



جامعة السلطان مولاي سليمان
Université Sultan Moulay Slimane

المدرسة الوطنية للعلوم التطبيقية - بني ملال
Ecole Nationale des Sciences Appliquées - Béni Mellal



المملكة المغربية
وزارة التعليم العالي
والبحث العلمي والابتكار

Ecole National des sciences appliquées de Beni Mellal

JAVA



Pr. Sara BAGHDADI

Docteur en Informatique et Intelligence Artificielle (IA)

Modalités de Validation du module



➤ *JAVA*

- *Examen : 60%*
- *Autres : 40% (Participation au TD, Compte rendu de TP, Projet, Présentation orale, devoirs, ...)*

$$\text{Note du module} = 0.6 \times \text{Note(Examen)} + 0.4 \times \text{Note(Autres)}$$





Attribution des Projets:

Chaque étudiant recevra un projet.

Les projets seront attribués en fonction des préférences ou par tirage au sort.

Une liste détaillée des projets sera fournie.

Projets

Projets avec Fichiers et Bases de Données

1. **Gestion de contacts avancée** → Ajouter, modifier, supprimer des contacts et les stocker en fichier ou en base de données.
2. **Application de gestion des étudiants** → CRUD (Create, Read, Update, Delete) des étudiants avec SQLite/MySQL.
3. **Système de gestion de stock** → Ajout, suppression et mise à jour de produits d'un magasin.
4. **Carnet de recettes** → Ajouter, modifier et rechercher des recettes de cuisine.
5. **Gestion de bibliothèque** → Stocker et gérer les emprunts et retours de livres.
6. **Gestionnaire de notes scolaires** → Saisie et calcul des moyennes des étudiants.
7. **Journal personnel sécurisé** → Un bloc-notes avec protection par mot de passe.
8. **Gestion de factures** → Génération et sauvegarde de factures sous format texte ou PDF.



Méthodologie de travail:

- *Planification.*
- *Développement.*
- *Documentation* (en parallèle de la réalisation de votre projet):
 - ✓ **Commenter le code.**
 - ✓ **Prendre des captures d'écran (des différentes étapes de votre projet (interface utilisateur)).**
 - ✓ **Rédiger un rapport décrivant le projet.**
- *Présentation.*

Évaluation des Projets

- *Fonctionnalités : Respect des exigences du projet.*
- *Qualité du code : Lisibilité, commentaires, structure.*
- *Design.*
- *Documentation : Rapport clair et détaillé.*
- *Présentation : Clarté, démonstration, réponses aux questions.*



À votre avis, quels critères font qu'un langage est populaire ?

Réponse :

- Facilité d'apprentissage
- Performances et efficacité
- Portabilité et compatibilité
- Large communauté et support
- Domaines d'application (web, mobile, IA, etc.)




Pourquoi pensez-vous que Java est encore utilisé après plus de 25 ans ?

Réponse :

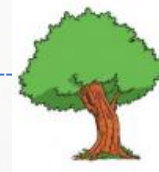
- Sa portabilité (WORA : Write Once, Run Anywhere)
- Son écosystème riche (bibliothèques, frameworks)
- Sa robustesse et sa sécurité
- Son usage dans les grandes entreprises et systèmes critiques

Historique et évolution de Java



- L'histoire du langage de programmation Java commence: Décembre, 1990.
- publié officiellement le 23 mai 1995.
- par  par une équipe dirigée par James Gosling

initialement baptisé Oak (Oak signifiant chêne)



- le nom Oak étant déjà utilisé
- Le nouveau nom : le café « **Java** » en argot américain. la boisson préférée des programmeurs de l'équipe.
- Le logo choisi = une tasse de café fumant



James Gosling, le père de Java





La société Sun a été ensuite rachetée en 2009 par la société Oracle qui détient et maintient désormais Java.

- Le 12 avril 2010, James Gosling, le créateur du langage de programmation Java démissionne d'Oracle pour des motifs qu'il ne souhaite pas divulguer.

- Versions majeures : Java SE (Standard Edition), Java EE (Enterprise Edition), Java ME (Micro Edition).
- Évolution avec des mises à jour régulières (Java 8, 11, 17, etc.).

