POO

JAVA LANGUAGE

SAGIM 2024/2025

Qu’est-ce qu’un objet en Java ?

En Java, un **objet** est une **instance d'une classe**. C’est une entité qui a :

- **des attributs (données)** → appelés **champs** ou **variables d’instance**
- **des comportements (actions)** → appelés **méthodes**

Qu’est-ce qu’un objet en Java ?

```
public class Voiture {
    String marque;
    String couleur;
    int annee;
 void demarrer() {
        System.out.println("La voiture démarre !");
    }
void afficherInfos() {
        System.out.println("Marque : " + marque);
        System.out.println("Couleur : " + couleur);
        System.out.println("Année : " + annee);
    }
}
```

Qu’est-ce qu’un objet en Java ?

```
public class Main {
    public static void main(String[] args) {
        Voiture v1 = new Voiture();
            v1.marque = "Toyota";
            v1.couleur = "Rouge";
            v1.annee = 2022;
            v1.demarrer();
            v1.afficherInfos();
    }
}
```

Qu’est-ce qu’un objet en Java ?

Terme Java	Signification
Classe	Modèle, plan de construction (ex : une recette)
Objet	Instance réelle (ex : un gâteau fait avec la recette)
Attributs	Données spécifiques à l'objet (ex : couleur, marque...)
Méthodes	Comportements que l’objet peut exécuter

JAVA TPs

EXEMPLE D'UTILISATION DES CLASSES

EXERCICE 1:

Créer une classe `Personne` avec les attributs nom, prénom et âge. Ajouter une méthode pour afficher les informations de la personne.

Énoncé :

1. Crée une classe `Personne` avec :
   - `String nom`
   - `String prenom`
   - `int age`
2. Ajoute une méthode `afficherInfos()` qui affiche les données de la personne.
3. Dans la classe principale (`Main`), crée 2 objets `Personne` avec des données différentes et appelle leur méthode `afficherInfos()`.

Personne.java

```
public class Personne {
    String nom;
    String prenom;
    int age;

void afficherInfos() {
        System.out.println("Nom : " + nom);
        System.out.println("Prénom : " + prenom);
        System.out.println("Âge : " + age + " ans");
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {

        Personne p1 = new Personne();
        p1.nom = "Ali";   p1.prenom = "Karim";   p1.age = 30;

        Personne p2 = new Personne();
        p2.nom = "Benali";  p2.prenom = "Sara";  p2.age = 25;

        System.out.println("=== Informations de la première personne ===");
        p1.afficherInfos();

        System.out.println("\n=== Informations de la deuxième personne ===");
        p2.afficherInfos();
    }
}
```

Encapsulation en Java

Public – Private - Protected

L’encapsulation :

L'**encapsulation** est un **principe fondamental de la programmation orientée objet** (POO). Elle consiste à **protéger les données** d'un objet en les **rendant inaccessibles directement depuis l'extérieur**, sauf via des **méthodes spéciales appelées getters et setters**.

Pourquoi utiliser l'encapsulation ?

1. **Sécurité des données** : on empêche les accès ou modifications incorrectes.
2. **Contrôle d'accès** : on peut vérifier ou transformer les données avant de les modifier.
3. **Facilité de maintenance** : on peut changer l'implémentation interne sans impacter l’extérieur.

Structure d'une classe avec encapsulation:

```
public class Compte {

    private double solde;

    public double getSolde() {
        return solde;
    }

    public void setSolde(double s) {
        if (s >= 0) {
            this.solde = s;
        } else {
            System.out.println("Solde invalide !");
        }
    }
}
```

Résumé :

Élément	Rôle
private	Rend l’attribut invisible à l’extérieur
getter (getX)	Permet de lire la valeur
setter (setX)	Permet de modifier la valeur de façon contrôlée

LE CONSTRUCTEUR

CONSTRUCTEUR EN JAVA

Constructeur en Java

En Java, un **constructeur** est une **méthode spéciale** utilisée pour **initialiser un objet** au moment de sa création.
Il ressemble à une méthode, **mais il n'a pas de type de retour** (pas même `void`) et **porte exactement le même nom que la classe**.

Pourquoi utiliser un constructeur ?

- Pour initialiser automatiquement les attributs d'un objet dès sa création.
- Pour éviter de devoir écrire plusieurs lignes comme objet.nom = ...;

après avoir créé l'objet.

Syntaxe de base

```
public class NomDeClasse {
    // Constructeur
    public NomDeClasse() {
        // Initialisation
    }
}
```

EXEMPLE :

```
public class Etudiant {
    public String nom;
    public int age;
     public Etudiant(String nom, int age) {
        this.nom = nom;
        this.age = age;
    }
     public void afficherInfos() {
        System.out.println("Nom : " + nom);
        System.out.println("Âge : " + age);
    }
}
public class Main {
    public static void main(String[] args) {
        Etudiant e = new Etudiant("Omar", 22);
        e.afficherInfos();
    }
}
```

CONSTRUCTEUR PAR DÉFAUT :

- Si **aucun constructeur** n'est défini, Java en crée un **automatiquement**, vide :

```
public Etudiant() { }
```

- Mais si tu déclares **un constructeur avec arguments**, Java **ne crée plus** le constructeur vide automatiquement.

