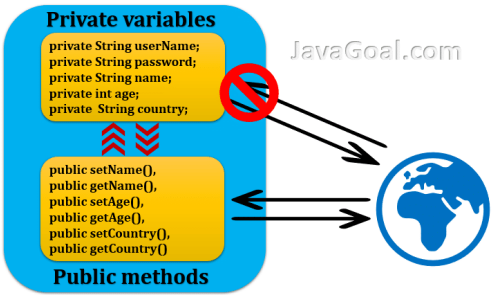
L'encapsulation dans Java



L'encapsulation est un concept fondamental de la programmation orientée objet (OOP) en Java. Elle consiste à regrouper les données (variables) et les méthodes (fonctions) qui manipulent ces données en une seule unité appelée classe. L'encapsulation permet de contrôler l'accès aux données et de protéger l'intégrité des objets.

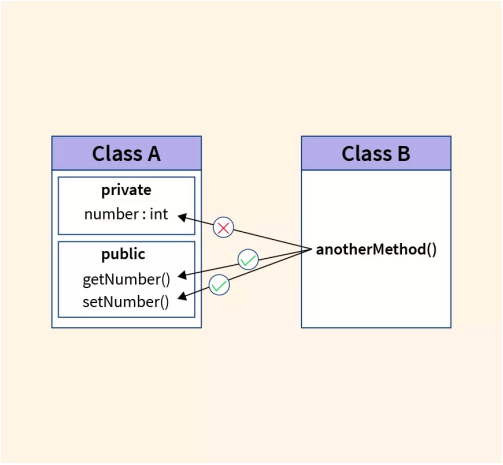
Voici les points clés de l'encapsulation en Java :

1. **Attributs privés** : Les variables d'une classe sont déclarées comme privées pour empêcher l'accès direct depuis l'extérieur de la classe. Cela garantit que les données ne peuvent être modifiées directement par des méthodes ou des classes non autorisées.

2. **Méthodes publiques** : Des méthodes publiques, souvent appelées getters et setters, sont fournies pour permettre l'accès et la modification des variables privées de manière contrôlée. Ces méthodes définissent comment les données peuvent être lues ou modifiées, permettant d'ajouter des validations ou d'autres logiques de contrôle.

3. **Encapsulation et abstraction** : L'encapsulation aide à implémenter l'abstraction en cachant les détails internes de l'objet et en ne présentant que les aspects essentiels de l'objet à l'extérieur.

La solution d'encapsulation avec les méthodes Getter/Setter :




```java
public class Person {
    // Variables privées
    private String name;
    private int age;

    // Constructeur
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Getter pour name
    public String getName() {
        return name;
    }

    // Setter pour name
    public void setName(String name) {
        this.name = name;
    }

    // Getter pour age
    public int getAge() {
        return age;
```

```
    }

    // Setter pour age
    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        } else {
            System.out.println("L'âge doit être positif.");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        // Création d'un objet Person
        Person person = new Person("Alice", 30);

        // Accès aux variables via les getters
        System.out.println("Name: " + person.getName());
        System.out.println("Age: " + person.getAge());

        // Modification des variables via les setters
        person.setName("Bob");
        person.setAge(35);

        // Affichage des nouvelles valeurs
        System.out.println("Updated Name: " + person.getName());
        System.out.println("Updated Age: " + person.getAge());

        // Tentative de définir un âge invalide
        person.setAge(-5);  // Affichera un message d'erreur
    }
}
```

Dans cet exemple :

- Les variables `name` et `age` sont privées et ne peuvent pas être accédées directement de l'extérieur de la classe `Person`.

- Les méthodes `getName`, `setName`, `getAge` et `setAge` sont publiques et

fournissent un moyen contrôlé d'accéder et de modifier les variables privées.

- Le setter `setAge` inclut une validation pour s'assurer que l'âge ne peut pas être défini sur une valeur négative.

Conclusion

L'encapsulation améliore la maintenabilité et la flexibilité du code en isolant les données et en contrôlant les interactions avec ces données.