Avantages et caractéristiques de Java



- Simple
- Apprentissage facile
 - Familier
- Syntaxe proche de celle de C/C++
 - Orienté objet
 - Java ne permet d'utiliser que des objets (hors les types de base)
 - Java est un langage objet de la famille des langages de classe comme C++



- Java est indépendant de l'architecture :
Le bytecode généré par le compilateur est indépendant de toute architecture.
Toute application peut donc tourner sur une plate-forme implémentant une machine virtuelle Java

« Ecrire une fois, exécuter partout »

- Sûr
 - Seul le bytecode est transmis, et «vérifié» par l'interpréteur.



Fiable

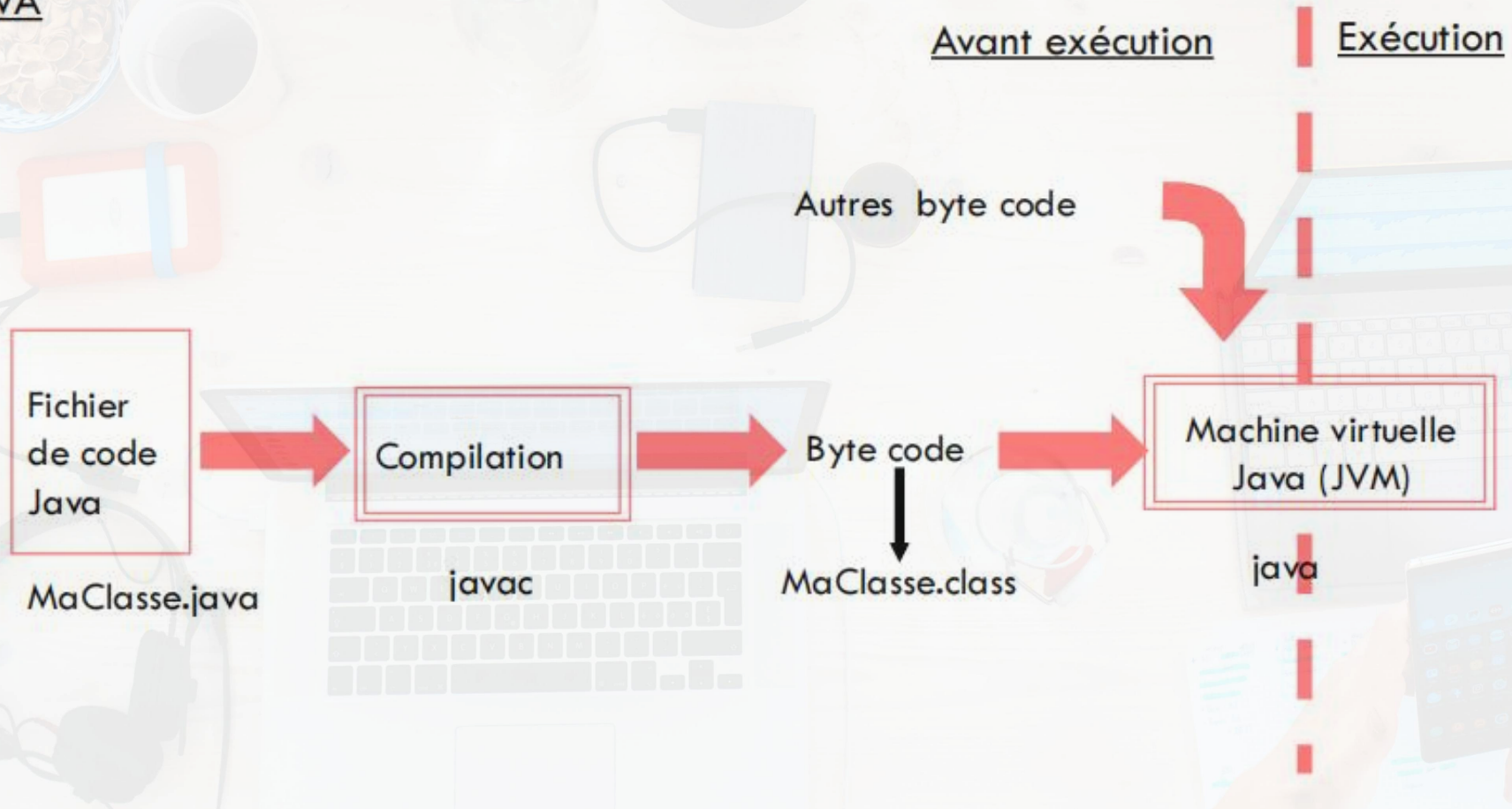
Gestion automatique de la mémoire
(*ramasse-miette* ou *"garbage collector"*)