Résumé:

Élément	Rôle
Constructeur	Initialise un objet lors de sa création
Nom du constructeur	Identique à la classe
Pas de type de retour	Ni void, ni autre type
Peut avoir des paramètres	Pour personnaliser l’objet dès le départ

JAVA TPs

ENCAPSULATION (getters – setters)

Exercice :

Création et gestion d’un compte bancaire en Java

Vous êtes chargé de modéliser un système de gestion de comptes bancaires très simple.
Chaque compte est associé à un **titulaire** et à un **solde**. Le solde ne peut jamais être négatif.

Travail à faire :

1.Créez une classe nommée CompteBancaire contenant :

- deux attributs privés : titulaire (String) et solde (double),
- un **constructeur** qui permet d’instancier un compte avec un titulaire et un solde initial (le solde doit être ≥ 0),
- des **getters et setters** pour chaque attribut.
  Le setter du solde doit **refuser** les valeurs négatives avec un message d’erreur,
- une méthode afficherInfos() qui affiche le nom du titulaire et le solde du compte.

2.Créez une classe Main avec une méthode main() qui :

- crée un compte au nom de "Ahmed" avec un solde initial de 500 DH,
- affiche les informations du compte,
- modifie le solde à 300 DH et affiche le nouveau solde,
- tente de modifier le solde à -100 DH et observe le comportement du programme.

Exercice :

Création et gestion d’un compte bancaire en Java

```
public class CompteBancaire {
    private String titulaire;
    private double solde;

    public CompteBancaire(String titulaire, double soldeInitial) {
        this.titulaire = titulaire;
        this.setSolde(soldeInitial);
    }
     public String getTitulaire() {
        return titulaire;
    }
     public void setTitulaire(String titulaire) {
        this.titulaire = titulaire;
    }
     public double getSolde() {
        return solde;
    }
```

Exercice :

Création et gestion d’un compte bancaire en Java

```
    public void setSolde(double solde) {
        if (solde >= 0) {
            this.solde = solde;
        } else {
            System.out.println("Erreur : le solde ne peut pas être négatif.");
        }
    }

    public void afficherInfos() {
        System.out.println("Titulaire : " + titulaire);
        System.out.println("Solde : " + solde + " dh");
    }
}
```

Exercice :

Création et gestion d’un compte bancaire en Java

```
public class Main {
    public static void main(String[] args) {
        CompteBancaire compte = new CompteBancaire("Ahmed", 500.0);
        compte.afficherInfos();

        compte.setSolde(300.0);
        System.out.println("Nouveau solde : " + compte.getSolde() + " DH");

        compte.setSolde(-100.0);
    }
}
```

JAVA TPs

Constructeur + accesseurs

Exercice :

Création et gestion d'une classe Livre en Java

1. Crée une classe `Livre` avec les **attributs privés** :
   - `String titre`
   - `String auteur`
   - `double prix`
2. Fournis les **getters** et **setters** pour chaque attribut :
   - Le prix doit être **positif**. Si on tente d'affecter une valeur négative, afficher un message d'erreur.
3. Crée un **constructeur** qui initialise les 3 attributs.
4. Ajoute une méthode `afficherInfos()` qui affiche les informations du livre.
5. Dans une classe `Main`, crée deux objets `Livre` et affiche leurs informations.

Exercice :

Création et gestion d'une classe Livre en Java

```
public class Livre {
    private String titre;
    private String auteur;
    private double prix;

 public Livre(String titre, String auteur, double prix) {
        this.titre = titre;
        this.auteur = auteur;
        setPrix(prix);
    }
 public String getTitre() {
        return titre;
    }
 public String getAuteur() {
        return auteur;
    }
     public double getPrix() {
        return prix;
    }
```

Exercice :

Création et gestion d'une classe Livre en Java

```
public void setTitre(String titre) {
        this.titre = titre;
    }
public void setAuteur(String auteur) {
        this.auteur = auteur;
    }
public void setPrix(double prix) {
        if (prix >= 0) {
            this.prix = prix;
        } else {
            System.out.println("Erreur : le prix ne peut pas être négatif !");
        }
    }
public void afficherInfos() {
        System.out.println("Titre : " + titre);
        System.out.println("Auteur : " + auteur);
        System.out.println("Prix : " + prix + " dh");
    }
}
```

Exercice :

Création et gestion d'une classe Livre en Java

```
public class Main {
    public static void main(String[] args) {


        Livre livre1 = new Livre (''java poo '', ''ahmed'', 10.5);
        livre1.afficherInfos();

        System.out.println();


        Livre livre2 = new Livre(''langage c'', ''laila'', -5.0);
        livre2.setPrix(12.0);
        livre2.afficherInfos();
    }
}
```