Université Sid Mohammed Ben Abdellah
Ecole Nationale des Sciences Appliquées
Fès

Programmation Orientée Objet en Java

Pr. Nabil EL AKKAD

Intitulé du module	*JAVA avancé*
Etablissement dont relève le module	*Ecole Nationale des Sciences Appliquées de Fès (ENSAF)*
Département d'attache	*Département Génie Electrique et Informatique (GEI)*
Master, (Semestre d'appartenance du module)	*Master Internet des Objets et Systèmes Mobiles (IOSM) (S2)*

Plan du cours

1. Caractéristiques du langage Java
2. Syntaxe et éléments de bases du langage Java
3. Concepts de la POO en Java
4. Gestion des exceptions
5. Flux d'entrées /Sorties
6. Accès aux bases de données avec l'API JDBC
7. Threads
8. Colletions
9. Programmation réseau
10. Applets
11. Interfaces graphiques

Caractéristiques du langage Java

Caractéristiques du langage Java

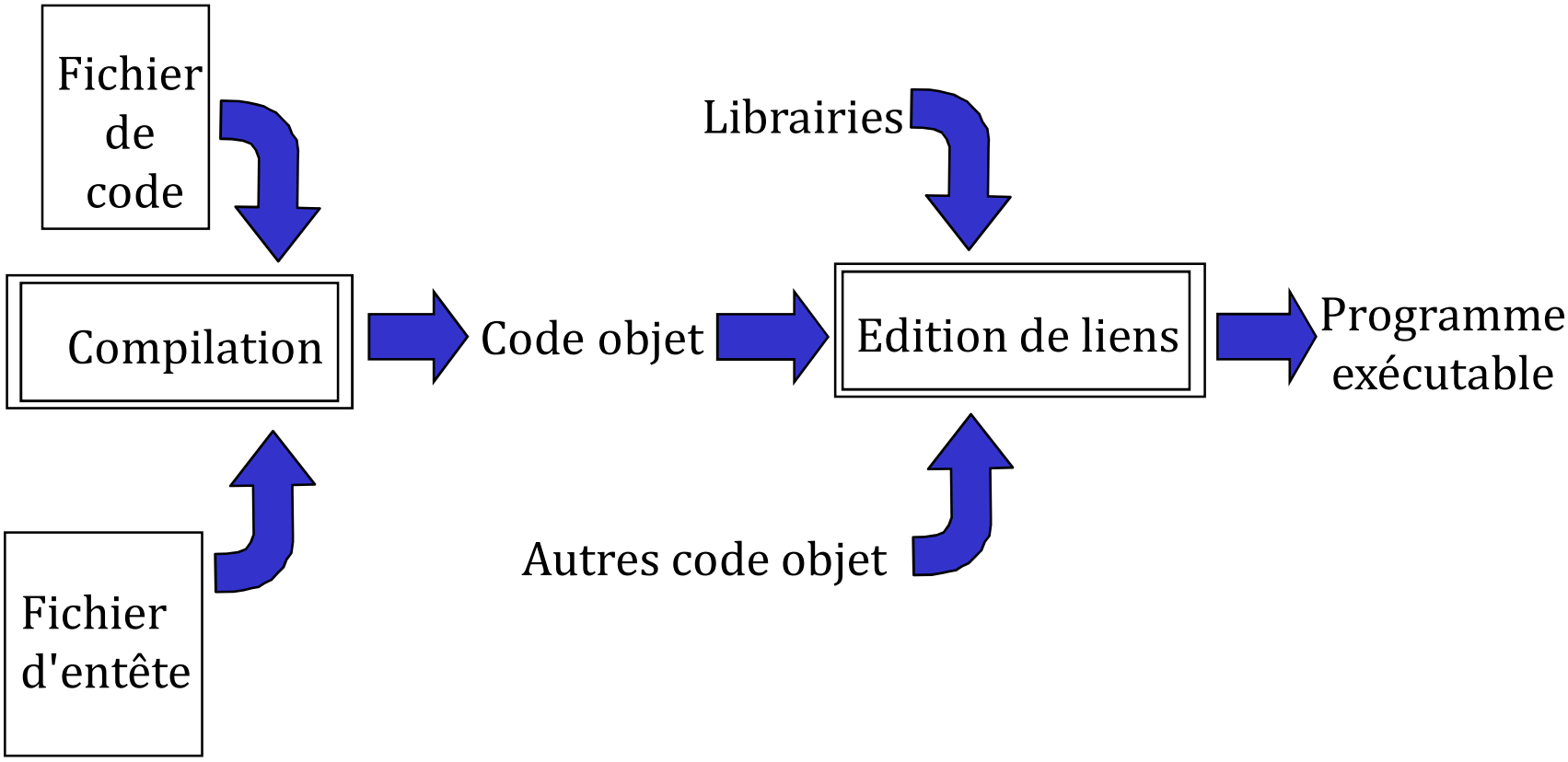
- ***Langage simple***
  - On peut l'apprendre facilement :
    - Peu de mots-clés
    - Facilité des fonctionnalités principales
  - Se base sur les concepts fondamentaux des langages C et C++.
- ***Langage orienté objet***
  - Java ne permet d'utiliser que des objets, sauf les types de base.
  - Java est un *langage objet* de la famille des langages de *classe* comme C++ ou SmallTalk.
  - Les grandes idées reprises sont : encapsulation, dualité classe /instance, attribut, méthode / message, visibilité, dualité interface/implémentation, héritage simple, surcharge des méthodes, polymorphisme.

Caractéristiques du langage Java (2)

- ***Langage fiable***
  - Sources d'erreurs limitées.
    - Pas de pointeurs,
    - Pas structures,
    - Pas héritage multiple,
    - pas de surcharge des opérateurs,
    - Vérifications faites par le compilateur facilitant une plus grande rigueur du code.
  - Gestion automatique de la mémoire *(ramasse-miette ou "garbage collector").*
  - Gestion des exceptions

