REPUBLIQUE DE GUINEE

Travail- Justice- Solidarité

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE LABE

FACULTE DE SCIENCES ET TECHNIQUES

DEPARTEMENT INFORMATIQUE

ANNEE UNIVERSITAIRE 2019 – 2020

OPTION : INFOMATIQUE

«Manuel de cours d'Introduction de JAVA. »

Présenté par **Abdoulaye sow**

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Plan du cours

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

I. Introduction au langage Java

La Programmation Orientée Objet (POO) consiste à modéliser informatiquement un ensemble d'éléments d'une partie du monde réel (que l'on appelle domaine) en un ensemble d'entités informatiques. Ces entités informatiques sont appelées objets. Il s'agit de données informatiques regroupant les principales caractéristiques des éléments du monde réel (taille, couleur, ...).

Le langage Java est un langage généraliste de programmation synthétisant les principaux langages existants lors de sa création en 1995 par *Sun Microsystems*. Il permet une programmation orientée-objet (à l'instar de SmallTalk et, dans une moindre mesure, C++), modulaire (langage ADA) et reprend une syntaxe très proche de celle du langage C.

Outre son orientation objet, le langage Java a l'avantage d'être **modulaire** (on peut écrire des portions de code génériques, c-à-d utilisables par plusieurs applications), **rigoureux** (la plupart des erreurs se produisent à la compilation et non à l'exécution) et **portable** (un même programme compilé peut s'exécuter sur différents environnements). En contrepartie, les applications Java ont le défaut d'être plus lentes à l'exécution que des applications programmées en C par exemple.

a. Caractéristiques du langage Java:

– Portabilité;

b. Indépendance par rapport aux plateformes.

– Sécurité et robustesse;

c. Le compilateur interdit toute manipulation en mémoire.

– Gratuité;

d. Les outils de développement Java sont fournis gratuitement.

– Richesse;

e. Disponibilité d'une vaste collection de bibliothèques de classes.

II. Environnement Java

Java est un langage interprété, ce qui signifie qu'un programme compilé n'est pas directement exécutable par le système d'exploitation mais il doit être interprété par un autre programme, qu'on appelle interpréteur.

Exemple :

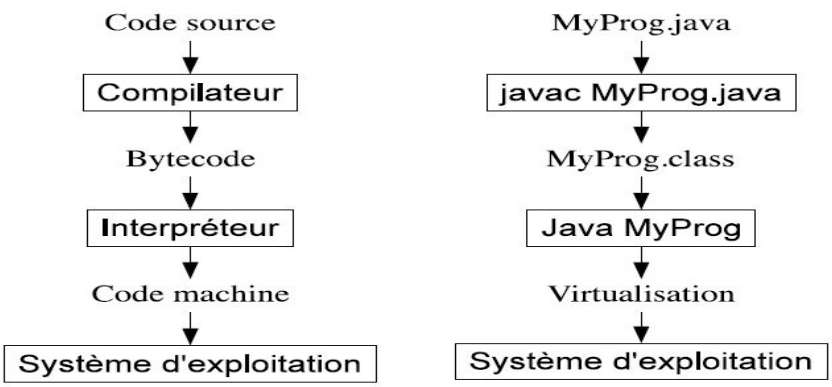
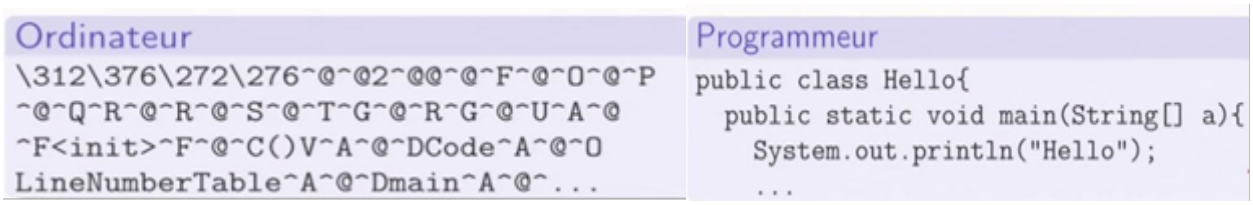


FIGURE 1.1 – Interprétation du langage

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Un programmeur Java écrit son code source, sous la forme de classes, dans des fichiers dont l'extension est **.java**. Ce code source est alors compilé par le compilateur **javac** en un langage appelé ***bytecode*** et enregistre le résultat dans un fichier dont l'extension est **.class**. Le **bytecode** ainsi obtenu n'est pas directement utilisable. Il doit être interprété par la **machine virtuelle** de Java qui transforme alors le code compilé en code machine compréhensible par le système d'exploitation.
C'est la raison pour laquelle Java est un langage portable : le **bytecode** reste le même quelque soit l'environnement d'exécution.
En 2009, Sun Microsystems est racheté par Oracle Corporation qui fournit dorénavant les outils de développement Java SE (*Standard Edition*) contenus dans le *Java Development Kit* (JDK).



Java à deux langages :

1. celui des humains
2. celui de l'ordinateur

Entre les deux un traducteur qu'on appelle programme
Ecrire le programme en langage source
Editeur de texte
Traduire le programme en langage cible
Compilateur
Exécuter le programme autant de fois qu'on veut
Lanceur d'application
A chaque étape, utilisation d'un programme
Outils pour la fabrication de programme
Editeur de texte, compilateur, lanceur d'application
Apprendre à utiliser les outils pour les taper, les compiler, les exécuter
Soit un outil pour débutant à installer : drjava
Soit un outil pour débutant en ligne : doppio
Les outils professionnels (éclipse, netbeans).

- Installez les outils de développement

L'un des principes phares de Java réside dans sa machine virtuelle : celle-ci assure à tous les développeurs Java qu'un programme sera utilisable avec tous les systèmes d'exploitation sur lesquels est installée une machine virtuelle Java.
Lors de la phase de compilation de notre code source, celui-ci prend une forme intermédiaire appelée byte code : c'est le fameux code inintelligible pour votre machine, mais interprétable par la machine virtuelle Java. Cette dernière porte un nom : on parle plus communément de JRE (Java Runtime Environment). Plus besoin de se soucier des spécificités liées à tel ou tel OS (Operating System, soit système d'exploitation). Nous pourrons donc nous consacrer entièrement à notre programme.
Afin de nous simplifier la vie, nous allons utiliser un outil de développement, ou IDE (Integrated Development Environment), pour nous aider à écrire nos futurs codes

source… Nous allons donc avoir besoin de différentes choses afin de pouvoir créer des programmes Java : la première est ce fameux JRE !

Installer les outils nécessaires

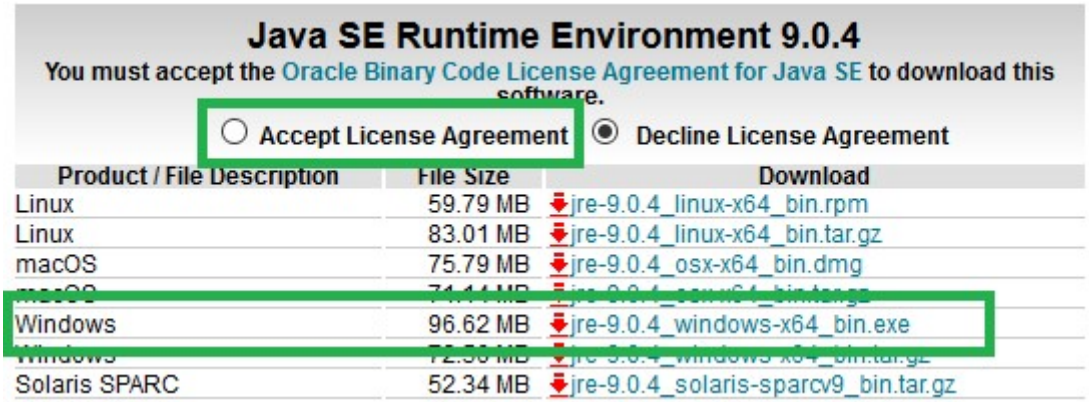
JRE ou JDK

Commencez par télécharger l'environnement Java sur le site d'Oracle, comme le montre la figure suivante. Choisissez la dernière version stable.

