

# Chapitre 1

- Généralités
- Premiers programmes
- Instructions de base
- Lectures
- Utilisation de la classe Clavier
- Compilation, exécution, documentation
- Type simples, opérations
- Classes enveloppes
- Conversions
- Tableaux
- Paquetages

*Tous les programmes donnés en  
exemples se trouvent à l'adresse :  
[http://www.math-info.univ-paris5.fr/  
~pastre/poo/cours](http://www.math-info.univ-paris5.fr/~pastre/poo/cours)*

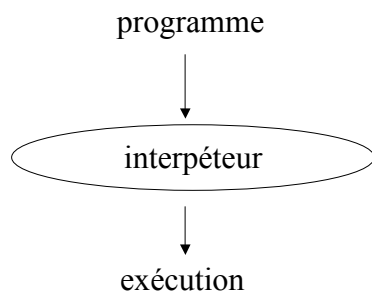
## Caractéristiques de Java

Java est un langage

- orienté objet : tout est objet
- simple
- robuste
- sécurisé
- doté d'une bibliothèque de classe très complète : API
- orienté réseau et distribué
- indépendant des architectures matérielles donc portable
- dynamique

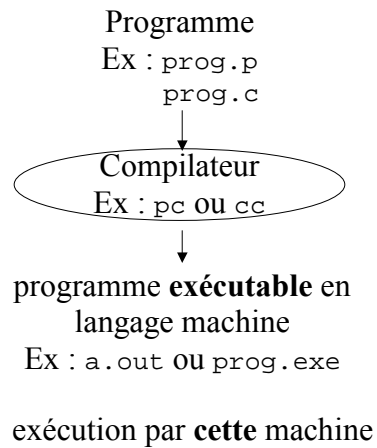
## Interpétation et compilation

- Interpétation



Ex: Scilab, Lisp, Prolog

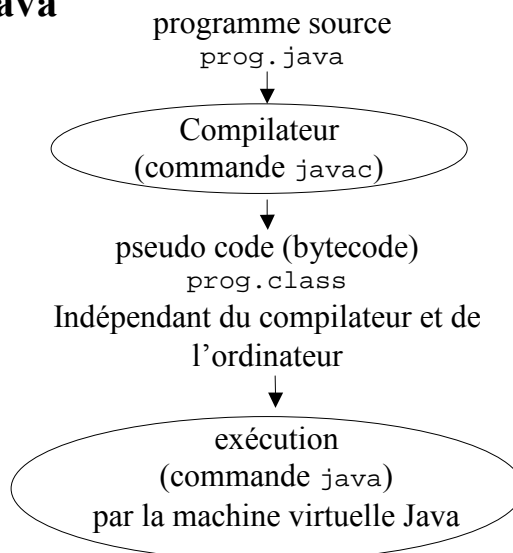
- compilation



Programmation Java 2003/2004 - D.Pastre

Chap.1.3

## en Java



Programmation Java 2003/2004 - D.Pastre

Chap.1.4

## Styles de programmation

### impératif (Pascal, C, Fortran)

- exécution d'instruction
- accès à la mémoire
- itératif et/ou récursif

### logique (Prolog)

- évaluation d'expressions logiques
- accès à la mémoire limité
- essentiellement récursif
- unification

### fonctionnel (Lisp, Prolog)

- évaluation d'expression
- accès à la mémoire limité
- essentiellement récursif

### objet (Smalltalk, C++, Java)

- composants autonomes et réutilisables
- utilisation de classes
- envois de messages
- héritage
- polymorphisme
- encapsulation

