Concepts de base de Java

# Introduction

Java est un langage de programmation largement utilisé pour coder des applications web. Il a été fréquemment choisi parmi les développeurs depuis plus de deux décennies, des millions d'applications Java étant utilisées aujourd'hui. Java est un langage multiplateforme, orienté objet et centré sur le réseau, qui peut être utilisé comme une plateforme à part entière.

Java est une technologie qui combine un langage de programmation et une plateforme logicielle. Pour créer une application en utilisant Java, vous devez télécharger le kit de développement Java (JDK), disponible pour Windows, macOS et Linux. Vous écrivez le programme dans le langage de programmation Java, puis un compilateur transforme le programme en bytecode Java, le jeu d’instructions destiné à la [machine virtuelle](https://www.ibm.com/fr-fr/topics/virtual-machines?_ga=2.31846846.1696084635.1710142763-2067957453.1707311480&_gl=1*12zdqfh*_ga*MjA2Nzk1NzQ1My4xNzA3MzExNDgw*_ga_FYECCCS21D*MTcxMDI5NzM0Ni43Ni4xLjE3MTAyOTc1MjMuMC4wLjA.) Java (JVM), qui fait partie de l’environnement d’exécution Java (JRE). Le bytecode Java s’exécute sans modification sur les systèmes prenant en charge les JVM, ce qui permet à votre code Java d’être exécuté partout.

# Les syntaxes et les concepts de base

**Les règles de base**

Java est sensible à la casse.

Les blocs de code sont encadrés par des accolades. Chaque instruction se termine par un caractère « ; » (point-virgule)

Exemple : int a = 7



**Les mots réservés du langage Java**

Java 9 définit 54 mots réservés qui ne peuvent pas être utilisés comme identifiant.

Historiquement, les mots réservés (reserved words) peuvent être regroupés en deux catégories :

* 51 mots clés (keywords) dont 3 ne sont pas utilisés
* 3 valeurs littérale (literals) : true, false et null.

# Comparaison entre Java et les autres langages que je maitrise (Java et JavaScript)

* JavaScript est un langage web, recommandé pour les applications et sites web, ce langage, historiquement réputé plus ”créatif”, est de ce fait dédié au développement d'applications dites front end. Il est depuis plusieurs années en pleine expansion. Pour JavaScript, il existe un nombre important de frameworks et de librairies, il peut donc sembler plus complexe à première vue, à appréhender. Depuis l’arrivée de [NodeJS](https://nodejs.org/fr/" \t "_blank), JavaScript permet le développement d’applications côté serveur, dédiées au développement back end.
* Java est un langage à la base destiné aux applications embarquées. C'est-à-dire des appareils qui ne sont pas considérés comme des ordinateurs avec des ressources (mémoire, disque dur) limités, comme le sont tablettes, ordinateurs de bord, etc. Réputé pour sa fiabilité, il a également très vite été utilisé pour des applications serveurs nécessitant une grande fiabilité - des serveurs de paiement par exemple.

|  |  |  |
| --- | --- | --- |
| Critère | Java | JavaScript |
| Type de langage | Langage de programmation orienté objet et multi-paradigme | Langage de script orienté objet |
| Compilation | Compilé en bytecode et exécuté sur la JVM (Java Virtual Machine) | Interprété par le navigateur ou un moteur JavaScript (comme Node.js) |
| Utilisation principale | Applications de bureau, applications mobiles (Android), serveurs, applications embarquées | Développement web (côté client), applications serveur (Node.js) |
| Syntaxe et typage | Syntaxe stricte, typage statique (avec vérification à la compilation) | Syntaxe plus souple, typage dynamique |
| Exécution | Fonctionne sur n'importe quel appareil avec une JVM | Exécuté dans un navigateur web ou un environnement comme Node.js |
| Multithreading | Supporte le multithreading natif | Monothreadé avec un modèle basé sur les événements (event-driven) |
| Interopérabilité | Ne peut pas être directement intégré dans du HTML | Peut être exécuté directement dans un navigateur web et manipuler le HTML/CSS |

# La programmation orientée objet

Chaque langage de programmation appartient à une « famille » de langages définissant une approche ou méthodologie générale de programmation. Par exemple, le langage c’est un langage de programmation procédurale car il suppose que le programmeur s’intéresse en priorité aux traitements que son programme devra effectuer. Un programmeur C commencera par identifier ces traitements pour écrire les fonctios qui les réalisent sur des données comme paramètres d’entrée.

