

Éléments de base du langage Java

Dr Khadim DRAME

Département Informatique
UFR Sciences et Technologies
Université Assane Seck de Ziguinchor

Mai 2022



Plan

- 1 Introduction
- 2 Types, déclarations, opérateurs, instructions
- 3 Structures conditionnelles
- 4 Structures itératives



Objectifs du cours

- Utiliser le langage Java pour implémenter et tester des algorithmes
- Compiler et exécuter des programmes Java via la ligne de commande
- Utiliser l'IDE Eclipse pour implémenter et exécuter des programmes Java



C'est quoi Java ?

- un langage de programmation orientée objet
- développé par [Sun](#) (1991), puis par [Oracle](#) (2009)
- nom Java en honneur à une boisson (café) provenant de l'île de Java, préférée des programmeurs
- 1994 : Navigateur Web HotJava supportant les applets



Caractéristiques de Java

- Un langage **simple** et **robuste**
- Un langage **portable** et **multi-plateforme**
- Une **bibliothèque très riche** avec des milliers de classes, offrant beaucoup de fonctionnalités
 - gestion des erreurs et exceptions
 - conception d'interfaces graphiques
 - accès aux bases de données
 - développement d'applications distribuées
 - développement d'applications Web



- **Java SE** (Java Standard Edition)
 - appelé aussi **JDK** (Java Development Kit)
 - utilisé pour le développement d'applications qui tournent sur un poste client
- **Java EE** (Java Enterprise Edition) / Jakarta EE
 - une extension de Java SE pour le développement d'applications d'entreprise
- **Java ME** (Java Micro Edition)
 - une distribution pour des plateformes embarquées (développement mobile)



- Un programme Java est un ensemble de fichiers «.java»
- Chaque fichier «.java» contient une ou plusieurs définitions de **classes**¹
- Un fichier «.java» contient au plus une définition de classe **publique**
- Le nom du fichier «.java» doit être celui de la classe publique

1. Nous allons revenir sur cette notion de classe ultérieurement.



Structure d'un programme java

```
public class Bienvenue {  
    public static void main(String [] args) {  
        System.out.println("Bonjour la classe!");  
        System.out.println("Bienvenue au cours de POO.");  
    }  
}
```

- Une application java a une classe publique (**Bienvenue**) qui contient une méthode **main()**
- Cette classe doit être placée dans un fichier de même nom (**Bienvenue.java**)
- Le nom d'une classe doit commencer par une majuscule et celui d'une méthode par minuscule



Compilation/exécution d'un programme java

- Compilation via la ligne de commande

`javac` Bienvenue.java

→ `Bienvenue.class`

- Exécution via la ligne de commande

`java` Bienvenue

```
C:\Users\HP>cd Documents  
  
C:\Users\HP\Documents>javac Bienvenue.java  
  
C:\Users\HP\Documents>java Bienvenue  
Bonjour la classe!  
Bienvenue au cours de POO.
```



Commentaires

`// un commentaire sur une seule ligne`

`/* un commentaire sur plusieurs
lignes */`

`/** un commentaire que javadoc va utiliser pour créer la
documentation sous forme HTML */`



Plan

- 1 Introduction
- 2 Types, déclarations, opérateurs, instructions
- 3 Structures conditionnelles
- 4 Structures itératives



Types primitifs

- Booléens
 - `boolean` (true ou false)
- Caractères
 - `char` (16 bits)
- Entiers
 - `byte` (8 bits), `short` (16 bits), `int` (32 bits), `long` (64 bits)
- Flottants
 - `float` (32 bits), `double` (64 bits)



Chaînes de caractères : String

- Une chaîne de caractères n'est pas un type primitif
- **String** est une classe java (Type d'objet) pour représenter les chaînes
- Un tel type de données utilise des méthodes spécifiques
- Exemple

```
1 public class TestChaine {
2     public static void main(String[] args) {
3         String pnom = "Amadou";
4         String nom = "Diop";
5         System.out.print( "Bonjour " + pnom + " " + nom);
6         int l = pnom.length();
7         System.out.print("Longueur prénom "+l); //6
8     }
9 }
```



Déclaration de variable

- Syntaxe

<type> <identificateur_variable>;

- Syntaxe (avec initialisation)

<type> <identificateur_variable> = <expression>;