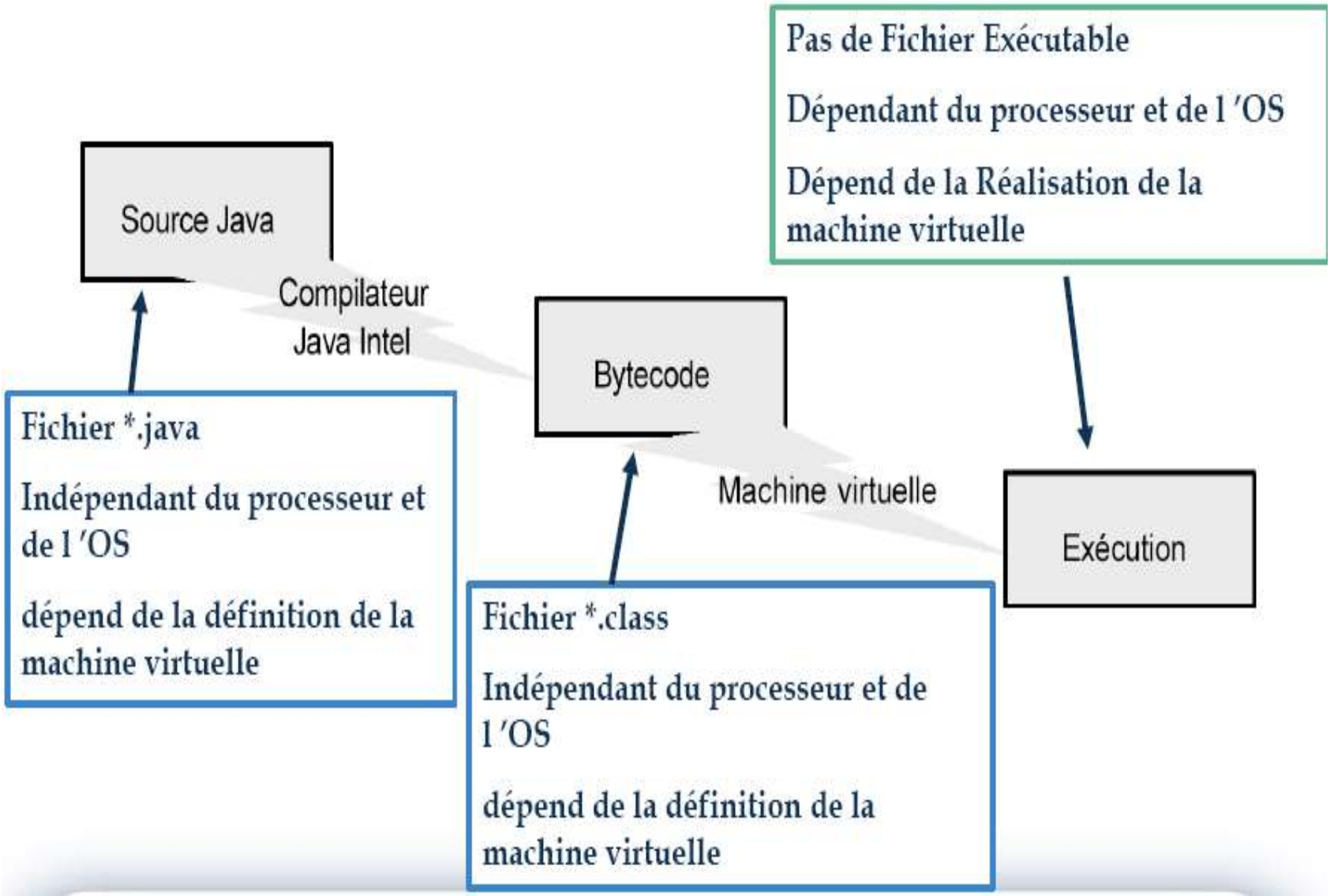
Caractéristiques du langage Java (3)

Langage compilé: étapes qui ont lieu avant l'exécution pour un langage de ce type comme C++.



Caractéristiques du langage Java (4)

***Langage interprété* :** cas de Java

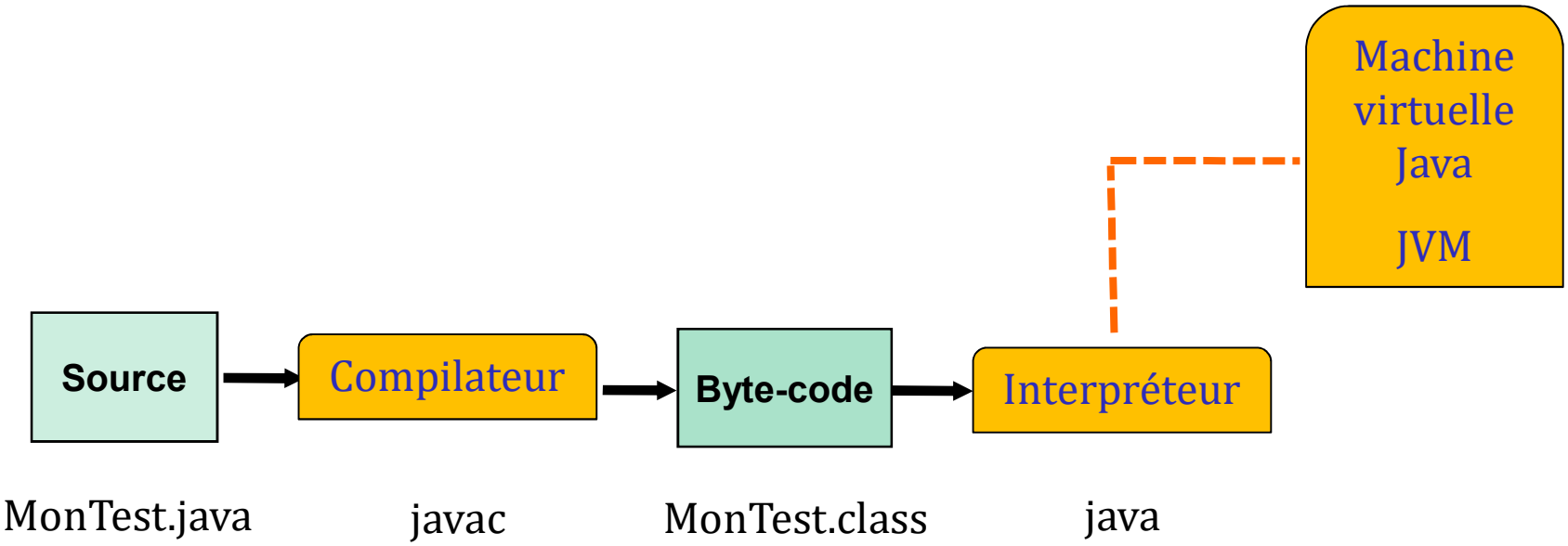


Caractéristiques du langage Java (5)

- ***Source Java***
  - ➢Fichier utilisé lors de la phase de programmation.
  - ➢Le seul fichier réellement intelligible par le programmeur.
- ***Byte-code Java***
  - ➢Code objet destiné à être exécuté sur toute « Machine virtuelle » Java
  - ➢Provient de la compilation du code source.
- ***Machines Virtuelles Java***
  - ➢Les machines virtuelles Java peuvent être:
    - ❖Des interpréteurs de byte-code indépendants (pour exécuter les programmes Java).
    - ❖Contenues au sein d'un navigateur (pour exécuter des applets Java).
  - ➢**Conclusion** : il suffit de disposer d'une « Machine virtuelle » Java pour pouvoir exécuter tout programme Java même s'il a été compilé avec un autre système d'exploitation.

