Introduction à Java

Équipe Pédagogique Informatique 2019

Janvier 2019
Version 1.0

Table des matières

Objectifs

À la fin de cette leçon, l'apprenant sera capable de :

- Présenter le fonctionnement de Java
- Identifier les caractéristiques du langage Java
- Définir les types primitifs et les variables
- Réaliser des opérations arithmétiques en java
- Écrire du code java pour interagir avec un utilisateur

Introduction

Programmer en Java nécessite d'abord la compréhension de certains mécanismes de base qui régissent ce langage. Dans cette leçon, l'apprenant aura connaissance des notions de base qui lui permettront de comprendre le fonctionnement du Langage Java et de mettre en œuvre des petits programmes capables d'interagir avec un utilisateur.

Les concepts de base du langage Java

Objectifs

- Présenter le fonctionnement de Java
- Identifier les caractéristiques du langage Java
- Définir les types primitifs et les variables
- Réaliser des opérations arithmétiques en java

1. Présentation du langage Java

Définition et caractéristiques

- Java est un *langage de programmation* à usage général, de *haut niveau* et *orienté objet* dont la *syntaxe est proche de celle du langage C*. Il se situe à la frontière des *langages compilés* et des *langages interprétés*.
- Java est *multiplate-forme* (portable), c'est à dire que le même programme écrit en langage Java une seule fois fonctionne sur toutes les plate-formes existantes (Windows, Linux, Mac OS, etc.).

2. Architecture et fonctionnement

Architecture d'exécution d'un programma java

Java est un langage de haut niveau ; de ce fait, il ne peut être exécuté directement par l'ordinateur. Tout programme java est au préalable écrit dans un fichier portant *l'extension ".java"*.

Le fichier portant l'extension ".java" est appelé *fichier source*. Le fichier source est *compilé* puis transformé en *bytecode java* qui est ensuite *interprété* par une *machine virtuelle de Java*. Le schéma suivant récapitule le schéma d'exécution

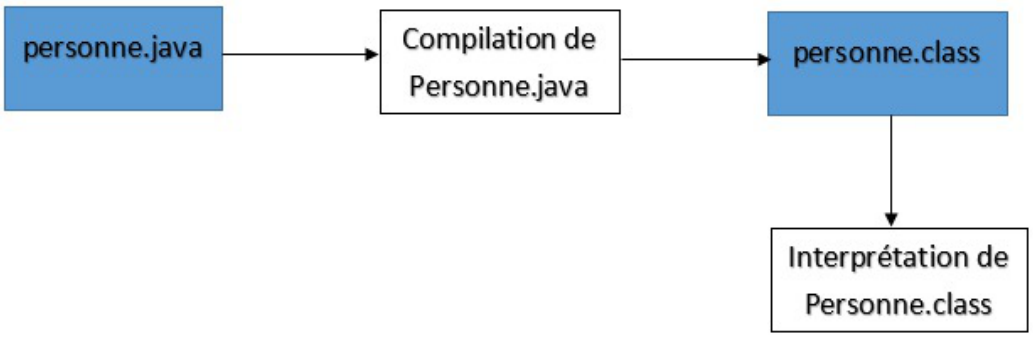

schéma d'exécution d'un programme java

Remarque

Le bytecode java généré après compilation du code source est contenu dans un fichier portant l'extension *".class"*

3. Le typage en java

Variable et type

Java est un langage fortement typé. avant d'utiliser une variable, celle ci doit être déclarée avec précision de son type. Toute déclaration suit la syntaxe suivante :

```
type_variable nom_variable
```

type primitif

Les types primitifs couramment utilisés sont :

- Les entiers : int
- Les flottants ou décimaux : double
- Les chaînes de caractère : String (notez bien le "S" majuscule, cela signifie que les String sont des objets et non de simples variables)
- Les caractères : char
- Les booléens : boolean

Exemple : Déclaration et initialisation de variable

```java
int age=24;
String nom="Adama"
double moyenne=17.5
char genre='F'
boolean majeur=true
```

Attention

- Il est important de noter qu'une variable est associé à un type, et en aucun cas cette variable ne peut recevoir une valeur d'un type différent. par exemple une variable de type String peut recevoir la valeur "Kouadio" mais ne peut jamais recevoir la valeur 10.
- Java est sensible à la casse (majuscule et minuscule).