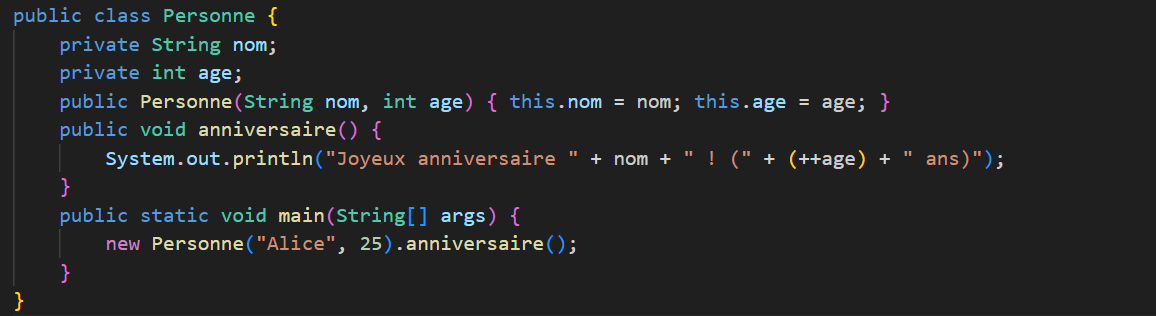
La programmation orientée-objet (introduite par le langage SmallTalk) propose une méthodologie centrée sur les données. Le programme Java va d’abord identifier un ensemble d’objets, tel que chaque objet représente un élément qui doit être utilisé ou manipulé par le programme, sous la forme d’ensembles de données. Ce n’est que dans un deuxième temps, que le programmeur va écrire les traitements, en associant chaque traitement à un objet donné. Un objet peut être vu comme une entité regroupant un ensemble de données et de méthodes (l’équivalent d’une fonction en C) de traitement.

**Classe**

Un objet est une variable (presque) comme les autres. Il faut notamment qu’il soit déclaré avec sont type. Le type d’objet est un type complexe (par proposition aux types primitifs entier, caractère, …) qu’on appelle une classe.

Donc une classe regroupe un ensemble de données (qui peuvent être des variables primitives ou des objets) et un ensemble de méthodes de traitement de ces données et/ou de données extérieures à la classe. On parle d’encapsulation pour désigner le regroupement de données dans une classe.

Exemple :

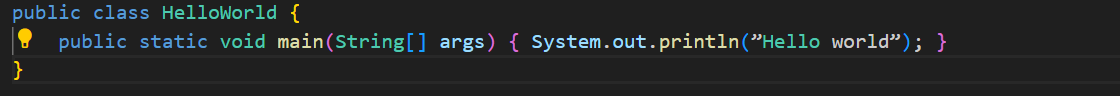


**L’encapsulation**

Lors de la conception d’un programme orienté-objet, le programmeur doit identifier les objets et les données appartenant à chaque objet mais aussi des droits d’accès qu’ont les autres objets sur ces données. L’encapsulation de données dans un objet permet de cacher ou non leur existence autres objets du programme. Une donnée peut être déclarée en accès :

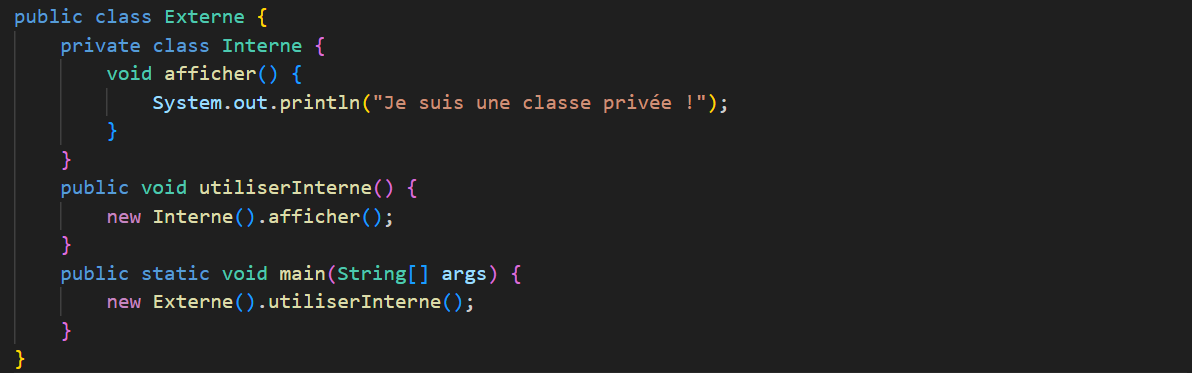
* Public : les autres objets peuvent accéder à la valeur de cette donnée ainsi que la modifier ;

Exemple :



**Privé** : les autres objets n’ont pas le droit d’accéder directement à la valeur de cette donnée (ni de la modifier). En revanche, ils peuvent le faire indirectement par des méthodes de l’objet concerné (si celles-ci existent).

