

## Devoir1 : Application Modulaire en Java

### Exercice 1 — Logique & Calcul

**Objectif :** Manipuler les bases du langage (variables, boucles, conditions, méthodes).

**Énoncé :**

Écrire un programme Java CalculStat.java qui :

1. Demande à l'utilisateur de saisir **N** nombres (au clavier).
  2. Stocke ces nombres dans un tableau.
  3. Calcule :
    - le **minimum**
    - le **maximum**
    - la **moyenne**
    - la **somme**
  4. Affiche les résultats sous une forme claire.
  - Trier le tableau (utiliser Arrays.sort).
- 

### Exercice 2 — Programmation Orientée Objet (POO)

**Objectif :** Créer et manipuler des classes, objets, encapsulation, héritage simple.

**Énoncé :**

Créer une classe Personne composée de :

- id (int)
- nom (String)
- age (int)

Créer ensuite une classe Etudiant **héritant** de Personne avec :

- niveau (String) (ex: L1, L2, Master, etc.)

Écrire une classe TestPersonne.java qui :

1. Crée un tableau d'au moins **5 étudiants**.
  2. Affiche la liste.
  3. Trouve et affiche l'étudiant le plus âgé.
  4. Recherche un étudiant par nom (méthode dédiée).
- 

### Exercice 3 — GUI (Swing ou JavaFX)

**Objectif :** Interface graphique, gestion d'événements, affichage dynamique.

**Énoncé (Swing suggéré) :**

Créer une interface graphique de gestion simple des personnes :

Zone	Fonction
TextField Nom	Saisir un nom
TextField Âge	Saisir un âge
Button Ajouter	Ajouter à une liste
JList / JTable Liste des personnes	Afficher les personnes ajoutées

**Actions :**

- Lorsque l'utilisateur clique sur **Ajouter**, l'élément est inséré dans la liste.
- Afficher un message d'erreur si un champ est vide ou incorrect.

**GUI :**

- Ajouter un bouton **Supprimer** l'élément sélectionné.
- Utiliser JTable au lieu de JList.

---

**Exercice 4 — API Java Avancée (Collections, Streams, Files)**

**Objectif :** Manipuler Java Collections & Streams, lecture et écriture fichiers.

**Énoncé :**

Créer une classe GestionEtudiants.java qui :

1. Charge une liste d'étudiants depuis un fichier texte etudiants.txt (format id;nom;age;niveau).
  2. Stocke les données dans une **List**.
  3. Filtre et affiche :
    - les étudiants dont l'âge > 20 (Stream filter)
    - les étudiants triés par nom (Stream sorted)
  4. Permet d'ajouter un nouvel étudiant à la liste puis **réécrit le fichier** avec les nouvelles données.
- Exporter également les étudiants dans un fichier CSV ou JSON.
-

## Structure conseillée du projet

tp/

└─ src/

| └─ ex1/CalculStat.java

| └─ ex2/Personne.java

| └─ ex2/Etudiant.java

| └─ ex2/TestPersonne.java

| └─ ex3/GestionPersonnesGUI.java

| └─ ex4/GestionEtudiants.java

└─ data/

└─ etudiants.txt

---