Rappels

Introduction à Java

Bibliothèques de classes Java standard :

- $java.lang$: Classes de base (importé par défaut) ; java.lang (Java Platform SE 8)
- $java.util$: Classes utilitaires ;
- $java.io$: Gestion des entrées/sorties ;
- $java.awt$: Classes graphiques et de gestion d'interfaces ;
- $javax.swing$: Autres classes graphiques.

Créer sa propre Javadoc :

1) Ecrire du code avec des commentaires Javadoc ;
2) Exécuter la commande : $javadoc \ -d\ doc\ Programme.java$
   - $-d\ doc$ → Indique le dossier de sortie ($doc/$) ;
   - $Programme.java$ → Fichier source à documenter.
3) Ouvrir le fichier $index.html$ dans le dossier doc.

Java et la Programmation Orientée Objet (POO)

A retenir : « $this$ » fait référence à l'objet qui appelle cette méthode.

Les mots clés $public$ et $private$

- Une variable déclarée $public$ est accessible de n'importe où dans le programme, à partir de n'importe quelle classe ;
- Une variable déclarée $private$:
  - Est accessible uniquement au sein de la classe où elle est définie ;
  - Ne peut être appelée qu'à l'intérieur de la classe dans laquelle elle est définie.

  Accesseur et modifieur :
  - Un **accesseur (getter)** est une méthode utilisée pour récupérer la valeur d'un attribut privé ;
  - Un **modifieur (setter)** est une méthode utilisée pour modifier la valeur d'un attribut privé.

  Quel mot clé utilisé ?
  - **Attributs** : Variable $private$ pour protéger les données sensibles lorsqu'on l'utilise dans d'autres classes.
  - **Méthodes** :
    - Publique → Usage mondial ;
    - Privée → Usage interne.

Quelques classes existantes en Java

- $ArrayList$: Une liste dynamique qui peut contenir des éléments. On peut ajouter, supprimer, et accéder à des éléments sans savoir comment la liste est implémentée en interne ;

- $HashMap$: Une collection qui associe des clés à des valeurs. On peut stocker et récupérer des paires clé-valeur sans connaître le fonctionnement interne de la table de hachage ;
- $String$: Représente une séquence de caractères. On peut manipuler des chaînes de caractères (concaténation, recherche, etc.) sans savoir comment Java gère les chaînes en mémoire.

La classe $String$ La classe $String$ en Java représente une séquence de caractères.

- Immutabilité

  Les objets $String$ sont immuables : Leur contenu ne peut pas être modifié après leur création. Sinon il créer un nouvel objet.
- Les String Pool en Java
  - **Principe :**
    - String Pool : Zone spéciale ou sont stockées les chaînes de caractères littérales (« $Bonjour$ ») ;
    - Si une chaîne identique existe déjà dans le pool, Java **réutilise la même référence** au lieu de créer un nouvel objet.
  - **Avantages :**
    - Economie de mémoire → Une seule copie de chaque littéral est conservée ;
    - Comparaisons plus rapides entre chaînes identiques (== fonctionne).
- Littéral vs $new\ String()$
  - $String\ s = "Java"$ → Utilise le String Pool ;
  - $String\ s = new\ String("Java")$ → Crée un nouvel objet distinct, même si « Java » existe déjà dans le pool.
- Quelques méthodes utiles de la classe String :
  - $\boldsymbol{length()}$ **:** Retourne la longueur de la chaîne ;
  - $\boldsymbol{charAt(int\ index)}$ **:** Retourne le caractère à l'index spécifié ;
  - $\boldsymbol{toUpperCase()/toLowerCase()}$ **:** Convertit la chaîne en majuscules ou minuscules ;
  - $\boldsymbol{replace(char\ oldChar, char\ newChar)}$ **:** Remplace les caractères dans la chaîne ;
  - $\boldsymbol{String.valueOf(int\ num)}$ **:** Conversion d'un nombre en chaîne.

Les packages en Java

Un package en Java est un **regroupement logique** de classes, d'interfaces et d'autres packages. Il permet d'organiser le code et d'éviter les conflits de noms.
Les packages sont similaires aux **dossiers** sur un disque dur, ils servent à organiser les fichiers.