## Héritage

hiérarchie de classes (dérivées), toutes sous-classes de la classe Object

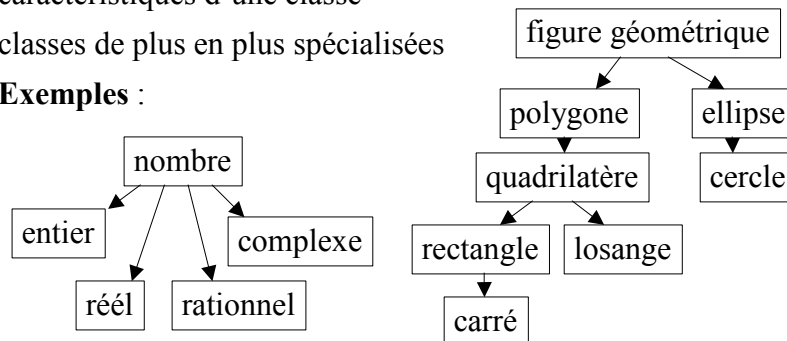
multiple (C++) ou non (Java mais interfaces)

redéfinitions possibles

caractéristiques d'une classe

classes de plus en plus spécialisées

**Exemples :**



## Polymorphisme

des fonctions peuvent avoir le même nom bien que ne s'appliquant pas aux mêmes objets

### Exemples :

si *r* est un rectangle (instance de la classe `Rectangle`)  
et *c* un cercle (instance de la classe `Cercle`),  
on peut définir et écrire : `r.dessiner()` et `c.dessiner()`  
si *e1* et *e2* sont des entiers, *r1* et *r2* des réels, *c1* et *c2* des complexes,  
on peut définir et écrire : `somme(4,5)`, `somme(6.7, 8.9)`,  
`somme(e1,e2)`, `somme(r1,r2)`, `somme(c1,c2)`  
`e1.somme(e2)`, `r1.somme(r2)`, `c1.somme(c2)`

## Encapsulation

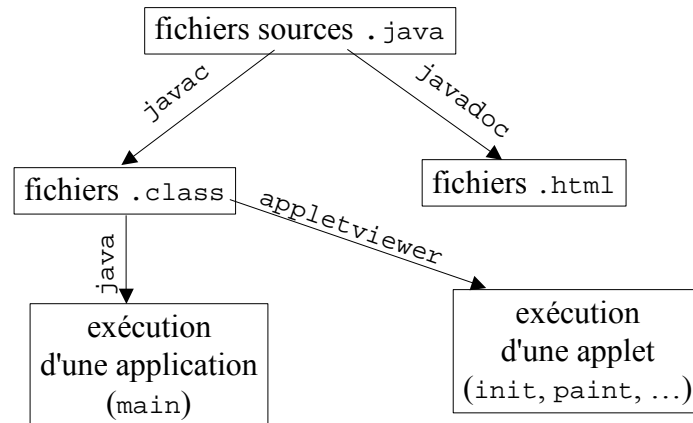
Attributs et méthodes peuvent être :

- visibles de l'extérieur (`public`)
- non visibles (`private`)
- partiellement visibles (`protected`)

La documentation d'une méthode publique  
**dit ce que fait** cette méthode mais **ne dit pas comment**

Les attributs **privés** d'un objet ne pourront être modifiés de l'extérieur (ce qui pourrait être source d'erreurs, par exemple, dans la relation *filz-père*, ajout d'un fils et oubli de l'ajout du père au fils)

## Compilation, documentation, exécutions



Programmation Java 2003/2004 - D.Pastre

Chap.1.9

## Premier programme

### Fichier Bonjour.java

```
/** Le programme Bonjour */
/* définition de la classe Bonjour */
public class Bonjour {
    /** ce programme dit bonjour */
    public static void main(String[] args) {
        System.out.println("bonjour tout le monde");
    }
}
```

The code is annotated with boxes: 'documentation' for the first comment, 'définition de classe' for the class definition, 'programme principal' for the main method, and 'écriture' for the print statement.