Encart de téléchargement

Vous avez sans doute remarqué qu'on vous propose de télécharger soit le JRE, soit le JDK (Java Development Kit). La différence entre ces deux environnements est écrite, mais pour les personnes fâchées avec l'anglais, sachez que le JRE contient tout le nécessaire pour que vos programmes Java puissent être exécutés sur votre ordinateur ; le JDK, en plus de contenir le JRE, contient tout le nécessaire pour développer, compiler…

L'IDE contenant déjà tout le nécessaire pour le développement et la compilation, nous n'avons besoin que du JRE. Une fois que vous avez cliqué surDownload JRE, vous arrivez sur la page représentée à la figure suivante.



Choix du système d'exploitation

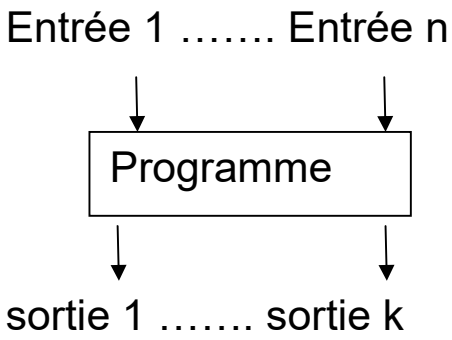
E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Cochez la case :Accept License Agreementpuis cliquez sur le lien correspondant à votre système d'exploitation (x86pour un système 32 bits etx64pour un système 64 bits). Une popup de téléchargement doit alors apparaître.
Je vous ai dit que Java permet de développer différents types d'applications ; il y a donc des environnements permettant de créer des programmes pour différentes plates-formes :
J2SE(Java 2 Standard Edition, celui qui nous intéresse dans cet ouvrage) : permet de développer des applications dites « client lourd », par exemple Word, Excel, la suite OpenOffice.org… Toutes ces applications sont des « clients lourds » . C'est ce que nous allons faire dans ce cours
J2EE(Java 2 Enterprise Edition) : permet de développer des applications web en Java. On parle aussi de clients légers.
J2ME(Java 2 Micro Edition) : permet de développer des applications pour appareils portables, comme des téléphones portables, des PDA…
Eclipse IDE
Avant toute chose, quelques mots sur le projet Eclipse. « Eclipse IDE » est un environnement de développement libre permettant de créer des programmes dans de nombreux langages de programmation (Java, C++, PHP…). C'est l'outil que nous allons utiliser pour programmer.
Eclipse IDE est lui-même principalement écrit en Java.
Je vous invite donc à télécharger Eclipse IDE. Une fois la page de téléchargement choisissezEclipse IDE for Java Developers, en choisissant la version d'Eclipse correspondant à votre OS (Operating System = système d'exploitation), comme indiqué à la figure suivante.

Version Windows d'Eclipse IDE
Sélectionnez maintenant le miroir que vous souhaitez utiliser pour obtenir Eclipse. Voilà, vous n'avez plus qu'à attendre la fin du téléchargement.
Pour ceux qui l'avaient deviné, Eclipse est le petit logiciel qui va nous permettre de développer nos applications ou nos applets, et aussi celui qui va compiler tout ça. Notre logiciel va donc permettre de traduire nos futurs programmes Java en langage byte code, compréhensible uniquement par votre JRE, fraîchement installé.
La spécificité d'Eclipse IDE vient du fait que son architecture est totalement développée autour de la notion de plugin. Cela signifie que toutes ses fonctionnalités sont développées en tant que plugins. Pour faire court, si vous voulez ajouter des fonctionnalités à Eclipse, vous devez :
télécharger le plugin correspondant ;
copier les fichiers spécifiés dans les répertoires spécifiés ;
démarrer Eclipse, et ça y est !
Lorsque vous téléchargez un nouveau plugin pour Eclipse, celui-ci se présente souvent comme un dossier contenant généralement deux sous-dossiers : un dossier « plugins » et un dossier « features ». Ces dossiers existent aussi dans le répertoire d'Eclipse. Il vous faut donc copier le contenu des dossiers de votre plugin vers le dossier correspondant dans Eclipse (pluginsdanspluginsetfeaturesdansfeatures).

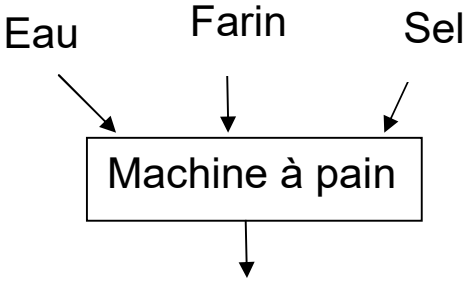
III. Qu'est –ce qu'un programme ?

- Suite d'instructions
- Comprend des données et des traitements
- Une partie des données vient de l'extérieur du programme (clavier, fichier, réseau, capteur) : les entrées
- Le programme calcule des résultats qui peuvent être transmis à l'extérieur (écran, fichier, réseaux, imprimante) : les sorties



Exemple : la machine à pain

Entrées :



Sortie : Pain

Traitements : pétrissage, levée, cuisson.

Langage

Programme écrits en utilisant un langage
Des règles de grammaire à respecter
Un sens associé à chaque programme
Il existe de nombreux langages, regroupés en familles
Pour nous : java, famille des langages objets

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

IV. Structure d'un programme

Comme la plupart des langages actuels, Java est un langage **structuré**. Un programme écrit en Java est constitué d'une suite de **méthodes** (**fonctions** ou **procédures**) bâties sur le modèle suivant :

a. Squelette d'une fonction

```
Type NomFonction (Paramètres) {
  ...
  //Corps de la fonction
  ...
}
```

- **Paramètres :** est une liste (éventuellement vide) de paramètres passés à la fonction lors de son appel. Même si aucun paramètre ne doit être passé à la fonction, vous devez cependant placer des parenthèses.
- **NomFonction :** est le nom de la fonction,
- **Type :** est le type de la valeur renvoyée par la fonction.

b. Squelette d'une procédure

Lorsqu'aucune valeur n'est renvoyée par la fonction, son type est **void**, et il est courant d'y faire référence sous le terme **procédure** :

```
void NomProcédure (Paramètres) {
  ...
  Corps de la fonction
  ...
}
```

Où **NomProcédure** est le nom de la procédure.

Attention :
Le nom d'une fonction doit respecter les règles suivantes :

1. Le premier caractère est obligatoirement une lettre ou le caractère de soulignement ;
2. Les caractères suivants peuvent être des lettres, des nombres ou des caractères de soulignement ;
3. La longueur du nom ne peut excéder 247 caractères ;
4. Une fonction ne peut avoir le même nom qu'un des mots réservés du langage.

La procédure **main()** joue un rôle particulier : c'est le point d'entrée d'une application Java. Elle peut être placée à un endroit quelconque du programme, mais sa première instruction est la première instruction exécutée dans l'application.

Exemples de méthodes

Voici, à titre purement informatif, le plus petit programme qui puisse exister en langage Java. Le programme principal n'effectue aucune action et il ne demande aucun paramètre :

```
public class Essai{

  public static void main (String args[]){
  }
}
```

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

La fonction ci-après demande et renvoie une valeur caractère. Remarquez l'instruction **return** qui désigne la valeur retournée par la fonction :

```
public class Essai{

  public char caractSuiv (char c); {
    return c + 1;
  }
}
```

La fonction ci-après renvoie sous forme entière le carré du nombre entier qui lui est passé :

```
public class Essai{

  int carre(int x) {
    return x*x;
  }
}
```

Les commentaires

- /* Commentaires sur une ou plusieurs lignes */

– Identiques à ceux existant dans le langage C.

- // Commentaires de fin de ligne

– Identiques à ceux existant en C++.

- /** Commentaires d'explication */

– Les commentaires d'explication se placent généralement juste avant une déclaration (d'attribut ou de méthode).