- Déclaration et utilisation

  Classe $MaClasse$ dans un package appeler $monPackage$: $package\ monPackage;$
  Utiliser une classe d'un autre package :

  Pour utiliser $MaClasse$ dans une autre classe, il faut l'importer :

  $$import\ monPackage.MaClasse;$$
- Avantages
  - **Organisation** : Les packages permettent de structurer le code de manière claire et hiérarchisée, facilitant ainsi la maintenance et la gestion de projet complexes ;

- **Eviter les Conflits de Noms** : En regroupant les classes dans les packages, les conflits entre des clases portant le même nom mais se trouvant dans des packages différents sont évités.

Modificateurs :

- $protected$: Accessibles dans le même package et par les sous-classes (même dans des packages différents).
  Utilisation : Favorise la réutilisation du code en permettant aux sous-classes d'accéder aux membres de la classe parente.
  En pratique : Le modificateur $protected$ est un compromis :
  - Plus ouvert que $private$;
  - Mais plus sécurisé que $public$.
- $default$:
  - Si aucun modificateur n'est spécifié, la variable a une accessibilité de package ($default$), ce qui signifie qu'elle est accessible par toutes les classes du même package, mais **pas par des classes de packages différents**.

Modificateur	Visibilité dans la même classe	Visibilité dans le même package	Visibilité dans les sous-classes	Visibilité partout
private	Oui	Non	Non	Non
default	Oui	Oui	Non	Non
protected	Oui	Oui	Oui	Non
public	Oui	Oui	Oui	Oui

Types primitifs

Les types primitifs de Java

- Java dispose d'un certain nombre de **types de base** dits primitifs ;
- Permettent de manipuler des entiers, des flottants, des caractères et des booléennes ;
- Ils se divisent en quatre catégories :
  - Nombres entiers ;
  - Nombres flottants ;
  - Caractères ;
  - Booléens.

- Les types entiers : Représente des nombres entiers relatifs.

Type	Taille (en octets)	Valeur minimale	Valeur maximale
byte	1	-128	127
short	2	-32,768	32,767
int	4	-2^31 (-2,147,483,648)	2^31 - 1 (2,147,483,647)
long	8	-2^63 (-9,223,372,036,854,775,808)	2^63 - 1 (9,223,372,036,854,775,807)

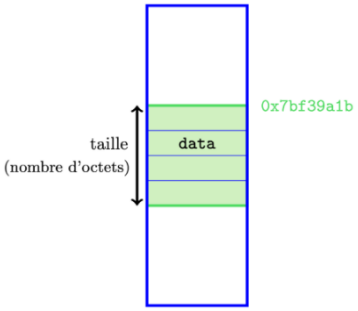
- Les types flottants : Représente, de manière approchée, une partie des nombres réels.

Type	Taille en octets	Précision (chiffres significatifs)	Valeur absolue minimale	Valeur absolue maximale
float	4	7	1.4e-45	3.4e+38
double	8	15	4.9e-324	1.7e+308

La mémoire

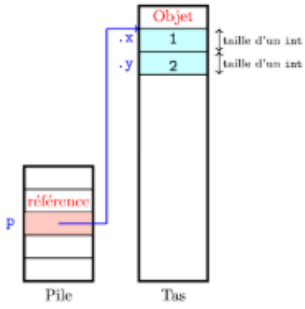
La mémoire dans un programme

- Allocation de mémoire :
  - Lorsqu'une variable est déclarée, un espace mémoire approprié est réservé ;
  - Cet espace correspond à la taille nécessaire pour stocker les données associées à la variable.
- Accès aux données :
  - Pour accéder aux données d'une variable, il suffit de connaître son adresse de début (référence) ;
  - La taille de la variable est généralement **déterminée par son type** (ex : int, $double$).



Suivant les cas, l'allocation peut avoir lieu sur la pile ou dans le tas.