4. Opérateurs primitifs

Opérations arithmétiques

Les opérateurs arithmétiques de java sont :

- Addition : +
- Soustraction : - (tiret de 6)
- Multiplication : *
- Division : /
- Modulo : % (reste de la division entière)

Exemple

```java
int somme = 1000*2 //la variable somme aura la valeur 2000
double div1 = 5/2  //la variable div1 aura la valeur 2.5
int div2 = 5/2   //la variable div2 aura la valeur 2
int reste = 5%2  //la variable reste aura la valeur 1
```

5. Exercice : Question1

[Solution n°1 p 13]

Pour exécuter une application java sur plusieurs plate-formes, combien de versions du programme le programmeur doit-il écrire ?

- ◯ une version pour chaque plate-forme
- ◯ deux versions seulement
- ◯ seulement une version

6. Exercice : Question2

[Solution n°2 p 13]

Classer les fichiers selon les différentes étapes d'exécution

1. class.java
2. poisson.class
3. oiseau.java
4. oiseau.class

Compilé	Interprété

7. Exercice : Question3

[Solution n°3 p 13]

Identifier les caractéristiques du langage Java

- ☐ Compilé
- ☐ Interprété
- ☐ multiplate-forme
- ☐ ni compilé, ni interprété
- ☐ à la fois compilé et interprété
- ☐ bas niveau et Orienté Objet
- ☐ Portable

8. Exercice : Question4

[Solution n°4 p 13]

Quel type de variable peut-on donner pour spécifier le groupe sanguin d'un individu sans précision des signes "+" et "-"

9. Exercice : Question5

[Solution n°5 p 13]

Quel est la valeur stockée dans la variable resultat ?

```
1 int x=8, y=5;
2 int resultat = x%y
```

10. Exercice : Question6

[Solution n°6 p 13]

Compléter le code suivant pour déclarer une variable de type chaîne de caractère et lui attribuer la valeur Zebié

var = "

Prise en main du langage Java

II

Objectifs

A la fin de cette activité, l'apprenant sera capable de :

- Écrire du code java capable d'interagir avec un utilisateur
- Décrire le processus de création d'un flux de lecture standard

1. squelette d'un programme java

Le code suivant défini le squelette d'un programme java:

```
1 class MonPremierProgramme{
2
3   public static void main(String[] args){
4      //Ecrire ici le programme à exécuter
5   }
6 }
```

Explication

Pour exécuter une application, la classe servant de point d'entrée (c'est-à-dire *MonPremierProgramme*) doit obligatoirement contenir une méthode ayant la *signature* `public static void main(String[] args)`. On dit que la méthode main est le point d'entrée de toute application Java.

La signature du point d'entrée signifie :

- *public* : la méthode est accessible à n'importe quel endroit du programme
- *static* : la méthode peut être exécutée sans créer au préalable une *instance* de la classe contenant la méthode main (dans notre cas MonPremierProgramme)
- *void* : la méthode ne retourne aucune valeur
- *main* : le nom de la méthode. Il est strictement interdit de changer le nom de la méthode sinon le code ne sera pas exécuté

Remarque

La leçon 4 aura pour objectif l'apprentissage du concept de classe

2. Afficher du texte à l'écran

Pour afficher une texte à l'écran en java, on utilise l'instruction suivante :

```
1 System.out.println(parametre)
```

Cette instruction peut être découpé en deux :

1. *System.out* : représente un objet qui est utilisé pour accéder aux méthodes d'affichage (sortie

standard)
2. *println* : est une méthode qui affiche une ligne de texte (paramètre) à l'écran

Exemple

```
1 class MonPremierProgramme{
2
3    public static void main(String[] args){
4       System.out.println("Bienvenue cher programmeur!");
5    }
6 }
```

Ce programme affichera à l'écran `Bienvenue cher programmeur`

Remarque

On peut constater à la fin de l'instruction d'affichage le symbole " ;". En Java, chaque ligne de code (instructions) doit se terminer par le symbole ";"

Attention

Il est strictement interdit d'utiliser le symbole ";" à la fin de la déclaration d'une méthode ou d'une classe. Ces deux éléments commencent et se terminent respectivement par une accolade ouvrante "{" et une accolade fermante "}"

