

**GENERALITE DE  
JAVA**

**INTRODUCTION**

1. **HISTORIQUE DE LA POO**
   1. Apparution de la POO
   2. Importance de la POO

(Convention de nommage ou dénomination à avoir)

1. **LES OBJETS**
2. **NOTION DE CLASSES**
3. **LES METHODES**
4. **ENCAPSULATION & ABSTRACTION**
5. **HERITAGE**
6. **POLYMORPHISME**
7. **INTERFACE**
8. **GESTION DES EXCEPTIONS**

* Opérateurs arithmétiques et logiques
* Variables et types de données
* Les conditions
* Les boucles
* Les Tableaux
* Les méthodes

**CONCLUSION**

. **REMERCIEMENTS**

**Q&A**

**II. LES OBJETS**

**1-Definition**

Une entité qui a un état et un comportement est connue sous le nom d'objet, par exemple une chaise, un vélo, un marqueur, un stylo, une table, une voiture, etc. Elle peut être physique ou logique (tangible et intangible). L'exemple d'un objet immatériel est le système bancaire.

Un objet a trois caractéristiques :

* **État :** représente les données (valeur) d'un objet.
* **Comportement :** représente le comportement (fonctionnalité) d'un objet tel que déposer, retirer, etc.
* **Identité :** une identité d'objet est un identifiant unique.

**2-Syntaxe**

|  |
| --- |
| ---code---  class Student{  int id=15;  String name="alband";    public static void main(String args[]){  Student s1=new Student(); //Creation de l’objet  System.out.println(s1.id); //acceder aux attributs de la classe  System.out.println(s1.name);  }  }  ---code--- |

**III. NOTION DES CLASSES**

* 1. **Définition d’une classe**

Une classe est un groupe d'objets qui ont des propriétés communes. Il s'agit d'un modèle ou d'un plan à partir duquel des objets sont créés. C'est une entité logique. Ça ne peut pas être physique.

Une classe en Java peut contenir :

* Des champs;
* Méthodes;
* Constructeurs;
* Blocs;
* Classe et interface imbriquées**;**
  1. **Syntaxe d’une classe**

Syntaxe pour déclarer une classe :

**class** <nom\_classe>{

    Attribut;

    méthode;

}

* 1. **Exemple**

|  |
| --- |
| ---code---  class Student{  int id=15;  String name="alband";    public static void main(String args[]){  Student s1=new Student();  System.out.println(s1.id);  System.out.println(s1.name);  }  } |

|  |
| --- |
| ---Resultat---  15  Alband  ---Resultat--- |

**IV. LES METHODES**

* 1. **Définition**

Une **méthode** est un bloc de code ou une collection d'instructions ou un ensemble de codes regroupés pour effectuer une certaine tâche ou une opération

Il est utilisé pour atteindre la**réutilisabilité**du code. Une méthode est écrite une fois et est utilisée plusieurs fois.

* 1. **Syntaxe**

Nom de la methode

|  |
| --- |
| **Public int sum (int a, int b)**  **{**  **//Methode**  **}** |

Liste de paramètre

* 1. **Les spécificateurs d’accès**

 Le spécificateur ou le modificateur d'accès est le type d'accès de la méthode. Il spécifie la visibilité de la méthode. Java fournit **quatre** types de spécificateurs d'accès :

* **Public :** la méthode est accessible par toutes les classes lorsque nous utilisons le spécificateur public dans notre application.
* **Private:** Lorsque nous utilisons un spécificateur d'accès privé, la méthode n'est accessible que dans les classes dans lesquelles elle est définie.
* **Protected :** lorsque nous utilisons un spécificateur d'accès protégé, la méthode est accessible dans le même package ou dans les sous-classes d'un package différent.
* **Par défaut :** lorsque nous n'utilisons aucun spécificateur d'accès dans la déclaration de méthode, Java utilise par défaut le spécificateur d'accès par défaut. Il n'est visible qu'à partir du même package uniquement.
  1. **Signature de la méthode**

Liste de paramètre

Nom de la methode

|  |
| --- |
| **Public int sum (int a, int b)**  **{**  **//Signature de la methode**  **}** |

* 1. **Methode Statique**
     + **Definition**

Une méthode qui a un mot-clé statique est appelée méthode statique. En d'autres termes, une méthode qui appartient à une classe plutôt qu'à une instance d'une classe est appelée méthode statique. Nous pouvons également créer une méthode statique en utilisant le mot-clé **static** avant le nom de la méthode.