- **La pile d'exécution**
  - Mémoire allouée pour l'exécution des méthodes ;
  - Stocke les types primitifs (int, $float$, etc.) et les références aux objets.
- **Le tas (heap)**
  - Le tas est une région de la mémoire où les objets Java sont stockés **dynamiquement à l'exécution** ;
  - Les objets sont créés dans le tas à l'aide de l'opérateur new ;
  - Le constructeur initialise ses champs ;
  - La mémoire du tas est partagée par tous les threads et gérée par le ramasse-miettes.



Variables de type objet

- Contrairement aux types primitifs, les objets sont **stockés différemment** en mémoire ;
- Les variables de types objet sont des références qui pointent vers des emplacements en mémoire où les objets sont stockés.

Objets et invocation de méthodes

Chaque objet d'une classe peut être manipulé avec les méthode visibles de la classe (notation .).

Tester l'égalité de deux objets avec $==$

- En Java, l'opérateur $==$ compare les références des objets, pas leur contenu ;
- Cela signifie qu'il vérifie si les deux variables pointent vers le même emplacement mémoire (la même instance), et **non si les objets eux-mêmes sont identiques**.

La référence $null$

- En Java, $null$ est une **valeur spéciale** qui signifie que la variable ne pointe vers aucun objet ;
- C'est l'absence de référence pour un type d'objet. Utiliser $null$ revient à dire que la variable est vide et ne contient aucune adresse mémoire d'un objet.

$NullPointerException$: Exception déclencher quand on essaye d'accéder à une méthode ou un attribut d'un objet qui est $null$.

Le garbage collector (ramasse-miettes)

- Le garbage collector est un mécanisme automatique en $Java$ qui gère la mémoire allouée aux objets sur le tas ;
- Il **libère la mémoire des objets qui ne sont plus référencés** par le programme, évitant ainsi les fuites de mémoire.

Variables et méthodes de classe

Les champs $static$ (variables de classe)

- Un champ $static$ est une variable de classe partagée entre toutes les instances d'une classe ;
- Contrairement aux variables d'instance, il **n'appartient pas à un objet** mais à la classe elle-même.

Caractéristiques :

- Il est initialisé une seule fois lors du changement de la classe ;
- Accessible via la classe elle-même sans qu'une instance ne soit nécessaire.

- Souvent utilisé pour définir les constantes.
- Restriction : Une méthode $static$ ne peut pas accéder directement aux variables d'instance ou appeler des méthodes non-statiques.

La méthode $main$

La méthode $main$ est toujours déclarée $static$.
Cela permet son invocation automatique à l'exécution, **sans création d'un objet** de la classe du $main$.

Le mot clé $final$ avec les variables

- Variable $final$: La valeur ne peut pas être modifiée après la première affectation ;
- La première affectation peut se faire lors de la compilation ou à l'exécution.

Le mot clé $final$ avec les tableaux

Attention : Avec un tableau (ou tout objet), $final$ rend la **référence** immuable, pas le contenu !

Constructeurs

Constructeurs

- Un constructeur est une méthode spéciale utilisée pour initialisée les objets ;
- Il a le **même nom que la classe** et n'a pas de type de retour (pas même $void$) ;
- Il est appelé automatiquement lors de la création d'un objet avec new ;
- Il est en général utilisé pour initialiser les attributs d'un objet dès sa création.

Constructeur par défaut

- Si aucun constructeur n'est défini dans la classe, Java génère automatiquement un constructeur par défaut ;
- Ce constructeur ne prend aucun paramètre et son rôle est simplement d'initialiser les attributs de l'objet avec leurs valeurs par défaut.

Les valeurs par défaut en Java sont :

- 0 pour les types primitifs numériques (int, $double$, etc.) ;
- $false$ pour les types booléens (boolean) ;
- $null$ pour les objets (références).

Règle de surcharge de constructeurs

En Java, chaque constructeur doit avoir une signature unique.

La **signature** d'un constructeur est définie par :

- Le nombres de paramètres ;
- Le types de paramètres ;
- L'ordre des paramètres.

Le mot clé $this$ pour les constructeurs

- Il est possible, grâce au mot clé $this$ d'appeler un constructeur de la classe depuis un autre constructeur de la même classe.
- A retenir :
  - $this()$ doit être la première dans l'instruction dans un constructeur ;
  - Eviter la **duplication de code** en réutilisant les constructeurs.