertinent.

Question 2

Modifiez votre programme de manière à ajouter une nouvelle méthode `indiceTrouve(v,t)` qui prend en argument une valeur à chercher et un tableau et retourne le premier où se trouve v dnas le tableau. La méthode reourne -1 si v ne se trouve pas dans le tableau. Modifiez le programme main : il doit afficher l'indice où une valeur lue se trouve dans le tableau, ou indiquer qu'elle n'a pas été trouvée.

Exercice 4 : méthodes qui retournent un tableau

Pour tous les sous-programmes demandés vous écrirez un main permettant de le tester.

1. Ecrivez ue méthode `sommeTableauxInt` qui prend deux tableaux d'entiers t1 et t2 et retourne un nouveau tableau qui contient la somme de leurs cases prises deux à deux. Les tableaux pourront être de tailles différentes. Exemple : si t1 = {1,3} et t2 = {1,7,8, 9}, la fonction doit retourner {2,10,8, 9}.
2. Ecrire une méthode `sousTabInt` qui prend un tableau t et un indice j supposé valide de t, et retourne un nouveau tableau composé de toutes les cases de t d'indices compris entre j et le dernier de t.
3. Ecrire une fonction `insereTabInt(v,i,t)` qui prend tableau t, un indice j et une valeur v. La fonction retourne un nouveau tableau contenant v à l'indice j ainsi que toutes les cases de t. Exemple : si t= {1,7,8, 9}, alors `insereTabInt(50,1,t)` retourne le tableau {1, 50, 7,8, 9}. Par ailleurs, `insereTabInt(50,4,t)` retourne {1, 7,8, 9, 50}.
4. Ecrire une méthode `enleveTabInt` qui prend tableau t et un indice j supposé valide de t. La fonction retourne un nouveau tableau composé de toutes les cases de t sauf celle d'indice j.

Exercice 5 : méthodes avec modifications en place

Pour tous les sous-programmes demandés vous écrirez un main permettant de le tester.

1. Ecrivez une procédure `void inverseTabInt` qui inverse les cases d'un tableau t passé en paramètre. L'inversion se fera sans le concours d'un tableau supplémentaire.
2. Ecrivez une méthode `void echangeDeuxCases` qui prend un tableau t et deux indices i et j supposés valides dans t, et échange les valeurs des case i et j dans t.
3. Modifiez le sous-programme `inverseTabInt` pour que l'inversion de chaque case se fasse par appel à `echangeDeuxCases`.
4. Ecrire une procédure qui réalise (en place) un déplacement à droite de toutes les cases d'un tableau passé en paramètre.