V.Instructions, blocs et blancs

- Les instructions Java se terminent par un « ; ».
- Les blocs sont délimités par deux accolades:
  { Pour le début de bloc
  } pour la fin du bloc
- Un bloc permet de définir un regroupement d'instructions. La définition d'une classe ou d'une méthode se fait dans un bloc.
- Les espaces, tabulations, sauts de ligne sont autorisés. Cela permet de présenter un code plus lisible.

Instructions possibles:

– Déclaration d'une variable.
– Appel de méthode.
– Affectation.
– Instruction de boucle (while, for…).
– Instruction de test (if, switch).

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Déclaration d'une variable

- Une variable possède un type et un nom.
- Le type peut être un type de base ou une classe.
- L'initialisation d'une variable peut se faire au moment de la déclaration.

```
{
int compteur; // Déclaration
int indice = 0; // Déclaration + Initialisation
Voiture golf;
Voiture twingo = new Voiture();
}
```

Portée d'une variable

La portée d'une variable s'étend jusqu'à la fin du bloc dans lequel elle est définie.

```
{
{
int compteur;
…
// compteur est accessible
}
// compteur n'est plus accessible
}
```

L'affectation

L'opérateur « = » permet d'affecter la valeur de l'expression qui se trouve à droite à la variable qui se trouve à gauche.

```
public class Test {
        public static void main(String[ ] args) {
                int calculer;
                int i = 0;
                int j = 6;
                i = (j + 5) * 3; // Instruction d'affectation
                calculer = i + j;
                //return calculer;
                System.out.println ("Le résultat de calculer est: " + calculer);
        }
}
```

VI. Les opérateurs arithmétiques élémentaires

- **Règles de priorité sur les opérateurs arithmétiques:**

Niveau	Symbole	Signification
1	()	Parenthèse
2	*	Produit
	/	Division
	%	Modulo
3	+	Addition ou concaténation
	-	Soustraction

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Les opérateurs de comparaison

Opérateur	Exemple	Renvoie TRUE si
>	v1 > v2	v1 plus grand que v2
>=	v1 >= v2	Plus grand ou égal
<	v1 < v2	Plus petit que
<=	v1 <= v2	Plus petit ou égal à
==	v1 == v2	égal
!=	v1 != v2	différent

Les opérateurs logiques

Opérateur	Usage	Renvoie TRUE si
&&	expr1 && expr2	expr1 et expr2 sont vraies
&	expr1 & expr2	Idem mais évalue toujours les 2 expressions
||	expr1 || expr2	Expr1 ou expr2, ou les deux sont vraies
|	expr1 | expr2	idem mais évalue toujours les 2 expressions
!	! expr1	expr1 est fausse
!=	expr1 != expr2	si expr1 est différent de expr2

Les opérateurs d'affectation

- L'opérateur de base est « = ».
- Il existe des opérateurs d'affectation qui réalisent à la fois une opération arithmétique, logique, ou bit à bit et l'affectation proprement dite:

Opérateur	Exemple	Équivalent à
+=	expr1 += expr2	expr1 = expr1 + expr2
-=	expr1 -= expr2	expr1 = expr1 – expr2
*=	expr1 *= expr2	expr1 = expr1 * expr2
/=	expr1 /= expr2	expr1 = expr1 / expr2
%=	expr1 %= expr2	expr1 = expr1 % expr2

Point d'entrée d'un programme Java

- Pour pouvoir faire un programme exécutable il faut toujours une classe qui contienne une méthode particulière: la méthode « main ».

- C'est le point d'entrée dans le programme: le microprocesseur sait qu'il va commencer à exécuter les instructions à partir de cet endroit.

```
public static void main (String arg[ ])
{
…/…

}
```

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Exemple

Fichier Bonjour.java

```
public class Bonjour

{  //Accolade débutant la classe Bonjour
        public static void main(String args[ ])
{ //Accolade débutant la méthode main
        /* Pour l'instant juste une instruction
*/
        System.out.println("bonjour");
} //Accolade fermant la méthode main

} //Accolade fermant la classe Bonjour
```

La classe est l'unité de base de nos programmes.
Le mot clé en Java pour définir une classe est class.

Fichier Bonjour.java

```
public class Bonjour

{
        public static void main(String args[ ])
{

        System.out.println("bonjour");
}

}
```

Accolades délimitant le début et la fin de la définition de la class Bonjour.

Accolades délimitant le début et la fin de la méthode main.

Les instructions se terminent par des ;

Fichier Bonjour.java

```
public class Bonjour

{
        public static void main(String args[ ])
{

        System.out.println("bonjour");
}

}
```

Une méthode peut recevoir des paramètres. Ici la méthode main reçoit le paramètre args qui est un tableau de chaine de caractères.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Compilation d'un Programme java dans une Console DOS :
Javac Bonjour.java
Génère un fichier Bonjour.class
Exécution du programme (toujours depuis la console DOS) sur la JVM
Java Bonjour
Affichage de « bonjour » Dans la console.

Fichier Bonjour.java

Le nom du fichier est nécessairement celui de la classe suivi de l'extension « .java ». Java est sensible à la casse des lettres (exemple : Bonjour # bonjour).

```
public class Bonjour {

        public static void main(String args[ ]) {

                System.out.println("bonjour");

        }

}
```

Compilation et exécution

- Pour résumer, dans une console DOS, si j'ai un fichier Bonjour.java pour la classe Bonjour:

- javac Bonjour.java

javac est la commande qui lance le compilateur Java.

- Compilation en bytecode java.

Indication des erreurs (éventuelles) de syntaxe.
Génération d'un fichier Bonjour.class s'il n'y a pas d'erreurs.

- java Bonjour

java est la commande qui lance la machine virtuelle (JVM).
Exécution du bytecode.

Une commande si le programme ne fonctionne pas sur la console :

```
set path=%path%;C:\Program Files\Java\jdk-9.0.1\bin
```

- **javac.exe** : compilation . Ex : javac Bonjour.java → cree fichier Bonjour.class
- **java.exe** : execution Ex : java Bonjour → affiche 'Bonjour Objis'
- **jar.exe** : creation livrables (.jar, .war, .ear...) → jar cvf bonjour.jar Bonjour.class
- **javadoc.exe** : creation documentation HTML → javadoc Bonjour.java
- **jvisualvm.exe** : surveillance memoire / jvm

- On peut utiliser ce qu'on appelle un environnement de développement intégré (IDE) pour compiler et exécuter des applications Java.

Un IDE Java est un éditeur spécifique du code Java ayant une interface intuitive qui permet de faciliter l'édition, la compilation, la correction d'erreurs et l'exécution des applications Java.

Exemples d'IDE Java:

- JCreator.
- netBeans (Open Source).
- BlueJ.
- JBuilder.

Remarque:

Nécessité de la méthode main, qui est le point d'entrée dans le programme.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Identificateurs

- Un identificateur permet de désigner une classe, une méthode, une variable…
- Règles à respecter pour les identificateurs:
- Interdiction d'utiliser les mots-clés (mots réservés de Java).
- Les identificateurs Commencent par:
  - Une lettre.
  - Un « $ ».
  - Un « _ » (underscore).
- Ne commencent pas par:
  - Un chiffre.
  - Un signe de ponctuation autre que « $ » ou « _ ».

Les mots réservés de Java

abstract	default	goto	null	synchronized
boolean	do	if	package	this
break	double	implements	private	throw
byte	else	import	protected	throws
case	extends	instanceof	public	transient
catch	false	int	return	true
char	final	interface	short	try
class	finally	long	static	void
continue	float	native	super	volatile
const	for	new	switch	while

Les types de base

- En Java, tout est objet sauf les types de base.
- Il y a huit types de base:

Un type booléen pour représenter les variables ne pouvant prendre que 2 valeurs (vrai et faux, 0 ou 1, etc.): Boolean avec les valeurs associées true et false.
Un type pour représenter les caractères: char.
Quatre types pour représenter les entiers de divers taille: byte, short, int et long.
Deux types pour représenter les réelles: float et double.
La taille nécessaire au stockage de ces types est indépendante de la machine.

- Avantage: portabilité.
- Inconvénient: « conversions » coûteuses.

