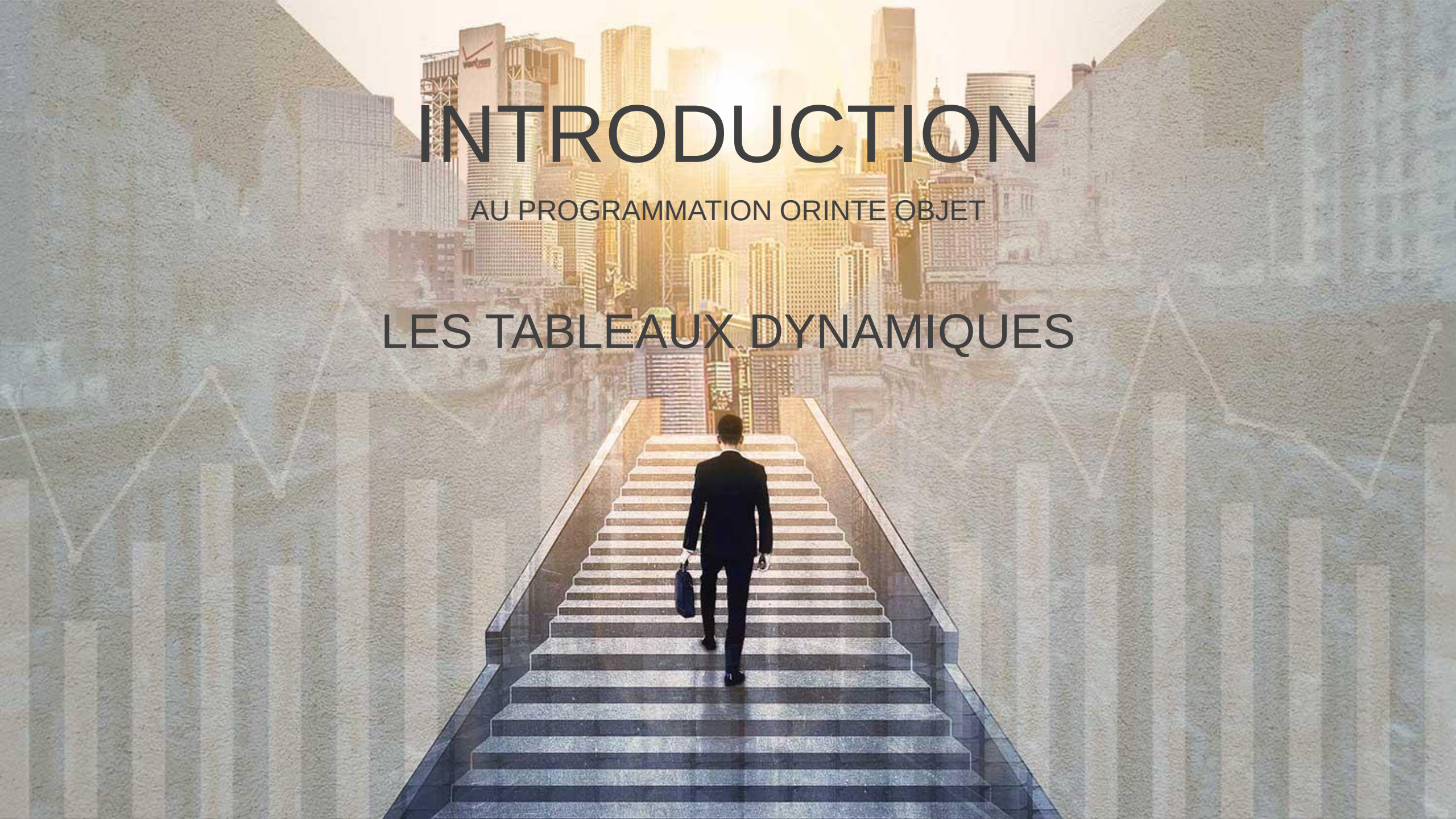
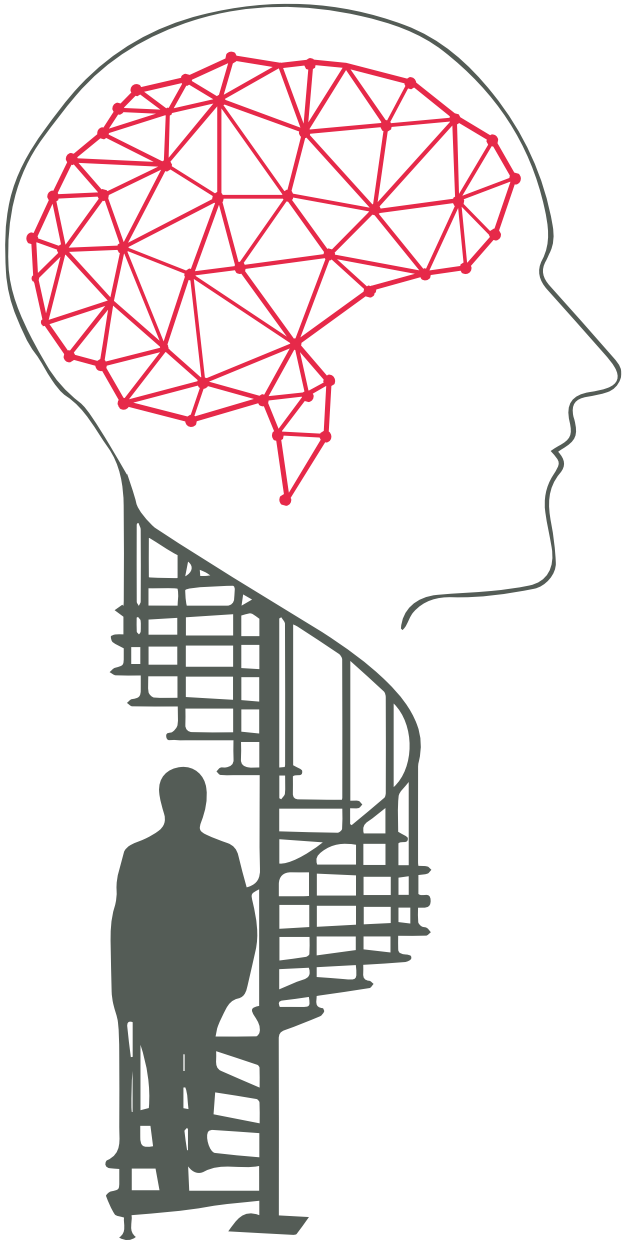