Le principal avantage d'une méthode statique est que nous pouvons l'appeler sans créer d'objet. Il peut accéder aux membres de données statiques et également en modifier la valeur. Il est utilisé pour créer une méthode d'instance. Il est appelé en utilisant le nom de la classe. Le meilleur exemple de méthode statique est la méthode **main()**.

* **Exemple**

Affichage   de la **classe publique**

{

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

{

Afficher () ;

}

**spectacle vide statique**()

{

System.out.println("C'est un exemple de méthode statique.") ;

}

}

* **Résultat**

**C’est un exemple de méthode Statique**

* C'est un exemple de méthode statique
  1. **Méthode d’instance**

La méthode de la classe est connue sous le nom de **méthode d’instance.**C'est une méthode **non statique** définie dans la classe. Avant d'appeler ou d'invoquer la méthode d'instance, il est nécessaire de créer un objet de sa classe. Voyons un exemple de méthode d'instance.

* **Exemple**

**class publique** InstanceMethodExample

{

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

{

//Création d'un objet de la classe

InstanceMethodExample obj =  **new** InstanceMethodExample();

//invoquer la méthode d'instance

System.out.println("La somme est : " +obj.add( 12 ,  13 ));

}

**int** s;

// méthode définie par l'utilisateur car nous n'avons pas utilisé de mot-clé statique

**public int** add(int a,  **int** b)

{

s = a+b ;

// retourne la somme

**retourner** s ;

}

}

* **Résultat**

**La somme est : 25**

* 1. **Surcharge de la methode**

Si une [classe a plusieurs méthodes ayant le même nom mais des paramètres différents, on parle de](https://www.javatpoint.com/object-and-class-in-java)**[surcharge de méthode](https://www.javatpoint.com/object-and-class-in-java)**[.Aussi](https://www.javatpoint.com/object-and-class-in-java)

[La surcharge de la methode augmente la lisibilité du programme](https://www.javatpoint.com/object-and-class-in-java)

[Il existe deux façons de surcharger la méthode en Java](https://www.javatpoint.com/object-and-class-in-java)

* [En changeant le nombre d'arguments](https://www.javatpoint.com/object-and-class-in-java)

|  |
| --- |
| [additionneur de classe  {](https://www.javatpoint.com/object-and-class-in-java)  [static int  add( int  a, int  b){ return  a+b;}](https://www.javatpoint.com/object-and-class-in-java)  [static int  add( int  a, int  b, int  c){ return  a+b+c;}](https://www.javatpoint.com/object-and-class-in-java)  [}](https://www.javatpoint.com/object-and-class-in-java)  [class  TestOverloading1{](https://www.javatpoint.com/object-and-class-in-java)  [public static void  main(String[] args){](https://www.javatpoint.com/object-and-class-in-java)  [System.out.println(Adder.add( 11 , 11 ));](https://www.javatpoint.com/object-and-class-in-java)  [System.out.println(Adder.add( 11 , 11 , 11 ));](https://www.javatpoint.com/object-and-class-in-java)  [}}](https://www.javatpoint.com/object-and-class-in-java) |

* [En changeant le type de données](https://www.javatpoint.com/object-and-class-in-java)

C C'est un exemple de méthode statique.

* C'est un exemple de méthode statique.

**V-ENCAPSULATION ET ABSTRACTION**

1. **Encapsulation**