

# TP N°3

## Chapitre 3 : Héritage et polymorphisme (Java)

Licence : INFO-AP | G-INFO

*Université Moulay Ismaïl – Faculté des Sciences Meknès*

*Pr. MAROUANE NAZIH & Pr. ISMAILI ALAOUI ELMEHDI*

### Énoncé du TP : Gestion d'une Faculté des Sciences

L'objectif de ce TP est de modéliser une partie du fonctionnement d'une Faculté des Sciences en utilisant l'héritage, le polymorphisme, la redéfinition de méthodes et les tableaux d'objets en Java.

Vous serez amenés à définir des classes représentant les acteurs de la faculté (personnes, étudiants, enseignants) ainsi que ses entités pédagogiques (filieres et modules).

### Travail demandé

#### Classe Personne

Créer une classe `Personne` avec les attributs suivants :

- `nom` : `String`
- `prenom` : `String`
- `age` : `int`

Travail à réaliser :

- Constructeur sans paramètre,
- Constructeur d'initialisation,
- Méthode `toString()`,
- Méthode `equals(Object o)` (égalité : même `nom` et même `prenom`).

#### Classe Etudiant (hérite de Personne)

Attributs supplémentaires :

- `cne` : `String`
- `filier` : `String`

Travail demandé :

- Constructeurs (vide et complet),
- Redéfinition de `toString()`,
- Redéfinition de `equals(Object o)` (égalité par `cne` uniquement).

#### Classe Enseignant (hérite de Personne)

Attributs supplémentaires :

- `grade` : `String`
- `specialite` : `String`

Travail demandé :

- Constructeurs,
- Redéfinition de `toString()`,
- Méthode `equals(Object o)` (égalité : même `nom`, `prenom` et `spécialité`).

### Classe Filiere (avec tableau d'étudiants)

Attributs :

- `nom` : `String`
- `responsable` : `Enseignant`
- `etudiants` : `Etudiant[]`
- `nbEtudiants` : `int`

Travail demandé :

- Constructeur d'initialisation (`nom`, `responsable`, capacité du tableau),
- Méthode `ajouterEtudiant(Etudiant e)`,
- Méthode `afficherEtudiants()`,
- Méthode `rechercherEtudiantParCne(String cne)`,
- Méthode `toString()`.

### Classe Module

Attributs :

- `intitule` : `String`
- `professeur` : `Enseignant`
- `coefficient` : `int`

Travail demandé :

- Constructeurs,
- Méthode `toString()`.

### Étude de cas : Classe de test GestionFaculte

La classe `GestionFaculte` doit permettre de tester toutes les fonctionnalités implémentées.

#### 6.1 Création des objets (exemples imposés)

a) **Enseignants** Créer les enseignants suivants :

- **Enseignant 1** : Kassimi Hamid, 50 ans, Professeur, spécialité Réseaux.
- **Enseignant 2** : Lahmidi Mohamed, 40 ans, MC, spécialité Intelligence Artificielle.

b) **Filière** Créer une filière :

- `Nom` : `Informatique`
- `Responsable` : Enseignant 1
- `Capacité` : 30 étudiants

c) **Étudiants** Créer les cinq étudiants suivants :

Nom	Prénom	Âge	CNE
Ali	Amine	20	CNE001
Sara	Nadia	21	CNE002
Omar	Yassin	19	CNE003
Hiba	Salma	22	CNE004
Karim	Ilyas	20	CNE005

Ajouter ces cinq étudiants à la filière.

## 6.2 Affichage et recherche

- Afficher les informations de la filière (`toString()`),
- Afficher la liste des étudiants inscrits,
- Rechercher l'étudiant portant le CNE `CNE003`,
- Tester la recherche d'un CNE inexistant (`CNE999`).

## 6.3 Tests de comparaison et polymorphisme

a) **Comparaison d'enseignants** Créer un troisième enseignant :

- Même nom : Kassimi
  - Même prénom : Hamid
  - Âge : 55 ans
  - Grade : Professeur
  - Spécialité : Réseaux
- Tester :
- `ens1.equals(ens3)` (doit être vrai),
  - `ens1.equals(ens2)` (doit être faux).

b) **Polymorphisme** Créer un tableau de `Personne` contenant :

- les enseignants : `ens1`, `ens2`, `ens3`,
- les étudiants : `e1`, `e2`, `e3`, `e4`.

Parcourir le tableau et afficher chaque élément pour illustrer la liaison dynamique.

## 6.4 Modules

Créer et afficher les modules suivants :

- **Module 1** : Programmation Java, professeur : Enseignant 2, coefficient 4.
- **Module 2** : Réseaux Informatiques, professeur : Enseignant 1, coefficient 3.