Éléments de base du langage Java

Dr Khadim DRAME
kdrame@univ-zig.sn

Département Informatique
UFR Sciences et Technologies
Université Assane Seck de Ziguinchor

Mai 2022

Plan

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

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

Éditions de Java

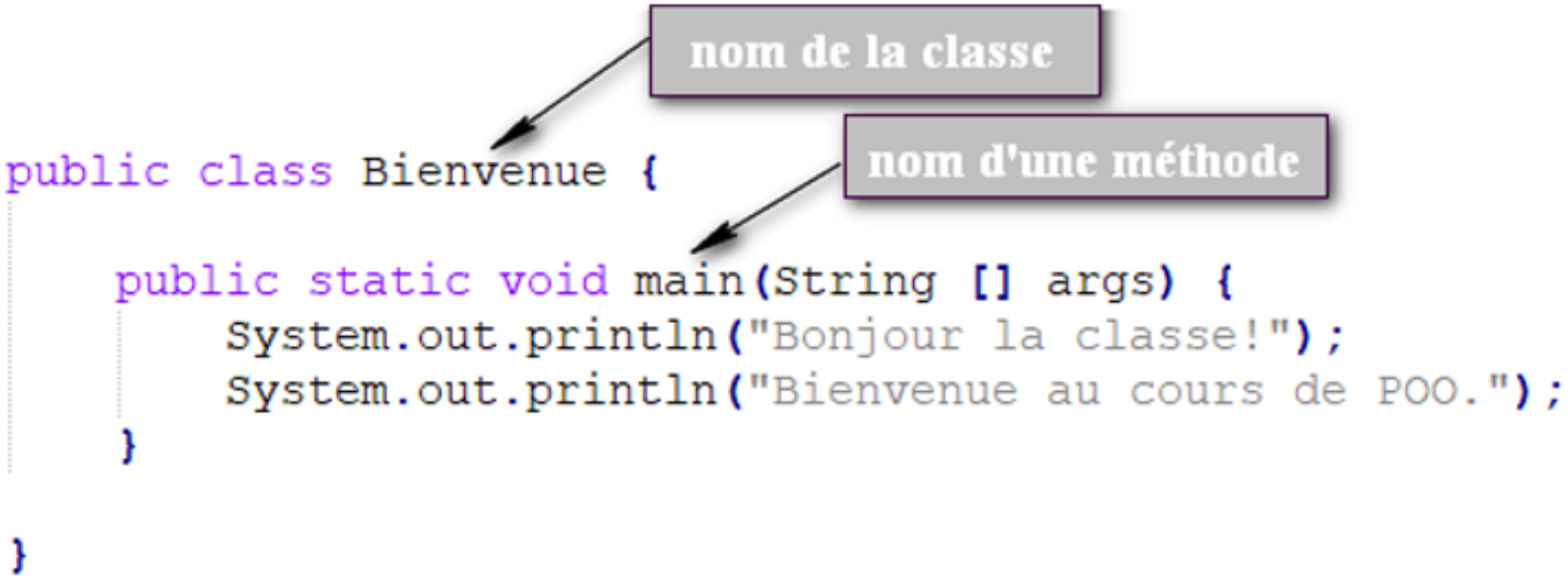
- Java SE (Java Standard Edition)
  - JDK (Java Development Kit) implémente la Java SE
  - utilisé pour le développement d'applications qui tournent sur un poste client
- Java EE (Java Enterprise Edition) / Jakarta EE
  - une extension de Java SE pour le développement d'applications d'entreprise
- Java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

Programme Java

- 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



- 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.
```

Plan

Commentaires

```
1 // un commentaire sur une seule ligne
2
3 /* un commentaire sur plusieurs
4 lignes
5 */
6
7 /** un commentaire que javadoc va utiliser pour géné
      rer la documentation au format HTML
8 */
```

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 chaines
- 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;// valeur par défaut 0
2 float poids, taille; // valeur par défaut 0
3 boolean boursier;// valeur par défaut false
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

- Opérateurs arithmétiques : +, -, *, / (entière et réelle), %
- Opérateurs de comparaison : <, <=, >, >=, ==, !=
- Opérateurs logiques : && (et), || (ou), ! (négation)
- Opérateurs d'incrémentation (++)/ de décrémentation (–)
  - incrémente de 1
  - Exemple

    x = 10 ;

    ++x et x++ équivalent à x = x + 1 // x vaudra 11.
  - préfixé : y=++x ⇔ x = x + 1 ; y = x ;
  - suffixé : y=x++ ⇔ y = x ; x = x + 1 ;

Opérateurs

- Opérateurs d'affectation : =, +=, -=, *=, /=, %=
  - <expression1> <op>= <expression2> équivaut à <expression1> = <expression1> <op> <expression2>
  - Exemple

    x += 5 équivaut à x = x + 5
- Opérateur ternaire : ?
  - <condition> ? <expression1> :<expression2>

    vaut expression1> si <condition> est vrai,

    vaut <expression2> sinon.
  - Exemples