INTRODUCTION

AU PROGRAMMATION ORINTE OBJET

LES TABLEAUX DYNAMIQUES

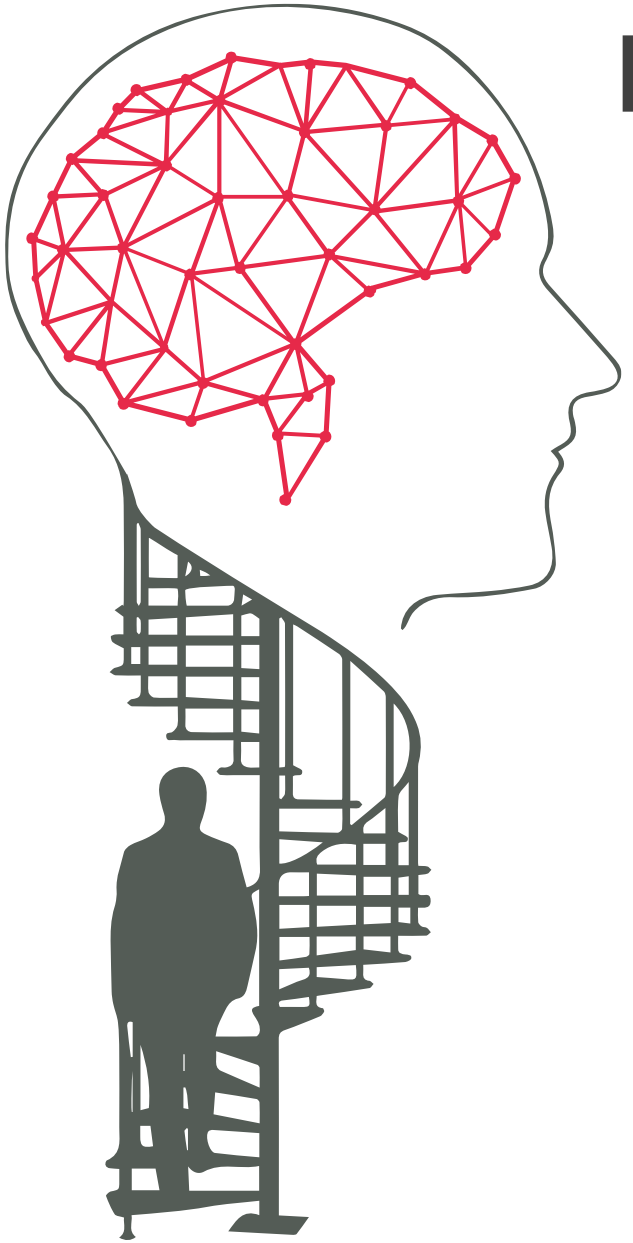


Introduction aux tableaux

Un tableau en Java est une structure de données statique qui permet de stocker plusieurs éléments de même type dans une seule variable.

Caractéristiques :

- Taille fixe dès la création.
- Accès rapide via index (de 0 à n-1).
- Peut contenir des types primitifs ou des objets.



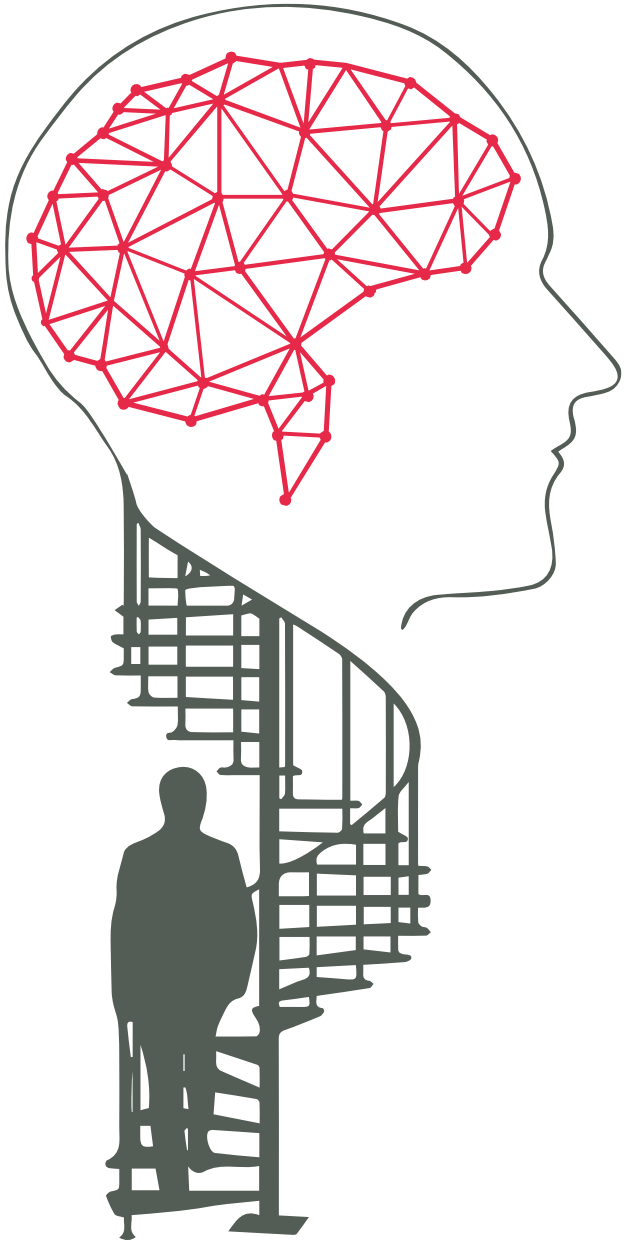
Déclaration et initialisation

Type primitif :

```
int[ ] notes = new int[5];
notes[0] = 10;
notes[1] = 15;
```

Type objet :

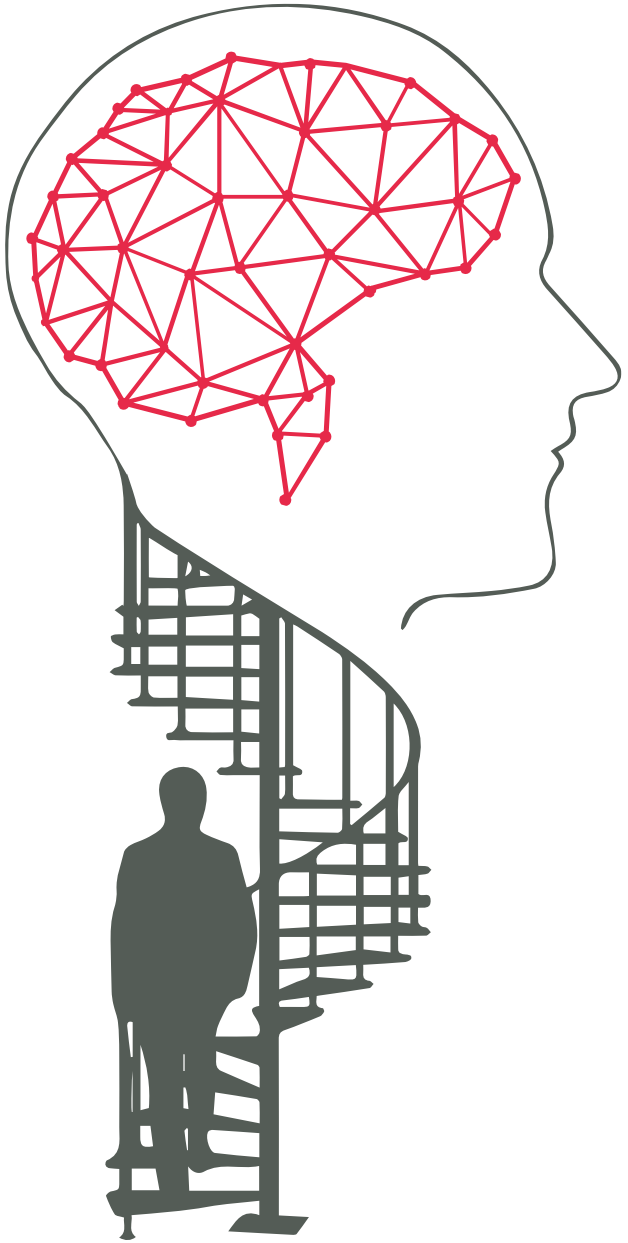
```
Etudiant[ ] liste = new Etudiant[3];
liste[0] = new Etudiant("Ali", 20);
```