Les entiers (avec signe):

- byte: codé sur 8 bits, peuvent représenter des entiers allant de -27 à 27 –1 (-128 à +127).
- short: codé sur 16 bits, peuvent représenter des entiers allant de -215 à 215 –1.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

- int: codé sur 32 bits, peuvent représenter des entiers allant de -231 à 231 –1.
- long: codé sur 64 bits, peuvent représenter des entiers allant de -263 à 263 –1.

Opérations sur les entiers:

- + : addition.
- - : soustraction.
- * : multiplication.
- / : division entière.
- %: reste de la division entière.

Exemples:

- 15 / 4 donne 3.
- 15 % 2 donne 1.

Les opérateurs d'incrémentation et de décrémentation ++ et -- :

- + + : ajoute 1 à la valeur d'une variable entière.
- - - : retranche 1 de la valeur d'une variable entière.
- n++; « équivalent à » n = n+1;
- n--; « équivalent à » n = n-1;

Exemple:

- int n = 12;
- n ++; // Maintenant n vaut 13.
- 8++; est une instruction illégale.

L'incrémentation (resp. la décrémentation) peut être utilisée de manière suffixée: ++n.
La différence avec la version préfixée se voit quand on les utilise dans les expressions.
En version suffixée l'incrémentation (resp. la décrémentation) s'effectue en premier, alors qu'elle s'effectue en dernier en version préfixée.

Exemple illustratif:

int m=7; int n=7;
int a=2 * ++m; // a vaut 16, m vaut 8
int b=2 * n++; // b vaut 14, n vaut 8

Les réels:

- float: codé sur 32 bits, peuvent représenter des nombres allant de -1035 à + 1035.
- double: codé sur 64 bits, peuvent représenter des nombres allant de -10400 à +10400.

Notation:

- 4.55 ou 4.55D: réel en double précision.
- 4.55f: réel en simple précision.

Les opérateurs sur les réels:

- Opérateurs classiques: +, -, *, /

Attention pour la division:

- 15 / 4 donne 3.
- 15.0 / 4 donne 3.75 (si l'un des termes de la division est un réel, la division retournera un réel).

Puissance:

- Utilisation de la méthode pow de la classe Math:

double y = Math.pow(x, a) « équivalent à » x^a (i.e. xa en notation mathématique), avec x et a étant de type double.

Les booléens:

boolean: contient soit vrai (true) soit faux (false).

Les opérateurs logiques de comparaison:

- Egalité: opérateur ==
- Différence: opérateur !=
- Supérieur et inférieur strictement à: opérateurs > et <
- Supérieur et inférieur ou égal: opérateurs >= et <=

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Exemple d'illustration:

```
boolean x;
x= true;
x= false;
x= (5==5); // l'expression (5==5) est évaluée et la valeur est affectée à x qui vaut alors vrai
x= (5!=4); // x vaut vrai, ici on obtient vrai car 5 est différent de 4
x= (5>5); // x vaut faux, 5 n'est pas supérieur strictement à 5
x= (5<=5); // x vaut vrai, 5 est bien inférieur ou égal à 5
```

Autres opérateurs logiques:

- Et logique: &&
- Ou logique: ||
- Non logique: !

Exemples:

```
boolean a,b, c;
a= true;
b= false;
c= (a && b); // c vaut false
c= (a || b); // c vaut true
c= !(a && b); // c vaut true
c=!a; // c vaut false
```

Les types de base (11): Les caractères

Les caractères:

« char » contient un seul caractère (lettre, symbole, ponctuation…). Le type char désigne des caractères en représentation Unicode. Codage sur 2 octets contrairement à ASCII/ANSI codé sur 1 octet.

Notation hexadécimale des caractères Unicode de ' \u0000 ' à ' \uFFFF '.

Remarque:

Le codage ASCII/ANSI est un sous-ensemble de la représentation Unicode. Pour plus d'information sur Unicode: www.unicode.org

Exemple d'illustration 1:

```
char a,b,c; // a,b et c sont des variables de type char
a='a'; // a contient la lettre « a »
b= '\u0022' // b contient le caractère guillemet "
c=97; // c contient le caractère de rang 97: 'a'
```

Exemple d'illustration 2:

```
int x = 0, y = 0;
float z = 3.1415F;
double w = 3.1415;
long t = 99L;
boolean test = true;
char c = 'a';
```

Remarque importante:

Java exige que toutes les variables soient définies et initialisées. Le compilateur sait déterminer si une variable est susceptible d'être utilisée avant initialisation et produit une erreur de compilation.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Les structures de contrôle :

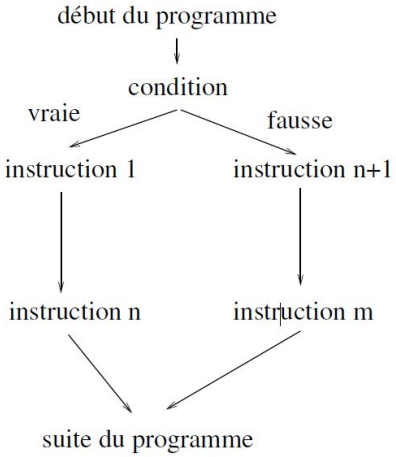
Les Structures de contrôles permettent d'arrêter l'exécution linéaire des instructions (de bas en haut et de gauche à droite).
Elles permettent d'exécuter conditionnellement une instruction ou de réaliser une boucle.
Le Java tout comme le C, propose plusieurs structures de contrôle différentes, qui nous permettent de nous adapter à tous les cas possibles. Il permet d'utiliser **des conditions** (structures alternatives), **des boucles** (structures répétitives), **des branchements conditionnels** ou non.
Certaines de ces structures de contrôle sont quasiment toujours utilisées ; d'autres le sont moins. Certaines sont très appréciées, d'autres sont à éviter.

- if, else
- switch, case, default, break
- for
- while
- do, while

- Instructions conditionnelles: if, else

On veut effectuer une ou plusieurs instructions seulement si une certaine condition est vraie:
if (condition) instruction; et plus généralement: if (condition) {bloc d'instructions}

- Condition doit être un booléen ou doit renvoyer une valeur booléenne.
- On veut effectuer une ou plusieurs instructions si une certaine condition est vérifiée sinon effectuer d'autres instructions:



if (condition) instruction1; else instruction2;
et plus généralement: if (condition) {1er bloc d'instructions} else {2ème bloc d'instructions}

Exemple :

Max.java

```
import java.io.*;
public class Max {
public static void main (String[ ] args) throws IOException {
BufferedReader in;
in = new BufferedReader (new InputStreamReader(System.in));
System.out.println ("Entrer le premier entier:");
String input1 = in.readLine();
int nb1 = Integer.parseInt(input1);
System.out.println("Entrer le second entier:");
String input2 = in.readLine();
int nb2 = Integer.parseInt(input2);
if (nb1 > nb2)
System.out.println("L'entier le plus grand est: " + nb1);
else
System.out.println("L'entier le plus grand est: " + nb2);
}
}
```

- **Les boucles :**

Les boucles vous permettent simplement d'exécuter des tâches répétitives.

- Il existe plusieurs sortes de boucles :

la boucle while (condition) {...} évalue la condition puis exécute éventuellement un tour de boucle (ou plus) ;
la boucle do {...} while (condition); fonctionne exactement comme la précédente, mais exécute un tour de boucle quoi qu'il arrive ;
la boucle for permet d'initialiser un compteur, une condition et un incrément dans sa déclaration afin de répéter un morceau de code un nombre limité de fois.
Tout comme les conditions, si une boucle contient plus d'une ligne de code à exécuter, vous devez l'entourer d'accolades afin de bien en délimiter le début et la fin.

- **Boucles indéterminées (while):**

On veut répéter une ou plusieurs instructions un nombre indéterminé de fois: On répète l'instruction ou le bloc d'instruction tant qu'une certaine condition reste vraie.

Syntaxe de la boucle while (tant que):

- while (condition) { bloc d'instructions }

Les instructions dans le bloc sont répétées tant que la condition reste vraie.
On ne rentre jamais dans la boucle si la condition est fausse dès le départ.

