Langage Java

Table des matières

Le langage Java est un langage de programmation informatique orienté objet dont la particularité est que les logiciels écrits dans ce langage sont très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX ou Windows, avec peu ou pas de modifications.

1. Premiers pas

1.1. Introduction

Le langage Java reprend en grande partie la syntaxe du langage C++, mais épuré des concepts du C++, tels que les pointeurs et références, ou l'héritage multiple contourné par l'implémentation des interfaces. Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.).

Java a donné naissance à un système d'exploitation (JavaOS), à des environnements de développement (eclipse/JDK), des machines virtuelles (JRE) applicatives multiplate-forme (JVM), une déclinaison pour les périphériques mobiles/embarqués (J2ME), une bibliothèque de conception d'interface graphique (AWT/Swing), des applications lourdes (Oracle SQL), des technologies web (servlets, applets), etc. La portabilité Java est assurée par la machine virtuelle Java, et éventuellement par des bibliothèques standard incluses dans un JRE.

1.2. Mon premier programme

Nous allons écrire un programme qui affichera « Hello world! ».

Tous les programmes Java sont composés d'au moins une classe. Elle doit contenir une méthode appelée main : ce sera le point de démarrage de notre programme.

```
public class myclass {
	public static void main(String[] args){
		// pour ajouter un saut de ligne, utilisez la méthode println
		System.out.print("Hello World !");
		}
}
```

System : appel de la classe « System » . C'est une classe utilitaire qui permet surtout d'utiliser l'entrée et la sortie standard, c'est-à-dire la saisie clavier et l'affichage à l'écran.

out : objet de la classe System qui gère la sortie standard.

print : méthode qui écrit dans la console le texte passé en paramètre (entre les parenthèses).

1.3. Les commentaires

Identique au langage C/C++.

2. Les variables et les opérateurs

Identique au langage C/C++ à trois exceptions près :

- Le type byte (1 octet) qui correspond au type char.
- Le type boolean qui ne peut contenir que deux valeurs : true (vrai) ou false (faux), sans guillemets.
- Le type String qui permet de gérer les chaînes de caractères (String n'est pas un type de variable, mais un objet).

3. La classe Scanner

Pour que Java puisse lire ce qui est tapé au clavier, il va falloir utiliser un objet de type Scanner qu'il faudra instancier. Pour faire ceci, il faut importer la classe Scanner qui se trouve dans le package java.util. Un package est un ensemble de dossiers et de sous-dossiers contenant une ou plusieurs classes.

```
//Ceci importe la classe Scanner du package java.util
import java.util.Scanner;
//Ceci importe toutes les classes du package java.util
import java.util.*;

public class myclass {
	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);

		// saisie d'un texte
		System.out.print("Veuillez saisir un mot : ");
		String str = sc.nextLine();
		System.out.println("Vous avez saisi : " + str);

		// saisie d'un nombre entier
		System.out.print("Veuillez saisir un nombre : ");
		int nbr = sc.nextInt();
		System.out.println("Vous avez saisi le nombre : " + nbr);
		}

}
```

Pour récupérer un type de variable, il vous suffit d'appeler next<*Type de variable commençant par une majuscule*> (convention de nommage Java).

Remarque : le type char n'est pas pris en compte par la classe Scanner. Pour cela, utilisez la méthode charAt(0) de la classe String qui renvoie le premier caractère de la chaîne.

4. Les conditions

Identique au langage C/C++.

5. Les boucles

Identique au langage C/C++.

6. Les tableaux

Identique au langage C/C++ à quelques exceptions près :

6.1. Déclarer un tableau

1. L'objet String, s'initialise de la façon suivante :

   String tableauChaine[] = {"chaine1", "chaine2", "chaine3" , "chaine4"};

2. Il existe deux méthodes pour initialiser un tableau vide :
   - int tableauEntier[] = new int[6];

- int[] tableauEntier = new int[6];

6.2. Parcourir un tableau

Pour accéder à un élément du tableau, il suffit de donner son indice dans le tableau (la méthode length donne la taille du tableau).

```
//Ceci importe la classe Scanner du package java.util
import java.util.Scanner;

public class myclass {
	public static void main(String[] args){
		int   i;
		char  tableau[] = {'a', 'b', 'c', 'd', 'e', 'f'};
		Scanner sc = new Scanner(System.in);

		// On répète tant que la lettre ne figure pas dans le tableau
		do {
			i = 0;
			System.out.println("Rentrez une lettre en minuscule : ");

			char carac = sc.nextLine().charAt(0);
			// Boucle de recherche dans le tableau
			while (i < tableau.length && carac != tableau[i])
				i++;

			// Si i < 7, la caractère a été trouvé
			if (i < tableau.length)
				System.out.println("La lettre " +carac+ " est correcte");
			else //Sinon
				System.out.println("Lettre " +carac+ " incorrecte");

		} while (i >= tableauCaractere.length);
	}
}
```