Caractéristiques du langage Java (6)

Compilation **Interprétation**



Caractéristiques du langage Java (7)

Etapes de développement

- **Création du code source**
  - A partir des spécifications (par exemple en UML)
  - Outil : éditeur de texte, IDE
- **Compilation en Byte-code**
  - A partir du code source
  - Outil : compilateur Java
- **Diffusion sur l'architecture cible**
  - Transfert du Byte-code seul
  - Outils : réseau, disque, …
- **Exécution sur la machine cible**
  - Exécution du Byte-code
  - Outil: Machine Virtuelle Java

Caractéristiques du langage Java (8)

- **Langage sûr**
  - Seul le **byte-code** est transmis, et «vérifié» par l'interpréteur.
  - Impossibilité d'accéder à des fonctions globales ou des ressources arbitraires du système.

Java est indépendant de l'architecture

- Le byte-code généré par le compilateur est indépendant de toute architecture. Toute application peut donc tourner sur une plate-forme implémentant une **machine virtuelle Java**.

« Ecrire une fois, exécuter partout ».

Java est multi-tâches

- Exécution de plusieurs processus effectuant chacun une tâche différente.
- Mécanismes de synchronisation.
- Fonctionnement sur des machines multiprocesseurs.

Caractéristiques du langage Java (9)

- ***Java est multithread :*** gestion de plusieurs tâches prévues.
- ***Deux types de programmation différents:***

➢**Applications** : programmes écrits en Java et exécutés sur une machine qui possède un interpréteur Java.

➢**Applets** (mini-programmes) : Programmes exécutables uniquement par l'intermédiaire d'une autre application.

- ***Navigateur web*** : Netscape, Internet explorer, Hotjava
- ***Application spécifique*** : Appletviewer.

➢Java est souvent considéré comme étant uniquement un langage pour écrire des applets alors que c'est aussi un vrai langage de programmation.

Caractéristiques du langage Java (10)

Java, un langage novateur?

- Java n'est pas un langage novateur : il a puisé ses concepts dans d'autres langages existants et sa syntaxe s'inspire de celle du C++.
- Cette philosophie permet à Java :
  - De ne pas dérouter ses utilisateurs en faisant "presque comme ... mais pas tout à fait"
  - D'utiliser des idées, concepts et techniques qui ont fait leurs preuves et que les programmeurs savent utiliser.
- En fait, Java a fait une synthèse efficace de bonnes idées issues de sources d'inspiration variées.
  - Smalltalk, C++, Ada, etc.

Caractéristiques du langage Java (11)

Java et ses versions

- **Quelques versions de la Machine Virtuelle**
  - Java 2 Micro Edition (Java ME): cible les terminaux portables.
  - Java 2 Standard Edition (Java SE) : cible les postes clients.
  - Java 2 Entreprise Edition (Java EE) : définit le cadre d'un serveur d'application.
- **Différentes finalités**
  - SDK (Software Development Kit) : fournit un compilateur et une machine virtuelle.
  - JRE (Java Runtime Environment) : fournit uniquement une machine virtuelle. Idéal pour le déploiement de vos applications.

Caractéristiques du langage Java (12)

Outil de développement : le JDK

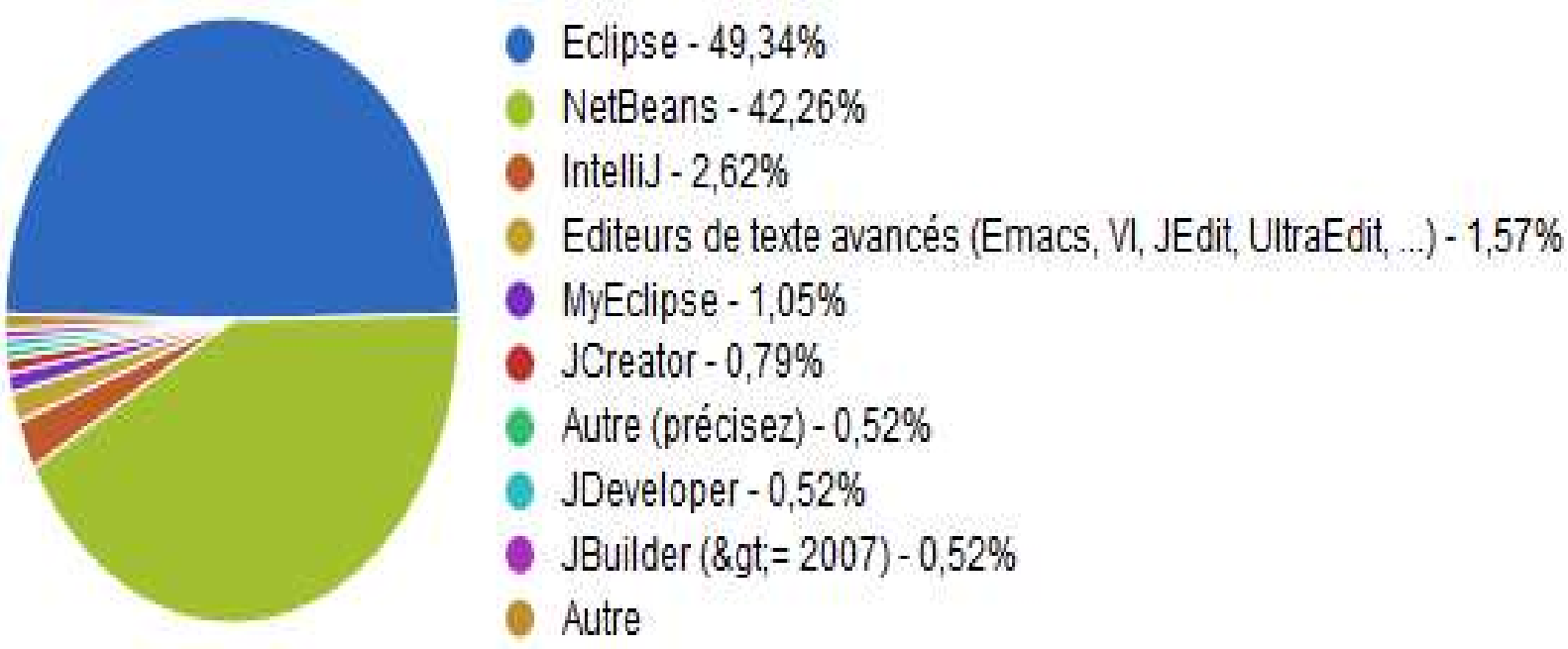
- Environnement de développement fourni par Sun
- JDK signifie Java Development Kit (Kit de développement Java).
- Il contient :
  - Les classes de base de l'API java (plusieurs centaines),
  - La documentation au format HTML
  - Le compilateur : **javac**
  - La JVM (machine virtuelle) : **java**
  - Le visualiseur d'applets : **appletviewer**
  - Le générateur de documentation : **javadoc**
  - etc.

Caractéristiques du langage Java (13)

- ***Les outils Java :***

Simples éditeurs ou environnements de développement

Quel EDI Java utilisez vous en 2019 ?