Exemple :

Facto1.java

```
import java.io.*;
public class Facto1 {
public static void main (String[ ] args) throws IOException {
int n, result, i;
BufferedReader in;
in = new BufferedReader (new InputStreamReader(System.in));
System.out.println ("Entrer une valeur pour n:");
String input = in.readLine();
n = Integer.parseInt(input);
result = 1; i = n;
while (i > 1)
{
result = result * i;
i--;
}
System.out.println ("La factorielle de " + n + " vaut " + result);
}
}
```

- **Boucles indéterminées (do, while):**

Syntaxe de la boucle do, while:
do {bloc d'instructions} while (condition)
Les instructions dans le bloc sont répétées tant que la condition reste vraie.
On rentre toujours au moins une fois dans la boucle: La condition est testée en fin de boucle.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

- **Boucles déterminées (for):**

On veut répéter une ou plusieurs instructions un nombre déterminé de fois: On répète l'instruction ou le bloc d'instructions pour un certain nombre d'itérations.

Syntaxe générale de la boucle for:

- for (int i = debut; i < fin; i++)
  Instructions;

Exemple:

```
for (int i = 0; i < 10; i++)
System.out.println(i); // affichage des nombres de 0 à 9
```

Remarque:

La syntaxe de la boucle for peut avoir d'autres variantes:

```
for (int i = fin; i > debut; i--)
Instructions;
```

Facto2.java

```
import java.io.*;
public class Facto2
{
public static void main (String[ ] args) throws IOException {
int n, result, i;
BufferedReader in;
in = new BufferedReader (new InputStreamReader(System.in));
System.out.println ("Entrer une valeur pour n:");
String input = in.readLine();
n = Integer.parseInt(input);
result = 1;
for(i =n; i > 1; i--)
{
result = result * i;
}
System.out.println ("La factorielle de " + n + " vaut " + result);
}
}
```

- **Sélection multiple (switch):**

L'utilisation de « if, else » peut s'avérer lourde quand on doit traiter plusieurs sélections ou effectuer un choix parmi plusieurs alternatives.
Pour cela, il existe en Java l'instruction « switch, case » qui est identique à celle de C/C++.
La valeur sur laquelle on teste doit être un char ou un entier (à l'exclusion d'un long).
L'exécution des instructions correspondant à une alternative commence au niveau du case correspondant et se termine à la rencontre d'une instruction break ou arrivée à la fin du switch.
Syntaxe général de l'instruction switch:

```
Switch (variable) {
Case valeur 1:
Instructions; break;
Case valeur 2:
Instructions; break;
…
Case valeur N:
Instructions; break;
Default:
Instructions;
}
```

Les structures de contrôle (11): switch

Alternative.java

```
import java.io.*;
public class Alternative {
   public static void main(String[] args) throws IOException {
            BufferedReader in;
            in = new BufferedReader (new InputStreamReader(System.in));
            System.out.println ("Entrer un entier:");
            String input = in.readLine();
            int nb = Integer.parseInt(input);

   switch(nb)
    {
      case 1:
        System.out.println("1"); break;
      case 2:
        System.out.println("2"); break;
      default:
       System.out.println ("Autre nombre"); break;
     }
   }
}
```

Première alternative : on affiche 1 et on sort du bloc du switch au break;

Deuxième alternative : on affiche 2 et on sort du bloc du switch au break;

Alternative par défaut: on réalise une action par défaut.

Les tableaux :

Un tableau est une variable contenant plusieurs données d'un même type.
Pour déclarer un tableau, il faut ajouter des crochets [ ] à la variable ou à son type de déclaration.
Vous pouvez ajouter autant de dimensions à votre tableau que vous le souhaitez, ceci en cumulant des crochets à la déclaration.
Le premier élément d'un tableau est l'élément 0.
Les valeurs contenues dans la variable sont repérées par un indice.
En langage java, les tableaux sont des objets.

Déclaration:

- Type Nom_Tableau[ ]; // Ou Type[ ] Nom_Tableau;

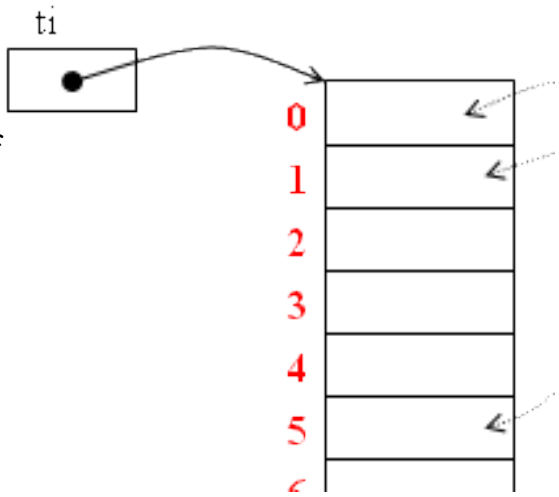
int tab[ ]; // Ou int[ ] tab;
String chaines[ ]; // Ou String[ ] chaines;

Création d'un tableau:

- Nom_Tableau = new Type[Taille_Tableau];

tab = new int [20]; // tableau de 20 entiers
chaines = new String [100]; // tableau de 100 chaînes

```
// ti est un tableau à une dimension de int
int ti [];
// tc est un tableau à une dimension de char
char[] tc;
```



Un tableau peut être initialisé :

```
int ti1 [] = { 1, 2, 3 , 4};
char[] tc = {'a', 'b', 'c'};
```

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Pour allouer l'espace nécessaire au tableau il faut utiliser **new** :

```
// ti est un tableau à une dimension de 10 int
ti = new int [10];
// tc est un tableau à une dimension de 15 char
tc = new char [15];
```

L'utilisation d'un tableau pour lequel l'espace n'a pas été alloué provoque la levée d'une exception `NullPointerException`

Les éléments d'un tableau sont indicés à partir de 0. Chaque élément peut être accédé individuellement en donnant le nom du tableau suivi de l'indice entre [ ]

```
ti [0] ;// premier élément du tableau ti
ti [9] ;// dernier élément du tableau ti
```

L'accès à un élément du tableau en dehors des bornes provoque la levée d'une exception `ArrayIndexOutOfBoundsException`

Un tableau possède un attribut ***length*** qui permet de connaître le nombre d'éléments d'un tableau.

- `ti.length` vaut 10.
- `tc.length` vaut 15.

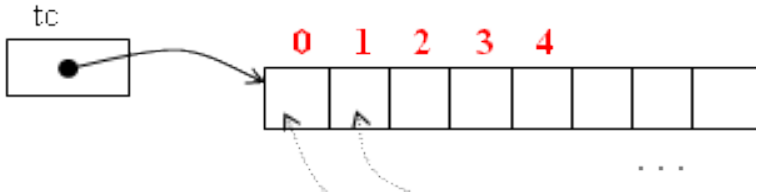


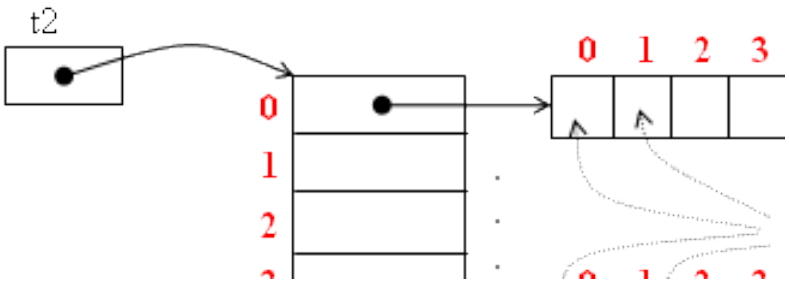
Tableau à plusieurs dimensions

Lors d'une définition de tableau le nombre de crochets indique le nombre de dimensions du tableau.