Programmation Java 2003/2004 - D.Pastre

Chap.1.10

## Compilation, exécution, documentation

```
$ javac Bonjour.java
    le fichier Bonjour.class est créé
$ java Bonjour
    exécution : affichage de
bonjour tout le monde

$ javadoc Bonjour.java
    création de plusieurs fichiers dont :
        Bonjour.html
        overview-tree.html
        index-all.html
        index.html
```

## Données et opérations

```
/** le programme Bonjour avec des données affectées dans le programme */
public class BonjourAvecDonnees {

    /** ce programme affiche bonjour suivi des données */
    public static void main(String[] args) {
        String prenom = "Jean";
        String nom = "Dupond";
        int annee = 1984;
        int age = 2003 - annee;
        double note1, note2;
        note1 = 15; note2 = 8.5;
        double moyenne = (note1 + note2)/2;
```

déclarations  
affectations  
opérations

### concaténation

```
System.out.println("bonjour "+prenom+" "+nom);
System.out.println("vous avez (ou aurez cette année) "+age+" ans");
System.out.println("vos notes sont : "+note1+" "+note2);
System.out.println("votre moyenne est : "+moyenne);
}
}
```

ou \_\_\_\_\_

```
String message = "bonjour "+prenom+" "+nom
                +"\nvous avez (ou aurez cette année) "+age+" ans"
                +"\nvos notes sont : "+note1+" "+note2
                +"\nvotre moyenne est : "+moyenne;
System.out.println(message);
```

ou \_\_\_\_\_

```
String message = "bonjour "+prenom+" "+nom;
message = message + "\nvous avez (ou aurez cette année) "+age+" ans";
message += "\nvos notes sont : "+note1+" "+note2;
message += "\nvotre moyenne est : "+moyenne;
System.out.println(message);
```

+

- Addition de nombres
- Concaténation de chaînes de caractères

Si a et b sont des nombres, a+b est leur somme  
Si a ou b est une chaîne, a+b est leur concaténation  
a+b+c est équivalent à (a+b)+c

D'où:

```
System.out.println("ab"+"cd") affiche abcd
System.out.println("a"+2003) affiche a2003
System.out.println(2003+"a") affiche 2003a
System.out.println(1999+4) affiche 2003
System.out.println("a"+1999+4) affiche a19994
System.out.println(1999+4+"a") affiche 2003a
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.15

## Passage d'arguments

```
/** le programme Bonjour avec des données en arguments */
public class BonjourAvecArgs {
    /** ce programme affiche bonjour suivi des arguments */
    public static void main(String[] args) {
        String prenom = args[0];
        String nom = args[1];
        int annee = Integer.parseInt(args[2]);
        int age = 2003 - annee;
        double note1, note2;
        note1 = Double.parseDouble(args[3]);
        note2 = Double.parseDouble(args[4]);
        double moyenne = (note1 + note2)/2;
        ...
    }
}
```

conversion  
String → int

conversion  
String → double

**Exécution :** java BonjourAvecArgs Paul Durand 1983 14.5 6

Programmation Java 2003/2004 - D.Pastre

Chap.1.16



## Lecture de caractères

```
/** le programme Bonjour avec deux initiales lues au clavier */
public class BonjourInitiales {

    /** affiche bonjour suivi des deux initiales lues au clavier */
    public static void main(String[] args) throws java.io.IOException {
        System.out.println(
            "tapez vos deux initiales sans espace suivies d'Entrée");
        char initiale1 = (char)System.in.read();
        char initiale2 = (char)System.in.read();
        System.out.println("bonjour " + initiale1+initiale2);
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.17

## ... avec définition d'une fonction lirechar

```
/** le programme Bonjour initiales avec une fonction lireChar*/
public class BonjourInitiales {
    public static void main(String[] args) throws java.io.IOException {
        System.out.println(
            "tapez vos deux initiales sans espace suivies d'Entrée");
        char initiale1 = lireChar();
        char initiale2 = lireChar();
        System.out.println("bonjour " + initiale1+initiale2);
    }
    /** fonction statique renvoyant un caractère lu au clavier */
    private static char lireChar() throws java.io.IOException {
        return((char)System.in.read());
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.18

## Lecture de chaines de caractères et de nombres, caractère par caractère

