Chapitre 5: Polymorphisme

1. Définition :

Le polymorphisme veut dire que le même service, aussi appelé opération ou méthode, peut avoir un comportement différent selon les situations.

2. Polymorphisme paramétrable :

a. Plusieurs signatures pour une même méthode (ad hoc)

On peut donner à une même méthode, plusieurs signatures pour implémenter des comportements différents selon les types des paramètres passés. La signature d'une méthode est composée du nom de celle ci, de sa portée, du type de donnée qu'elle renvoie et enfin du nombre et du type de ses paramètres.

```
public class A {

  private int a;

  public A() { //constructeur 1
    System.out.println("Création de A");
  }

  public A(int a) { //constructeur 2 par surcharge
    this.a = a;
    System.out.println("Création de A");
  }

  public int getter() {
    return this.a;
  }

  public void setter(int a) {
    this.a = a;
  }

  public static void main(String[] args) {
    A premierA = new A(); //construction par 1
    A secondA = new A(1); //construction par 2
  }

}
```

b.

Proposer le passage d'un nombre inconnu de paramètres

Dans la signature d'une méthode , on peut préciser qu'il est possible de passer plus de 1 paramètre du même type en suffixant le type du paramètre avec « ... ».

```
// supposons la méthode suivante :
public String concatenation(String... elements) {
    // pour l'implementation, il faut considérer le paramètre comme un
tableau
    String resultat = "";
    for (String element : elements) {
        resultat += element;
    }
    return resultat;
}

// elle peut être appelée ainsi
concatenation("abc", "de", "f"); // renvoie "abcdef"
```

3. Polymorphisme d'héritage

En redéfinissant une méthode dans une sous-classe, on peut spécialiser le comportement d'une méthode.

```
public class B extends A {

  private int b;

  public B() {
    super();
    System.out.println("Création de B");
  }

  public B(int a, int b){
    super(a);
    this.b = b;
    System.out.println("Création de B");
  }

  public int getter() {
    return this.b;
  }

  public void setter(int a, int b) {
```

```
    super.setter(a);
    this.b=b;
  }

  public static void main(String[] args) {
      A[] table = new A[2];
      table[0] = new A(10);
      table[1] = new B(20,30);

     for(A a : table){
        System.out.println("* " + a.getter());
      }

    /* les résultats sur console:
    Création de A
    Création de A
    Création de B
    * 10
    * 30
    */
  }

}
```