Laboratoire 2 – Programmation orientée objet
ITI 1521. Introduction à l'informatique II
1-5 Février 2021
Dû en ligne après une semaine du Lab
/10

Objectifs

- Classes et Objets simples
- Constructeurs
- Documentation API
- Tableaux

I. Partie I : Programmation orientée objet

Question 1 : (1 POINT)
Pour la classe `Book` suivante, corrigez quelques petites erreurs et ajoutez une méthode `main` pour créer 2 Objets de type `Book` (2 livres), et faire afficher les auteurs de ces 2 livres.

Lancez l'exécution de la classe `Book`.

/***Classe `Book`***/
```
public class Book {
  // Variables
  private String title, author;

  // Constructeur
  public Book (String a, String t) {
    author = a;
    title = t;
  }

  // Accesseur
  public String getAuthor() {
    return author;
  }
```

Question 2 : (1 POINT)
Ecrivez une classe `TestBook` dans un fichier `TestBook.java`. Cette classe a une seule méthode `main()` qui fait ce que fait la méthode `main()` de la classe `Book`.

Faites exécuter la méthode `main` de la classe `TestBook` .

Question 3 : (1 POINT)
Modifiez la classe `Book`:

- Ajoutez un accesseur pour la variable `title`.
- Ajouter un modificateur pour les variables `author` et `title`.

Supprimez la méthode `main()` de la classe `Book` et compilez le fichier `Book.java`.

Faites exécuter la méthode `main` de la classe `TestBook`.

Question 4 : (1 POINT)

a) Enlevez le seul constructeur de la classe `Book` . Sans ajouter de nouveau constructeur, peut-on quand même créer un nouveau objet de type `Book` dans la méthode `main` ? Si c'est possible, créez un Objet de type `Book` et faites afficher son titre et son auteur.
Remettez le constructeur que vous avez enlevé. Est-ce que le code de la méthode `main` de la question précédente fonctionne toujours?

b) Ajoutez un constructeur pour avoir 2 constructeurs dans la classe :
   a. un sans paramètre,
   b. un qui prend en paramètre l'auteur et le titre du livre,

Utilisez les 2 constructeurs (et d'autres méthodes) pour créer 2 objets `Book` dans la méthode `main` de `TestBook`.

Testez en lançant l'exécution.

Question 5 : (2 POINTS)

a) Dans la classe `Book`, ajoutez une méthode `affiche()` qui affiche une description du livre (auteur et titre). Utilisez `affiche()` dans la méthode `main` de `TestBook`. Ajoutez l'instruction `System.out.println(book)` où `book` désigne un des objets `Book` que vous avez créés. Lancez l'exécution de la classe `TestBook`.

b) Ajoutez une méthode `toString()` qui renvoie une chaîne de caractères qui décrit le livre. Donnez à la méthode `toString()` le même profil que la méthode de même nom de la classe `java.lang.Object` (cherchez dans les API du JDK). Exécutez à nouveau la classe `TestBook`. Voyez ce qui est affiché maintenant par l'instruction `System.out.println(book)`.

`println()` utilise automatiquement la méthode `toString()` de la classe de l'objet qu'il a à imprimer.

Il faut savoir chercher dans la documentation de l'API (javadoc). En partant de la classe `java.lang.System` et en cliquant sur les liens, retrouvez dans la

documentation que `System.out.println(objet)` affiche ce que retourne la méthode `toString` de la classe de `objet`.

Modifiez la méthode `affiche()` pour utiliser `toString()`.

Vérifiez le bon fonctionnement de la méthode en lançant l'exécution de la classe `TestBook`.

II. Partie II : Tableaux

Copier et exécuter le code java suivant. Il implémente l'algorithme de tri par sélection.

Le **tri par sélection** (ou **tri par extraction**) est un algorithme de tri par comparaison. Cet algorithme est simple et son principe est le suivant :

- rechercher le plus petit élément du tableau, et l'échanger avec l'élément d'indice 0 ;
- rechercher le second plus petit élément du tableau, et l'échanger avec l'élément d'indice 1 ;
- continuer de cette façon jusqu'à ce que le tableau soit entièrement trié.

La méthode *trier* déclarée à la ligne 2 implémente l'algorithme de tri ; Cette méthode a un paramètre, nommé *tab*, de type référence vers un tableau d'entiers, int[].
À la ligne 5, la valeur de la variable *min* sera utilisée pour désigner la position du plus petit élément du tableau entre les positions i et tab.length-1.
À la ligne 7, le test de l'énoncé *if* compare le contenu des positions j et *min* du tableau. Si la valeur à la position j est inférieure à celle de la case *min*, on sauvegarde la valeur de j dans la variable *min*, ligne 8. C'est la position de la plus petite valeur jusqu'à maintenant.
Lignes 12-14 : Une fois la boucle imbriquée terminée, *min* contient la position de la plus petite valeur pour le segment du tableau situé entre les positions, i et tab.length-1. Nous interchangeons le contenu des positions i et *min* du tableau.
La méthode principale (*main*) possède un paramètre, nommé *args*, une référence vers un tableau de chaînes de caractères.

```
/*Code Tri.java*/
public class Tri {
public static void trier( int [ ] tab ){
int i , j , min , tmp ;        // déclaration de 4 variables de type entier (int).
for ( i = 0 ; i < tab.length - 1 ; i++) {
        min = i;            // Ligne 5.
        for ( j = i + 1 ; j < tab.length ; j++) {          //ligne 6: boucle for imbriquée
                if (tab[j] <tab[ min ]  ) {                // ligne 7
                min =  j   ;                               // ligne 8
                 }
        }
        tmp = tab[min] ;                  // ligne 12
        tab[min] = tab[ i ] ;             // ligne 13
        tab [ i ] = tmp ;                 // ligne 14
        }
 }
```

```
public static void main ( String [ ] args ) {
 int [ ] notes ;    /*déclaration d'une variable (notes) référence vers un tableau d'entiers*/
 notes = new int [ ] { 125 , 3 , 272 , 5 , 80 , 87 , 74 } ;/* crée un tableau d'entier et l'initialise. */
 trier(notes) ;                          /*Appel à la méthode trier pour le tableau notes*/
 for ( int i =0; i<notes.length ; i++) {
/*Impression du contenu du tableau trié*/
          if ( i >0) {
          System.out.print ( " , " ) ;
           }
           System.out.print (notes [ i ] ) ;
 }
System.out.println ( ) ;
 }
 }
```