```
/** le programme Bonjour avec données lues caractère par caractère */
public class BonjourAvecLireChar {
    public static void main(String[] args) throws java.io.IOException {
        System.out.println("Toutes les données devront être suivies d'Entrée");
        System.out.println("tapez votre nom");
        String nom = lireLigne();
        System.out.println("tapez votre prenom");
        String prenom = lireLigne();
        System.out.println("tapez votre année de naissance");
        int annee = lireInt();
        System.out.println(
            "tapez vos deux notes (séparées par des blancs ou par Entrée)");
        double note1 = lireDouble();
        double note2 = lireDouble();
    }
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.19

```
/** lecture d'un caractère au clavier */
private static char lireChar() throws java.io.IOException {
    return((char)System.in.read());
}

/** lecture d'une ligne de caractères au clavier */
private static String lireLigne() throws java.io.IOException {
    String chaine = "";
    char caractere;
    caractere = lireChar();
    while (caractere != '\n') {
        chaine = chaine + caractere;
        caractere = lireChar();
    }
    return chaine;
}
```

ou

```
while ((caractere = lireChar()) != '\n') {
    chaine = chaine + caractere;
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.20

```

/** lecture d'une chaine de caractères au clavier */
private static String lireString() throws java.io.IOException {
    à compléter
}

/** lecture d'un entier au clavier */
private static int lireInt() throws java.io.IOException {
    String chaine = lireString();
    return Integer.parseInt(chaine);
    // ou return (Integer.valueOf(chaine)).intValue();
}

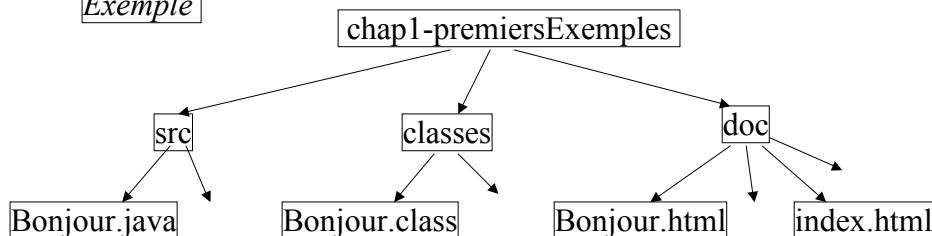
/** lecture d'un double au clavier */
private static double lireDouble() throws java.io.IOException {
    à compléter
}

```

## Organisation des fichiers (recommandations)

Regrouper tous les sources .java dans un répertoire **src**  
 et tous les pseudo-codes .class dans un répertoire **classes**  
**src** et **classes** étant au même niveau de l'arborescence de fichiers  
 On rangera la documentation dans un troisième répertoire **doc**  
 au même niveau

*Exemple*



## Nouvelle forme des commandes

dans le répertoire src

```
$ javac -d ../classes Bonjour.java  
$ java -classpath ../classes Bonjour  
$ javadoc -d ../doc Bonjour.java
```

dans le répertoire classes

```
$ java Bonjour
```

## Utilisation de la classe Clavier

```
import Clavier;  
/** Le programme Bonjour utilisant pour les lectures  
 * la classe Clavier dont le pseudo-code Clavier.class se trouve  
 * dans la bibliothèque du module (répertoire bibliotheque/classes)  
 */  
public class BonjourAvecClasseClavier {  
    public static void main(String[] args) {  
        String nom = Clavier.readString("tapez votre nom : ");  
        String prenom = Clavier.readString("tapez votre prenom : ");  
        int annee = Clavier.readInt("tapez votre année de naissance : ");  
        int age = 2003 - annee;  
        double note1 = Clavier.readDouble(  
            "tapez vos deux notes (séparées par au moins un blanc ou Entrée) : ");  
        double note2 = Clavier.readDouble();  
        ...  
    }  
}
```

*mieux*