Sondage: https://java.developpez.com/actu/705/Sondage-Quel-EDI-Java-utilisez-vous-et-pourquoi/

API de Java

- Java fournit de nombreuses bibliothèques de classes remplissant des fonctionnalités très diverses : c'est l'API Java
  - **API** (Application Programming Interface /Interface de programmation des applications) : Ensemble de librairies permettant une programmation plus aisée, car les fonctions deviennent indépendantes du matériel.
- Ces classes sont regroupées, par catégories, en paquetages (ou "packages").

Un package est un ensemble de classes. En fait, c'est un ensemble de dossiers et de sous-dossiers contenant une ou plusieurs classes.

API de Java (2)

- Les principaux paquetages (packages)
  - java.util : structures de données classiques (vecteurs, hashtable).
  - java.io : entrées / sorties (flux, fichiers).
  - java.lang : chaînes de caractères, interaction avec l'OS, threads.
  - java.math : gestion de grands nombres.
  - java.applet : les applets sur le web.
  - java.awt : interfaces graphiques, images et dessins.
  - javax.swing : package récent proposant des composants « légers » pour la création d'interfaces graphiques.
  - java.net : classes de support réseau (URL, sockets).
  - java.rmi : (Remote Method Invocation), contient les méthodes invoquées à partir des machines virtuelles non locales.
  - java.sql : fournit le package JDBC.

API de Java (3)

La documentation de Java

- La documentation de Java est standard, que ce soit pour les classes de l'API ou pour les classes utilisateur.
  - Possibilité de génération automatique avec l'outil Javadoc.
- Elle est au format HTML.
  - Intérêt de l'hypertexte pour naviguer dans la documentation .
- Pour chaque classe, il y a une page HTML contenant :
  - L'hiérarchie d'héritage de la classe.
  - Une description de la classe et son but général.
  - La liste des attributs de la classe (locaux et hérités).
  - La liste des constructeurs de la classe (locaux et hérités).
  - La liste des méthodes de la classe (locaux et hérités).
  - Puis, chacune de ces trois dernières listes, avec la description détaillée de chaque élément.

API de Java (4)

Documentation de l'API Java SE 8 :

http://docs.oracle.com/javase/8/docs/api/

Avantages et inconvénients du langage Java

- ***Avantages :***
  - ***Portabilité***
    - Des machines virtuelles Java existent pour de nombreuses plates-formes dont : Linux, Windows, MacOS
  - ***Développement plus rapide***
    - Courte étape de compilation pour obtenir le byte-code.
    - Pas d'édition de liens.
    - Débogage plus aisé.
  - ***Le byte-code est plus compact que les exécutables***
    - Pour voyager sur les réseaux.

Avantages et inconvénients du langage Java (2)

- ***Inconvénients :***
  - Nécessite l'installation d'un interpréteur pour pouvoir exécuter un programme Java.
    - ✓ **un interpréteur** est un outil ayant pour tâche d'analyser, de traduire et d'exécuter les programmes écrits dans un langage informatique.
  - L'interprétation du code ralentit l'exécution.
  - Les applications ne bénéficient que du dénominateur commun des différentes plate-formes.
    - ✓ Limitation, par exemple, des interfaces graphiques.
  - Gestion gourmande de la mémoire.
  - Impossibilité d'opérations de « bas niveau » liées au matériel.

Syntaxe et éléments de base du langage Java

Instructions, blocs et blancs

- Les instructions Java se terminent par un ;
- Les blocs sont délimités par :

  { 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.

Point d'entrée d'un programme Java

Pour pouvoir faire un programme exécutable, il faut toujours une classe contenant 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 args[ ])
 {
          .../...
 }
```

Premier exemple de programme Java

Fichier PremTest.java

```
public class PremTest
{   //Accolade débutant la classe PremTest
 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 PremTest
```

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

Les instructions se terminent par des ;

Premier exemple de programme Java (2)

Fichier PremTest.java

```
public class PremTest
{
 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 chaîne de caractères.

- **public class** premTest**{…..}**: ce bloc correspond la définition d'une classe nommé PremTest .
- **public static void main (String []** args**){…}** : ce bloc permet de définir la méthode particulière nommée main.
- Le **main** () est une fonction interne de la classe.

Premier exemple de programme Java (3)

- public : obligatoire pour que votre programme puisse s'exécuter.
- static précise que la méthode main de la classe **PremTest** n'est pas liée à une instance (objet) particulière de la classe.
- String[] args : permet de récupérer des arguments transmis au programme au moment de son lancement. On peut lancer un programme sans fournir d'arguments, mais l'indication String args[] est obligatoire.

- **System.out.println ("Bonjour") ;** cette ligne sert à définir le contenu de notre programme qui va afficher le message **Bonjour** suivi d'un retour à la ligne suivante dans la fenêtre de console.

Compilation et exécution

Fichier PremTest.java

Le nom du fichier doit être le même que celui de la classe avec l'extension .java en plus. Java est sensible à la casse des lettres.

Compilation en bytecode java dans une console DOS:

