

# POO et JAVA - Concepts de base

---



Lydia YATAGHENE  
lydia.yataghene@junia.com

# Les niveaux de langage

---

- Echelle de placement entre l'humain et la machine
- Langage le + bas niveau : binaire
- Langage le + haut niveau : langage naturel

## Intérêt:

### Bas niveau

- Proximité machine
- Ordres directs
- Performances

### Haut niveau

- Facilité à conceptualiser
- Fonctions évoluées
- Problématiques complexes

# Java

---

# Java

---

- Apparu en 1995 chez Sun Microsystems
- Acquisition de Sun par Oracle en 2009
- Version actuelle Java SE 17



# Java

---

- Orienté objet
- Fortement typé
  - Toute variable doit être déclarée avec un type
  - Le compilateur vérifie que les utilisations des variables sont compatibles avec leur type
  - Les types sont d'une part fournis par le langage, mais également par la définition des classes
- Compilé: En bytecode, code intermédiaire indépendant de la machine
- Interprété : Le bytecode est interprété par une machine virtuelle Java

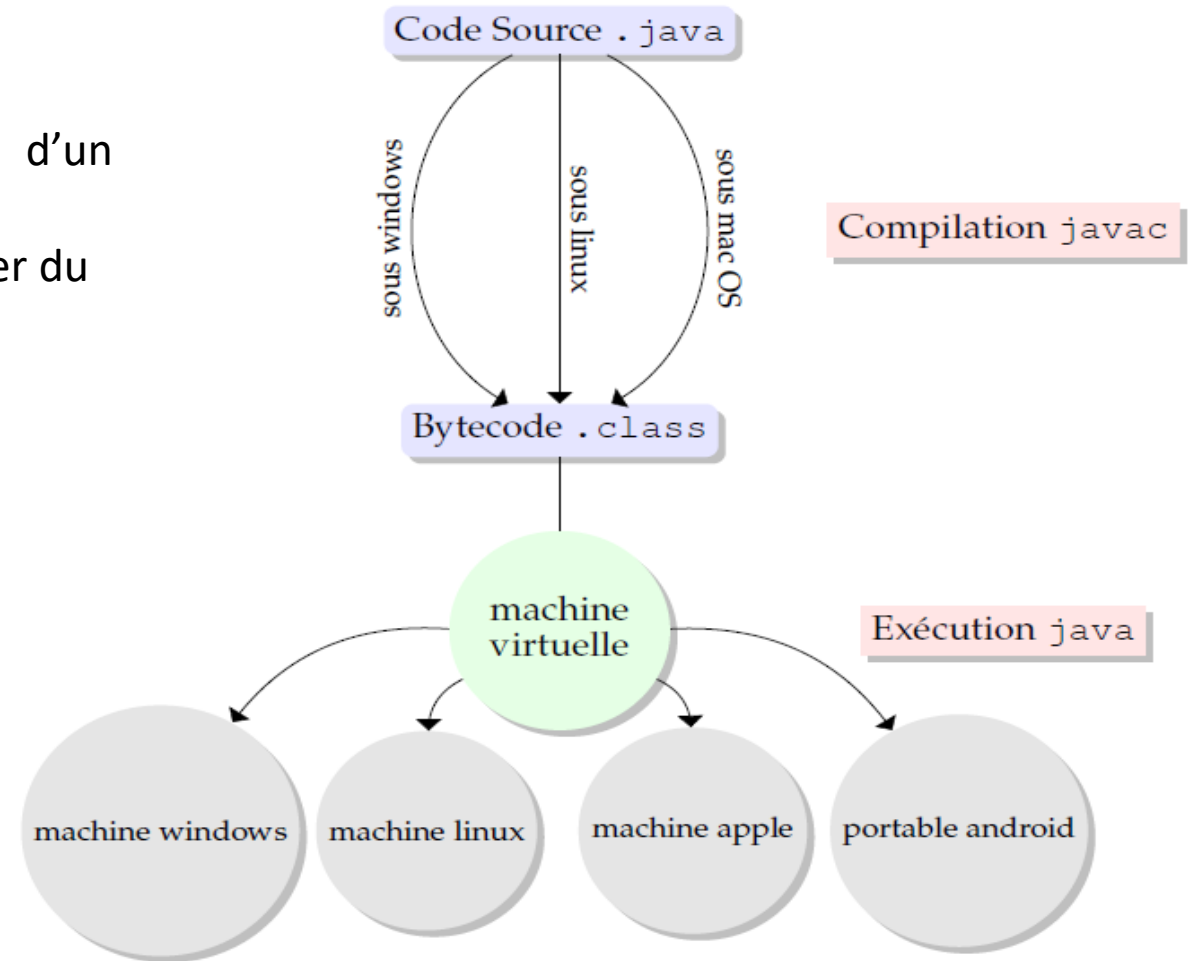
# Les objectifs de Java

---

- La portabilité
- L'intégration du modèle objet
- Syntaxe claire
- Approche générique
- Facilité d'apprentissage

# Compilation et exécution

- Java n'est pas seulement la description d'un langage et une bibliothèque de classe.
- Java dispose d'outils pour générer et exécuter du code



# Compilation et exécution

## L'environnement Java

---

- **JDK**

Java Development Kit   Outils de développement

- **JVM**

Java Virtual Machine

- **JRE**

Java Runtime Environment



# Compilation et exécution

## Bytecode

---

- Le bytecode est une liste d'instructions précompilée conçue pour une exécution optimale par la JVM.
  - Fichiers .class
  - Editable par le développeur (rare)
  - Etape intermédiaire vers le binaire

# Compilation et exécution

## JVM

---

- Java fournit une machine virtuelle : c'est un programme qui lit du code en bytecode et interprète ce code dans le langage de la machine pour l'exécuter
  - Le code est portable : On peut écrire, compiler et exécuter sur des machines d'architectures différentes. (ordinateur windows, mac, linux ...).
  - La machine virtuelle permet de partager d'une manière sécurisée une machine.
  - Le code est généralement plus compact (pas besoin d'inclure les bibliothèques comme en C ou C++).
  - Coût en ressources de la machine virtuelle.