```
1 y = x >= 0 ? x : -x;// valeur absolue de x
2 m = x <= y ? x : y;// minimun 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
```

Lecture/Écriture

- System.out.print : fonction pour imprimer sur l'écran
- System.out.prinf : 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 }
```

Lecture/Écriture

- Classe [2] 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

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_defaut> ;}

  }

- <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

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 }
```

Exercices d'application

Exercice 2 (*nombres premiers*)
Écrire un programme qui demande à l'utilisateur de saisir un entier positif n, puis indique si c'est un nombre premier ou pas.

Exercices d'application

Correction Exercice 2

```
1  import java.util.Scanner;
2  public class TestPremier{
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 i = 2; boolean b = true;
8      while(i <= n/2 && b){
9        if (n % i == 0)
10         b = false;
11         i = i + 1;
12     }
13     if (b)
14       System.out.println(n+" est premier");
15     else
16       System.out.println(n+" n'est pas premier");
17   }
18 }
```

Exercices d'application

Exercice 3 (*boucles imbriquées*)
Écrire un programme qui demande à l'utilisateur de saisir un entier positif n, puis affiche un triangle d'étoiles. Par exemple, pour n = 10, il affiche le triangle suivant :

Plan

Déclaration d'un tableau à une dimension

- Syntaxe

  <type> [ ] <identificateur_tableau> ;

- Syntaxe (déclaration et création)

  <type> [ ] <identificateur_tableau> =

  new <type>[<taille>] ;

- Syntaxe (avec initialisation)

  <type> [ ] <identificateur_tableau> = <valeurs> ;

- Exemples

```
1 int [] tab1;// tableau d'entiers
2 int [] tab2 = new int[10];// tableau de 10 entiers
3 int [] tab3 = {1, 2, 3, 4, 5};
4 float [] notes;// tableau de flottants
5 String [] noms;// tableau de chaînes de caractères
6 int tab4 [];// crochets après
```

- Les crochets peuvent être avant ou après le nom du tableau

Déclaration d'un tableau à deux dimensions

- Syntaxe

  <type> [ ][ ] <identificateur_tableau> ;

- Syntaxe (déclaration et création)

  <type> [ ][ ] <identificateur_tableau> =

  new <type>[<ligne>][<colonne>] ;

- Syntaxe (avec initialisation)

  <type> [ ][ ] <identificateur_tableau> = <valeurs> ;

- Exemples

```
1 // tableau d'entiers de 10 lignes et 5 colonnes
2 int [][] mat1 = new int[10][5];
3 // tableau d'entiers de 3 lignes et 4 colonnes
4 int [][] mat2 = {{1, 2, 3, 4},
5                  {5, 6, 7, 8},
6                  {9, 10, 11, 12}};
```

Initialisation d'un tableau

- Avec new, les éléments du tableau sont initialisés selon leur type
  - **0** pour les nombres (entiers et flottants)
  - **'\0'** pour les caractères
  - **false** pour les booléens
  - **null** pour les chaînes de caractères et les autres types

Accès aux éléments d'un tableau

- Accès aux éléments d'un tableau via leurs indices

  <identificateur_tableau>[<indice>];
- En Java, les indices d'un tableau commencent par 0
- Taille d'un tableau : length
- Un accès a un élément avec un indice en dehors des bornes inférieure et supérieure du tableau lève une exception
- Exemples

```
1 int [] tab = {1, 2, 3, 4, 5};
2 tab[0];// 1
3 tab[2];// 3
4 tab.length;// 5
```

Accès aux éléments d'un tableau

- Accès aux éléments d'un tableau à 2 dimensions

  <identificateur_tableau>[<indice_lig>][<indice_col>] ;

- Exemples

```
1 int [][] mat = {{1, 2, 3, 4},
2                 {5, 6, 7, 8},
3                 {9, 10, 11, 12}};
4 mat[0][0];// 1
5 mat[1][2];// 7
6 mat.length;// 3
```

Parcours d'un tableau

- Avec une boucle for ou avec une boucle for each
- for each introduite depuis la version 5 du JDK, très utile

```
1  public class ParcoursTableau{
2      public static void main(String [] args){
3          int [] tab = {1, 2, 3, 4, 5};
4          //avec un for traditionnel
5          for (int i=0; i<tab.length; i++)
6              System.out.println(tab[i]);
7
8          //avec un for each
9          for (int val : tab){
10             System.out.println(val);
11         }
12     }
13 }
```

Parcours d'un tableau à deux dimensions

- Avec une boucle for