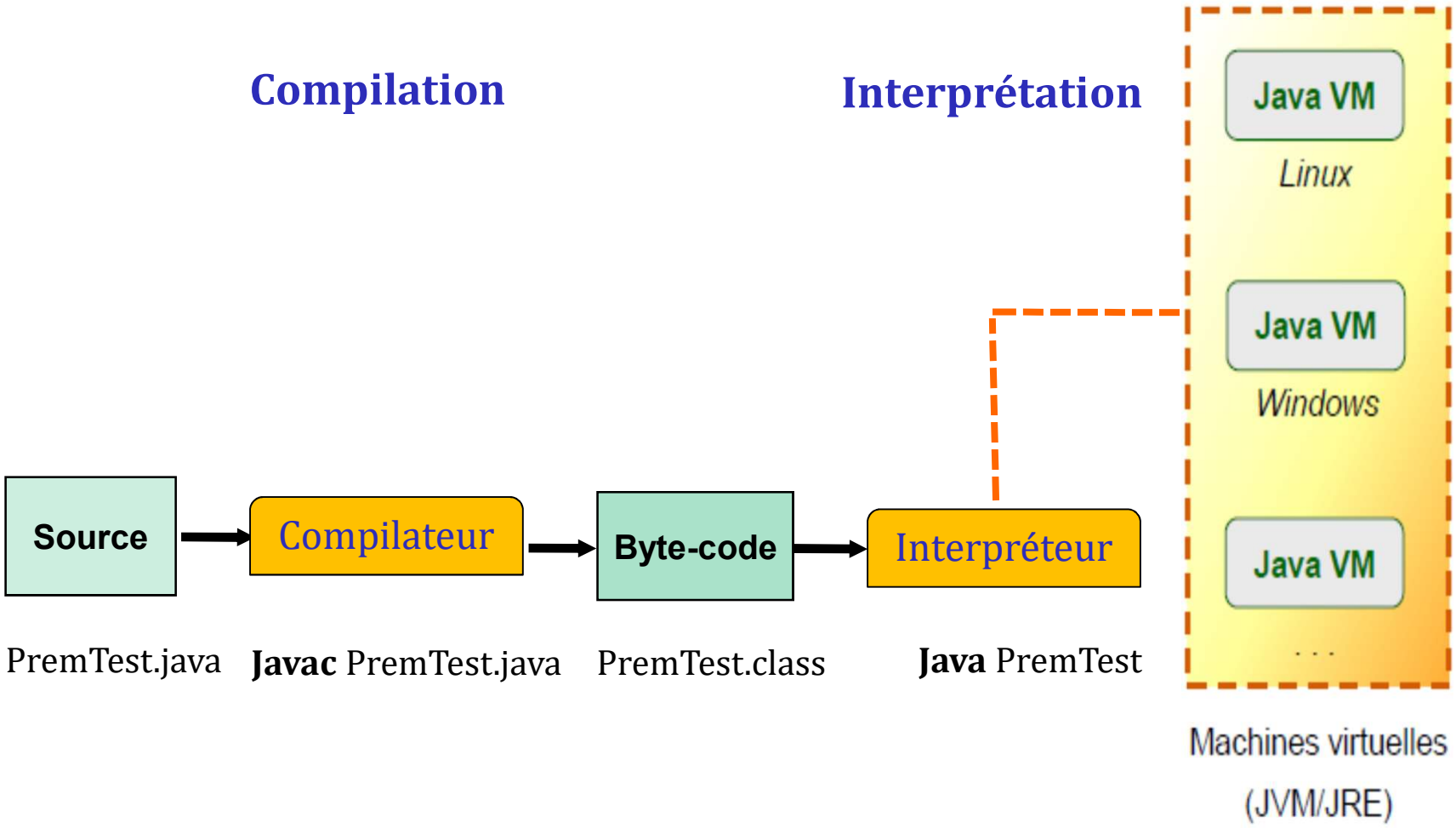
javac PremTest.java
Génère un fichier PremTest.class

Exécution du programme (toujours depuis la console DOS) sur la JVM :

java PremTest
Affichage de "Bonjour" dans la console

```
public class PremTest
{
 public static void main(String[] args)
 {
  System.out.println("Bonjour");
 }
}
```

Compilation et exécution (2)



Compilation et exécution (2)

Pour résumer, dans une console **DOS**, on *la méthode main ()* dans la classe *PremTest* :

- **javac** PremTest.java
  - Compilation en Byte-code java.
  - Indication des erreurs de syntaxe éventuelles.
  - Génération d'un fichier *PremTest.class* si pas d'erreurs
- **java** PremTest
  - Java est la machine virtuelle.
  - Exécution du Byte-code.
  - Nécessité de la méthode main, qui est le point d'entrée dans le programme.

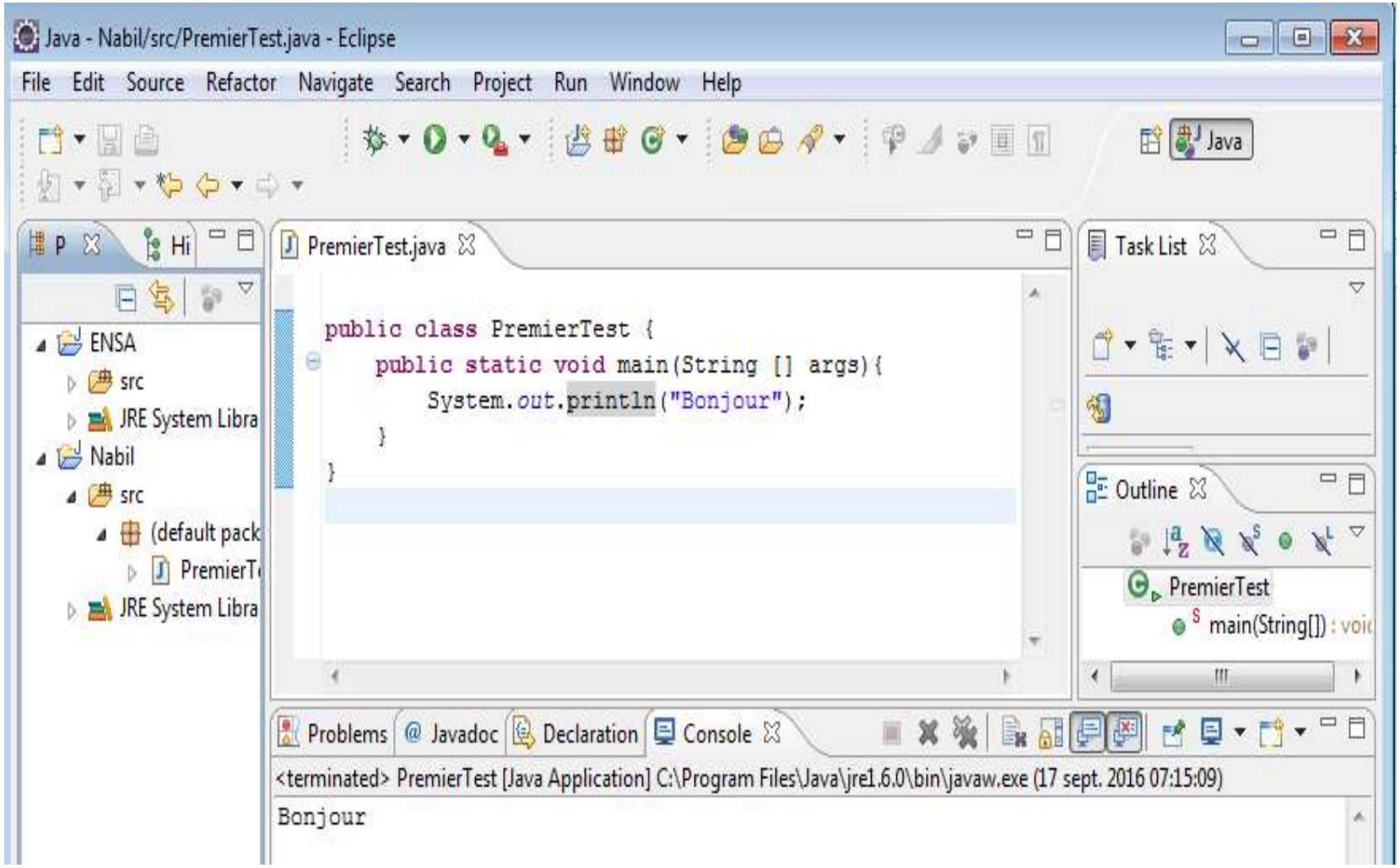
Exécution en mode enivrement de développement

- Le traitement et l'exécution des différents programmes Java seront effectués par le fameux environnement de développement, qui est **Eclipse**.
- Donc, on va utiliser désormais la dernière version d'Eclipse (**Eclipse IDE for Java Developers**), qu'on peut la télécharger de la page: **http://www.eclipse.org/downloads**.
- Eclipse a besoin d'un JRE ou d'un JDK pour fonctionner. Donc, installer JDK au lieu d'un JRE pour bénéficier de la Javadoc et du code source de l'API Java Standard. La version de la JDK (8) est disponible sur le site suivant:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Exécution en mode enivrement de développement (2)

- Premier programme réalisée dans IDE Eclipse:



Lecture d'informations au clavier

- Pour la saisie de données au clavier, on va utiliser la classe **Scanner** (java.util.Scanner) qui regroupe plusieurs méthodes pratiques :
  - nextInt(),
  - nextDouble(),
  - nextFloat(),
  - nextLong(),
  - nextLine(),
  - ...

