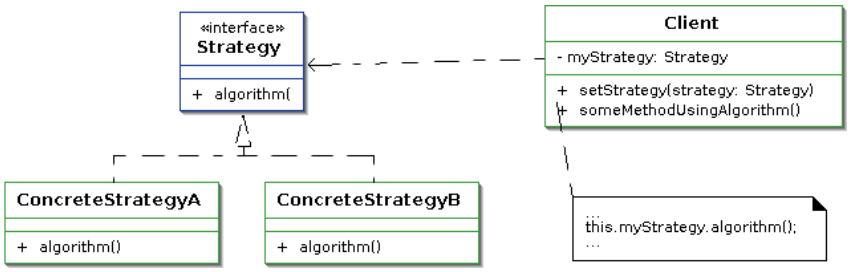
Design Pattern : strategy

Intent

Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

Structure



Eléments caractéristiques

- un attribut représentant la stratégie qui est d'un type abstrait
- l'invocation des méthodes de la stratégie, en faisant varier le type concret de la stratégie on change le comportement apparent de la classe cliente.

On remplace la variation du comportement que l'on peut obtenir par héritage par une variation du comportement par composition via la stratégie.

Exemples rencontrés

API Java : Layout Manager

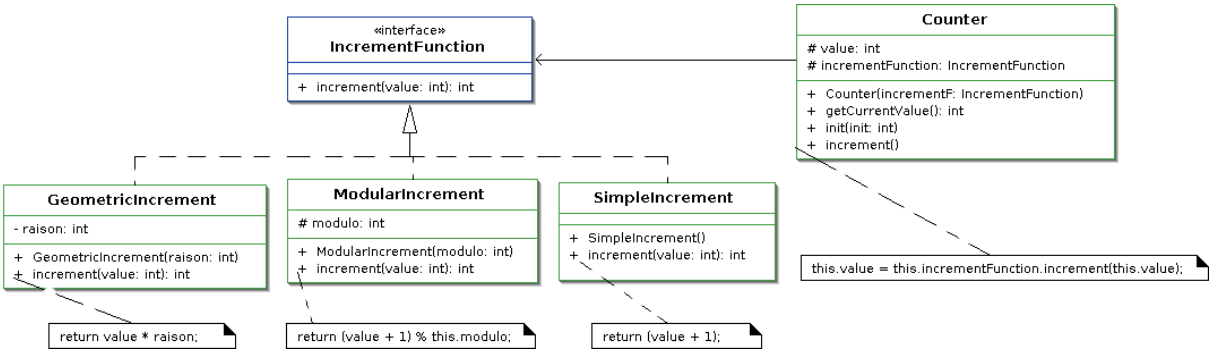
Dans les api awt/swing, un objet `Container` gère la disposition des différents objets graphiques qu'il contient grâce à un `LayoutManager`

- `public interface LayoutManager` Defines the interface for classes that know how to lay out Containers.

  Et les classes : `java.awt.FlowLayout`, `java.awt.GridLayout`, `java.awt.BorderLayout`, ...
- Dans `java.awt.Container` : `public void setLayout(LayoutManager mgr)` Sets the layout manager for this container.

Compteurs et fonctions d'incrémentation

Les différents types de compteurs sont obtenus en faisant varier la stratégie (fonction) d'incrémentation de la valeur du compteur..



Pierre-feuille-ciseaux : joueurs et stratégie

Une seule classe de joueur et plusieurs classes de stratégie. Lorsque le joueur joue c'est en fait la stratégie qui fait le choix du coup à jouer. En changeant la stratégie d'un joueur à l'autre on obtient des joueurs qui jouent différemment