```
import Clavier;
/** Le programme Bonjour utilisant pour les lectures
 * la classe Clavier dont le pseudo-code Clavier.class se trouve
 * dans la bibliothèque du module (répertoire bibliotheque/classes)
 */
public class Bonjour6AvecClasseClavier {
    public static void main(String[] args) {
        String nom = Clavier.readString("tapez votre nom : ");
        String prenom = Clavier.readString("tapez votre prenom : ");
        int annee = Clavier.readInt("tapez votre année de naissance : ");
        int age = 2003 - annee;
        double note1 = Clavier.readDouble(
            "tapez vos deux notes (séparées par au moins un blanc ou Entrée) : "
            ,0,20);
        double note2 = Clavier.readDouble(0,20);
        double moyenne = (note1 + note2)/2;
    }
}
```

...  
Programmation Java 2003/2004 - D.Pastre

Chap.1.25

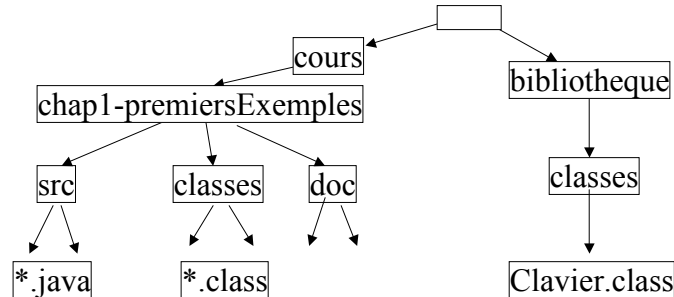
```
/** Cette classe permet de lire des données au clavier
 * (int, char, long, double, float, String). */
public class Clavier {
    /** Renvoie une chaîne entrée au clavier. */
    public static String readString() {...}
    /** Renvoie une chaîne entrée au clavier après avoir affiché 'prompt'. */
    public static String readString(String prompt) {...}
    /** Renvoie un entier entré au clavier. */
    public static int readInt() {...}
    /** Renvoie un entier entré au clavier après avoir affiché 'prompt'. */
    public static int readInt(String prompt) {...}
    /** Renvoie un entier entré au clavier, entre 'min' et 'max'. */
    public static int readInt(int min, int max) {...}
    /** Renvoie un entier entré au clavier, entre 'min' et 'max'
     * après avoir affiché 'prompt'. */
    public static int readInt(String prompt, int min, int max) {...}
}
...
}
```

*Documentation*

...  
Programmation Java 2003/2004 - D.Pastre

Chap.1.26

## Nouvelle forme des commandes



dans le répertoire src

```
$ javac -d ../classes -classpath
    ../../bibliotheque/classes BonjourAvecClasseClavier.java
$ java -classpath ../classes:../../bibliotheque/classes
    BonjourAvecClasseClavier
```

## Forme générale des commandes

```
javac -d < rép. où seront les nouveaux .class >
    -classpath < rép. où se trouvent les .class nécessaires
                pour la compilation séparés par des : (linux)
                ou des ; (windows) >
    < fichiers .java séparés par des espaces >

javadoc -d < rép. où sera la documentation >
    < fichiers .java ou paquetages séparés par des espaces >

java -classpath < rép. où se trouvent les .class nécessaires
                pour l'exécution, séparés par des : (linux)
                ou des ; (windows) >
    < nom complet de la classe principale >
```

## Variables, nombres, types simples, opérations élémentaires

*Variables* : déclaration (obligatoire) et initialisation

`int a;`  
`a=10;`    ou    `int a=10;`

	<i>Types de base</i>	<i>opérations</i>
entiers	byte -128 à 127 (8 bits) short -32768 à 32767 (16 bits) int -2 <sup>31</sup> à 2 <sup>31</sup> -1 (32 bits) long -2 <sup>63</sup> à 2 <sup>63</sup> -1 (64 bits)	+ - * / %
réel	float 1.40239846*10 <sup>-45</sup> à 3.40282347*10 <sup>38</sup> (32 bits) double 4.940656...*10 <sup>-324</sup> à 1.7976931...*10 <sup>308</sup> (64 bits)	+ - * /
caractères	char (16 bits)	
booléens	boolean (1 bit)	&&,   , ! == != < ...
Programmation Java 2003/2004 - D.Pastre		Chap.1.29

## Priorité des opérateurs

→	.	( ) [ ]
←	! ~ ++ --	-unaire
→	*	/ %
→	+	- binaire
→	<	<= > >= instanceof
→	==	!=
→	&	
→	^	
→		
→	&&	
→		
→	?	:
←	=	*= /= %= += -=

Programmation Java 2003/2004 - D.Pastre

Chap.1.30

## Conversion des types simples

implicites *ou* obligatoirement explicites si elles peuvent apporter une perte de précision

### Exemples