Exemple:

```
public class AfficheEntier{
public static void main(String[] args) {
   Scanner sc=new Scanner(System.in);//System.in: Entrée Standard
   System.out.println("Entrer un entier:");
   int a=sc.nextInt();
   System.out.println(" le nombre entré est :" + a);
 }}
```

Lecture d'informations au clavier (2)

Noté bien: si vous avez invoqué **nextInt()**, **nextDouble()** et que vous invoquez directement après **nextLine**(), celle-ci ne vous invite pas à saisir une chaîne de caractères : elle vide la ligne commencée par les autres instructions. En effet, celles-ci ne repositionnent pas la tête de lecture, l'instruction nextLine() le fait à leur place.

Exemple:

```
import java.util.Scanner;
public class Main {
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    System.out.println("Saisissez un entier : ");
    int i = sc.nextInt();
    System.out.println("Saisissez une chaîne : ");
    String str = sc.nextLine();
    System.out.println("FIN ! ");
   }}
```

Lecture d'informations au clavier (3)

Le pg précédent ne vous demande pas de saisir une chaîne et affiche directement « Fin ». Pour pallier ce problème, il suffit de vider la ligne après les instructions ne le faisant pas automatiquement :

```
import java.util.Scanner;
public class Main {
public static void main(String[] args){
Scanner sc = new Scanner(System.in);
System.out.println("Saisissez un entier : ");
int i = sc.nextInt();
System.out.println("Saisissez une chaîne : ");
//On vide la ligne avant d'en lire une autre
sc.nextLine();
String str = sc.nextLine();
System.out.println("FIN ! ");
} }
```

Lecture d'informations au clavier (4)

En résumé

- L'affichage des données se fait par **System.out.println** ().
- La lecture des entrées clavier se fait via l'objet **Scanner.**
- Ce dernier se trouve dans le package **java.util** que vous devez importer.
- Pour pouvoir récupérer ce que vous allez taper dans la console, vous devrez initialiser l'objet **Scanner** avec l'entrée standard, **System.in.**
- Il y a une méthode de récupération de données pour chaque type (sauf les char) : **nextLine()** pour les String, **nextInt()** pour les int **...**

Identificateurs

- On a besoin de nommer les classes, les variables, les constantes, etc. ; on parle *d'identificateur*.
- Les identificateurs commencent par une lettre, _ ou $

  Attention : Java distingue les *majuscules* des *minuscules* .
- **Conventions sur les identificateurs :**

  Si plusieurs mots sont accolés, mettez d'une part une majuscule à chacun des mots sauf le premier et d'autre part suivez ces directives:

  - La première lettre est majuscule pour les classes et les interfaces.
    - ✓ **exemples** : MaClasse, UneJolieFenetre
  - La première lettre est minuscule pour les méthodes, les attributs et les variables.
    - **exemples** : setLongueur, i, uneFenetre.
  - Les constantes sont entièrement en majuscules.
    - **exemple** : LONGUEUR_MAX

Les commentaires

//…Texte…

- Commence dès // et se termine à la fin de la ligne.
- Sur une seule ligne.
- A utiliser de préférence pour les commentaires généraux.

/* …Texte…*/

- Le texte entre /* et */ est ignoré par le compilateur.
- Peuvent s'étendre sur plusieurs lignes.
- Ne peuvent pas être imbriqués.
- Peuvent être utilisés pour désactiver, temporairement, une zone de code.

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 bases

- 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 ou 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 diverses tailles : **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 types de bases (2)

- Quelques opérations sur les entiers
  - les opérateurs d'incrémentation **++** et de décrémentation et **--**
    - ajoute ou retranche 1 à une variable

      int **n** = **12**;

      n ++; //Maintenant **n** vaut **13**
    - **n++**; « équivalent à » **n = n+1**; **n--**; « équivalent à » **n = n-1**;
    - **8++**; est une instruction illégale.
    - Peut s'utiliser de manière suffixée : **++n**. La différence avec la version préfixée se voit quand on les utilisent dans les expressions.

      En version suffixée l'incrémentation ou la décrémentation s'effectue en premier.

Exemples:

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 opérateurs

Opérateurs arithmétiques

	Symbole	Description	Exemple
Opérateurs arithmétiques	-	*soustraction*	*x-y*
	*	*multiplication*	*3 *x*
	/	*division*	*4/2*
	%	*modulo (reste de la division)*	*5%2*

Opérateurs d'affectations

	Symbole	Description	Exemple
Opérateurs d'affectation	**=**	*Affectation*	***x=2***
	-=	*Soustraction et affectation*	***x-=2***
	+=	*Addition et affectation*	***x+=2***

Les opérateurs (2)

Opérateurs d'incrémentations et décrémentations

	Symbole	Description	Exemple
Opérateurs d'incrémentations et décrémentations	*++*	*Pré-incrémentation*	*++x*
	++	*Post-incrémentation*	*x++*
	--	*Pré-décrémentation*	*--x*
	--	*Post-décrémentation*	*x--*

Opérateurs relationnels

	Symbole	Description	Exemple
Opérateurs relationnels	*==*	*égal à*	*x==2*
	<	*inférieur à*	*x<2*
	<=	*inférieur ou égal à*	*x<=3*
	>	*supérieure à*	*x>2*
	>=	*supérieur ou égal à*	*x>=3*
	!=	*différent de*	*a !=b*

Les opérateurs (3)

Opérateurs logiques

	Symbole	Description	Exemple
Opérateurs logiques	*&&*	*et*	*a && b*
	||	*ou*	*a ||b*
	!	*non*	*!a*

Opérateurs bit à bit

	Symbole	Description	Exemple
Opérateurs relationnels	*&*	*et*	*a&b*
	|	*ou*	*a|b*
	^	*ou exclusif*	*a^b*
	~	*nom*	*~x*
	<<	*décalage à gauche*	*a<<3*
	>>	*décalage à droite*	*b>>2*

Les structures de contrôle

- Les structures de contrôle classiques existent en Java :
  - **if, else**
  - **switch, case, default, break**
  - **for**
  - **while**
  - **do, while**