Tableaux et Programmation Orientée Objet

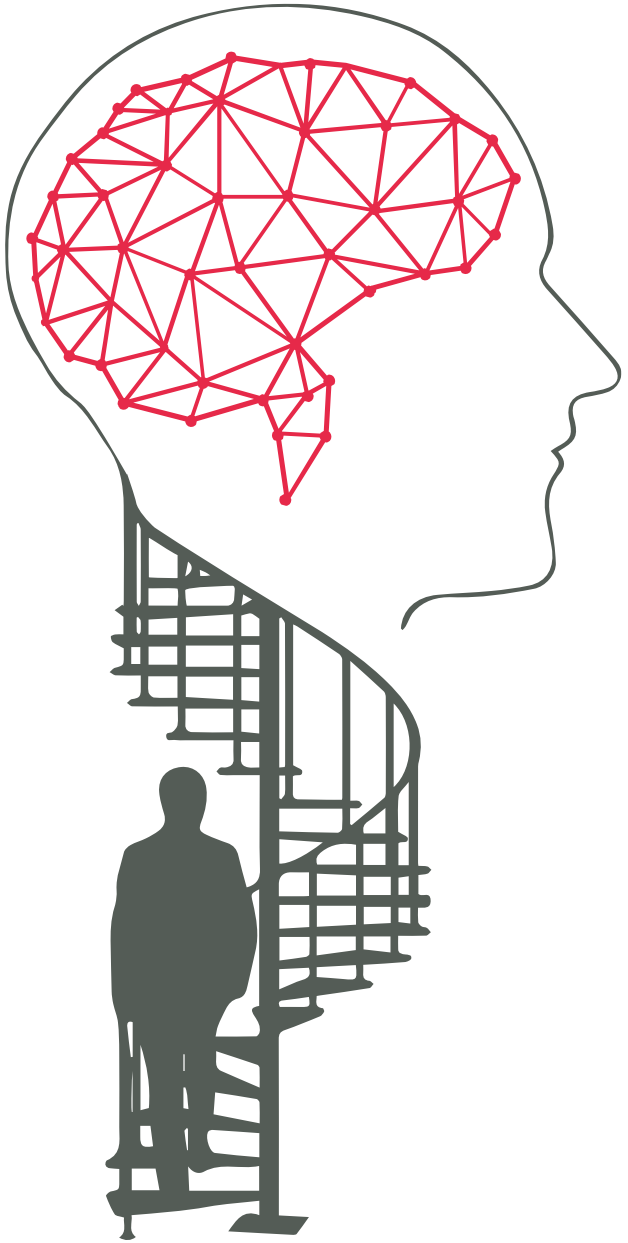
En POO, on peut stocker des objets dans un tableau.

Cela permet par exemple de gérer un ensemble **d'étudiants**, de **produits**, de **clients**, etc.



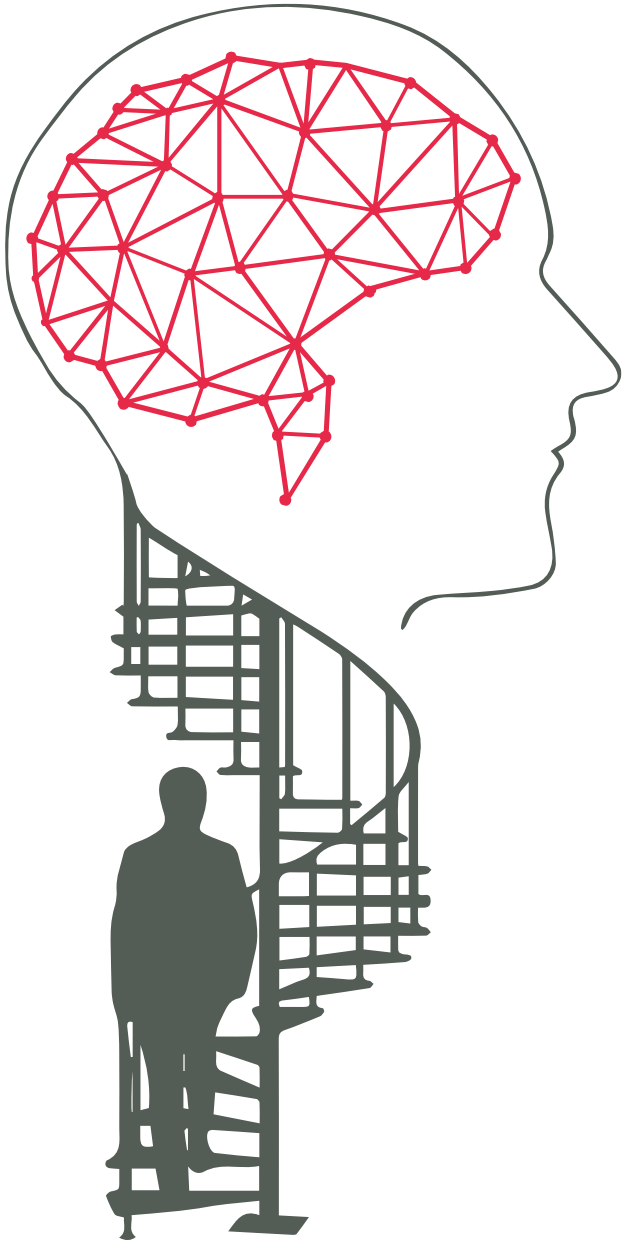
Exemple complet avec une classe Produit

```
public class Produit {
    private String nom;      private double prix;
    public  Produit(String nom, double prix) {
        this.nom = nom;        this.prix = prix;
    }
    public String getNom() {    return nom;   }
    public double getPrix() {    return prix;    }
    public String toString() {
                        return  nom + " - " + prix + " dh";  }
}
```



Exemple complet avec une classe Produit

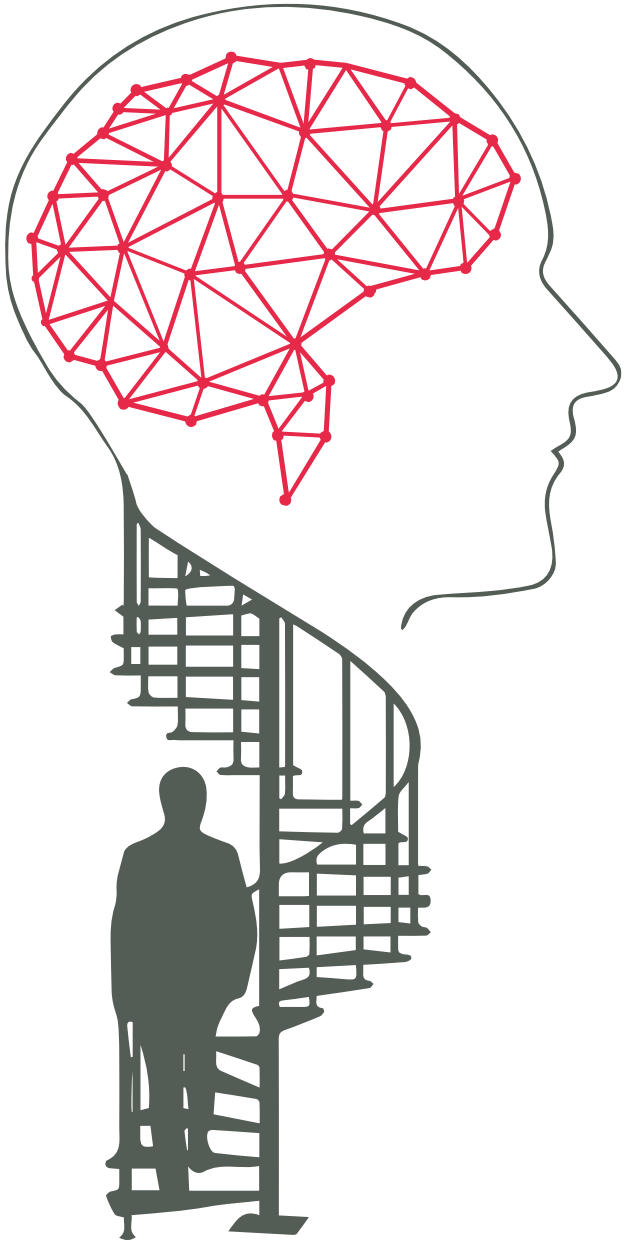
```
public class Magasin {
    public static void main(String[] args) {
        Produit[] stock = new Produit[3];
        stock[0] = new Produit(“PRD 1", 30.0);
        stock[1] = new Produit(“PRD2", 15.0);
        stock[2] = new Produit(“PRD3", 25.0);
    System.out.println("Liste des produits :");
        for (int i = 0; i < stock.length; i++) {
            System.out.println(stock[i]);
        }
```