```
1  public class ParcoursMatrice{
2      public static void main(String [] args){
3          int [][] mat = {{1, 2, 3, 4},
4                          {5, 6, 7, 8},
5                          {9, 10, 11, 12}};
6          //Parcours avec 2 for traditionnel
7          for(int i=0; i<mat.length; i++){
8            for(int j=0; j<mat[i].length; j++)
9                System.out.print(mat[i][j]+" ");
10             System.out.println();
11         }
12     }
13 }
```

Exercices d'application

Exercice 4 (*compréhension de code*)

Qu'affiche ce programme ? pour tab = {10, -5, 12, -9, 0, 8, -13, 11} ?

```
1  public class Mystere{
2    public static void main(String [] args){
3      int [] tab = {10, 12, 9, 8, 11, 0};
4      //int [] tab = {10, -5, 12, -9, 0, 8, -13, 11};
5      int somme = 0;
6      for (int i=0; i < tab.length; i++){
7        if (tab[i] == 0) break;
8        if (tab[i] < 0) continue;
9        somme += tab[i];
10     }
11     System.out.println(somme);
12   }
13 }
```

Exercices d'application

Exercice 5 (*moyenne d'un tableau*)
Écrire un programme qui permet de saisir des notes dans un tableau, de calculer la moyenne et d'afficher le nombre de notes supérieures à la moyenne.

Exercice 6 (*comparaison de tableaux*)
Écrire un programme qui permet de saisir deux tableaux, puis de les comparer.

Exercices d'application

Correction Exercice 5

```
1  import java.util.Scanner;
2  public class Moyenne{
3    public static void main(String [] args){
4      double [] notes;
5      double som = 0, moy; int i, n;
6      Scanner scanner = new Scanner(System.in);
7      System.out.println("Donner le nombre de notes");
8      n = scanner.nextInt();
9      notes = new int[n];
10     for(i=0; i<n; i++){// saisie du tableau
11       System.out.println("notes["+i+"]");
12       notes[i] = scanner.nextDouble();
13     }
14     for(i=0; i<notes.length; i++)// calcul moyenne
15       som += notes[i];
16     moy = som / notes.length;
17     System.out.println("Moyenne = "+moy);
18   }
19 }
```

Plan

Méthodes statiques

- Une méthode statique est une fonction rattachée à un type de données (classe)
- La méthode main est un exemple de méthode statique
- Syntaxe

  <visibilité> **static** <type> <id_methode>(<args>){

  <bloc_instructions>

  }

  - <visibilité> : public, private ou protected
  - <type> : type de retour, void si aucun résultat n'est retourné
  - <id_methode> doit commencer par une minuscule

Méthodes statiques

- Exemple

```java
import java.util.Scanner;
public class TestMethodes1{
  public static int mini(int x, int y){
    return x < y ? x : y;
  }
  public static int maxi(int x, int y){
    return x > y ? x : y;
  }
  public static void main(String [] args){
    int a = 5, b = 8;
    int p = mini(a, b);
    int g = maxi(a, b);
    System.out.println("Le minimum est "+p);
    System.out.println("Le maximum est "+g);
  }
}
```

Lecture de paramètres sur l'invite de commande

- Exemple

```
1  public class LectureParametre{
2    public static int mini(int x, int y){
3      return x < y ? x : y;
4    }
5    public static void main(String [] args){
6        if (args.length!=2){
7          System.out.println("Veuillez saisir deux entiers ");
8          System.exit(0);
9        } else {
10         int a = Integer.parseInt(args[0]);
11         int b = Integer.parseInt(args[1]);
12         System.out.println("Le minimum est "+mini(a, b));
13       }
14   }
15 }
```

Méthodes à nombre variable de paramètres

- Exemple

```
1  import java.util.Scanner;
2  public class TestMethodes2{
3    public static int mini(int... values){
4          int m = values[0];
5          for(int i=1; i<values.length; i++){
6              if (values[i] < m) m = values[i];
7          }
8          return m;
9      }
10     public static void main(String [] args){
11         System.out.println(mini(5, 3));//3
12         System.out.println(mini(5, 3, 8));//3
13         System.out.println(mini(5, 3, 8, 2));//2
14         System.out.println(mini(5, 3, 8, 2, 1));
15     }
16 }
```

Méthodes statiques récursives

- Exemple

```
1  import java.util.Scanner;
2  public class TestMethodesRec{
3    public static int modulo(int n, int m){
4          if (n < m)
5            return n;
6          else
7            return modulo(n - m, m);
8      }
9      public static void main(String [] args){
10         System.out.println(modulo(17, 5));//2
11         System.out.println(modulo(36, 2));//0
12     }
13 }
```