Il existe une autre syntaxe pour parcourir un tableau depuis la version du JDK 1.5 (cette syntaxe ne fonctionnera pas sur les versions antérieures au JDK 1.5) :

```
public class myclass {

	public static void main(String[] args){
		String tab[][]={{"pim", "pam", "poum"}, {"1", "2", "3", "4", "5"}};
		parcourirTableau(tab);
	}

	static void parcourirTableau(String[][] tab)
	{
		// syntaxe valable uniquement depuis JDK 1.5
		for (String tab2[] : tab)
		{
			for (String str : tab2)
				System.out.println(str);
		}
	}
}
```

7. Les méthodes de classe

Il existe énormément de méthodes dans le langage Java. Elles sont équivalentes aux fonctions des langages qui ne sont pas orientés objet.

Voici quelques méthodes utiles concernant les chaînes de caractères :

Méthode	**description**
toLowerCase()	permet de transformer tout caractère alphabétique en son équivalent minuscule.
toUpperCase()	transforme une chaîne de caractères en capitales.
length()	renvoie la longueur d'une chaîne de caractères (en comptant les espaces).
equals()	permet de vérifier (donc de tester) si deux chaînes de caractères sont identiques.
charAt()	méthode d'extraction de caractère sur un objet String.
substring()	extrait une partie d'une chaîne de caractères.
indexOf()	explore une chaîne de caractères à la recherche d'une suite donnée de caractères.
lastIndexOf()	explore en partant de la fin.

Les méthodes suivantes nécessitent la classe Math, présente dans java.lang. Elle fait donc partie des fondements du langage. Par conséquent, aucun import particulier n'est nécessaire pour utiliser la classe Math qui regorge de méthodes utiles :

Méthode	**description**
random()	Retourne un nombre aléatoire compris entre 0 et 1.
sin(), cos(), tan()	fonctions sinus, cosinus, tangente
abs()	valeur absolue (retourne le nombre sans le signe)
pow()	fonction exposant

7.1. Créer sa propre méthode

Il en existe trois grands types de méthodes :

- celles qui ne renvoient rien. Les méthodes de ce type n'ont pas d'instruction return, et elles sont de type void.
- celles qui retournent des types primitifs (double, int etc.). Elles sont de type double, int, char etc. Celles-ci possèdent une instruction return.
- celles qui retournent des objets. Par exemple, une méthode qui retourne un objet de type String. Celles-ci aussi comportent une instruction return.

Voici un exemple de code :

```
public class myclass {

	public static void main(String[] args){
		String[] tab = {"pim", "pam", "poum"};
```

```
            parcourirTableau(tab);
            System.out.println(toString(tab));
            }

      // cette méthode ne retourne rien
      static void parcourirTableau(String[] tab)
      {
            int i;

            // parcours du tableau
            for (i=0; i < tab.length; i++)
                  System.out.println(tab[i]);
      }

      // cette méthode retourne un objet String
      static String toString(String[] tab)
      {
            int i;
            String retour = "";

            // parcours du tableau
            for (i=0; i < tab.length; i++)
                  retour += tab[i] + "\n";

            System.out.println("Méthode toString() !\n----------");

            return retour;
      }
}
```

7.2. La surcharge de méthode

La surcharge de méthode consiste à garder le nom d'une méthode (donc un type de traitement à faire) et à changer la liste ou le type de ses paramètres.

Exemple :

```
// parcours d'un tableau de chaînes de caractères.
static void parcourirTableau(String[] tab)
{
      for (String str : tab)
            System.out.println(str);
}
// parcours d'un tableau d'entiers.
static void parcourirTableau(int[] tab)
{
      for (int str : tab)
            System.out.println(str);
}
```

JVM se charge d'invoquer l'une ou l'autre méthode : vous pouvez donc créer des méthodes ayant le même nom, mais avec des paramètres différents, en nombre ou en type.

8. Les objets

Les objets sont des variables plus complexes qui peuvent être constitués d'une ou plusieurs autres variables (attributs) et d'une ou de plusieurs fonctions (méthodes). Ainsi, un objet peut lui-même

contenir un objet ! Un objet peut représenter absolument ce que l'on veut : une chaise, une voiture, un concept philosophique, une formule mathématique, etc.