Exemple complet avec une classe Produit

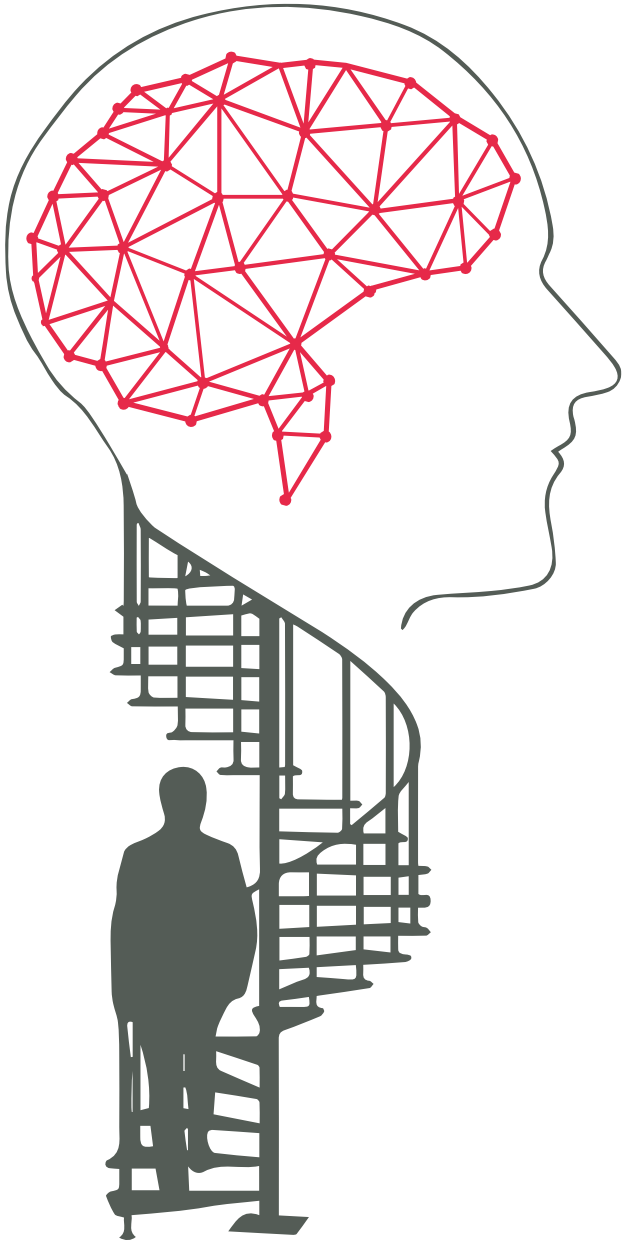
```
        double total = 0;
        for (int i = 0; i < stock.length; i++) {
            total += stock[i].getPrix();
        }

        System.out.println("Valeur totale du stock : “
                                + total + " DH");
    }
}
```



Avantages et inconvénients des tableaux en POO

Avantages	Inconvénients
Simple à utiliser	Taille fixe
Accès rapide aux éléments	Moins flexible qu’un ArrayList
Moins gourmand en mémoire	Pas de méthodes intégrées pratiques



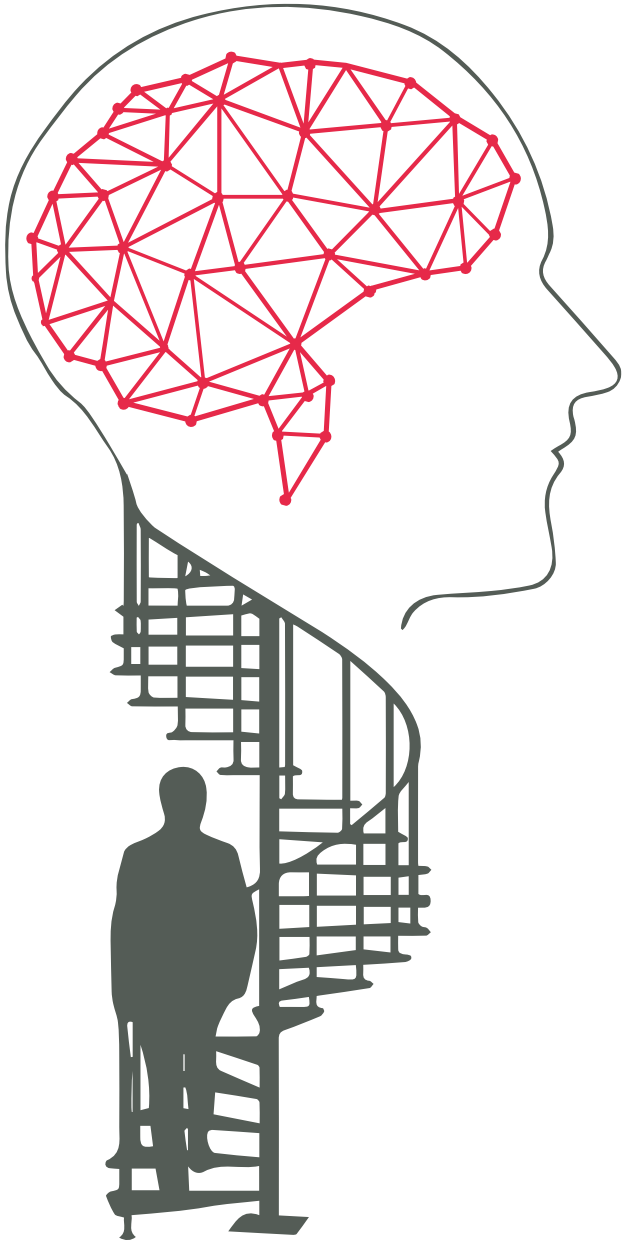
Quand utiliser un tableau ou un ArrayList ?

Tableau : si la taille est connue à l'avance et fixe (ex. jours de la semaine, 12 mois).

ArrayList : si la taille peut varier à l'exécution (ex. liste de clients, produits ajoutés par l'utilisateur).