# Programmation orientée Objet

---

# Styles de programmation

---

- Style impératif : Fondé sur l'exécution d'instructions qui modifient l'état de la mémoire
  - Utilise beaucoup les itérations et autres structures de contrôle
  - Types et structures
  - Pointeurs de fonctions
  - Les structures de données sont fondamentales
  - Exemple: Fortran, C
- Le style objet
  - Les objets disposent de ressources et de moyens d'interactions entre eux.
  - Les objets représentent des données qui sont modélisées par des classes qui définissent des types
  - Les classes définissent les actions qu'ils peuvent prendre en charge et la manière dont ces actions affectent leur état ce sont des "méthodes"
  - Exemple: Java, C++...

# Programmation objet

---

- Représenter les objets par des concepts (Personnage, voiture, ...)
  - Attributs
  - Méthodes (fonctions)
- Exemple: Objet voiture

Attributs	Méthodes
Marque	Démarrer
Modèle	Arrêter
Couleur	Accélérer
Nb chevaux	Freiner

# Programmation objet

---

- Ecriture d'attributs et de méthodes
- Création d'objets
- Appels de méthodes entre objets

# Classe

---

- Une classe est un descripteur de données composé d'attributs (propriétés) et de méthodes (actions).
- Une classe est un type abstrait caractérisé par des attributs et méthodes communes à un ensemble d'objets et permettant de créer des objets ayant ces propriétés.
- Une classe sert de modèle à la création d'objets.
- Elle est utilisée comme un type d'élément.

## ➤ Important

- Par convention, le nom d'une classe commence toujours par une majuscule.
- Ce qui n'est pas le nom d'une classe commence par une minuscule

# Les classes en Java

---

- Le mot clé **class** indique que l'on crée une classe, il est suivi par le nom de la classe et le code de la classe se trouvera entre les accolades { et }.
- une classe représente un objet.
- Une classe = un fichier **.java**

```
class Personnage {  
  
}
```



# Les packages

---

- Java permet de regrouper les classes en ensembles appelés packages afin de faciliter la modularité
- Organiser les classes
- Packages et sous-packages
- Mots-clés package et import
- En-tête de fichier

# Attributs de classe

---

- Un attribut est une propriété de classe
- Exemple : âge d'un personnage, couleur d'une voiture, ...

```
class Personnage {  
    int age ;  
    float taille ;  
}
```