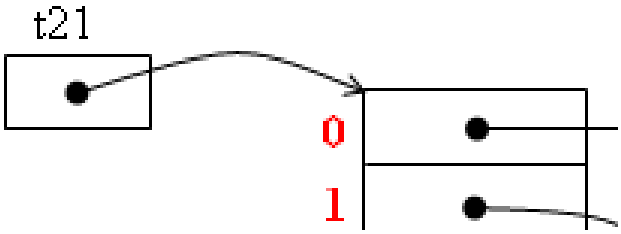
```
int t2[][] = new int[5][10] ;
```

définit un tableau à 2 dimensions 5 lignes sur 10 colonnes (ou l'inverse).

`t2[i]` désigne le $i^{ème}$ tableau à une dimension d'entiers.



Un tableau à plusieurs dimensions peut être initialisé :




```
int t21[][] = {{1, 2, 3},
               {4, 5, 6}};
```

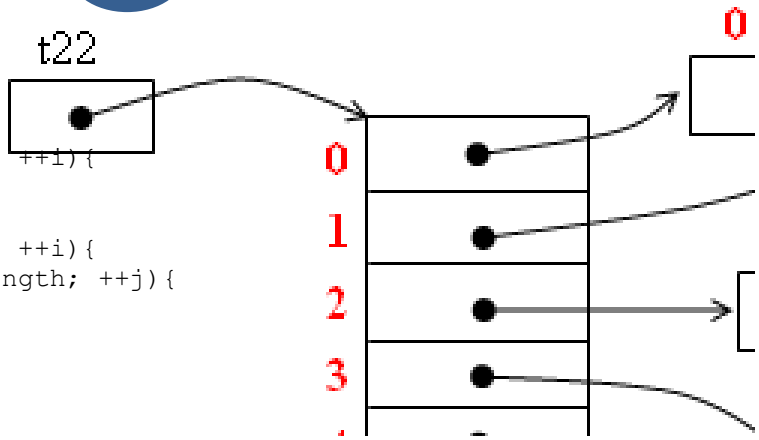
Définit un tableau dont la première dimension va de l'indice 0 à l'indice 1 et la deuxième dimension de l'indice 0 à l'indice 2.

Toutes les lignes d'un tableau à 2 dimensions n'ont pas forcément le même nombre d'éléments :

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

```
int t22[][] ;

t22 = new int[5][];
for( int i = 0; i< t22.length; ++i){
   t22[i]= new int [i+1];
}
for( int i = 0; i< t22.length; ++i){
   for( int j = 0; j<t22[i].length; ++j){
      //accès à t22[i][j]
   }
}
```



Tableaux en paramètre :

La spécification d'un paramètre tableau se fait en écrivant autant de couples de [] que de dimensions du tableau, mais sans donner la taille de chaque dimension. Cette taille peut être obtenue dans la méthode à l'aide de l'attribut *length*.
Le contenu des éléments peut être modifié, mais pas le tableau lui-même.

Utilitaires pour les tableaux :

System.arraycopy : la classe *System* a une méthode de copie rapide de tableaux :

```
public static void arraycopy(Object src, int srcPos, Object dest,
int destPos, int l)
```

Copie un tableau depuis *src*, et partir de la position *srcPos*, dans la destination *dest* à partir de *destPos* et sur une longueur *l*. La copie n'est pas une copie profonde : seules les références sont copiées.

- Si *src* et *dest* sont le même tableau, tout se passe comme si les éléments à copier étaient d'abord copiés dans un tableau auxiliaire.
- Si *src* ou *dest* vaut *null*, une *NullPointerException* est levée
- Une *ArrayStoreException* est levée si :
  - *src* ou *dest* n'est pas un tableau
  - *src* et *dest* sont des tableaux dont les éléments ne sont pas de même type.
- Une *IndexOutOfBoundsException* est levée si :
  - *srcPos* ou *destPos* ne sont pas « corrects »
  - *destPos+l* ou *srcPos+l* sortent du tableau

La classe *java.lang.reflect.Array* contient des méthodes statiques permettant :

- de récupérer un élément d'un tableau, en le convertissant dans un autre type
- d'affecter un tableau avec un élément
- de créer, à l'exécution, un tableau

`static Object get(Object t, int index)`	retourne l'objet d'indice *index* dans le tableau *t*. Cet élément est convertit en un des types enveloppants s'il est de type primitif (*Integer* pour *int*, *Long* pour *long*, etc...).
`static xxx getXxx(Object t, int index)`	retourne l'objet d'indice *index* dans le tableau *t*. Cet élément est convertit en le type primitif *xxx* si c'est possible

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

`static void set(Object t, int index, Object valeur)`	affecte une nouvelle valeur à l'élément d'indice *index* du tableau *t*. La valeur est convertie en le bon type avant l'affectation
`static void setXxx(Object t, int index, xxx z)`	affecte une nouvelle valeur à l'élément d'indice *index* du tableau *t*.

Les méthodes précédentes peuvent lever les exceptions suivantes :

- *NullPointerException* si l'objet au rang *index* vaut *null*.
- *IllegalArgumentException* si le premier argument n'est pas un tableau.
- *ArrayIndexOutOfBoundsException* si *index* est en dehors des bornes du tableau

`static Object newInstance(Class componentType, int l)`	Crée un nouveau tableau dont les éléments sont de type *componentType* et la longueur est *l*. Cette méthode peut lever les exceptions : • *NullPointerException* si le *componentType* vaut *null*. • *nullNegativeArraySizeException* si *l* est négative.
`static Object newInstance(Class componentType, int[] dimensions)`	Crée un nouveau tableau dont les éléments sont de type *componentType* : • Si *componentType* est une class ou une interface (pas tableau) le nouveau tableau a dimensions.length dimensions. Le nombre d'éléments dans la $i^{ème}$ dimension est égal à *dimensions[i]*. • Si *componentType* représente une classe tableau, le nombre de dimensions du nouveau tableau est égal à la somme du nombre de dimensions de *componentType* plus *dimensions.length* .

Le nombre d'éléments du tableau est mémorisé. Java peut ainsi détecter, lors de l'exécution, le dépassement d'indice et générer une exception.

Mot clé pour récupérer la taille d'un tableau: length

Syntaxe:

Nom_Tableau.length;

Exemple:

int taille = tab.length; // taille vaut 20

Comme en C/C++, les indices d'un tableau commencent à ' 0 '. Donc un tableau de taille 100 aura ses indices qui iront de 0 à 99.

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Initialisation:
tab[0] = 5;
tab[1] = 3; // etc.
chaines[0] = new String("Pierre");
chaines[1] = new String("Paul"); // etc.
Création et initialisation simultanées:
int tab [ ] = {5, 3};
String chaines [ ] = {"Pierre", "Paul"};

Les tableaux (4)

Tab1.java

```
public class Tab1
{
  public static void main (String args[ ])
  {
    int tab[ ] ;
    tab = new int[4];
    tab[0]=5;
    tab[1]=3;
    tab[2]=7;
    tab[3]=tab[0]+tab[1];
  }
}
```

Pour déclarer une variable tableau on indique le type des éléments du tableau et le nom de la variable tableau suivi de [ ].

On utilise new <type> [taille]; pour initialiser le tableau.

