

JAVA NOTES

Table of Contents

1. Java est un langage.....	1
2. Quelle phrase au sujet de Java est correcte ?.....	1
3. Que signifient les mots-clés public, private et protected ?.....	2
4. Que signifie le mot-clé void ?.....	2
5. Que signifie le mot-clé return ?.....	3
6. Comment faire une boucle avec foreach ?.....	3
7. Que signifie le mot-clé static ?.....	4
8. Que signifie le mot-clé final ?	6
Design pattern en JAVA.....	6

1. Java est un langage...

☐ Interprété

☐ Compilé

☒ Les deux

☐ Ni l' un, ni l'autre

2. Quelle phrase au sujet de Java est correcte ?

☐ En Java, les types de base ne sont pas des objets

☒ Java supporte l'héritage multiple entre les classes

☐ En Java, on peut affecter soi-même l'adresse d'un pointeur

☐ Java supporte la surcharge des opérateurs

a. Quels sont les différents mots-clés du langage Java ?

public

private

protected

3. Que signifient les mots-clés public, private et protected ?

Ces trois mots-clés du langage Java (public, private, protected) définissent la portée d'une variable, d'une méthode ou d'une classe.

Il existe en fait quatre modificateurs d'accessibilité ; le quatrième est le modificateur dit par package-private. Pour ce dernier modificateur, on n'écrit rien : il n'y a pas de modificateur devant le nom de la variable, de la méthode ou de la classe. Attention, il ne faut pas confondre ce dernier modificateur avec le modificateur public.

Voici les caractéristiques de ces modificateurs, du plus permissif au plus restrictif :

public

Les variables, méthodes ou classes publiques sont accessibles par tout objet. Il ne peut y avoir qu'une seule classe publique par .java et celle-ci doit obligatoirement porter le nom du fichier .java

Les variables, méthodes ou classes définies sans modificateur sont accessibles par toute classe appartenant au même package. Attention : les variables sans modificateur ne sont pas accessibles aux classes filles définies dans un autre package.

protected

Les variables, méthodes ou classes définies comme protégées ne sont accessibles que par les classes filles et classes du même package.

private Les variables, méthodes ou classes définies comme privées ne sont accessibles que par la classe dans laquelle elles sont définies. Il est fortement conseillé de déclarer comme privés tous les attributs d'une classe, et de créer des méthodes de type getter et setter pour y accéder.

Naturellement, toute méthode, variable ou classe est accessible dans la classe où elle est définie.

Remarque : il y a deux cas particuliers où l'absence de mot-clé de visibilité ne correspond pas à une visibilité package-private : tous les membres (attributs et méthodes) d'une interface ou d'une annotation sont obligatoirement public ; tous les constructeurs d'une Enum sont obligatoirement private.

4. Que signifie le mot-clé void ?

Le mot-clé `void` (avec un petit `v`) signifie littéralement « rien » ou « néant ». Ce mot-clé indique qu'une méthode ne retourne aucune valeur et sert donc à distinguer les méthodes qui ne retournent pas de valeur (appelées « procédures » dans d'autres langages de programmation) de celles qui en retournent une (appelées « fonctions » dans d'autres langages).

5. Que signifie le mot-clé `return` ?

Le mot-clé `return` permet de terminer une méthode et de la quitter. Lorsqu'une méthode ne retourne pas de valeur (déclarée `void`), le mot-clé `return` doit être utilisé tel quel sans spécifier de valeur. Lorsque cette méthode retourne une valeur, ce mot-clé doit être suivi d'une valeur de retour ; cela permet de faire remonter ce résultat dans la méthode appelante. Combiné avec des tests, le mot-clé `return` peut être utilisé pour sortir prématurément d'une méthode.

