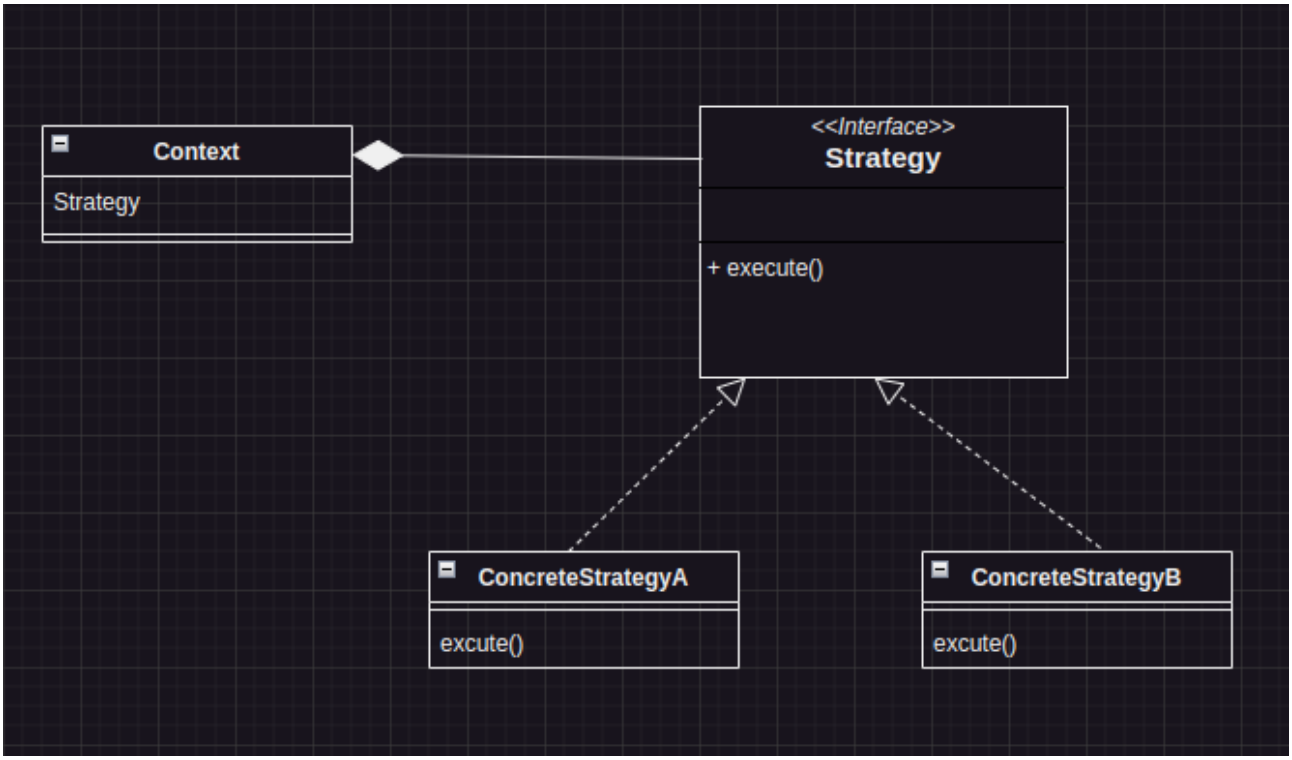
Le patron Strategy

Le pattern Strategy: Description

- Propose de définir un ensemble d'implémentations(des classes) d'un concept spécifique(interface).
- Les implémentations sont exécutées en fonction de l'appelant.
- Chaque implémentation représente une façon spécifique de résoudre le problème de l'appelant.
- C'est un pattern qui permet d'adapter la façon de faire en fonction de la situation
- Donc plus de flexibilité

Le pattern Strategy: Structure



Le pattern Strategy: Participants

- **Context** : c'est la classe qui définit l'objet dont le comportement doit être modifié dynamiquement.

- **Strategy** : l'interface ou classe abstraite qui définit les méthodes communes à tous les algorithmes pouvant être utilisés par l'objet Context.

- **ConcretStrategyA** et **ConcretStrategyB** sont les classes qui implémentent l'interface Strategy et fournit l'implémentation réelle de l'algorithme.A l'exécution, un objet Context utilise un objet ConcretStrategy sélectionné lors de sa création pour effectuer ses opérations.

Le pattern Strategy: Exemple

On souhaite réaliser une application de sauvegarde d'images

des utilisateurs. Pour cela, l'application doit :

• Compresser l'image en utilisant un algorithme de

compression selon le format de l'image(JPEG, PNG, GIF, …)

• Appliquer différents filtres suivant un large choix de filtres