Exemple :



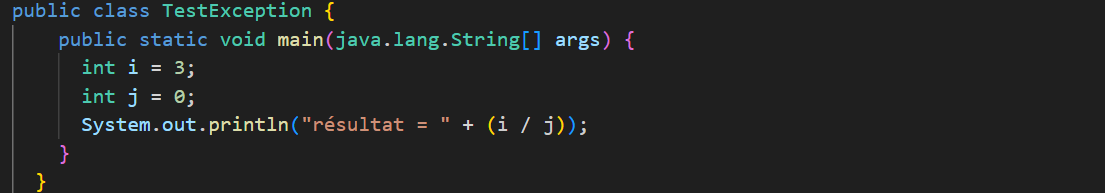
# Gestion des exceptions avec des exemples

Les exceptions représentent le mécanisme de gestion des erreurs intégré au langage Java. Il se compose d'objets représentant les erreurs et d'un ensemble de trois mots clés qui permettent de détecter et de traiter ces erreurs (try, catch et finally) mais aussi de les lever ou les propager (throw et throws).

Lors de la détection d'une erreur, un objet qui hérite de la classe Exception est créé (on dit qu'une exception est levée) et propagé à travers la pile d'exécution jusqu'à ce qu'il soit traité.

Ces mécanismes permettent de renforcer la sécurité du code Java.

Exemple :



# Les collections et streams

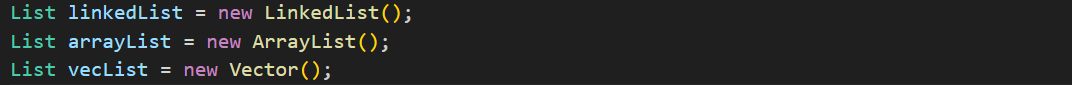
Les **collections** sont des objets qui permettent de gérer des ensembles d'objets. Ces ensembles de données peuvent être définis avec plusieurs caractéristiques : la possibilité de gérer des doublons, de gérer un ordre de tri.

Une collection est un regroupement d'objets qui sont désignés sous le nom d'éléments.

L'API Collections propose un ensemble d'interfaces et de classes dont le but est de stocker de multiples objets. Elle propose quatre grandes familles de collections, chacune définie par une interface de base :

List : collection d'éléments ordonnés qui accepte les doublons.

Exemple :



Set : collection d'éléments non ordonnés par défaut qui n'accepte pas les doublons

Map : collection sous la forme d'une association de paires clé/valeur

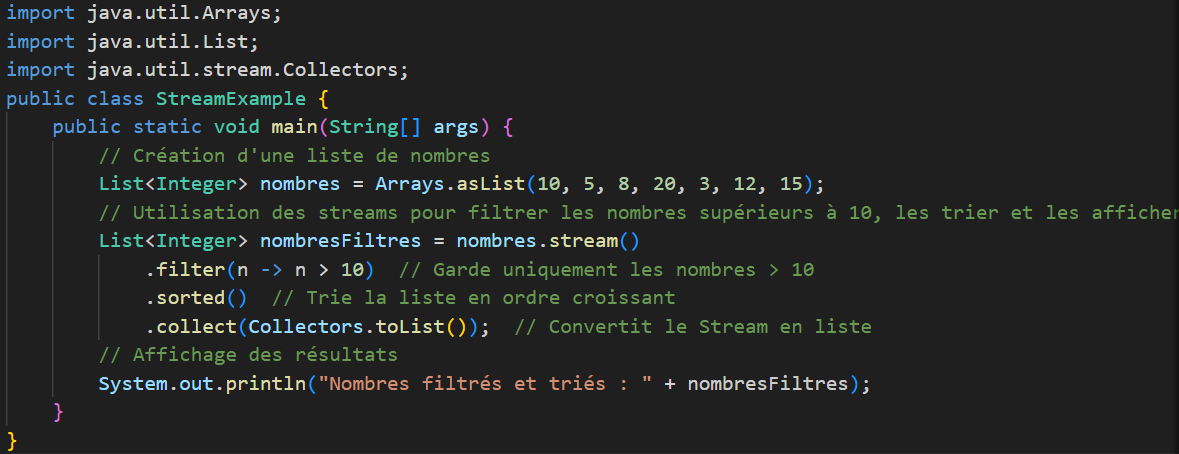
Queue : collections qui stockent des éléments dans un certain ordre avant qu'ils ne soient extraits pour traitement

**Streams** en programmation fonctionnelle, on décrit le résultat souhaité mais pas comment on obtient le résultat. Ce sont les fonctionnalités sous-jacentes qui se chargent de réaliser les traitements requis en tentant de les exécuter de manière optimisée.

Ce mode de fonctionnement est similaire à SQL : le langage SQL permet d’exprimer une requête mais c’est le moteur de la base de données qui choisir la meilleure d’obtenir le résultat décrit.

Le concept de Stream existe déjà depuis longtemps dans l'API I/O, notamment avec les interfaces InputStream et OutputStream. Il ne faut pas confondre l'API Stream de Java 8 avec les classes de type xxxStream de Java I/O. Les streams de Java I/O permettent de lire ou écrire des données dans un flux (sockets, fichiers, ...).