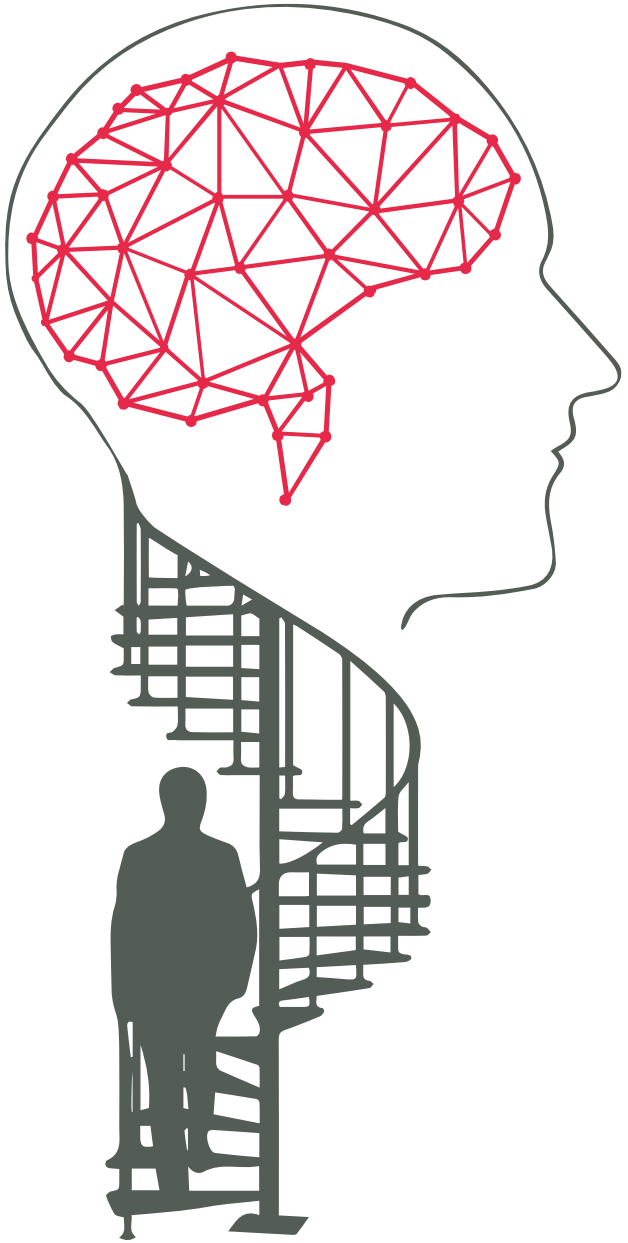
TP DE SYNTHESE

Gestion des étudiants



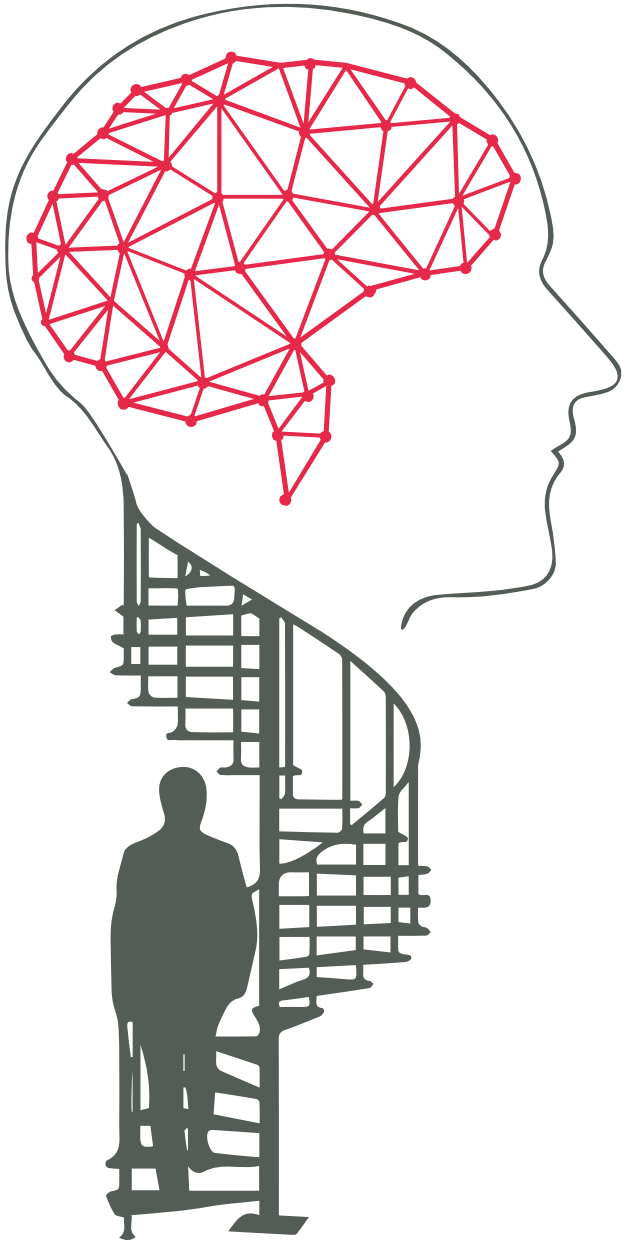
EXERCICE DE SYNTHESE

1. Créer une classe **Etudiant** avec nom, moyenne.
2. Créer un tableau de 5 étudiants.
3. Afficher les étudiants et leur moyenne.
4. Afficher le nom de l'étudiant ayant la meilleure moyenne.
5. Calculer la moyenne générale de la classe.



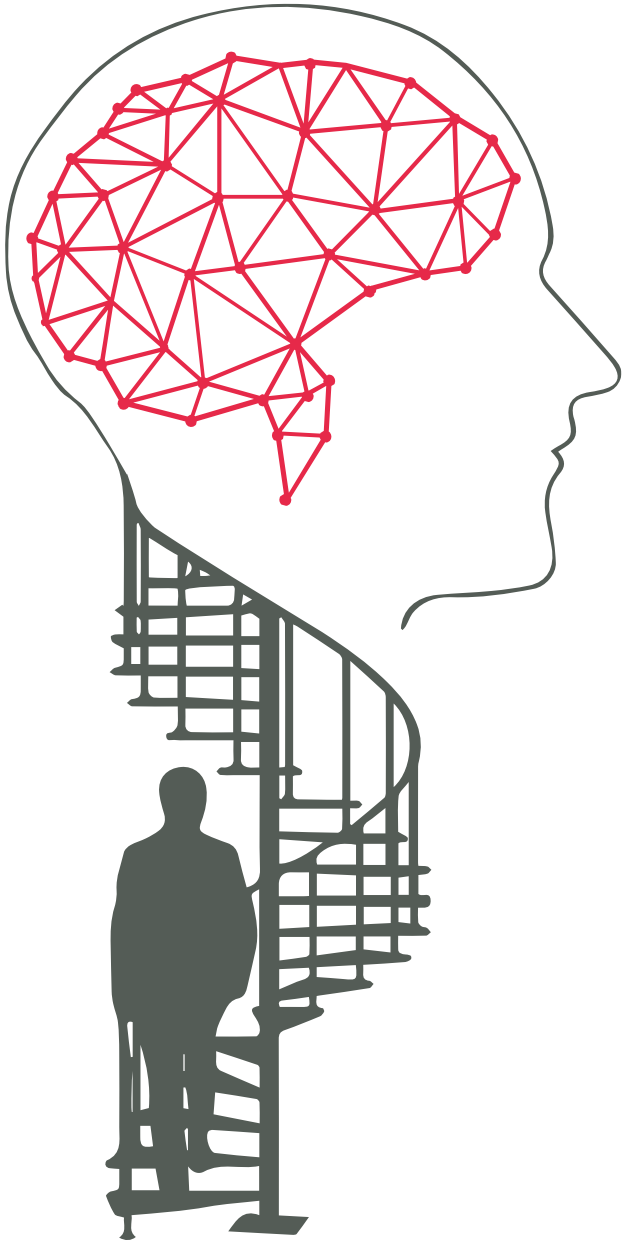
EXERCICE DE SYNTHESE

```
public class Etudiant {
    private String nom;      private double moyenne;
    public Etudiant(String nom, double moyenne) {
        this.nom = nom;      this.moyenne = moyenne;
    }
    public String getNom() {    return nom;   }
    public double getMoyenne() {   return moyenne; }
    public String toString() {
        return "Nom : " + nom + ", Moyenne : " + moyenne;
    }
}
```



