ChatGPT

Voici un résumé des parties mentionnées dans le cours avec des exemples simples :

---

2. Classes et Objets

Une classe est un modèle pour créer des objets. Elle regroupe des **attributs** (caractéristiques) et des **méthodes** (comportements).

Exemple :

```java
class Etudiant {
    String nom;
    String prenom;

    void afficher() {
        System.out.println("Nom : " + nom + ", Prénom : " + prenom);
    }
}

public class Main {
    public static void main(String[] args) {
        Etudiant et = new Etudiant();
        et.nom = "Ali";
        et.prenom = "Mohamed";
        et.afficher();
    }
}
```

---

3. Constructeurs

Un constructeur initialise un objet lorsqu'il est créé. Il porte le même nom que la classe et n’a pas de type de retour.

Exemple :

```java
class Etudiant {
    String nom;
    String prenom;

    // Constructeur
    Etudiant(String nom, String prenom) {
        this.nom = nom;
        this.prenom = prenom;
    }

    void afficher() {
        System.out.println("Nom : " + nom + ", Prénom : " + prenom);
    }
}

public class Main {
    public static void main(String[] args) {
        Etudiant et = new Etudiant("Ali", "Mohamed");
        et.afficher();
    }
}
```

---

4. Héritage

L'héritage permet de créer une classe (sous-classe) à partir d'une autre classe (super-classe). La sous-classe hérite des attributs et des méthodes de la super-classe.

Exemple :

```java
class Personne {
    String nom;
    String prenom;

    void afficher() {
```

```
        System.out.println("Nom : " + nom + ", Prénom : " + prenom);
    }
}

class Etudiant extends Personne {
    String cne;

    void afficher() {
        super.afficher(); // Appelle la méthode de la super-classe
        System.out.println("CNE : " + cne);
    }
}

public class Main {
    public static void main(String[] args) {
        Etudiant et = new Etudiant();
        et.nom = "Ali";
        et.prenom = "Mohamed";
        et.cne = "A12345";
        et.afficher();
    }
}
```

---

5. Polymorphisme

Le polymorphisme permet à une méthode d'avoir plusieurs formes. Il est souvent utilisé avec l'héritage pour redéfinir ou surcharger des méthodes.

Exemple de redéfinition :

java

```
class Animal {
    void parler() {
        System.out.println("L'animal fait un bruit.");
    }
}

class Chien extends Animal {
    @Override
```

```
    void parler() {
        System.out.println("Le chien aboie.");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal animal = new Chien(); // Liaison dynamique
        animal.parler(); // "Le chien aboie."
    }
}
```

---

6. Tableaux et Collections

- Les **tableaux** sont des structures de données fixes. Exemple :

```java
int[] nombres = {1, 2, 3};
for (int nombre : nombres) {
    System.out.println(nombre);
}
```

- Les **collections** (comme `ArrayList`) sont dynamiques et plus flexibles :

```java
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> noms = new ArrayList<>();
        noms.add("Ali");
        noms.add("Mohamed");

        for (String nom : noms) {
            System.out.println(nom);
        }
```

```
    }
}
```

Ces concepts sont fondamentaux pour la programmation orientée objet et la gestion efficace des données dans Java.