Les structures de contrôle (2)

Structure *if*

```
if (expression logique) instruction;
```

```
if (expression logique)
    {       instruction;
            instruction;
            . . .
    }
```

Structure *if else*

```
if (expression logique) instruction;
            else instruction;
```

```
if (expression logique)
    {       instruction;
            instruction;
            . . .
     }
else
          {  instruction;
               instruction;
               . . .
           }
```

Les structures de contrôle (3)

Boucle *for*

```
for (initialisation; condition ; incrémentation ou décrémentation)
{
        instruction_1;
        instruction_2;
        . . .
        instruction_n;
}
```

Boucle *while*

```
while (expression booléenne)
{
        instruction_1;
        instruction_2;
        . . .
        instruction_n;
}
```

Les structures de contrôle (4)

Boucle *do while*

```
do
{
        instruction_1;
        instruction_2;
        . . .
        instruction_n;
}  while (condition);
```

Exemple: calcul de la somme des dix premiers entiers positifs.

```
public class Counter {
    public static void main (String args[]){
        int somme=0, indice=1;
        do{
            somme += indice++;
        } while (indice<=10) ;
        String str= "La somme des dix premiers entiers positifs est : ";
        System.out.println(str + somme);
    }}
```

Les structures de contrôle (4)

L'instruction *switch :* sélectionne un morceau de code parmi d'autres en se basant sur la valeur d'une expression entière

```
switch (variable){
        case valeur1 : instructions1 ;
        case valeur2 : instructions2 ;
                        ...
        default : instructions ;
}
```

Exemples:

```
for(int i = 0; i < 100; i++) {
    char c = (char)(Math.random() * 26 + 'a');
    System.out.print (c + ": ");
    switch(c) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        case 'y': System.out.println(" voyelle "); break;
        default:  System.out.println(" consonne");
    }}
```

Les structures de contrôle (5)

L'instruction *break* : Permet de terminer l'exécution d'une boucle.

Exemple :// Impression des nombres premiers entre 2 et 50.

```
int n = 50;
boolean Premier = true;
for (int i = 2; i <= n; i++)
{
        Premier = true;
        for (int j = 2; j < i; j++)
        {
                if (i % j == 0)
                {
                        Premier = false;
                        break;
                }
        }
        if (Premier) System.out.println(i);
}
```

Les structures de contrôle (6)

L'instruction *continue* : permet l'interruption d'une itération en cours et retourne au début de la boucle avec exécution de la partie incrémentation.

Exemple : la somme des entiers impairs.

```
int somme = 0;
        for (int i = 0; i < 100; i++)
        {
                if (i % 2 == 0) continue;
                somme += i;
        }
```

Les tableaux

Les tableaux Java sont des structures pouvant contenir un nombre fixe d'éléments de même nature. Il peut s'agir d'objets ou de primitives.

Tableaux unidimensionnels

Déclaration : ***type nomTableau [] ;*** ou ***type [] nomTableau ;***

Exemple : int t[] ; peut aussi s'écrire : int [] t ;

La différence entre les deux formes :

int [] t1, t2 ; équivalent à int t1[], t2[] ;

int t1[], n, t2[] ; // mélange entre tableaux d'entiers et primitive entier

Les éléments d'un tableau peuvent être d'un type primitif ou d'un type objet:

Point tp [] ; // tp est une référence à un tableau d'objets de type Point

Point a, tp[], b ; // a et b sont des références à des objets de type Point.
tp est une référence à un tableau d'objets de type Point

Les tableaux (2)

Tableaux unidimensionnels (2)

Création

La création d'un tableau dans java est réalisée par l'opérateur **new** :

nomTableau ***=*** **new** ***type*** *[*dimension*];*

Exemple : int t[]= **new** int[50]; ou int [] t = **new** int[50] ;

Initialisation

1- **Automatique**: Chaque élément du tableau est initialisé selon son type par l'instruction ***new*** :

- ***0*** pour les numériques,
- ***\0*** pour les caractères,
- ***false*** pour les booléens
- **null** pour les chaînes de caractères et les autres objets.

2- On peut initialiser un tableau au moment de sa déclaration par la syntaxe suivante :

int [] x={1, 2, 3, 4, 5} ;// tableau de cinq éléments de type ***int*** initialisé par{1, 2, 3, 4, 5}

Les tableaux (3)

Tableaux unidimensionnels (3)

Taille d'un tableau

Un tableau est un objet possédant l'attribut ***length*** : c'est la taille du tableau.

Exemple :

```
int t[] = new int[5] ;
System.out.println ("taille de t : " + t.length) ;   // affiche 5
t = new int[3] ;
System.out.println ("taille de t : " + t.length) ;   // affiche 3
```

Utilisation d'un tableau

Accès individuelle à chaque élément du tableau

Exemple: voici un programme qui utilise un tableau de réels pour déterminer le nombre d'élèves d'une classe ayant une note supérieure à la moyenne de la classe.

Les tableaux (4)

Tableaux unidimensionnels (4)

```
public class Moyenne{
 public static void main (String args[]) {
 int i, nbEl, nbElSupMoy ;
double somme ;
double moyenne ;
Scanner sc=new Scanner (System.in);
System.out.print ("Combien d'élèves ") ;
 nbEl = sc.nextInt();
double notes[] = new double[nbEl] ;
for (i=0 ; i<nbEl ; i++){
    System.out.print ("donnez la note numéro " + (i+1) + " : " ) ;
    notes[i] = sc.nextDouble() ;}
    for (i=0, somme=0 ; i<nbEl ; i++) {
      somme += notes[i] ;}
      moyenne = somme / nbEl ;
      System.out.println ("\n moyenne de la classe  " + moyenne) ;
      for (i=0, nbElSupMoy=0 ; i<nbEl ; i++ )
       if (notes[i] > moyenne)
         nbElSupMoy++ ;
     System.out.println (nbElSupMoy + " élèves ont plus de cette moyenne") ;
 }}
```

```
Combien d'élèves 5
donnez la note numéro 1 : 12
donnez la note numéro 2 : 14.5
donnez la note numéro 3 : 10
donnez la note numéro 4 : 9
donnez la note numéro 5 : 16
moyenne de la classe  12.3
2 élèves ont plus de cette moyenne
```