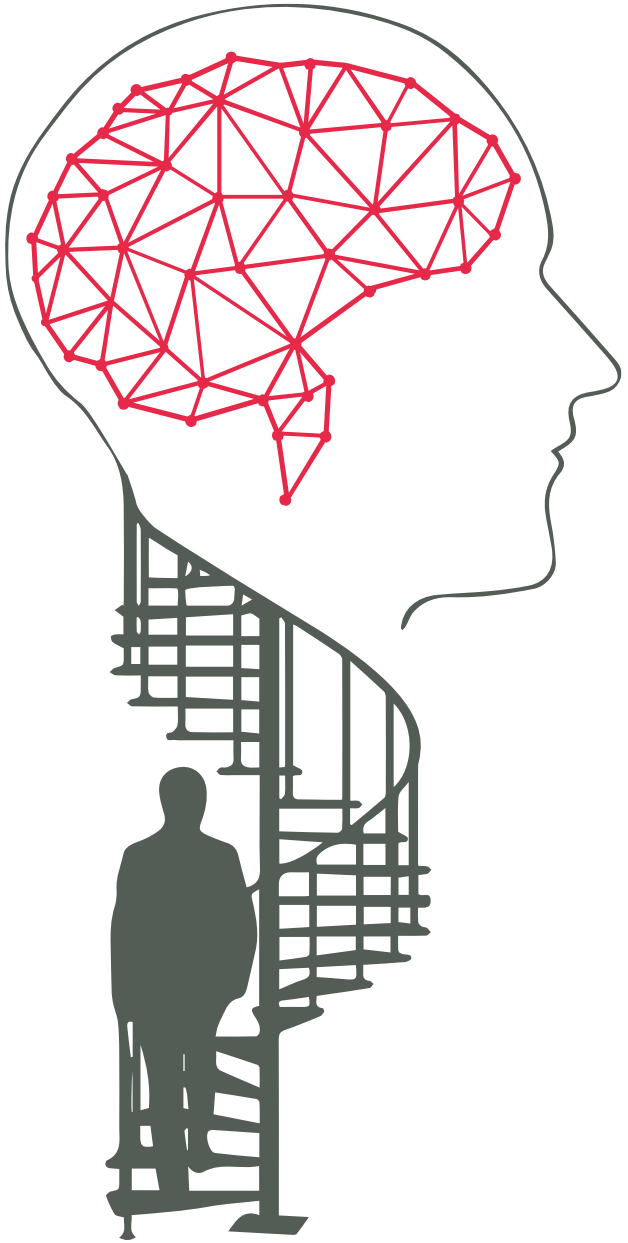
EXERCICE DE SYNTHESE

```
public class GestionClasse {
    public static void main(String[] args) {
        Etudiant[] classe = new Etudiant[5];
                classe[0] = new Etudiant("Ali", 14.5);
                classe[1] = new Etudiant("Fatima", 16.2);
                classe[2] = new Etudiant("Yassine", 12.0);
                classe[3] = new Etudiant("Imane", 17.8);
                classe[4] = new Etudiant("Karim", 15.0);
        System.out.println("Liste des étudiants :");
        for (int i = 0; i < classe.length; i++) {
                        System.out.println(classe[i]);    }
```



EXERCICE DE SYNTHESE

```
        double maxMoyenne = classe[0].getMoyenne();
        String meilleurNom = classe[0].getNom();
for (int i = 1; i < classe.length; i++) {
            if (classe[i].getMoyenne() > maxMoyenne) {
                maxMoyenne = classe[i].getMoyenne();
                meilleurNom = classe[i].getNom();
            }
        }
System.out.println("\nMeilleur étudiant : " + meilleurNom
                                    + " avec " + maxMoyenne);
```

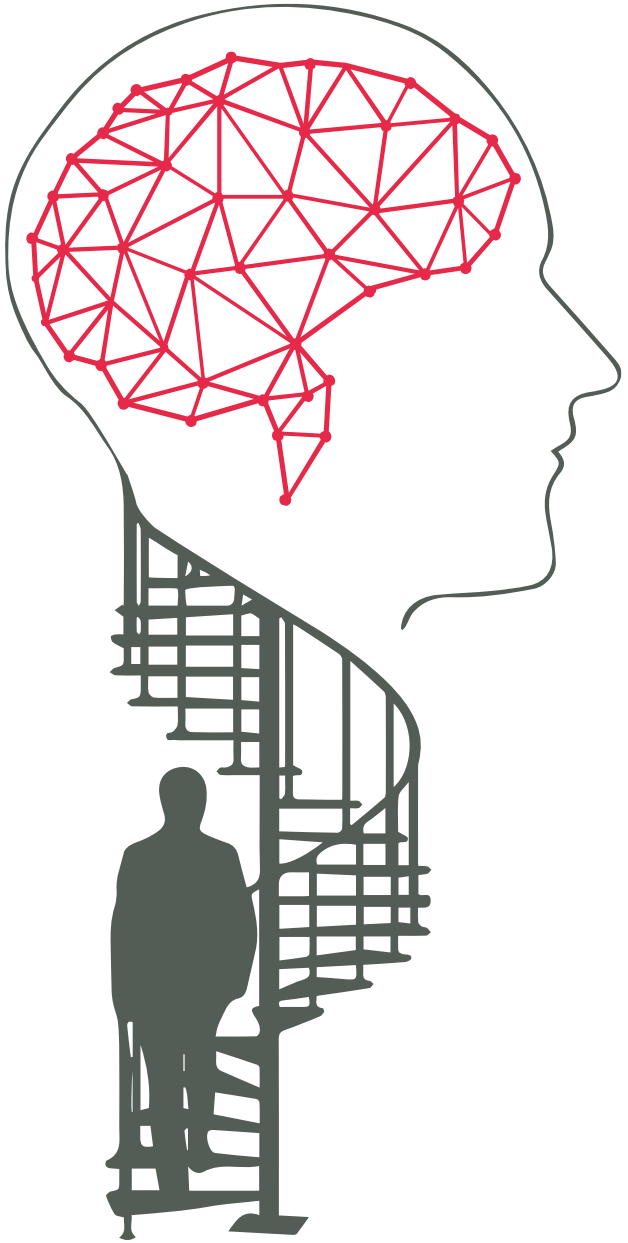


EXERCICE DE SYNTHESE

```
double somme = 0;
        for (int i = 0; i < classe.length; i++) {
            somme += classe[i].getMoyenne();
        }
double moyenneGenerale = somme / classe.length;
        System.out.println(
                                    "Moyenne générale de la classe : "
                                    + moyenneGenerale);
        }
}
```

Introduction à ArrayList

Les tableaux dynamiques



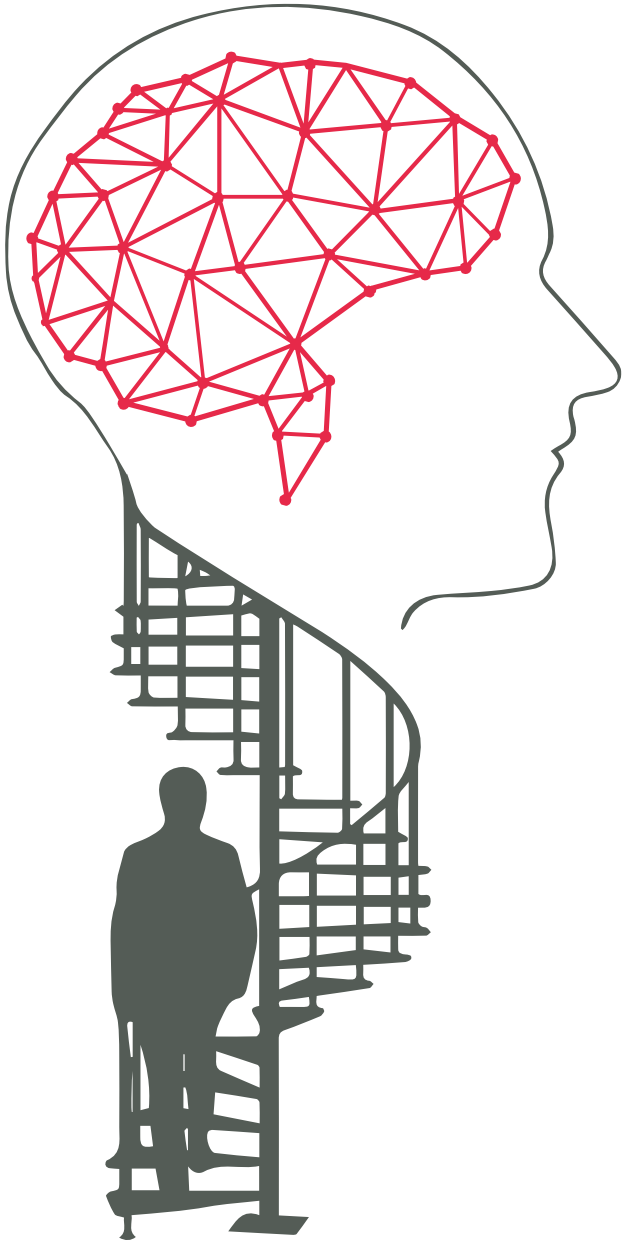
DEFINITION :

Une ArrayList est une classe fournie par Java dans le package java.util

Elle permet de stocker une collection d'objets dynamiquement, c'est-à-dire que sa taille peut changer à l'exécution (contrairement aux tableaux).