```
int n ; float x ; double y ; char c ;
n=12 ; x=n ; y=n ;
System.out.println("n="+n+" x="+x+" y="+y);
y = 12.5 ; n = (int) y ; x = (float) y;
System.out.println("y="+y+" n="+n+" x="+x);
x = 23.5f ; n = (int) x ; y = x ;
System.out.println("x="+x+" n="+n+" y="+y);
x = (float) 23.5 ;
System.out.println("x="+x);
c='A' ; n = c;
System.out.println("c="+c+" n="+n);
n=66; c = (char) n ;
System.out.println("n="+n+" c="+c);
```

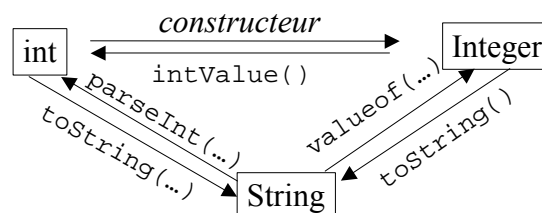
```
n=12 x=12.0 y=12.0
y=12.5 n=12 x=12.5
x=23.5 n=23 y=23.5
x=23.5
c=A n=65
n=66 c=B
```

## Classes enveloppes

Il existe une classe enveloppe pour chaque type de base, ce qui permet de travailler avec des objets.

De plus String n'est pas une type de base mais une classe (un peu particulière).

Des méthodes permettent de manipuler objets et types de base





## Exemples

```
int i; float x; Integer I; String ch;
...

i = (int) x; // float → int
I = new Integer(i); // int → Integer
i = I.intValue(); // Integer → int
I = Integer.valueOf(ch); // String → Integer
i = Integer.parseInt(ch); // String → int
// ou
I = Integer.valueOf(ch);
i = I.intValue();
// ou directement
i = Integer.valueOf(ch).intValue();
ch = Integer.toString(i); // int → String
ch = I.toString(); // Integer → String
```

## Tableaux

Un tableau est une collection ordonnée de variables de même type ou même classe

Déclaration : `int[] tab;`

Création : `int n=...; tab=new int[n];`

Initialisation : `tab[0]=...; tab[1]=... ;...`

ou `for (int i= 0;i<n;i++) tab[i]=...;`

Déclaration et création : `int[] tab=new int[n];`

Déclaration, création et initialisation :

```
int[] tab = {6,3,6,8,9}; // uniquement des constantes
String[] tabMois={"janv","fev","mars",...}
```

Nombre d'éléments : `tab.length` (constante)

## Exemples

```
public class Tab {  
    /** calcule la moyenne de notes données dans le programme */  
    public static void main(String[] args) {  
        double[] notes = {12,13,15,7,19};  
        int nbNotes = notes.length;  
  
        double somme = 0;  
        for (int i=0;i<nbNotes;i++) somme += notes[i];  
        double moyenne = somme/nbNotes;  
        System.out.print("\nnotes :");  
        for (int i=0;i<nbNotes;i++) System.out.print(" "+notes[i]);  
        System.out.println("\nmoyenne : "+ moyenne);  
    }  
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.35

```
public class Tab {  
    /** calcule la moyenne de notes données en arguments */  
    public static void main(String[] args) {  
  
        int nbNotes = args.length;  
        double[] notes;  
        notes = new double[nbNotes];  
        for (int i=0;i<nbNotes;i++) {  
            notes[i] = Double.parseDouble(args[i]);  
        }  
        ...  
    }  
}
```