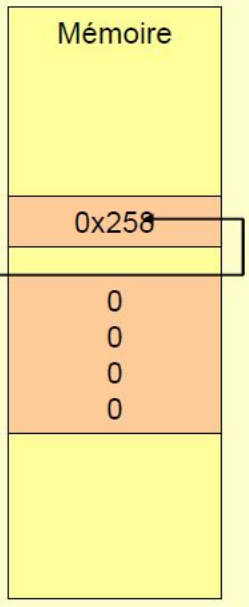
On peut ensuite affecter des valeurs au différentes cases du tableau: <Nom_Tableau>[indice] = Valeur

Les indices vont toujours de 0 à (taille-1)

Les tableaux (5)

Tab1.java

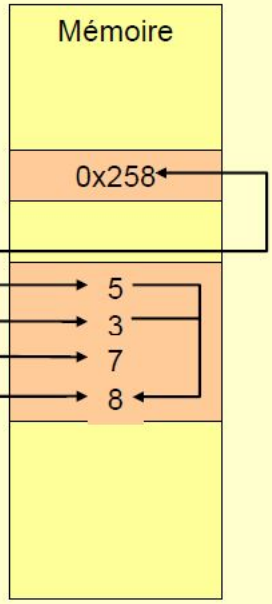
```
public class Tab1
{
  public static void main (String args[ ])
  {
    int tab[ ];
    tab = new int[4];
    tab[0]=5;
    tab[1]=3;
    tab[2]=7;
    tab[3]=tab[0]+tab[1];
  }
}
```



Les tableaux (6)

Tab1.java

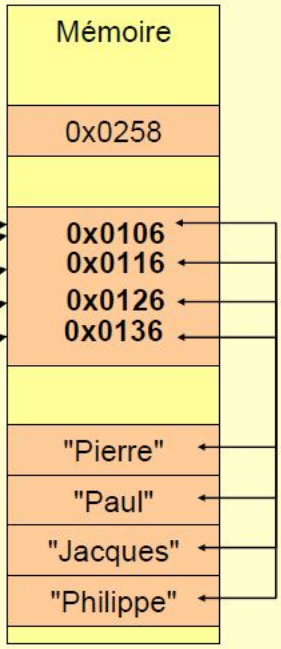
```
public class Tab1
{
 public static void main (String args[ ])
 {
  int tab[ ] ;
  tab = new int[4];
  tab[0]=5;
  tab[1]=3;
  tab[2]=7;
  tab[3]=tab[0]+tab[1];
 }
}
```



Les tableaux (7)

Tab2.java

```
public class Tab2
{
 public static void main (String args[ ])
 {
  String chaines[ ] ;
 chaines = new String[4];
 chaines[0]=new String("Pierre");
 chaines[1]=new String("Paul");
 chaines[2]=new String("Jacques");
 chaines[3]=new String("Philippe");
 }
}
```



E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Les tableaux (8)
Tab2.java
Modification du programme
pour afficher le contenu
du tableau.

```
public class Tab2
{
public static void main (String args[ ])
{
String chaines[ ] ;
chaines = new String[4];
chaines[0]=new String("Pierre");
chaines[1]=new String("Paul");
chaines[2]=new String("Jacques");
chaines[3]=new String("Philippe");
for (int i=0;i< chaines.length;i++)
{
System.out.println( " chaines[ " + i + " ] = "
+ chaines[i]);
}
}
}
```

Les tableaux (9)

 Copie de la référence d'un tableau:

– En Java, un tableau est un objet et le nom du tableau contient la référence de cet objet.

– L'affectation d'un tableau à un autre ne le duplique pas, mais affecte simplement une autre référence au même objet.

 Exemple:

– CopieRef.java (voir page suivante).

Pr. Mohamed NAIMI | Cours_Java | ESTB 67

Les tableaux (10)

```
public class CopieRef {
public static void main(String[] args) {
int[] a = {22, 44, 66, 55, 33};
System.out.println("a: " + a);
print(a);
int[] aa;
aa = a;
System.out.println("aa: " + aa);
print(aa);
a[3] = 88;
print(a);
print(aa);
aa[1] = 11;
print(a);
print(aa);
}
public static void print(int[] a) {
for (int i=0; i<a.length; i++)
System.out.print(a[i] + " ");
System.out.println();
}
}
```

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

Les tableaux (11)
Sortie du programme CopieRef.java:
a: [I@47858e
22 44 66 55 33
aa: [I@47858e
22 44 66 55 33
22 44 66 88 33
22 44 66 88 33
22 11 66 88 33
22 11 66 88 33

Les tableaux (12)
 Copie d'un tableau:
– Pour copier le tableau lui-même au lieu de la référence, on doit copier chaque élément qui le compose.
– Copie intuitive:
 Supposons que l'on dispose de 2 tableaux a[ ] et aa[ ] et que l'on désire copier les éléments de a[ ] dans aa[ ]:
– aa[0] = a[0];
– aa[1] = a[1]; // etc.
Pr. Mohamed NAIMI | Cours_Java | ESTB 70
Les tableaux (13)
– Copie en utilisant la méthode arraycopy( ):
 La méthode arraycopy( ) de la classe System permet de copier rapidement un tableau:
 System.arraycopy(Src, SrcPos, Dest, DestPos, Length);
– Src: le tableau source.
– SrcPos: l'index du premier élément du tableau source à copier.
– Dest: le tableau cible.
– DestPos: l'index où le premier élément doit être copié dans le tableau cible.
– Length: le nombre d'éléments à copier.
 Exemple:
– CopieTab.java (voir page suivante).

Les tableaux (14)

```
public class CopieTab {
public static void main(String[ ] args) {
int[ ] a = {22, 33, 44, 55, 66, 77, 88, 99};
System.out.println("a: " + a);
print(a);
int[ ] aa = new int[a.length];
System.out.println("aa: " + aa);
print(aa);
System.arraycopy(a, 0, aa, 0, a.length);
System.out.println("aa: " + aa);
print(aa);
aa[1] = 11;
print(a);
print(aa);
aa = new int[12];
System.arraycopy(a, 0, aa, 3, 8);
System.out.println("aa: " + aa);
print(aa);
System.arraycopy(aa, 3, aa, 1, 5);
```

```
System.out.println("aa: " + aa);
print(aa);
}
public static void print(int[ ] a) {
for (int i=0; i<a.length; i++)
System.out.print(a[i] + " ");
System.out.println();
}
}
```

Les tableaux (15)
Sortie du programme CopieTab.java:

```
a: [I@47858e
22 33 44 55 66 77 88 99
aa: [I@19134f4
0 0 0 0 0 0 0 0
aa: [I@19134f4
22 33 44 55 66 77 88 99
22 33 44 55 66 77 88 99
22 11 44 55 66 77 88 99
aa: [I@2bbd86
0 0 0 22 33 44 55 66 77 88 99 0
aa: [I@2bbd86
0 22 33 44 55 66 55 66 77 88 99 0
```

Les tableaux (16)

- La classe Arrays:

– La classe Arrays permet de traiter les tableaux. Elle est définie dans le paquetage java.util.
– Cette classe contient plusieurs méthodes utilitaires:
- sort(a): permet de trier les éléments du tableau a.
- equals(a, b): renvoie « true » si les tableaux a et b sont égaux.
- etc.

– Exemple:
- TestArrays.java (voir page suivante).

Les tableaux (17)

```
import java.util.Arrays;
public class TestArrays {
public static void main(String[] args) {
int[] a = {44, 77, 55, 22, 99, 88, 33, 66};
print(a);
Arrays.sort(a);
print(a);
int[] b = new int[8];
print(b);
System.arraycopy(a, 0, b, 0, a.length);
print(b);
System.out.println("Arrays.equals(a,b): " + Arrays.equals(a,b));
}
public static void print(int[] a) {
for (int i=0; i<a.length; i++)
System.out.print(a[i] + " ");
System.out.println();
}
```

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.

```
}
```

Les tableaux (18)
Sortie du programme TestArrays.java:
44 77 55 22 99 88 33 66
22 33 44 55 66 77 88 99
0 0 0 0 0 0 0 0
22 33 44 55 66 77 88 99
Arrays.equals(a,b): true

Les tableaux (19)
 Tableaux bidimensionnels:
– Les éléments d'un tableau peuvent être n'importe quel type d'objet.
– Ces éléments peuvent être donc des tableaux. Le cas échéant, nous obtenons ce qu'on appelle un tableau bidimensionnel (ou matrice).
– Syntaxe de déclaration et de création:
Type[ ][ ] Nom_Tableau = new Type[Nombre_Lignes][Nombre_Colonnes];
– Exemple:
 UneMatrice.java (voir page suivante).

