Paradigmes de programation 3/ Programmation Impérative vs Déclarative

Programmation Impérative vs Déclarative

---

1. Programmation Impérative

La programmation impérative consiste à **décrire chaque étape** de la manière dont un problème doit être résolu. On donne des instructions explicites à l'ordinateur sur la manière d'exécuter les opérations, en contrôlant le flux d'exécution via des boucles, des conditions, etc.

Exemple en Java :

```java
import java.util.ArrayList;
import java.util.List;

public class ImperativeExample {
    public static void main(String[] args) {
        List<Integer> numbers = List.of(1, 2, 3, 4, 5);
        List<Integer> evenNumbers = new ArrayList<>();

        // Boucle impérative pour sélectionner les nombres pairs
        for (int number : numbers) {
            if (number % 2 == 0) {
                evenNumbers.add(number); // Ajoute le nombre pair à la liste
            }
        }

        System.out.println(evenNumbers); // Résultat = [2, 4]
    }
}
```

Concept : Le code ici suit une approche impérative, où on décrit explicitement comment boucler sur les nombres et sélectionner les pairs.

Exemple en JavaScript :

```javascript
const numbers = [1, 2, 3, 4, 5];
let evenNumbers = [];

// Boucle impérative pour sélectionner les nombres pairs
for (let i = 0; i < numbers.length; i++) {
    if (numbers[i] % 2 === 0) {
        evenNumbers.push(numbers[i]); // Ajoute le nombre pair au tableau
    }
}

console.log(evenNumbers); // Résultat = [2, 4]
```

Concept : Ici, on utilise une boucle `for` pour parcourir le tableau et vérifier chaque élément. Cette approche est impérative car on spécifie étape par étape ce qu'il faut faire.

---

2. **Programmation Déclarative**

La programmation déclarative consiste à **décrire le résultat souhaité** sans préciser exactement comment y arriver. On exprime **le "quoi"** (ce qu'on veut obtenir) plutôt que **le "comment"** (les étapes pour l'obtenir).

Exemple en Java :

```java
import java.util.List;
import java.util.stream.Collectors;

public class DeclarativeExample {
    public static void main(String[] args) {
        List<Integer> numbers = List.of(1, 2, 3, 4, 5);

        // Utilisation des streams pour filtrer les nombres pairs (approche
déclarative)
        List<Integer> evenNumbers = numbers.stream()
                                           .filter(n -> n % 2 == 0) // Filtre pour
les pairs
                                           .collect(Collectors.toList());

        System.out.println(evenNumbers); // Résultat = [2, 4]
    }
}
```

Concept : Ici, l'approche déclarative est utilisée avec un **stream** pour filtrer les nombres pairs, sans avoir à gérer explicitement la boucle ou les conditions.

Exemple en JavaScript :

```javascript
const numbers = [1, 2, 3, 4, 5];

// Utilisation de la méthode filter pour sélectionner les nombres pairs (approche
déclarative)
const evenNumbers = numbers.filter(number => number % 2 === 0);

console.log(evenNumbers); // Résultat = [2, 4]
```

Concept : Ici, la méthode `filter` est utilisée pour filtrer les nombres pairs. C'est une approche déclarative car on exprime uniquement ce que l'on veut sans préciser comment faire la boucle.

---

3. **Différences Clés entre Impératif et Déclaratif**

- **Impératif** : Vous **décrivez comment** accomplir la tâche étape par étape, en contrôlant explicitement le flux d'exécution. C'est plus **procédural** et exige une gestion fine du contrôle de la machine.
  - Exemple : Une boucle `for` qui parcourt un tableau et ajoute des éléments un par un.
- **Déclaratif** : Vous **décrivez ce que** vous voulez obtenir, et le système détermine la manière d'exécuter la tâche. Il se focalise sur le **résultat** final sans gérer les détails d'exécution.
  - Exemple : Utiliser `filter` ou des **streams** pour sélectionner directement les éléments souhaités.

Comparaison avec Commentaires :

Impératif (JavaScript) :

```
const numbers = [1, 2, 3, 4, 5];
let evenNumbers = [];

// Approche impérative : boucle explicitement sur le tableau
for (let i = 0; i < numbers.length; i++) {
    if (numbers[i] % 2 === 0) {
        evenNumbers.push(numbers[i]); // Ajoute l'élément si pair
    }
}

console.log(evenNumbers); // Résultat : [2, 4]
```

Commentaire : Ici, on indique clairement chaque étape pour obtenir les nombres pairs. On contrôle le **"comment"** (boucle, condition, ajout).

Déclaratif (JavaScript) :

```
const numbers = [1, 2, 3, 4, 5];

// Approche déclarative : exprime seulement le résultat souhaité
const evenNumbers = numbers.filter(number => number % 2 === 0);

console.log(evenNumbers); // Résultat : [2, 4]
```

Commentaire : En utilisant `filter`, on déclare simplement qu'on veut les nombres pairs, sans se soucier de comment parcourir le tableau.

Conclusion :

La programmation impérative est utile lorsque vous devez contrôler finement chaque étape du processus, tandis que la programmation déclarative simplifie le code en se concentrant sur le résultat. Dans de nombreux langages modernes, comme Java et JavaScript, on encourage souvent une approche plus déclarative pour sa concision et sa lisibilité.
