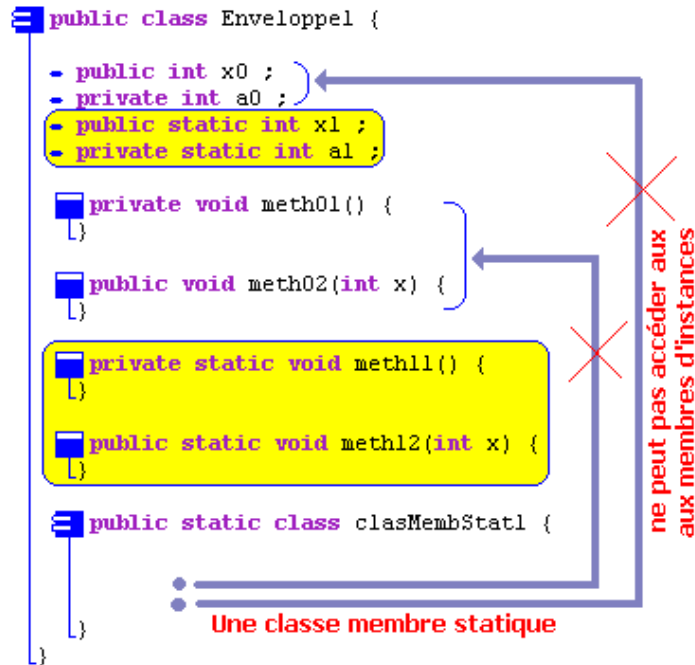


Java Statique et Objet



En Java, les concepts de membres statiques et d'objets sont fondamentaux pour comprendre la gestion de la mémoire et l'organisation du code. Voici une explication détaillée de ces concepts.

Membres statiques (static)

Les membres statiques

Les membres statiques (attributs ou méthodes)

Un attribut statique

- Un attribut membre statique c'est une variable de la classe
- La valeur de la variable statique est indépendante de l'objet
 - Quelque soit le nombre d'objet créés, cette variable a une seule valeur
 - Cette variable peut avoir une valeur même si aucun objet n'existe

Définition

Les membres statiques sont associés à la classe elle-même plutôt qu'aux instances (objets) de la classe. Cela signifie qu'ils sont partagés entre toutes les instances de cette classe.

Caractéristiques

1. **Variables statiques:** Elles sont déclarées avec le mot-clé `static`. Une variable statique appartient à la classe et non aux instances de la classe.

```
class MyClass {  
    static int staticVariable = 0;  
}
```

Toutes les instances de `MyClass` partageront la même copie de `staticVariable`.

2. **Méthodes statiques:** Elles sont également déclarées avec le mot-clé `static`. Une méthode statique peut être appelée sans créer une instance de la classe.

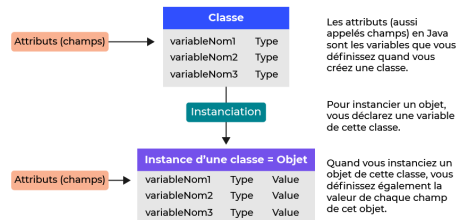
```
class MyClass {  
    static void staticMethod() {  
        System.out.println("This is a static method.");  
    }  
}
```

Vous pouvez appeler cette méthode avec `MyClass.staticMethod()`.

3. **Blocs statiques:** Un bloc de code statique s'exécute une seule fois lors du chargement de la classe en mémoire.

```
class MyClass {  
    static {  
        System.out.println("Static block executed.");  
    }  
}
```

Objets



Définition

Un objet est une instance d'une classe. Chaque objet possède ses propres valeurs pour les variables d'instance définies dans la classe.

Caractéristiques

1. **Variables d'instance:** Elles sont propres à chaque instance d'une classe. Chaque objet a sa propre copie de ces variables.

```
class MyClass {  
    int instanceVariable = 0;  
}
```

```
MyClass obj1 = new MyClass();  
MyClass obj2 = new MyClass();
```

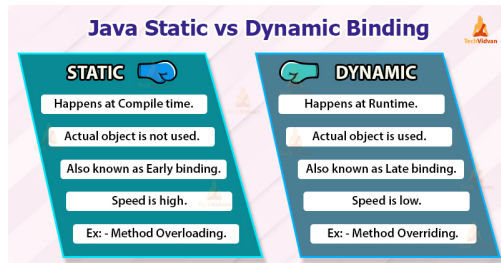
```
obj1.instanceVariable = 1;  
obj2.instanceVariable = 2;
```

2. **Méthodes d'instance:** Elles opèrent sur les variables d'instance et peuvent accéder à `this`, qui est une référence à l'objet courant.

```
class MyClass {  
    int instanceVariable = 0;  
  
    void instanceMethod() {  
        System.out.println("Instance variable: " + this.instanceVariable);  
    }  
}
```

```
MyClass obj = new MyClass();
obj.instanceMethod();
```

Comparaison et interaction



- Les **méthodes statiques** ne peuvent pas accéder directement aux variables ou méthodes d'instance car elles n'ont pas de référence à un objet spécifique (`this`).

```
class MyClass {
    int instanceVariable = 0;
    static int staticVariable = 0;

    static void staticMethod() {
        // Erreur : ne peut pas accéder à instanceVariable
        // System.out.println(instanceVariable);

        // Correct : peut accéder à staticVariable
        System.out.println(staticVariable);
    }
}
```

- Les **méthodes d'instance** peuvent accéder aux variables et méthodes statiques.

```
class MyClass {
    int instanceVariable = 0;
    static int staticVariable = 0;

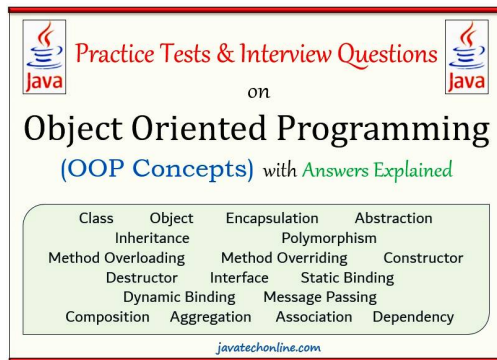
    void instanceMethod() {
        // Correct : peut accéder à staticVariable
        System.out.println(staticVariable);
    }
}
```

```

        // Correct : peut accéder à instanceVariable
        System.out.println(instanceVariable);
    }
}

```

Exemple complet



Voici un exemple illustrant l'utilisation des membres statiques et d'instance.

```

class Example {
    static int staticCounter = 0;
    int instanceCounter = 0;

    Example() {
        staticCounter++;
        instanceCounter++;
    }

    static void printStaticCounter() {
        System.out.println("Static counter: " + staticCounter);
    }

    void printInstanceCounter() {
        System.out.println("Instance counter: " + instanceCounter);
    }

    public static void main(String[] args) {
        Example obj1 = new Example();
        Example obj2 = new Example();
    }
}

```

```
        obj1.printInstanceCounter(); // Instance counter: 1
        obj2.printInstanceCounter(); // Instance counter: 1
        Example.printStaticCounter(); // Static counter: 2
    }
}
```

Conclusion

Dans cet exemple, `staticCounter` est partagé entre toutes les instances de `Example`, tandis que `instanceCounter` est spécifique à chaque instance.

Cet exemple résume bien la différence fondamentale entre les membres statiques et leurs instances de classe qui font la spécificité de la Programmation Orientée Objet (POO).