- Exemples

```
1 int age;
2 float poids, taille;
3 boolean boursier;
4 float hauteur = 3.5f;
5 double montant = 1000.0;
6 char c1='M', c2 = '\t';
7 String mat = "Programmation objet";
```



Déclaration de constante

- Syntaxe

`final` <type> <identificateur_constante> = <valeur> ;

- Par convention, le nom d'une constante est toujours en majuscules

- Exemples

```
1 final double PI = 3.14;  
2 final double GRAVITATION = 9.8;
```



- Opérateurs arithmétiques : +, -, *, / (entière et réelle), %
- Opérateurs de comparaison : <, <=, >, >=, ==, !=
- Opérateurs logiques : && (et), || (ou), ! (négation)
- Opérateurs d'incrémentement (++)/ de décrémentation (--)
 - incrémente de 1
 - Exemple
 $x = 10;$
 $++x$ et $x++$ équivalent à $x = x + 1$ // x vaudra 11.
 - préfixé : $y=++x \Leftrightarrow x = x + 1; y = x;$
 - suffixé : $y=x++ \Leftrightarrow y = x; x = x + 1;$



Opérateurs

- Opérateurs d'affectation : $=$, $+=$, $-=$, $*=$, $/=$, $\%=$
 - $\langle \text{expression1} \rangle \langle \text{op} \rangle = \langle \text{expression2} \rangle$ équivaut à $\langle \text{expression1} \rangle = \langle \text{expression1} \rangle \langle \text{op} \rangle \langle \text{expression2} \rangle$
 - Exemple
 $x += 5$ équivaut à $x = x + 5$
- Opérateur ternaire : $?$
 - $\langle \text{condition} \rangle ? \langle \text{expression1} \rangle : \langle \text{expression2} \rangle$
vaut $\langle \text{expression1} \rangle$ si $\langle \text{condition} \rangle$ est vrai,
vaut $\langle \text{expression2} \rangle$ sinon.

- Exemples

```
1 y = x >= 0 ? x : -x; // valeur absolue de x
2 m = x <= y ? x : y; // minimum de x et y
3 i = i < 4 ? i+1 : i-1;
```



Conversion de type : *cast*

- Syntaxe

(<type>) <objet>

- Exemple

```
1 double moyenne;  
2 int somme = 100, nombre = 12;  
3 moyenne = (double) somme / nombre;  
4 //conversion de somme en flottant
```



- `System.out.print` : fonction pour imprimer sur l'écran
- `System.out.printf` : idem (comme en C)
- Exemple

```
1 public class Affichage{
2     public static void main(String[] args){
3         int num = 2, den = 3;
4         System.out.println(num+"/"+den); //affiche 2/3
5         System.out.printf("%d/%d", num, den); // idem
6     }
7 }
```



- Classe² utilitaire `Scanner`³ pour lire des données sur l'entrée standard (clavier).

```
1 import java.util.Scanner;
2 public class TestScanner{
3     public static void main(String[] args){
4         Scanner scanner = new Scanner(System.in);
5         System.out.println("Veuillez saisir le nom ");
6         String nom = scanner.nextLine();
7         System.out.println("Veuillez saisir l'âge ");
8         int age = scanner.nextInt();
9     }
10 }
```

2. Nous allons voir cette notion de classe ultérieurement.
3. <https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html> ▶



Plan

- 1 Introduction
- 2 Types, déclarations, opérateurs, instructions
- 3 Structures conditionnelles**
- 4 Structures itératives



Structures conditionnelles

- Syntaxe (Sélection simple)

```
if(<condition>){  
    <bloc_instructions>  
}
```

- Syntaxe (Sélection avec alternative)

```
if(<condition>){  
    <bloc_instructions1>  
}else{  
    <bloc_instructions2>  
}
```



Structures conditionnelles

● Exemple

```
1 int a = 15, b = 12;
2 double q;
3 if(b!=0){
4     q = (double)a/b;
5     System.out.print(" Le quotient est "+q) ;
6 }else{
7     System.out.print("Division par 0 impossible");
8 }
```



Structures conditionnelles imbriquées

- Syntaxe

```
if(<condition1>){  
    <bloc_instructions1>  
}else if(<condition2>){  
    <bloc_instructions2>  
...  
}else{  
    <bloc_instructionsn>  
}
```

- Exemple

```
1 float moy;  
2 char mention;  
3 if (moy>=16)  
4     mention='T';  
5 else if (moy>=14)  
6     mention='B';  
7 else if (moy>=12)  
8     mention='A';  
9 else  
10    mention='N';  
11 // N pour Néant, pas de  
    mention
```