Exercice :  
reprendre le programme BonjourAvecArgs avec un tableau de notes

Programmation Java 2003/2004 - D.Pastre

Chap.1.36

```

public class Tab {
    /** calcule la moyenne de notes lues au clavier */
    public static void main(String[] args) {
        int nbNotes = Clavier.readInt("nombre de notes : ");
        double[] notes;
        notes = new double[nbNotes];
        System.out.println(
            "tapez les notes séparées par des blancs ou des \"Entrée\"");
        for (int i=0;i<nbNotes;i++) {
            notes[i] = Clavier.readDouble();
        }
        ...
    }
}

```

```

/** définitions de fonctions et moyenne de notes */
public class TabAvecFonctions {
    /** renvoie la somme des éléments du tableau 'tab' */
    public static double somme(double [] tab) {
        double somme = 0;
        int n = tab.length;
        for (int i=0;i<n;i++) somme += tab[i];
        return somme;
    }
    /** renvoie la moyenne des éléments du tableau 'tab' */
    public static double moyenne(double[] tab) {...}
    /** renvoie une chaine représentant 'tab' */
    public static String toString(double[] tab) {...}
    /** remplit 'tab' par des lectures au clavier */
    public static void lire(double[] tab) {...}
}

```

à compléter  
en exercice

```

/** calcule la moyenne de notes lues au clavier */
public static void main(String[] args) {
    double[] notes;
    int nbNotes = Clavier.readInt("nombre de notes : ");
    notes = new double[nbNotes];
    System.out.println(
        "tapez les notes séparées par des blancs ou des \"Entrée\"");
    lire(notes);
    System.out.print("\nnotes : " + toString(notes));
    System.out.println("\nmoyenne : " + moyenne(notes));
}
}

```

### *variante*

```

/** crée un tableau de double de dimension 'n'
 * et le remplit par des lectures au clavier */
public static double[] creerEtLire(int n) {...}

/** calcule la moyenne de notes lues au clavier */
public static void main(String[] args) {
    int nbNotes = Clavier.readInt("nombre de notes : ");
    System.out.println(
        "tapez les notes séparées par des blancs ou des \"Entrée\"");
    double[] notes = creerEtLire(nbNotes);
    ...
}

```

à compléter  
en exercice

## Tableaux de tableaux

Déclaration : `int[][] tab;`

Création : `tab=new int[3][];`

`tab[0]=new int[4];`

`tab[1]=new int[5];`

`tab[2]=new int[2];`

Initialisation : `tab[0][0]=...;tab[0][1]=...;..`

ou : `for (int i=0;i<tab.length;i++)`  
`for (int j=0;j<tab[i].length;j++)`  
`tab[i][j]=...;`

Déclaration et création : `int[][] tab=new int[n][m];`

Déclaration, création et initialisation :

`int[][] tab = {{6,3,6},{8,9},{3,6,8;2}};`

## Tableaux à plusieurs dimensions

Ce sont des tableaux de tableaux (exemple : matrices)

Déclaration : `int[][] mat;`

Création : `n=...;m=...;`

`mat=new int[n][m];`

Initialisation : `mat[0][0]=...;mat[0][1]=...;..`

ou : `for (int i=0;i<n;i++)`  
`for (int j=0;j<m;j++)`  
`mat[i][j]=...;`

Déclaration et création : `int[][] mat=new int[n][m];`

Déclaration, création et initialisation :

`int[][] mat = {{1,0,0},{0,1,0},{0,0,1}};`

## Exemples

```
public class Matrice {  
    /** lit un tableau à 2 dimensions et calcule la somme des éléments */  
    public static void main(String[] args) {  
        int dim1 = Clavier.readInt("première dimension : ");  
        int dim2 = Clavier.readInt("deuxième dimension : ");  
        System.out.println("les" + dim1*dim2 + "éléments : ");  
        int[][] matrice = new int[dim1][dim2];  
        for (int i=0;i<dim1;i++)  
            for (int j=0;j<dim2;j++) matrice[i][j] = Clavier.readInt();  
        int somme = 0;  
        for (int i=0;i<dim1;i++)  
            for (int j=0;j<dim2;j++) somme += matrice[i][j] ;  
        for (int i=0;i<dim1;i++) {  
            for (int j=0;j<dim2;j++) System.out.print(" "+matrice[i][j]);  
            System.out.println("");  
        }  
        System.out.println("somme = "+somme) }  
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.43