- Sources d'erreurs limitées
 - typage fort,
 - pas d'héritage multiple,
 - pas de manipulations de pointeurs, etc.
- Java est un langage interprété
 - La compilation d'un programme Java crée du code java: le "byte-code"
 - Sur n'importe quelle plate-forme, une machine virtuelle Java peut interpréter le code afin qu'il soit exécuté



Langage interprété

CAS DE JAVA





- Java fournit de nombreuses librairies de classes remplissant des fonctionnalités très diverses : c'est l'API Java

API (Application and Programming Interface /Interface pour la programmation d'applications) : Ensemble de bibliothèques permettant une programmation plus aisée car les fonctions deviennent indépendantes du matériel

- Ces classes sont regroupées, par catégories, en paquetages (ou "packages").
- possibilité de génération automatique avec l'outil Javadoc.



- Elle est au format **HTML**.

Où trouver les informations sur les classes de l'API

- sous le répertoire `jdk1.x/docs/api` dans le **JDK**

- Sur le site :

<http://docs.oracle.com/javase/7/docs/api/>



Environnement de développement Java

- **JDK (Java Development Kit)** : Ensemble des outils pour développer en Java.
- **JRE (Java Runtime Environment)** : Environnement d'exécution des programmes Java.
- **IDE (Environnements de développement)** : Eclipse, IntelliJ IDEA, NetBeans, VS Code.



Comment un ordinateur exécute-t-il un programme écrit en Java ?

Qu'est-ce qu'un commentaire dans un programme ? À quoi ça sert ?

Pouvez-vous citer quelques types de données en programmation ?

Quelle est la différence entre un tableau et une variable simple ?



Exécution d'un programme Java

Un programme Java suit plusieurs étapes :

- 1. Écriture du code source (fichier .java)*
- 2. Compilation en bytecode (fichier .class)*
- 3. Interprétation par la JVM (Java Virtual Machine)*

Exemple de programme simple :

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```




Commentaires en Java

- Commentaire monoligne : `// Ceci est un commentaire`
- Commentaire multilignes :

`/*`

`Ceci est un commentaire
sur plusieurs lignes`

`*/`

- Javadoc (Documentation du code) : Commentaires d'explication

`/**`

`* Cette classe affiche un message de bienvenue.`

`*/`



Types de données en Java

- Types primitifs : int, double, char, boolean...
- Types référencés : String, objets...

```
int age = 25;  
double prix = 19.99;  
boolean estActif = true;  
char lettre = 'A';  
String nom = "Java";
```



Opérateurs en Java

- Arithmétiques : +, -, *, /, %
- Relationnels : ==, !=, <, >, <=, >=
- Logiques : &&, ||, !
- Affectation : =, +=, -=, *=, /=



Structures de contrôle

- Conditionnelles : if/else, switch
- Boucles : for, while, do-while



Les tableaux

- Les tableaux permettent de stocker plusieurs valeurs de même type dans une variable.
 - Les valeurs contenues dans la variable sont repérées par un indice
 - En langage java, les tableaux sont des objets
- Déclaration
 - `int tab []; String chaines[];`
- Création d'un tableau
 - `tab = new int [20]; // tableau de 20 int`
 - `chaines = new String [100]; // tableau de 100 chaine`



Les tableaux

- Le nombre d'éléments du tableau est mémorisé. Java peut ainsi détecter à l'exécution le dépassement d'indice et générer une exception. Mot clé **length**
 - Il est récupérable par `nomTableau.length`
`int taille = tab.length; //taille vaut 20`
- Comme en C/C++, les indices d'un tableau commencent à '0'. Donc un tableau de taille 100 aura ses indices qui iront de 0 à 99.



Les tableaux

- Initialisation

```
tab[0]=1;
```

```
tab[1]=2; //etc.
```

```
noms[0] = new String( "Boule");
```

```
noms[1] = new String( "Bill");//etc
```

- Création et initialisation simultanées

```
String noms [ ] = {"Boule","Bill"};
```

```
Point pts[ ] = { new Point (0, 0), new Point (10, -1)};
```



Exercices

- 1.Écrivez un programme Java qui déclare et affiche les valeurs d'un tableau de 5 nombres entiers.
- 2.Implémentez un programme qui vérifie si un nombre est pair ou impair.
- 3.Réalisez un programme qui calcule la somme des nombres de 1 à 10 en utilisant une boucle for.