3. lire les entrées saisie au clavier par l'utilisateur

la classe Scanner

Java offre au programmeur une palette de méthode qui permette de lire les entrées saisies par l'utilisateur. Parmi ces méthodes, nous découvrirons les méthodes de *l'objet Scanner*. Elle est la méthode la plus facile à utiliser. Puisque la classe Scanner n'appartient pas aux classes standards chargées par Java comme par exemple System.out, il faudrait l'importer dans le programme grâce à la syntaxe suivante.

```
1 import java.util.Scanner
```

pour bien comprendre ce code, veuillez suivre ce *lien*

étapes de création d'un outil de lecture standard

Cela se fait en deux étapes :

1. Créer un objet de type scanner, c'est-à-dire créer une instance de la classe Scanner
2. Faire appel à une des méthodes de lecture appartenant à la classe Scanner :
    - nextInt() :permet de lire un entier
    - nextDouble() :permet de lire un décimal
    - nextLine() :permet de lire une ligne entière, cela peut être une chaîne de caractère
    - next() :permet de lire un mot
    - nextBoolean() :permet de lire un booléen

Le code suivant montre ces deux étapes :

```
1 Scanner lire = new Scanner(System.in); //notre objet prend en paramère le flux de
  lecture standard System.in
2 lire.nextLine();
```

Exemple : code Java qui affiche ce qu'un utilisateur a saisi au clavier

```
1 import java.util.Scanner ;
```

```
2
3 class MaClasse{
4    public static void main(String[] args){
5       Scanner lire = new Scanner(System.in);
6       String saisi = lire.nextLine();
7       System.out.println("Vous avez écrit :"+ saisi);
8    }
9 }
```

La ligne 5 permettra à l'utilisateur de saisir quelque chose au clavier.

Le symbole "+" à la ligne 7 permet de *concaténer* les deux chaînes de caractère.

4. Exercice : Question1

[Solution n°7 p 14]

Quelle méthode est le point de départ de tout programme Java ?

5. Exercice : Question2

[Solution n°8 p 14]

Quelle méthode affiche du texte dans un programme Java

- ◯ System.printText()
- ◯ out.writeText()
- ◯ System.out.println()
- ◯ System.out()

6. Exercice : Question3

[Solution n°9 p 14]

Compléter le code suivant pour écrire un programme Java valide

```
      Bienvenue{
    public static void      (String[] args){
          System.     .println("Bienvenue les amis !")
  }
}
```

7. Exercice : Question4

[Solution n°10 p 14]

Complétez le code suivant pour permettre à un utilisateur de saisir du texte au clavier

```
import java.util.Scanner ;
class Test{
    public static void main(String[] args){
          sc= new Scanner(System.in);
       st= sc.nextLine()
  }
}
```

Conclusion

A l'issue de cette leçon, l'apprenant a découvert les fondamentaux du langage Java et s'est initié à la programmation en java. La leçon suivante sera consacrée à la gestion des flux d'entrées, de sorties et des exceptions

Solutions des exercices

> Solution n°1

Exercice p. 7

- ○ une version pour chaque plate-forme
- ○ deux versions seulement
- ◉ seulement une version

> Solution n° 2

Exercice p. 7

Compilé
oiseau.java class.java
Interprété
oiseau.class poisson.class

> Solution n°3

Exercice p. 7

- ☐ Compilé
- ☐ Interprété
- ☑ multiplate-forme
- ☐ ni compilé, ni interprété
- ☑ à la fois compilé et interprété
- ☐ bas niveau et Orienté Objet
- ☑ Portable

> Solution n° 4

Exercice p. 7

char

> Solution n° 5

Exercice p. 8

3

Exercice p. 8

> Solution n° 6

```
String var = " Zebié"
```

> Solution n° 7

Exercice p. 11

main

> Solution n°8

Exercice p. 11

- ○ System.printText()
- ○ out.writeText()
- ◉ System.out.println()
- ○ System.out()

> Solution n° 9

Exercice p. 11

```
class Bienvenue{
    public static void main (String[] args){
            System. out .println("Bienvenue les amis !") ;
    }
}
```

> Solution n° 10

Exercice p. 11

```
import java.util.Scanner ;
class Test{
    public static void main(String[] args){
          Scanner sc= new Scanner(System.in);
       String st= sc.nextLine() ;
    }
}
```