Pour cela, il faut déclarer ce qu'on appelle une classe et lui donner un nom.

Exemple :

```
// déclaration de la classe Voiture
class Voiture {
	int		roue = 4;		// le nombre de roues
	float		vitesse;		// On ne connaît pas la vitesse
	Couleur		carrosserie;		// la couleur est représentée par une classe
	}
```

8.1. Les constructeurs

Parmi les différents types de méthode, il existe un type particulier : les constructeurs. Ces constructeurs sont des méthodes qui construisent l'objet désigné par la classe. Un constructeur porte le nom de la classe.

Exemple :

```
// déclaration de la classe Voiture
class Voiture {
	int		roue = 4;		// le nombre de roues
	float		vitesse;		// On ne connaît pas la vitesse
	Couleur		carrosserie;		// la couleur est représentée par une classe

	// Ce constructeur prend en paramètre la couleur de la carrosserie
	Voiture(Couleur ma_carosserie) {
		// Quand on construit une voiture, elle a une vitesse nulle
		vitesse		= 0;
		carrosserie = ma_carosserie;
		}
	}
```

On construit une voiture avec cette syntaxe : Voiture v = new Voiture(rouge);

Construire un objet s'appelle l'instanciation.

8.2. L'héritage

Il existe certains objets dont l'instanciation n'aurait aucun sens. Par exemple, un objet de type Véhicule n'existe pas vraiment dans un jeu de course. En revanche il est possible d'avoir des véhicules de certains types, par exemple des voitures ou des motos. Une moto doit avoir deux roues et une voiture doit en avoir 4 : dans les deux cas elles ont des roues. Dans les cas de ce genre, c'est-à-dire quand plusieurs classes ont des attributs en commun, on fait appel à l'héritage. Quand une classe A hérite d'une classe B, on dit que la classe A est la fille de la classe B et que la classe B est le parent (ou la superclasse) de la classe A.

Exemple :

```
// Classe qui ne peut pas être instanciée
abstract class Vehicule {
	int	nombre_de_roues;
	float vitesse;
	}
```

```
// Une Voiture est un Vehicule
class Voiture extends Vehicule {
	}

// Une Moto est aussi un Vehicule
class Moto extends Vehicule {
	}

// Un Cabriolet est une Voiture (et par conséquent un Véhicule)
class Cabriolet extends Voiture {
	}
```

Le mot-clé abstract signifie qu'une classe ne peut être instanciée.

Une méthode peut aussi être abstract, auquel cas il n'est pas besoin d'écrire son corps. En revanche, toutes les classes héritant de la classe qui contient cette méthode devront décrire une implémentation de cette méthode.

Pour contrôler les capacités des classes à utiliser les attributs et méthodes les unes des autres, il existe à trois niveaux d'accessibilité :

- public, pour qu'un attribut ou une méthode soit accessible à tous.
- protected, pour que les éléments ne soient accessibles qu'aux classes filles.
- private, pour que les éléments ne soient accessibles à personne si ce n'est la classe elle-même.

Exemple :

```
// Cette classe est accessible à tout le monde
public abstract class Vehicule {
	// attribut accessible à toutes les filles de la classe Vehicule
	protected int	roue;

	abstract private void decelerer();  // Personne n'a accès à cette méthode.
	}
```

Remarque : il existe un type de classe mère particulier appelé interfaces. Une interface est impossible à instancier et toutes les classes filles de cette interface devront instancier les méthodes de cette interface — elles sont toutes forcément abstract.

```
//Interface des objets qui peuvent voler
interface PeutVoler {
	void décoller();
	}

class Avion extends Vehicule implements PeutVoler {
	//Implémenter toutes les méthodes de PeutVoler
	//et les méthodes abstraites de Vehicule
	}
```

TEST

1 - Qu'est-ce qu'un IDE ?

- Un outil permettant de développer des programmes.
- Un environnement permettant d'exécuter des programmes écrits en Java.
- Une instruction du langage Java.
- Un site web référençant les fonctionnalités d'un langage de programmation.

2 - Qu'est-ce qu'un JRE ?

- Un outil permettant de développer des programmes.
- Un environnement permettant d'exécuter des programmes écrits en Java.
- Une instruction du langage Java.
- Un site web référençant les fonctionnalités d'un langage de programmation.

3 - Pouvez-vous développer un programme Java sans IDE ?