Point d'entrée d'un programme Java

- Pour pouvoir faire un programme exécutable il faut toujours une classe qui contient une méthode particulière, la méthode « main »
 - c'est le point d'entrée dans le programme : le microprocesseur sait qu'il va commencer à exécuter les instructions à partir de cet endroit

```
public static void main(String arg[ ])  
{  
    .../  
}
```




Écriture sur l'écran

- Pour écrire dans l'écran on utilise la ligne de commande suivante
 - `System.out.println(Texte)`
- Le texte est sous format alphanumérique
- Le texte peut être stocké dans une variable



Lecture clavier

- Pour lire à partir du clavier, on utilise la classe Scanner
- La classe Scanner a des fonctionnalités plus avancées que l'affichage à l'écran
- Nous limiterons son utilisation dans ce cours à l'affichage



Exemple

Fichier Bonjour.java

```
public class Bonjour
{ //Accolade débutant la classe Bonjour
  public static void main(String args[])
  { //Accolade débutant la méthode main
    /* Pour l'instant juste une instruction */
    System.out.println("bonjour");
  } //Accolade fermant la méthode main
} //Accolade fermant la classe Bonjour
```

La classe est l'unité de base de nos programmes. Le mot clé en Java pour définir une classe est **class**

Identificateurs

- On a besoin de nommer les classes, les variables, les constantes, etc. ; on parle d'identificateur.
- Les identificateurs commencent par une lettre, _ ou \$

Attention : Java distingue les majuscules des minuscules

- Conventions sur les identificateurs :
 - Si plusieurs mots sont accolés, mettre une majuscule à chacun des mots sauf le premier.
 - exemple : uneVariableEntiere
 - La première lettre est majuscule pour les classes et les interfaces
 - exemples : MaClasse, UneJolieFenetre

Identificateurs

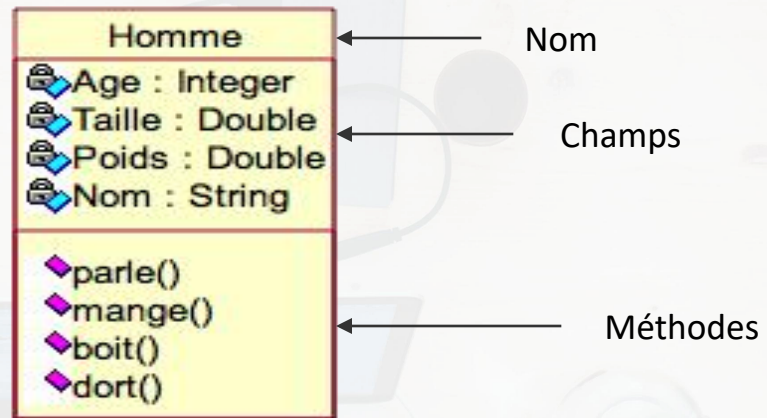
- Conventions sur les identificateurs :
 - La première lettre est minuscule pour les méthodes, les attributs et les variables
 - exemples : setLongueur, i, uneFenetre
 - Les constantes sont entièrement en majuscules
 - exemple : LONGUEUR_MAX

Les mots réservés de Java

<code>abstract</code>	<code>default</code>	<code>goto</code>	<code>null</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>package</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>private</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>protected</code>	<code>throws</code>
<code>case</code>	<code>extends</code>	<code>instanceof</code>	<code>public</code>	<code>transient</code>
<code>catch</code>	<code>false</code>	<code>int</code>	<code>return</code>	<code>true</code>
<code>char</code>	<code>final</code>	<code>interface</code>	<code>short</code>	<code>try</code>
<code>class</code>	<code>finally</code>	<code>long</code>	<code>static</code>	<code>void</code>
<code>continue</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>volatile</code>
<code>const</code>	<code>for</code>	<code>new</code>	<code>switch</code>	<code>while</code>

- La POO est une méthode d'implémentation dans laquelle les programmes sont organisés sous forme de collections coopératives d'objets.
- Chaque objet représente une instance d'une classe quelconque et dont toutes les classes sont membres d'une hiérarchie de classes unies à travers des relations d'héritage.

- Représentation graphique d'une classe



Une classe représentée avec la notation UML
(Unified Modeling Language)

- La **classe** est la machine qui fabrique les **objets**
- Un moule ou un modèle à partir du quel on peut créer des objets.
- Des objets créés à partir de la même classe auront des aspects semblables (**mêmes attributs et méthodes**).
- Un objet d'une classe est aussi appelé **instance** de cette classe.