# Types élémentaires

---

Nom	Taille	Description
char	16 bits	Caractère ASCII ou Unicode
byte	8 bits	Octet
short	16 bits	Entier signé de 16 bits ( $[-32768; 32767]$ )
int	32 bits	Entier signé de 32 bits ( $[-231; 231]$ )
long	64 bits	Entier signé de 64 bits ( $[-263; 263]$ )
float	32 bits	Nombre réel signé ( $[-1.4 \cdot 10^{-45}; 3.4 \cdot 10^{38}]$ )
double	64 bits	Nombre réel signé ( $[4.9 \cdot 10^{-324}; 1.7 \cdot 10^{308}]$ )
boolean	1 octet	Valeur binaire (true/false)

# Types primitifs

---

- TOUT peut être représenté par un objet
- Certains types ont été conservés : int, float, boolean, ...
- On les appelle les types primitifs
- Ils possèdent leur équivalent en objet
- Intérêt : application des opérateurs fondamentaux

# String

---

- String est l'objet java permettant de manipuler des chaines de caractères
- Package java.lang.String

```
public class Personnage {  
    private int age ;  
    private String name ;  
}
```

- Méthodes usuelles: charAt, substring, length, compareTo, ...

# Les instances en Java

---

- Une entité d'objet est une instance de classe
- Créer une instance : mot-clé **new**

```
Personnage asterix = new Personnage() ;  
Personnage obelix = new Personnage() ;  
asterix.age = 35;  
System.out.println ("Astérix a:" + asterix.age + " ans ");
```

- **new** : allouer l'espace mémoire nécessaire Gestion automatique de la mémoire
- Destruction d'une instance : Garbage Collector

# Accès aux attributs

---

- Accès par le signe .
- Valeur différente à chaque instance
- Mot-clé **this**
- **this** permet d'accéder aux attributs et méthodes de l'instance courante

```
asterix.age = 35;  
class Personnage {  
    void setAge(int age){  
        this.age=age;  
    }  
}
```

# La portée d'un attribut

---

- Permettre l'accès mémoire à un attribut depuis différents endroits du code
  - **public** : L'attribut est accessible partout
  - **protected** : L'accès est limité au package
  - **private** : l'accès est limité à la classe

```
public class Personnage {  
public String name ;  
private int age ;  
protected float taille;  
}
```



# Les méthodes de classe

---

- Les méthodes permettent de définir des fonctions propres à une classe.
- Une fonction dans une classe est appelée une méthode de classe.

```
public class Personnage {  
    private String name ;  
    public int age ;  
    public String getName(){  
        return name ;  
    }  
    public void birthday(){  
        age++ ;  
    }  
}
```

# Les méthodes de classe

---

- **Exemple:** Appel d'une méthode de classe sur une instance

```
System.out.println(asterix.getName());  
asterix.birthday();
```

# Les méthodes de classe

---

- Une méthode aussi possède son propre scope !
- L'accès à une méthode peut donc être restreint
- Scope d'une méthode:
  - **public** : Accessible depuis partout
  - **protected** : Limité au package
  - **private** : Limité à la classe

# Les méthodes de classe

---

```
class Personnage {  
    public string getName () {}  
    protected void anniversaire () {}  
    private void setAge (in age) {}  
}  
  
Personnage asterix = new Personnage ();  
asterix.getName(); // Partout  
asterix.anniversaire(); // Depuis une autre classe du package  
asterix.setAge(35) ; // Depuis une instance de Chat
```

# Les accesseurs

---

- Les accesseurs (getters/setters) sont des méthodes de classe dont le rôle est de renvoyer/modifier la valeur d'un attribut de classe
- Intérêt : garantir l'accès aux attributs sans créer de dépendance
- getters (fonction get) : Connaître la valeur d'un attribut (lecture)
- setters (fonction set) : Modifier la valeur d'un attribut (écriture)

# Les accesseurs

---

```
class Personnage {  
    private int age ;  
  
    public int getAge () {  
        return this. age;  
    }  
  
    public void setAge ( int ageP) {  
        this.age = ageP;  
    }  
}  
  
Personnage asterix = new Personnage(30);
```