```
public class TabDeTab {  
    /** calcule la somme des éléments d'un tableau de tableaux */  
    public static void main(String[] args) {  
        int[][] tab2 = { {9,13,1}, {6,7}, {8,5,13,5} };  
        int somme = 0;  
        for (int i=0;i<tab2.length;i++)  
            for (int j=0; j<tab2[i].length;j++) somme += tab2[i][j];  
        for (int i=0; i<tab2.length;i++) {  
            for (int j=0; j<tab2[i].length;j++) System.out.print(" "+tab2[i][j]);  
            System.out.println("");  
        }  
        System.out.println("somme = "+somme);  
    }  
}
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.44

## Paquetages

Il est préférable de définir les classes dans des paquetages.

Les paquetages permettent

- de regrouper les classes qui vont ensemble
- de trouver facilement des classes utilitaires (prédéfinies ou mises à disposition ou personnelles)
- de pouvoir utiliser les mêmes noms dans des paquetages différents
- de contrôler l'accès

On définira ainsi de nouvelles classes Bonjour... dans un paquetage *bonjour* (par exemple) et les classes travaillant sur les tableaux dans un paquetage *tab*.

On utilisera une autre classe Clavier dans un paquetage *es* (entrées/sorties).

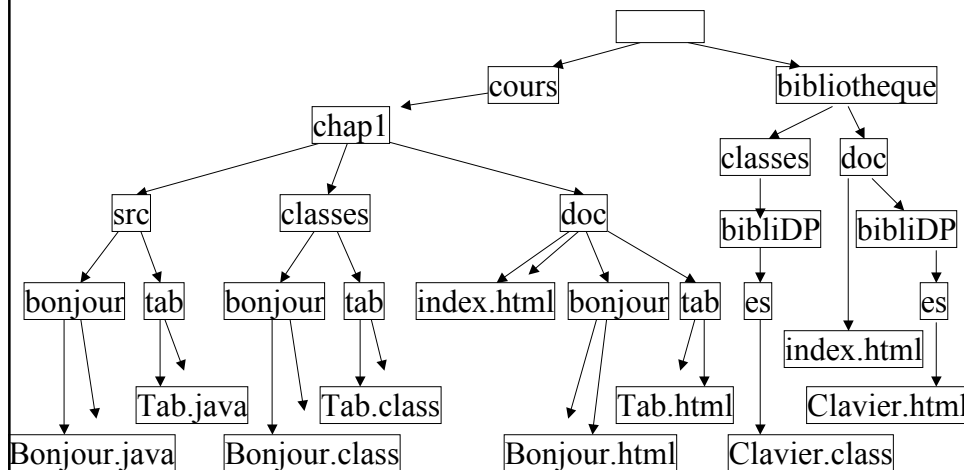
## Nouveaux codes

```
package bonjour;  
public class Bonjour {  
    ...  
}
```

```
package tab ;  
public class Tab {  
    ...  
}
```

```
package tabtab ;  
public class Matrice {  
    ...  
}
```

## Nouvelle arborescence



Programmation Java 2003/2004 - D.Pastre

Chap.1.47

## Nouvelles commandes

dans le répertoire src/bonjour

```
$ javac -d ../../classes Bonjour.java
```

```
$ java -classpath ../../classes bonjour.Bonjour
```

```
$ javadoc -d ../../doc Bonjour.java
```

```
$ javac -d ../../classes -classpath
```

```
    ../../../../bibliotheque/classes BonjourAvecClasseClavier.java
```

```
$ java -classpath ../../classes:../../../../bibliotheque/classes
```

```
    bonjour.BonjourAvecClasseClavier
```

dans le répertoire src

```
$ javadoc -d ../../doc bonjour tab
```

Programmation Java 2003/2004 - D.Pastre

Chap.1.48



## Exercices

définir des fonctions pour lire, sommer, représenter sous forme de chaîne les éléments de la matrice et du tableau de tableaux