Caractéristiques :

- **Elle ne peut contenir que des objets (pas de types primitifs comme int, double).**
- **Elle conserve l'ordre d'insertion.**
- **Elle accepte les doublons.**
- **Elle permet l'accès direct par l'index.**

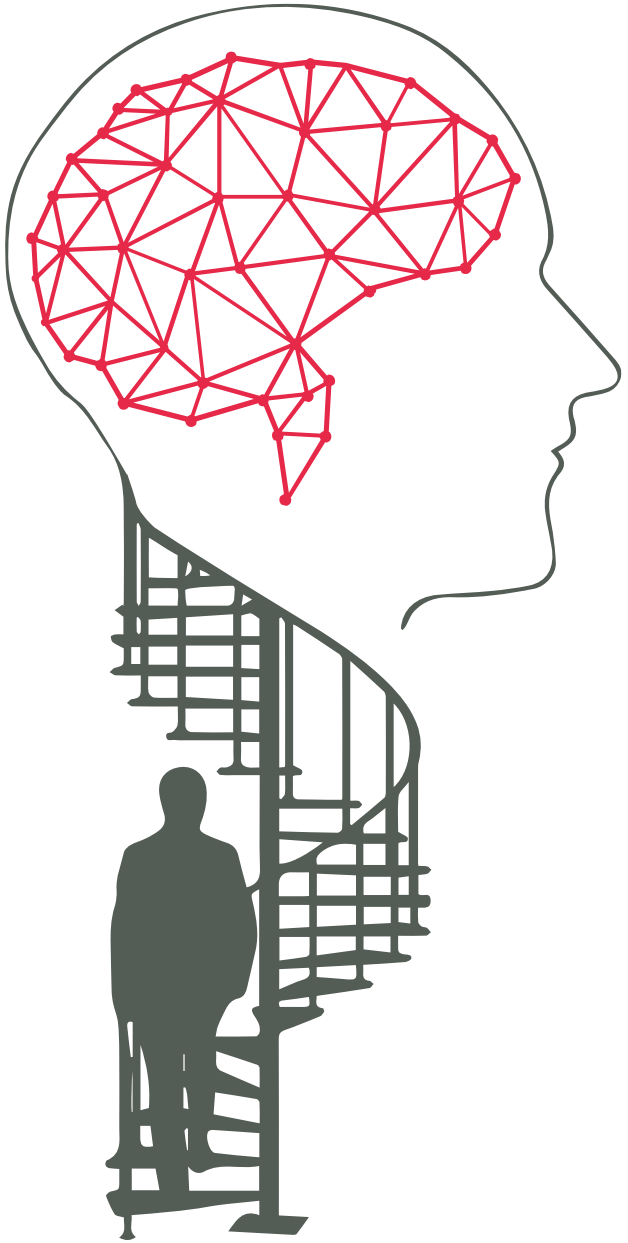


Déclaration et utilisation d'une ArrayList

```
import java.util.ArrayList;

ArrayList<Type> nomListe =
                    new ArrayList<>();
```

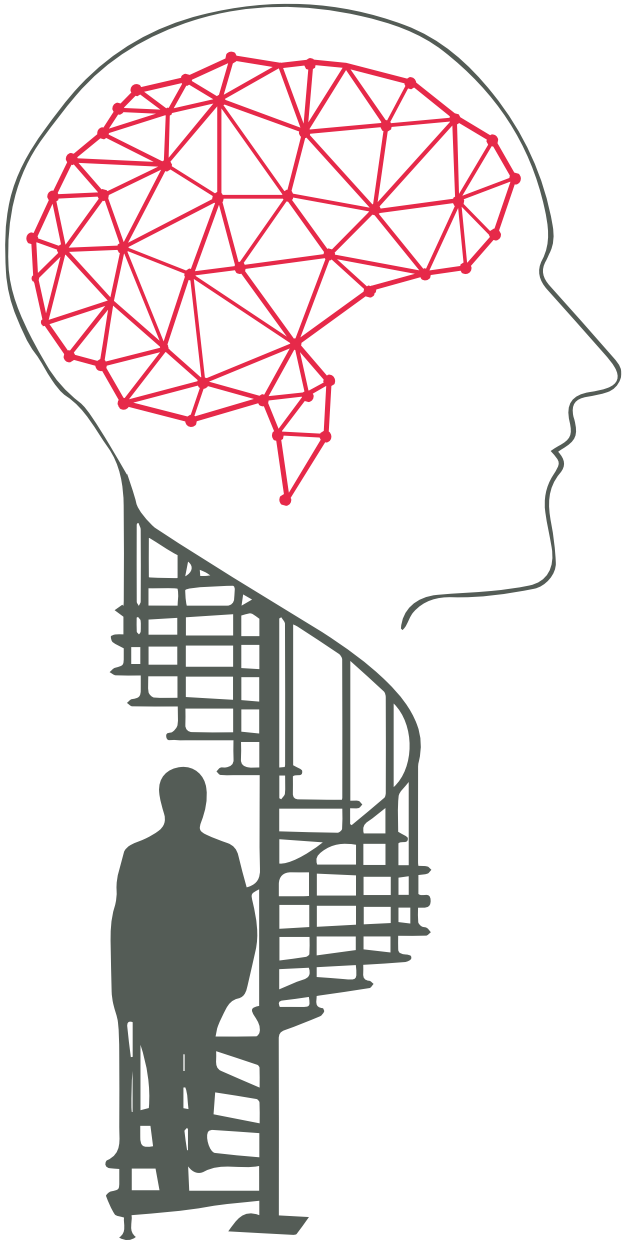
On utilise les génériques (<>) pour spécifier le type d'objet stocké.



Exemple d'utilisation d'une ArrayList

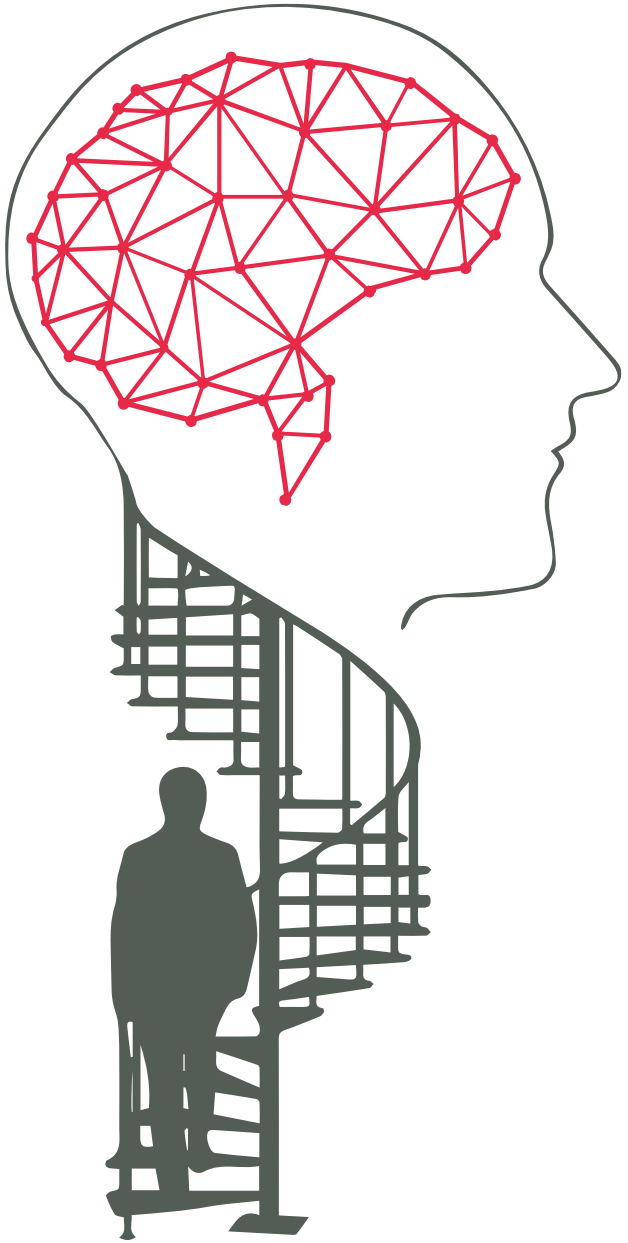
```
ArrayList<String>  noms =
                        new ArrayList<>();


noms.add("Ahmed");
noms.add("Fatima");
noms.add("Youssef");
```



