**JAVA**

Table des matières

[Chapitre 1 INTRODUCTION & DEFINITION : 2](#_Toc98009589)

[Chapitre 2 SYNTHAXE DE JAVA : 2](#_Toc98009590)

[A)Les commentaires : 2](#_Toc98009591)

[B)Instructions , blocs et blancs : 2](#_Toc98009592)

[B)Les identificateurs : 3](#_Toc98009593)

[C)Les mots réservés : (mots clés) : 3](#_Toc98009594)

[D)Les opérateurs : 3](#_Toc98009595)

[Chapitre 3 : QUELQUES RACOURCIS : 4](#_Toc98009596)

# 

# INTRODUCTION & DEFINITION :

* Avec Java tout est objet
* Il y a des objets qui manipulent les données et il y a des objets qui exposent des services (les API intégrées dans java.
* Le Bytecode est le fichier exécutable = .Class ou .Java
* Java est fiable car il ne transmet que la bytecode en production, il est aussi fiable car grace à son algorithme « ramasse miette » les variables déclarée n’occupent plus la mémoire dans une machine. (contraire à C++) ;
* Java limite les sources d’erreurs grace à son typage fort (on ne peut pas faire une addition entre + et > par exemple.
* Un pointeur = adresse mémoire d’une variable. Donc java est fiable aussi car on ne manipule pas les adresses mémoires chose qui était très difficile (compliquée car il faut comprendre l’architecture système pour pouvoir coder) dans C++.
* Le java est est indépendant de l’architecture système car le byte code (le livrable) est installable sur tous les systèmes d’exploitation à condition d’installer une JVM (machine virtuelle).
* Java est multi taches (multi Threadings) (thread = processus ~ Comme les branches dans git (le parallélisme dans le travail)) : on peut lancer autant de processus qu’on veut et chaque processus va concurrencer les autres. Exemple : pour lire un fichier de 1 million de lignes il faut 2 heures pour lire avec la méthode classique (en une seule lecture) mais grâce à java (API Thread) on peut subdiviser le fichier en plusieurs parties (chaque partie est constituée de 50 000 lignes par exemple), et chaque API se occupe de lire 50000 lignes et à la fin on aurait gagné au moins 1H30 en terme de temps (ressemble au calculs distribués).
* Classe composée de d’attributs et méthodes. Attributs = variables, méthodes = fonctions (bloc d’instructions).
* Ma connaissance de java dépend du nombre d’API qu’il existe dans java.

# SYNTHAXE DE JAVA :

## A)Les commentaires :

3 Types :

Commentaires pour la ligne **// commentaire ligne**, commentaires sur plusieurs lignes **/\*…..\*/ commentaire sur une ou plusieurs lignes**, documentation sur les classes **/\*\*…..\*/ commentaire d’explication**

## B)Instructions , blocs et blancs :

On met dans un bloc les méthodes des classes

Un bloc est délimité par { } ;

La signature d’une méthode : exemple public static void main ( String [] args) 🡺 point d’entrée d’un code Java (méthode main). On ne peut pas créer dans une classe deux méthodes avec les mêmes structures (nom) et les mêmes paramètres.

Le code d’une class doit être stockée dans un fichier .Java qui porte le même nom que la classe.

## B)Les identificateurs :

Dans Java il existe une rigueur de nomenclature des attributs, classes, variables, constantes.

On peut débuter la nomenclature par : lettre, \_ ou $.

1. Si la variable est composée (méthode, attribut, variables) 1ere lettre de chaque mots est en majuscule excepté la première lettre du nom de la variable.
2. Avec la classe on applique exactement la même chose comme les attributs et les méthodes sauf qu’on rajoute une majuscule au début du nom de la classe.
3. Constante : toutes les lettres d’une constante sont en majuscule, si le nom est composé de plusieurs mots alors on met un \_ entre les mots.

## C)Les mots réservés : (mots clés) :

Types de variables : boolean (true, false), chiffres (integer, byte, double, long, float, short), char pour une seule lettre.

Modificateurs : public, static, void, final, native, abstract, synchronized, private, protected, class, transient, volatile, method, attribute

Algorithmie: goto, if, do, break, for, return, while, switch, continue,

Gestion des exceptions: throw, throws, catch, finally, try

Heritage: extends, implements, super, instance of, interface, package pour définir l’appartenance.

Objet: import : pour interpreter une librairies (classe)

New : créer un objet à partir d’une classe

This : pointeur vers la classe en cours

Null : objet vide

## D)Les opérateurs :

+, -, \*, /

15 /4 🡺 3 15.00/4 🡺3.75 15 %2 🡺 1 ( % le reste de la division)

++ incrémentation, i - - décrémentation,

N= 3, n++ 🡺 n = n+1 =4, n - - 🡺 n = n-1 = 3-1=2, 8++ illégale

System.in : entrée standard (rentrer des données depuis un clavier)

System.out : sortie (console)

Scanner scan = new Scanner(System.in)

Nb1 = scan.nextInt(); (méthode bloquante car on attend que l’utilisateur rentre une donnée depuis le clavier pour que le programme puisse continuer à s’exécuter).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Factorielle = cpt \* n | Initialisé à 1 | = 1 \* (2)=2 | = 2 \* 3 =6 | = 6\*4=24 |
| N | Argument (choisi dans le programme) exemple n = |  | | |
| Cpt | Initialisé à 2 | 3 | 4 | 5 |
| Cpt < = n « condition » | oui | oui | oui | Non 🡺 fin de programme |

-A chaque fois qu’on crée une nouvelle méthode dans java, il la considère comme nouveau type

# : QUELQUES RACOURCIS :

Ctrl shif T : faire une recherche des classes dans java

« text » ctrl espace : permet de choisir ce qu’on veut rédiger directement (raccourcis)

Ctrl + clique droit 🡺 « choisir ouvrir une nouvelle fenêtre de commande ici ».

Ctrl shift C : commentaire sur l’ensemble des lignes sélectionnées.

Ctrl D : supprimer la ligne entière.

Ctrl 1 créer une nouvelle classe.

Ctrl shift W : fermeture de toutes les fenêtres de eclipse déjà ouvertes.

Alt shft S S : générer une classe à partir d’un mots texte.