Structures conditionnelles multiples : **switch/case**

- Syntaxe

```
switch (<expression>){  
    case <valeur_1> : {<bloc_instructions_1> ; break ;}  
    case <valeur_2> : {<bloc_instructions_2> ; break ;}  
    ...  
    case <valeur_n> : {<bloc_instructions_n> ; break ;}  
    default : {<bloc_instructions_default> ;}  
}
```

- <expression> doit être de type **char**, entier (**byte**, **short**, **int**, **long**), de type **énuméré**, ou une chaîne de caractères.
- Switch plus optimisé qu'un ensemble de **if** imbriqués.



Structures conditionnelles multiples : **switch/case**

● Exemple

```
1  class JourSemaineDemo {
2      public static void main(String[] args) {
3          int jour = 5;
4          switch (jour) {
5              case 1: System.out.println("Lundi"); break;
6              case 2: System.out.println("Mardi"); break;
7              case 3: System.out.println("Mercredi"); break;
8              case 4: System.out.println("Jeudi"); break;
9              case 5: System.out.println("vendredi"); break;
10             case 6: System.out.println("Samedi"); break;
11             case 7: System.out.println("Dimanche"); break;
12             default: System.out.println("Jour inconnu");
13         }
14     }
15 }
```



Structures conditionnelles multiples : **switch**

- Une nouvelle syntaxe de switch depuis Java SE 14
- Plusieurs valeurs pour une branche **case**
- Exemple

```
1 public class Chiffre {  
2     public static void main( String [] args ) {  
3         int val = (int) (Math.random() * 12);  
4         switch(val) {  
5             case 0, 1, 2, 3, 4 -> System.out.println("Petit chiffre");  
6             case 5, 6, 7, 8, 9 -> System.out.println("Grand chiffre");  
7             default -> System.out.println("Pas un chiffre , mais un nombre");  
8         }  
9     }  
10 }
```



Plan

- 1 Introduction
- 2 Types, déclarations, opérateurs, instructions
- 3 Structures conditionnelles
- 4 Structures itératives**



Structures itératives : **boucle for**

- Syntaxe

```
for (<initialisation> ; <condition> ; <pas>){  
    <bloc_instructions>  
}
```

- Exemple

```
1 int i, somme = 0;  
2 for(i = 1; i <= 10; i++){  
3     System.out.println(i);  
4     somme += i;  
5 }  
6 System.out.println("La somme est "+somme);
```



Structures itératives : **boucle while**

- Syntaxe

```
while (<condition>){  
    <bloc_instructions>  
}
```

- Exemple

```
1 int i = 1, somme = 0;  
2 while(i <= 10){  
3     System.out.println(i);  
4     somme += i;  
5     i = i + 1; // ou i++ ou encore i += 1  
6 }  
7 System.out.println("La somme est "+somme);
```



Structures itératives : **boucle do..while**

- Syntaxe

```
do {  
    <bloc_instructions>  
} while (<condition>);
```

- Exemple

```
1 int i = 1, somme = 0;  
2 do{  
3     System.out.println(i);  
4     somme += i;  
5     i++;  
6 }while(i <= 10);  
7 System.out.println("La somme est "+somme);
```



Branchements inconditionnels : **break**, **continue**

- Exemple 1 (avec break)

```
1 for(int i=1; i<=5; i++){
2     System.out.println(i);
3     if (i==2)
4         break;
5 }
```

- Résultat exemple 1

1
2

- Exemple 2 (avec continue)

```
1 for(int i=1; i<=5; i++){
2     if (i==2)
3         continue;
4     System.out.println(i);
5 }
```

- Résultat exemple 2

1
3
4
5



Exercices d'application

Exercice 1 (*compréhension de code*)

Quelle est la valeur de s si l'utilisateur donne le nombre 2705 ? 426 ?
Que fait ce programme ?

```
1 import java.util.Scanner;
2 public class Mystere{
3     public static void main(String [] args){
4         Scanner scanner = new Scanner(System.in);
5         System.out.println("Donner un entier positif ");
6         int n = scanner.nextInt();
7         int s = 0;
8         while(n > 0){
9             s = s + n % 10;
10            n = n / 10;
11        }
12        System.out.println("s = "+s);
13    }
14 }
```