6. Comment faire une boucle avec `foreach` ?

Depuis Java 1.5, il est possible d'utiliser le mot-clé `for` dans une boucle nommée `enhanced for loop` (boucle `for` améliorée) également appelée `for-each` dans d'autres langages. Ce type de boucle effectue une itération automatique sur un ensemble de valeurs ; il n'est donc pas besoin de manipuler un indice manuellement comme dans le `for` classique. Si un conteneur de données est un tableau ou hérite de l'interface `java.lang.Iterable<T>`, il est possible de faire une boucle `for-each` en utilisant la syntaxe suivante :

```
Code Java :  
for (valeur : conteneur) {  
    // Code à exécuter dans la boucle.  
}
```

Par exemple :
Code Java :

```
final int[] valeurs = { 1, 2, 3 };  
for (final int valeur : valeurs) {  
    System.out.println(valeur);  
}
```

Ou encore :

Code Java :

```
final List<Integer>valeurs = Arrays.asList(1, 2, 3);
for (final int valeur : valeurs) {
    System.out.println(valeur);
}
```

Ces deux bouts de code auront comme sortie :
Code Console :

Ici, dans le premier code valeurs est un tableau d'entiers tandis que dans le second bout de code c'est un objet de type List<Integer> qui étend donc l'interface Iterable<Integer>. Il est donc possible d'utiliser la syntaxe for-each dans les deux cas.

7. Que signifie le mot-clé static ?

Devant une variable ou méthode :

Le mot-clé static devant une variable (ou méthode) indique que celle-ci n'appartient pas à une instance particulière de la classe. Les variables ou méthodes statiques appartiennent à la classe elle-même. On peut ainsi les utiliser sans avoir une instance créée. De nombreuses classes ont des membres ou méthodes statiques. Par exemple la classe java.lang.Math :

Code Java :

```
System.out.println(Math.PI);
// Affiche la valeur de PI.
System.out.println(Math.abs(-1));
// Affiche la valeur absolue de -1.
Voici quelques remarques :
```

on peut aussi manipuler une variable ou méthode statique à partir d'une instance de la classe ;
pour rendre des variables statiques comme des constantes, il faut combiner le mot-clé static avec le mot-clé final ;
les méthodes statiques, étant indépendantes de toute instance, n'ont pas accès aux variables ou méthodes non statiques.

Devant un bloc de code :

Le mot-clé `static` devant un bloc de code indique que celui-ci ne sera exécuté qu'une fois. L'exécution se fait lors du chargement de la classe par le `ClassLoader`. On peut utiliser ces blocs, par exemple, pour initialiser des variables statiques complexes.

Code Java :

```
public class MaClasse {  
    public static Map<String, String> uneVariableStatique = new HashMap<String, String>();  
    static{  
        // Initialisation du contenu de uneVariableStatique lors du chargement de la classe.  
        uneVariableStatique.put("une clef","une valeur");  
        uneVariableStatique.put("une autre clef","une autre valeur");  
        // Etc .  
    }  
}
```

Devant une classe interne :

Pour plus d'informations sur ce cas, reportez-vous à la QR Quels sont les différents types de classes internes (nested classes) ? .

Le mot-clé `static` est utilisable pour des variables, méthodes, classes internes ou blocs de code.

1. Puis-je utiliser des méthodes statiques dans une interface ?

Depuis le JDK8, les méthodes statiques sont autorisées dans les interfaces. Dans les versions antérieures, seule la définition de variables statiques était autorisée.

Dans l'exemple ci-dessous, une interface `Person` déclare une méthode statique `sayHello()`.

Code Java :

```
interface Person {  
    static void sayHello() {  
        System.out.println("Hello there!");  
    }  
}
```

8. Que signifie le mot-clé final ?

Le mot-clé final est utilisable pour des variables, méthodes, classes, classes internes ou des classes internes statiques.

Devant une méthode :

On indique que cette méthode ne pourra plus être redéfinie dans une classe fille. Ce qui entraîne une certaine optimisation dans les appels à cette méthode.

Design pattern en JAVA

Design Pattern

Patron de conception Creation - Fabrique - Singleton - Builder Patron de conception Structurel - Adapter - Proxy - Composite Patron de conception Comportemental - Strategy - Iterator - Template

Patron de conception Creation - Fabrique: Creation d'un objet dont le type depend du context - Singleton: Creation sera une seule instance d'objet - Builder: crostruir un objet en fonction de son contexte Patron de conception Structurel - Adapter: adaptation d'un objet à partir d'un autre - Proxy: Representation d'un objet à partir d'un autre - Composite: composition d'objet Patron de conception Comportemental - Iterator - Template - Strategy