Exemple :



# Manipulation des fichiers entrée-sortie

Les flux d’entrées/sorties sont un aspect fondamental de la programmation qui permet aux programmes d’interagir avec leur environnement. Les entrées/sorties peuvent prendre plusieurs formes, notamment la lecture et l’écriture de fichiers.

Exemple :

****

# Notion de tests unitaires

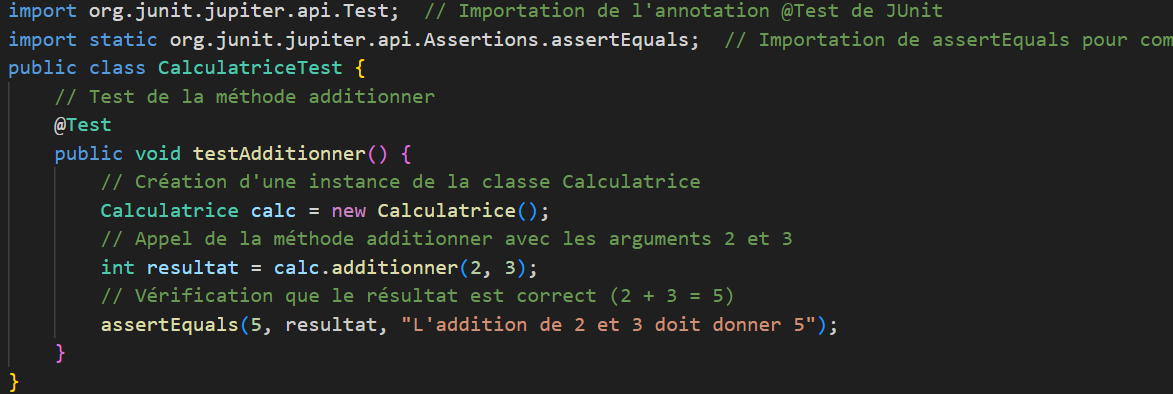
Les tests unitaires sont une bonne pratique fondamentale dans le développement de logiciels qui garantit la fiabilités et l’exactitude de votre code. Dans cet, nous explorerons les concepts clés des tests unitaires Java, y compris des exemples de code simples, le paramétrage, les tests d’exceptions, les annotations telles que @Before, @BeforeEach, @After et @AfterEach.

Les tests unitaires consistent à tester des composants individuels ou des unités de code de manière isolée pour vérifier leur exactitude. L'objectif principal est de détecter et de corriger les bogues dès le début du processus de développement, en garantissant que chaque unité de code se comporte comme prévu.

Des annotations telles que @Before, @BeforeEach, @After et @AfterEach sont utilisées pour configurer et supprimer l'environnement de test. Ces annotations aident à gérer les tâches courantes d'initialisation et de nettoyage des tests.

* @Before et @After s'exécutent respectivement une fois avant et après toutes les méthodes de test de la classe de test.
* @BeforeEach et @AfterEach s'exécutent avant et après chaque méthode de test.

Exemple :



# Multithreading et les comparables

**Multithreading**

Java est un langage de programmation Multithread, ce qui signifie que nous pouvons développer des programmes Multithreads en utilisant Java. Un programme multithread contient deux ou plusieurs parties qui peuvent s’exécuter simultanément et chaque partie qui peuvent en même temps, en optimisant l’utilisation des ressources disponibles, en particulier lorsque votre ordinateur dispose de plusieurs processeurs.

Le multithreading vous permet d'écrire de manière à ce que plusieurs activités puissent se dérouler simultanément dans le même programme. Pour réaliser le multithreading (ou écrire du code multithread), vous avez besoin de [la classe java.lang.Thread](https://www.tutorialspoint.com/java/lang/java_lang_thread.htm) .

**Comparables**

Comparable en Java est un objet qui se compare à un autre objet. Il permet de trier la liste des objets personnalisés. Java.lang. L’interface comparable doit être implémentée par une classe afin de comparer ses instances. Un tableau d'objets implémentant une interface comparable est automatiquement trié par Arrays.sort et Collections.sort méthodes.

**Méthode utilisée dans Comparable**

Je voudrais souligner une méthode importante que j'utilise fréquemment dans l'interface Comparable :

**Comparer aux():**

La méthode CompareTo() est utilisée pour effectuer un tri naturel sur une chaîne. Le tri naturel désigne l'ordre de tri qui s'applique à l'objet, par exemple l'ordre numérique pour le tri des entiers, l'ordre alphabétique pour les chaînes, etc.

La syntaxe de CompareTo() La méthode est la suivante:

int compareTo(T obj)

CompareTo() la méthode compare l'objet avec T obj.

Haut du formulaire

Exemple :