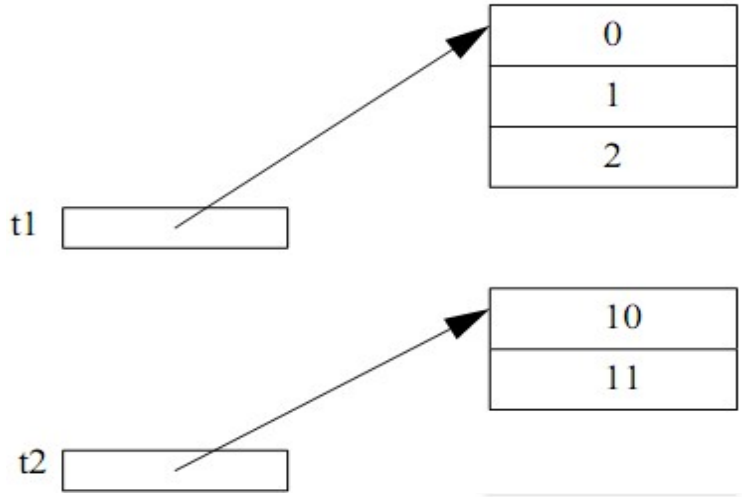
Les tableaux (5)

1. Tableaux unidimensionnels (5)

accès globale de tableau: affectation

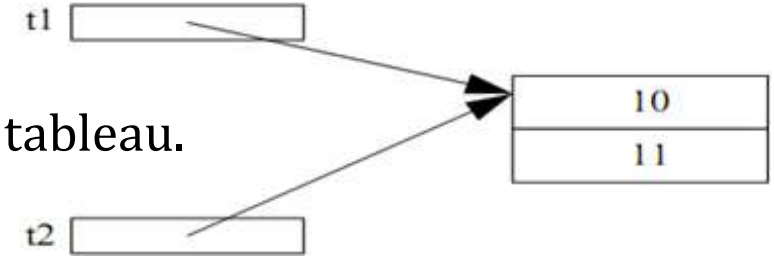
Exemple:

```
int [] t1 = new int[3] ;
for (int i=0 ; i<3 ; i++)
        t1[i] = i ;
int [] t2 = new int[2] ;
for (int i=0 ; i<2 ; i++)
        t2[i] = 10 + i ;
```



On exécute maintenant l'affectation

t1 = t2 ; // la référence contenue dans t2 est recopiée dans t1



Dorénavant, t1 et t2 désignent le même tableau.

Donc, si on écrit: t1[0] = 5 ;

Alors l'instruction : System.out.println (t2[0]) affichera la valeur 5, et non 10.

Les tableaux (6)

Tableaux multidimensionnels

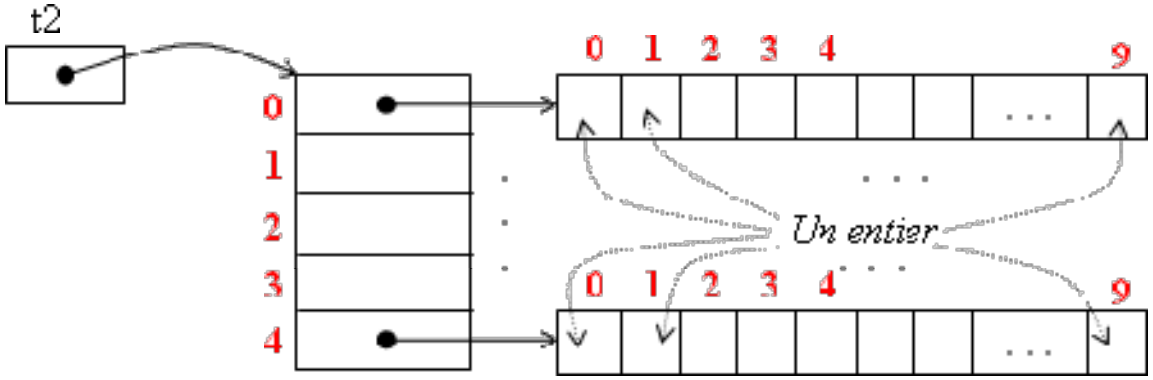
Déclaration et création

type nomTableau [] []= new type[n1][n2];

ou

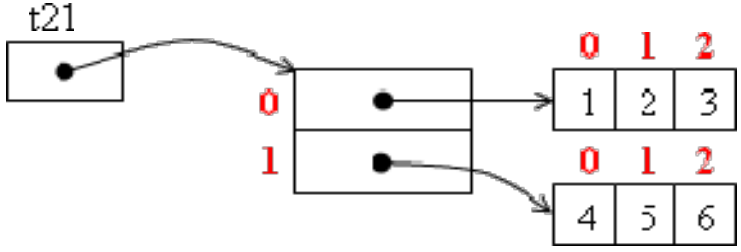
type [] [] nomTableau = new type[n1][n2];

Exemple : int t2[][]= new int [5][10] ;



Un tableau à plusieurs dimensions peut être initialisé comme suit:

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



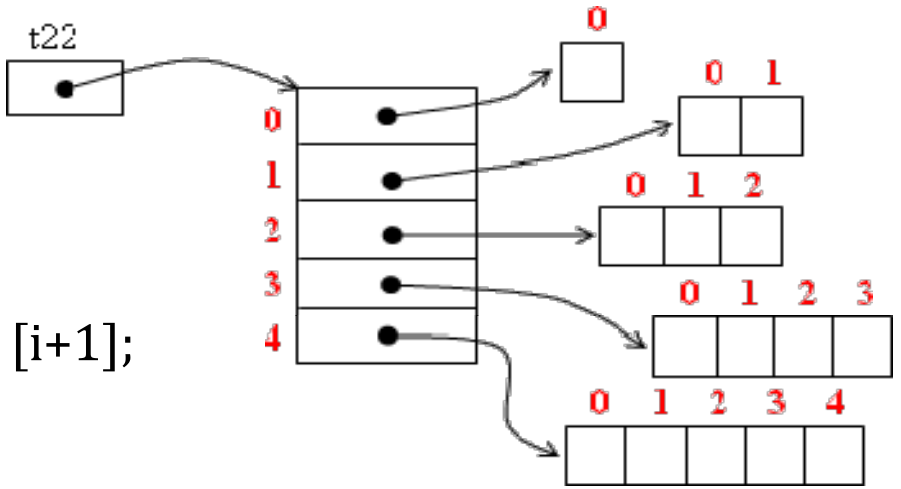
Les tableaux (7)

Tableaux multidimensionnels (2)

Déclaration et création

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

```
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]
                    }
}
```



Fonctions mathématiques

La classe Math

- **Math.pow ( )** : x^a (x puissance a)
- **Math.sqrt( )** : détermine la racine carrée d'un nombre
- **Math.abs ( )** : |x| (valeur absolue de x)
- **Math.min ( )** : détermine le minimum de deux nombres
- **Math.max( )** : détermine le maximum de deux nombres
- **Math.random( )** : donne un nombre au hasard entre 0 et 1

Fonctions mathématiques (2)

- **Math. sin ( )** : calcule le sinus d'un angle.
- **Math.cos ( )** : calcule le cosinus d'un angle.
- **Math.tan ( )** : calcule la tangente d'un angle.
- **Math.exp ( )** : calculer l'exponentielle d'un nombre.
- **Math.log ( )** : calcule le logarithme d'une valeur.
- **Math.floor( )** : arrondit à l'entier inférieur.