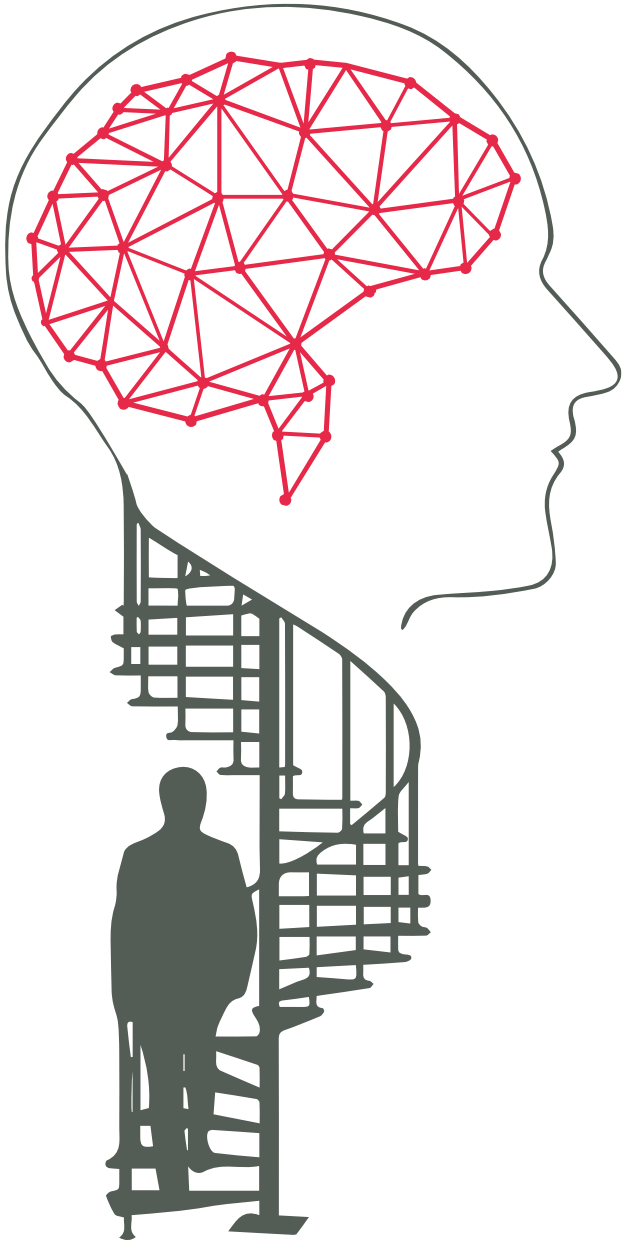
Méthodes principales de ArrayList

Méthode	Description
add(obj)	Ajoute un élément à la fin de la liste
add(index, obj)	Insère à une position spécifique
get(index)	Récupère un élément
set(index, obj)	Modifie un élément
remove(index)	Supprime un élément à une position
remove(obj)	Supprime le premier objet égal()
size()	Retourne la taille
contains(obj)	Vérifie si un objet est présent
clear()	Supprime tous les éléments
isEmpty()	Vérifie si la liste est vide



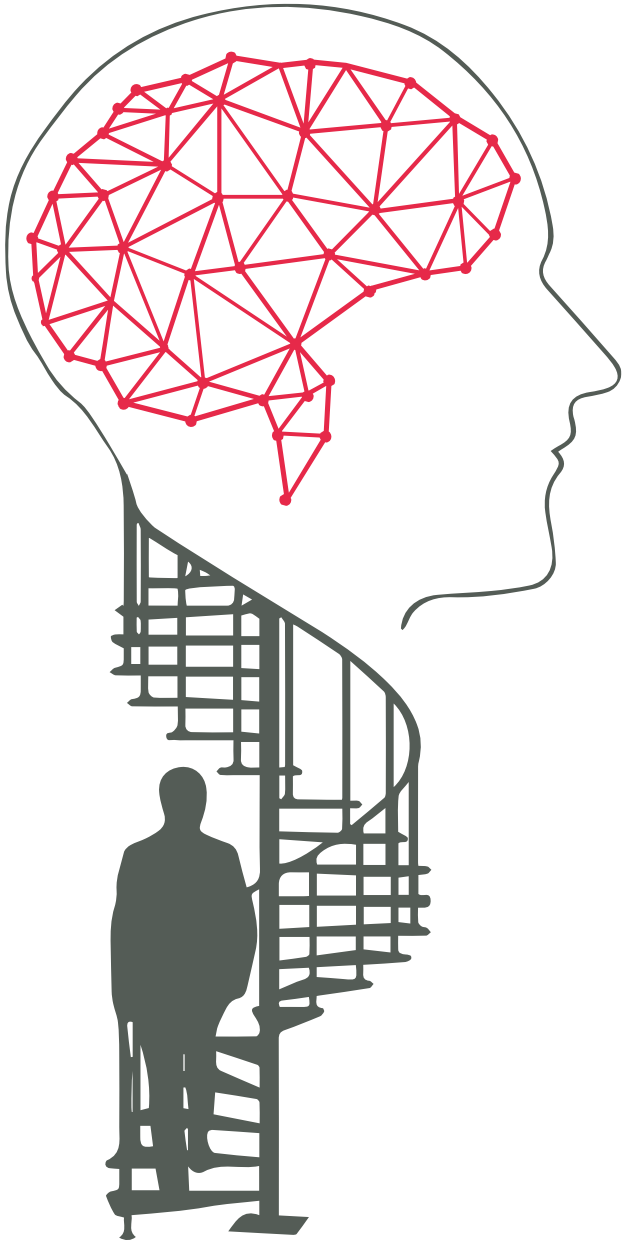
Exemple complet :

```
public class Etudiant {
    private String nom;      private int age;
    public Etudiant(String nom, int age) {
        this.nom = nom;     this.age = age;
    }
    public String getNom() {    return nom;   }
    public int getAge() {    return age;    }
    public String toString() {
        return    nom + " (" + age + " ans)";
    }
}
```



Exemple complet :

```
import java.util.ArrayList;
public class GestionEtudiants {
    public static void main(String[] args) {
        ArrayList<Etudiant> liste = new ArrayList<>();
        liste.add(new Etudiant("Ahmed", 20));
        liste.add(new Etudiant("Yasmine", 22));
        liste.add(new Etudiant("Karim", 21));
for (Etudiant e : liste) {
            System.out.println(e);
        }
```



Exemple complet :

```
for (Etudiant e : liste) {
        if (e.getNom().equals("Yasmine")) {
            System.out.println("Yasmine trouvée !");
        }
    }
     liste.remove(1);
    System.out.println("Après suppression :");
    for (Etudiant e : liste) {
        System.out.println(e);
    } } }
```