Question 6 : (1 POINT)

Écrire un programme `TriCmd` qui prend une liste d'entiers sur la ligne de commande, les recopie dans un tableau d'entiers, trie le tableau en ordre croissant puis finalement affiche le contenu du tableau (vous pouvez vous inspirer de la méthode *trier* ci-dessus).

Exemple de sortie :
>javac TriCmd.java
>java TriCmd 15 2 0 40
Le tableau trié est : {0,2,15,40}

Question 7 : (3 POINTS)
Nous allons effectuer quelques opérations simples sur des tableaux :

a) Écrire un programme nommé *Tab* (classe *Tab*) pour déclarer les tableaux suivants :
```
int tab1 [ ] = { 3,55,7, 1, 88, 9 , 4, -10 };
int[] tab2;
tab2 = new int[]{10,34,62,56,82,7,95};
```

Implémenter dans la classe *Tab*, une méthode *trier()* qui trie les éléments d'un tableau dans l'ordre croissant (utiliser la méthode trier précédente). Appeler cette méthode pour les tableaux tab1 et tab2 dans le programme *Tab* et afficher les éléments triés des deux tableaux.

b) Y ajouter une méthode *moyenne*() qui calcule et affiche la valeur moyenne d'un tableau. La méthode *moyenne* possède un seul paramètre, c'est la référence d'un tableau dont les éléments sont de type int. La méthode doit retourner la moyenne des valeurs du tableau. Vous devez modifier la méthode principale (main) pour y insérer un appel à la méthode *moyenne* pour le tableau tab2.
Vous devrez sauvegarder la valeur de retour dans une variable de type int. Finalement, vous devez afficher le résultat.

c) Nous souhaitons maintenant créer un nouveau tableau au moment de l'exécution (dynamiquement), comme *tab2* ci-haut. Ce tableau sauvegardera des valeurs de type int. Le tableau, désigné par la variable référence *petitTab*, contiendra toutes les valeurs qui sont inférieures à la valeur *moyenne* d'un tableau. Pour créer dynamiquement un tableau, nous devrons en connaître la taille exacte (size). Nous déclarons d'abord une variable référence afin de le désigner :

int[] petitTab;

Ensuite, lorsque la taille est connue, nous créeons le tableau comme suit :

petitTab = new int[size];

Nous devons connaître le nombre d'éléments d'un tableau ayant une valeur inférieure à une valeur donnée. Pour se faire, nous allons créer la méthode *count*, possédant deux paramètres, la référence d'un tableau de valeurs tab, et la valeur donnée val :

public static int ***count****(int[] tab, int val)*

Cette méthode retourne un entier (int) qui représente le nombre d'éléments du tableau désigné par *tab* dont la valeur est strictement inférieure à une valeur donnée *val*.

On appliquera cette méthode par la suite aux paramètres actuels *tab2* donné ci-haut et sa valeur *moyenne*.
Nous aurons ainsi le nombre d'éléments qui doivent être sauvegardés dans le tableau désigné par la variable référence *petitTab*. Nous pourrons ainsi créer le tableau dynamiquement.

Finalement, nous devrons créer la méthode *split* qui aura trois paramètres :

public static void ***split****(int[] tab,int[] petit, int val)*

Les deux premiers paramètres sont des références vers des tableaux de type *int*, le dernier paramètre est de type *int* . Le but de la méthode est de sauvegarder dans le tableau désigné par *petitTab* toutes les valeurs du tableau désigné par *tab* dont la valeur est inférieure à sa valeur moyenne (sera copiée dans le paramètre *val*).
Modifier le programme *Tab* pour y inclure les méthodes *count* et *split* décrites ci-dessus. Il faudra aussi modifier la méthode principale afin de calculer la moyenne, compter le nombre d'éléments inférieurs à la moyenne, créer le tableau nécessaire *petitTab*, et appeler la méthode *split* afin de remplir petitTab. Finalement, vous devez imprimer les valeurs de ces deux tableaux.

Exemple de sortie :
Les éléments de tab1 et tab2 triés sont :
-10 , 1 , 3 , 4 , 7 , 9 , 55 , 88
7 , 10 , 34 , 56 , 62 , 82 , 95
La valeur moyenne de tab2 est 49
Les valeurs de petitTab2 sont : 7, 10, 34

Créer et soumettre un fichier zip comme d'habitude (Q1, …Q7)

Resources

https://docs.oracle.com/javase/tutorial/reallybigindex.html
https://docs.oracle.com/javase/tutorial/getStarted/application/index.html
https://docs.oracle.com/javase/tutorial/getStarted/cupojava/win32.html
https://docs.oracle.com/javase/tutorial/getStarted/cupojava/unix.html
https://docs.oracle.com/javase/tutorial/getStarted/problems/index.html
http://docs.oracle.com/javase/8/docs/api/overview-summary.html
http://docs.oracle.com/javase/8/docs/api/java/lang/package-summary.html
http://docs.oracle.com/javase/8/docs/api/java/lang/String.html