Passer à la POO Avancée avec Les Princip SOLID

Ibrahim Ahmad,
FullStack Developer

Comprendre le Princip SOLID

SOLID est un acronyme qui représente les cinq principes de base de la programmation orientée objet. Ces principes ont été développés pour aider les développeurs à écrire un code plus propre, maintenable et évolutif. Les principes SOLID peuvent être appliqués à tous les langages de programmation orientée objet, y compris C#.

L'acronyme SOLID

L'acronyme SOLID représente les cinq principes de base de la programmation orientée objet :

- **[S]** RP : **S**ingle Responsibility Principle (Principe de responsabilité unique)
- **[O]** CP : **O**pen-Closed Principle (Principe ouvert-fermé)
- **[L]** SP : **L**iskov Substitution Principle (Principe de substitution de Liskov)
- **[I]**SP : **I**nterface Segregation Principle (Principe de ségrégation d'interface)
- **[D]** IP : **D**ependency Inversion Principle (Principe d'inversion de dépendance)

- **[S]** RP : **S**ingle Responsibility Principle (Principe de responsabilité unique)

Le premier principe est le principe de responsabilité unique (Single Responsibility Principle ou SRP), qui stipule qu'une classe ne doit avoir qu'une seule raison de changer.

Cela signifie qu'une classe ne doit avoir qu'une seule responsabilité ou fonctionnalité.

Si une classe a plusieurs responsabilités, cela rend le code plus difficile à comprendre, à maintenir et à étendre.

- **[O] CP : Open-Closed Principle (Principe ouvert-fermé)**

Le deuxième principe est le principe ouvert-fermé (Open-Closed Principle ou OCP), qui stipule qu'une classe doit être ouverte à l'extension mais fermée à la modification.

Cela signifie qu'une classe doit être facilement étendue pour ajouter de nouvelles fonctionnalités, mais sans modifier le code existant.

Cela permet d'éviter les répercussions imprévues sur le reste de l'application.

- **[L] SP : Liskov Substitution Principle (Principe de substitution de Liskov)**

Le troisième principe est le principe de substitution de Liskov (Liskov Substitution Principle ou LSP), qui stipule qu'un objet de type dérivé doit être substituable par un objet de type de base sans altérer le comportement du programme.

Cela signifie que les classes dérivées doivent respecter le contrat de la classe de base et ne pas introduire de nouveaux comportements qui pourraient entraîner des erreurs.

- [I]SP : Interface Segregation Principle (Principe de ségrégation d'interface)

Le quatrième principe est le principe de ségrégation d'interface (Interface Segregation Principle ou ISP), qui stipule qu'une classe ne doit pas être obligée de mettre en œuvre des méthodes dont elle n'a pas besoin.

Cela signifie que les interfaces doivent être définies de manière à ce que les classes n'aient à implémenter que les méthodes nécessaires à leur fonctionnement.

- ## [D] IP : Dependency Inversion Principle (Principe d'inversion de dépendance)

Le cinquième et dernier principe est le principe d'inversion de dépendance (Dependency Inversion Principle ou DIP), qui stipule que les modules de haut niveau ne doivent pas dépendre des modules de bas niveau, mais plutôt des abstractions.

Cela signifie que les dépendances doivent être inversées pour permettre une plus grande flexibilité et une meilleure extensibilité.

Projet Concret .Net Console

Projet Git Appliquant les Princip SOLID :

Telecharger ici :
https://github.com/ibrahmad18/Employee-Management-System.git