# Constructeurs de classe

---

- Un constructeur est une méthode qui porte le nom de la classe et qui n'a pas de type de retour.
- Un constructeur permet d'initialiser les attributs de classe et de créer une instance
- En cas d'absence de constructeur, Java se charge de donner automatiquement un constructeur par défaut: celui-ci initialisera toutes les variables à leur valeur par défaut (donc soit 0, false ou null selon le type).
- Si vous avez déjà défini un constructeur avec des arguments, JAVA ne mettra pas à votre disposition un constructeur par défaut : si vous voulez aussi avoir un constructeur par défaut, il faudra alors le définir vous-mêmes.
- Lorsqu'on déclare une variable d'instance, on peut lui donner une valeur par défaut lors de la déclaration. Cette affectation est effectuée avant l'appel du constructeur. Un constructeur pourra donc mettre à jour la valeur par défaut.
- Les arguments sont définis par l'utilisateur.

# Constructeurs de classe

---

```
public class Personnage {  
    // constructeur par défaut  
    public Personnage(){  
        name="unknown" ;  
    }  
    // constructeurs  
    public Personnage(String name){  
        this.name = name ;  
    }  
}
```



# Java

Boucles, condition, opérateurs et tableaux

---

# Modèles de boucles en Java

---

- Une boucle est une structure dont le début et l'arrêt sont conditionnés permettant la répétition d'une séquence d'instructions.

```
for (i =0;i< LIMIT ;i++) {}
```

```
while ( condition ) {}
```

```
do { } while ( condition );
```

```
for ( String elt : tabString ) {} //For each Itérations basées sur les éléments d'un ensemble
```

# Structures conditionnelles

---

- **If** Exécution conditionnelle d'une séquence d'instructions

```
if ( condition ){} else{}
```

- **Switch** Ensemble de conditions liées à des instructions

```
switch ( valeur ){  
case "A" :... break ;  
case "B" :... break ;  
default :... break ;  
}
```

# Opérateur conditionnel ternaire

---

`result = uneCondition ? value1 : value2;`

**Si** le test (une expression booléenne) `uneCondition` est vérifié,

**alors** la variable `result` prend la valeur `value1`,

**sinon** elle prend la valeur `value2`.

```
float x,y, r=1.0;
```

```
...
```

```
boolean interieur = x*x + y*y < r ? true : false
```

# Opérateurs fondamentaux en Java

---

## Opérateurs

+, -, /, \*, %

<, ==, >, >=, <=, !=

++, --

!, &&, ||

~, <<, >>, >>=, &, |, ^

+=, -=, /=, \*=, &=, |=

## Description

Opérateurs arithmétiques

Opérateurs de comparaison

Sucre syntaxique

Opérateurs logiques

Opérateurs binaires

Opérateurs composés

# Opérateurs binaires en Java

---

Opérateur	Description	Exemple
~	Non binaire	00101 = 11010
<<	Décalage gauche	110 << 1 = 1100
>>	Décalage droit avec signe	0110 >> 1 = 0011
>>>	Décalage droit avec padding	0011 >>> 1 = 0001
&	Et logique	a&b    ou    a AND b
	Ou logique	a b    ou    a OR b
^	Ou exclusif	a^b    ou    a XOR b

# Les tableaux

---

- Les tableaux sont des ensembles typés et de taille fixe d'éléments.

```
Personnage [] tab = new Personnage [50];  
Personnage [] tab = new Personnage[ LIMITE ];
```

- Insertion d'un objet

```
tab [12] = new Personnage ();
```

- Lecture

```
if(tab [8] == null ){  
if(tab.length > 10) {}
```

# Les tableaux

---

- Il est possible d'initialiser un tableau à la création (contenu par défaut)

```
int [] tab = {10 ,20 ,30 ,40};  
char [] tabP = {'a','b','c','d'};  
Personnage [] tabPers = {new Personnage (), new Personnage(), new Personnage ()};
```

- Il est possible de créer des tableaux à 2 dimensions (matrices) et plus

```
float [][] tab =new float [10][12];  
float [][] tab = { {12.3 ,14.5 ,11.5} , {11.0 ,42.0 ,13.37} };  
float [][][] tab3D =new float [10][12][14];
```