L'instanciation

- Instanciation : concrétisation d'une classe en un objet « *concret* ».
- Instance
 - représentant physique d'une classe
 - obtenu par moulage du dictionnaire des variables et détenant les valeurs de ces variables.
 - Son comportement est défini par les méthodes de sa classe
- Exemple :
 - si nous avons une classe voiture, alors votre voiture est une instance particulière de la classe voiture.
 - Classe = concept, description
 - Objet = représentant **concret** d'une classe

L'encapsulation

- Regrouper des données(attributs)et un comportement (méthodes) dans une même classe
- Réglementer l'accès aux données de l'extérieur (par d'autres objets).



- Impossible d'agir directement sur les données d'un objet
- Il est nécessaire de passer par ses méthodes.

- L'encapsulation

- L'appel de la méthode d'un objet



- *envoi de message à l'objet.*

- L'encapsulation permet de définir des niveaux de visibilité des éléments de la classe.



- Définir les droits d'accès aux données.

• L'encapsulation/Niveaux de visibilité

- Publique: plus bas niveau de protection.
 - les fonctions de toutes les classes peuvent accéder aux données ou aux méthodes
- Protégée: niveau moyen de protection
 - l'accès aux données est réservé aux fonctions des classes héritières et dérivées des classes dérivées.
- Privée: le plus élevé niveau de protection
 - l'accès aux données est limité aux méthodes de la classe elle-même

Les constructeurs

- L'appel de **new** pour créer un nouvel objet déclenche, dans l'ordre :
 - L'allocation mémoire nécessaire au stockage de ce nouvel objet et l'initialisation par défaut de ces attributs,
 - L'initialisation explicite des attributs, s'il y a lieu,
 - L'exécution d'un constructeur.
- Un constructeur est une méthode d'initialisation.

```
public class Application
{
    public static void main(String args[])
    {
        Personne p = new Personne()
        p.setNom(" ahmed");
    } }
```

Le constructeur est ici celui par défaut (pas de constructeur défini dans la classe Personne)

Les constructeurs

- Le constructeur est une méthode :
 - de même nom que la classe,
 - sans type de retour.
- Toute classe possède au moins un constructeur. Si le programmeur ne l'écrit pas, il en existe un par défaut, sans paramètres, de code vide.

Les constructeurs

- Pour une même classe, il peut y avoir plusieurs constructeurs, de signatures différentes.
- L'appel de ces constructeurs est réalisé avec le **new** auquel on fait passer les paramètres.
 - **p1 = new Personne(" ahmed", " radi", 33);**
- Attention
 - Si le programmeur crée un constructeur (même si c'est un constructeur avec paramètres), le constructeur par défaut n'est plus disponible.

Les constructeurs

Personne.java

```
public class Personne
{
    public String nom;
    public String prenom;
    public int age;
    public Personne(String unNom,
                    String unPrenom,
                    int unAge)
    {
        nom=unNom;
        prenom=unPrenom;
        age = unAge;
    }
}
```

Va donner une erreur à la compilation

Définition d'un Constructeur. Le constructeur par défaut (Personne()) n'existe plus. Le code précédent occasionnera une erreur

```
public class Application
{
    public static void main(String args[])
    {
        Personne p = new Personne()
        p.setNom(" ahmed");
    }
}
```

Les constructeurs

Personne.java

```
public class Personne
{
    public String nom;
    public String prenom;
    public int age;
    public Personne()
    {
        nom=null; prenom=null;
        age = 0;
    }
    public Personne(String unNom,
                    String unPrenom, int unAge)
    {
        nom=unNom;
        prenom=unPrenom; age = unAge;
    } }

```

Redéfinition d'un
Constructeur sans paramètres

On définit plusieurs constructeurs
qui se différencient uniquement
par leurs paramètres (on parle
de leur signature)

EXERCICE

A top-down view of a cluttered wooden desk. In the center, a silver laptop is open. To its left is another laptop with a keyboard. Below the central laptop are a pair of black headphones. To the right of the central laptop is a pair of glasses. Above the central laptop is a small bowl of snacks and a cup of coffee. In the bottom right corner, a person's hands are visible, holding a smartphone. The desk is covered with various cables and other small items, creating a busy, workspace-like environment.