Les tableaux (20)

```
public class UneMatrice {
public static void main(String[ ] args) {
int[ ][ ] M = new int[3][5];
for (int i=0; i<3; i++)
for (int j=0; j<5; j++)
M[i][j] = 10*(i+1) + j;
for (int i=0; i<3; i++)
print(M[i]);
System.out.println("M[2][4] = " + M[2][4]);
}
public static void print(int[ ] a) {
for (int i=0; i<a.length; i++)
System.out.print(a[i] + " ");
System.out.println();
}
}
```

Pr. Mohamed NAIMI | Cours_Java | ESTB 78
Les tableaux (21)
Sortie du programme UneMatrice.java:
10 11 12 13 14
20 21 22 23 24
30 31 32 33 34
M[2][4] = 34

Les chaînes: La classe String (1)
 Attention ce n'est pas un type de base. Il s'agit d'une classe défini dans l'API Java (Dans le package java.lang).
 Déclaration:
– String Nom_chaine;
 Déclaration et initialisation:
– String s = "aaa"; // s contient la chaîne "aaa"
– String s = new String("aaa"); // s contient la chaîne "aaa"

Les chaînes: La classe String (2)
 La concaténation:
– l'opérateur + permet de concaténer un nombre fini de chaînes de type String:
– Exemple:
 String str1 = "Bonjour ! ";
 String str2 = null;
 str2 = "Comment vas-tu ?";
 String str3 = str1 + str2; /* str3 contient "Bonjour ! Comment vas-tu ?" */

Les chaînes: La classe String (3)
 Longueur d'un objet String:
– La méthode length( ) renvoie la longueur d'une chaîne de type String.
– Exemple:
 String str1 = "bonjour";
 int n = str1.length( ); // n vaut 7

Les chaînes: La classe String (4)
 Les sous-chaînes:
– La méthode substring(int debut, int fin) permet d'extraire une sous-chaîne à partir d'une chaîne de type String.
 Extraction de la sous-chaîne depuis la position debut jusqu'à la position fin non-comprise.
 Exemple:
String str1 = "bonjour";
String str2 = str1.substring(0,3); // str2 contient la valeur "bon"
– Remarque:
 Le premier caractère d'une chaîne occupe la position 0.
 Le deuxième caractère occupe la position 1 et ainsi de suite.
Pr. Mohamed NAIMI | Cours_Java | ESTB 83
Les chaînes: La classe String (5)
 Récupération d'un caractère dans une chaîne:
– La méthode charAt(int Pos) permet de renvoyer le caractère situé à la position Pos dans la chaîne de caractères appelant cette méthode.
– Exemple:
String str1 = "bonjour";
char UNj = str1.charAt(3); // UNj contient le caractère 'j'

Les chaînes: La classe String (6)
 Modification de la casse des lettres dans une chaîne:
– La méthode toLowerCase( ) permet de transformer toutes les lettres majuscules en des lettres minuscules dans la chaîne appelant cette méthode.
– Exemple:
String str1 = "BONJOUR";
String str2 = str1.toLowerCase( ) // str2 contient la chaîne "bonjour"
– De même, la méthode toUpperCase( ) permet de transformer toutes les lettres minuscules en des lettres majuscules dans la chaîne appelant cette méthode.

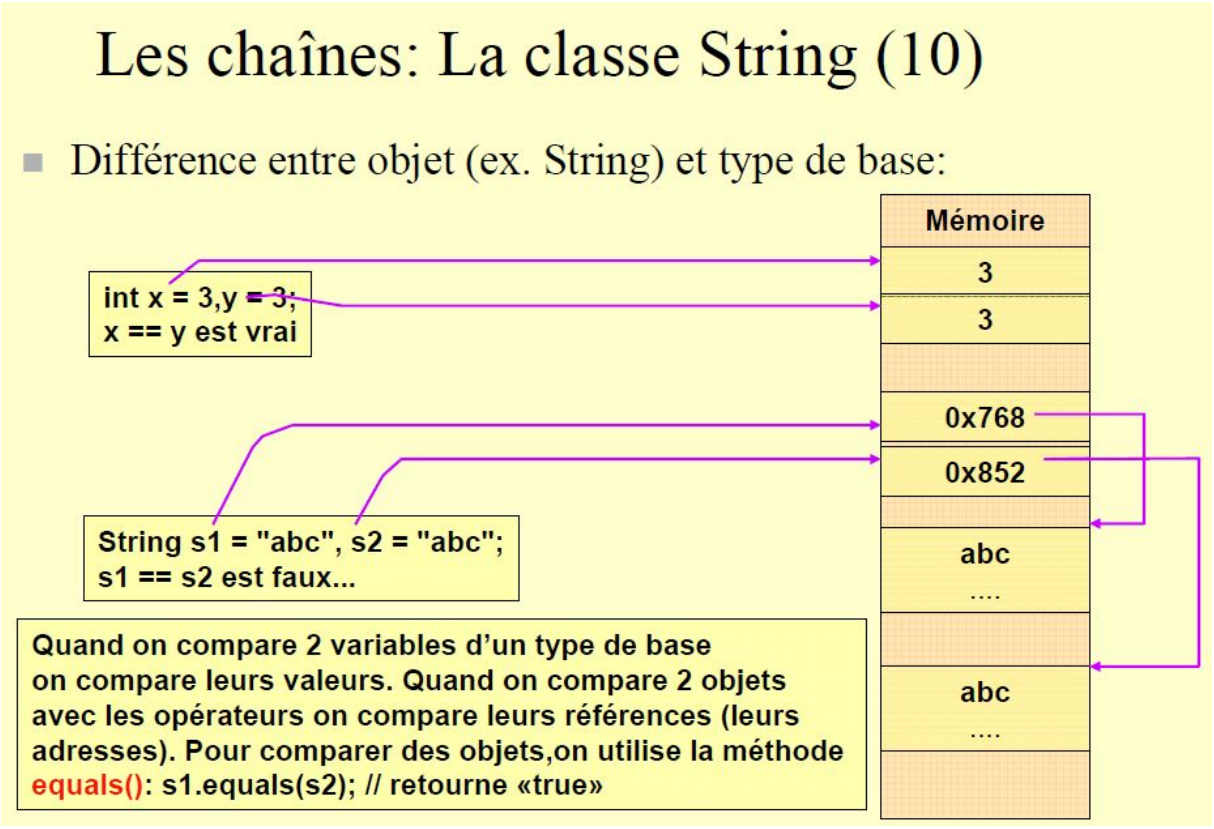
Les chaînes: La classe String (7)

 Modification des objets String:
– Les objets String sont immuables en Java, ce qui signifie qu'ils ne peuvent pas être modifiés.
– Remarque:
 Quand on a besoin de manipuler directement les chaînes de caractères, on peut utiliser la classe StringBuffer.

Les chaînes: La classe String (8)

```
class TestClasseString {
public static void main(String[ ] args) {
String alphabet = "ABCITALIEFGHIJKLMNOPQRSTUVWXYZ";
System.out.println("Cette chaine est: " + alphabet);
System.out.println("Sa longueur est: " + alphabet.length());
System.out.println("Le caractère de l'indice 4 est: "
+ alphabet.charAt(4));
System.out.println("Sa version en minuscules est: "
+ alphabet.toLowerCase());
System.out.println("Cette chaine est toujours: " + alphabet);
}
}
```

Exemple récapitulatif:



Les nombres aléatoires: La classe Random
 La classe Random est une classe utilitaire de l'API Java permettant de générer des nombres aléatoires.
 La classe Random est définie dans le paquetage java.util.
 Les méthodes les plus utilisées de la classe Random:
– La méthode nextInt( ) permet de renvoyer des nombres aléatoires

entiers répartis de façon uniforme dans l'intervalle comportant les entiers de type int (actuellement, de -2 147 483 648 à 2 147 483 647).
– La méthode nextInt(n) permet de renvoyer des nombres aléatoires entiers répartis de façon uniforme dans l'intervalle de 0 à n – 1.
– La méthode nextDouble( ) permet de renvoyer des nombres aléatoires de type double qui sont répartis uniformément dans l'intervalle de 0 à 1.

Les nombres aléatoires: La classe Random

 Exemple:

```
import java.util.Random;
public class EntierAleatoire {
public static void main(String[ ] args) {
Random random = new Random( );
int n = random.nextInt( );
System.out.println("n = " + n);
if (n < 0)
System.out.println("**** n < 0");
else
System.out.println("**** n >= 0");
}
}
```

E-mail : tibousow@gmail.com ; elabdoulayesow30@yahoo.fr;
Tel : 622 29 43 67 / 666 82 89 70.