- Non
- Oui, avec Word
- Oui, avec n'importe quel éditeur de fichier texte (Notepad++, bloc note Windows…)
- Oui, avec Netbeans

4 - Qu'est-ce qu'une variable ?

- Un conteneur permettant de stocker des données (entier, caractère, booléen…)
- Une instruction du langage Java
- Une entité permettant de manipuler des nombres
- Une entité permettant de manipuler des caractères

5 - Ce code est-il correct ?

```
int i = 10;
int j = 12
int k = 0;
k = (i*i)*(j*j)/j+i;
```

- Oui, rien ne manque.
- Non, les noms de variables doivent avoir au moins deux caractères.
- Non, la variable k doit impérativement être de type double.
- Non, une des variables est mal déclarée !

6 - Que vaut la variable result :

```
double i = 10;
double j = 3;
int result = i / j;
```

- 3.333333333333333333...

- 3.4
- 3
- Rien, il y a une erreur dans le programme...

7- Qu'est-ce qui ne va pas ici :

string str = 'ma chaîne de caractères';

- Les caractères accentués sont interdits dans le type String.
- Il s'agit de l'objet String et non string.
- Il faut initialiser la variable avec des double quotes ("") et non des single quotes ('').
- Il s'agit de l'objet String et non string.

8 - Que devez-vous utiliser pour récupérer les saisies clavier ?

- La classe scanner
- La classe sanner
- La classe Sanner
- La classe Scanner

9 - Que faut-il absolument faire lorsque vous utilisez des objets présents dans des packages autres que java.lang ?

- Rien de particulier
- Il faut importer tout le package.
- Il faut importer la classe dont vous avez besoin.

10 - Qu'est-ce qui ne va pas, ici ?

```
Scanner sc = new Scanner(System.in);
System.out.println("Veuillez saisir un entier : ");
double d = sc.nextInt();
System.out.println("Vous avez saisi le nombre : " + d);
```

- Rien.
- Tout va bien.
- L'objet Scanner est mal initialisé.
- Il y a une incohérence entre la variable d et le type de retour de l'objet Scanner.
- La méthode nextInt() n'existe pas...

11 - Quelle instruction peut-on utiliser avec une condition du type if ...else afin de rajouter des conditions ?

- then
- elseif
- else if
- then if
- if else

12 - Que va retourner ce programme ?

```
int nbre = 999;
if (nbre < 1000 && nbre > 10000)
        System.out.println("c'est bon !");
else
        System.out.println("ce n'est pas bon");
```

- C’est bon !
- Ce n’est pas bon !

13 - Quel sera le résultat de ce code :

```
int a = 10, b = 20;
int max = (a < b) ? ((b < 20) ? b * 2 : ((b > 20) ? b % 3 : b / 4) ) : ((a == 10) ? a / 2 : a % 3);
```

- 5
- 10
- 20
- 30
- 40

14 - Quelle sera la valeur de la variable nbre après ces boucles ?

```
int i = 0, nbre = 0;
while(i <= 9)
 {
    for (int j = 0; j < 10; j++)
      nbre++;

    i++;
}
System.out.print(nbre);
```

- 10
- 20
- 50
- 100

15 - Quel indice d’un tableau permet de récupérer son premier élément ?

- 0
- 1
- 2
- x

16 - Qu’est-ce qui ne va pas ici :

```
int tableau{} = ['1','2','3','3','3'];
```

- Rien du tout

- Les crochets sont à utiliser sur la variable et les accolades dans l'initialisation du tableau.
- Des caractères sont mis à la place des entiers !

17 - Cette déclaration est-elle correcte ?

```
int entier [] [] = {{1,2,3,4,5}{1,2,3,4,5}};
```

- Oui
- Non, la variable est mal déclarée : il faut utiliser des double.
- Non, les deux tableaux ont la même taille : c'est interdit !
- Non, il manque une virgule entre l'initialisation des deux tableaux !

18 - Que va afficher ce programme :

```
String tab[][] = {{"pim", "pam", "poum"}, {"chapi", "chapo", "tralala"}};
for (String str[] : tab)
{
        for(String str2 : str)
        {
                System.out.println("La valeur est  = " + str2);
        }
}
```

- Rien, il y a une erreur dans le code.
- Seulement le premier tableau
- Seulement le deuxième tableau
- L'intégralité du tableau bidimensionnel

19 - Que faut-il faire pour surcharger une méthode ?

- Modifier le nombre ou le type de ses paramètres.
- Modifier le type de retour de la nouvelle méthode.
- Modifier le nom de la méthode.