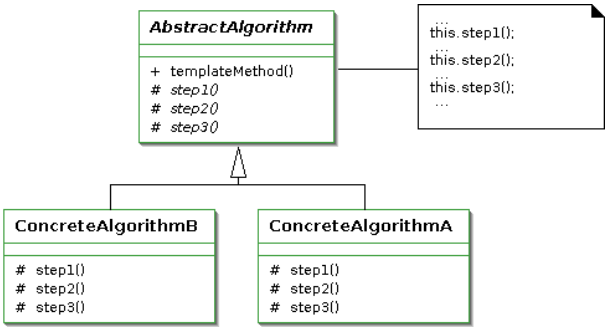
Design Pattern : template method

Intent

> *Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.*

Structure

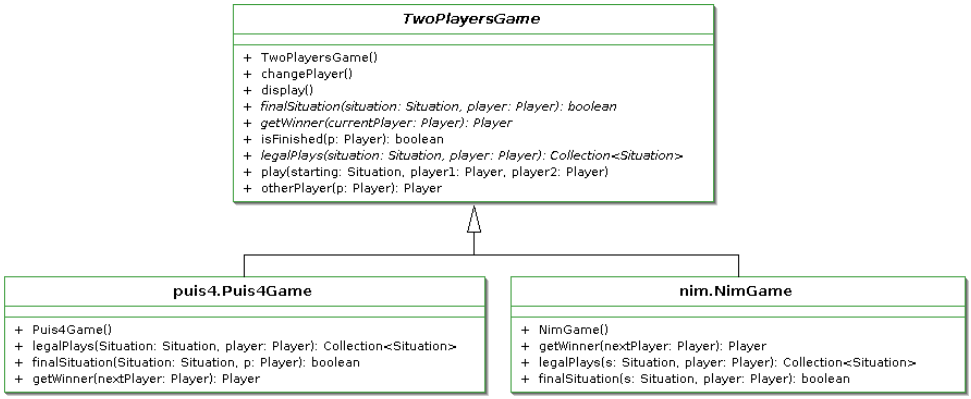


Eléments caractéristiques

- une méthode décrivant un algorithme qui fait appel à des méthodes abstraites de la classe, ces méthodes sont définies dans les sous-classes, permettant à ces sous-classes de faire varier le comportement de l'algorithme.

Exemples rencontrés

Cours : exemple de conception, les jeux à 2 joueurs.



La méthode `play()` s'appuie sur les méthodes abstraites `isFinalSituation()`, `legalPlays()` et `getWinner()` qui sont définies dans les sous-classes, et elles ne pourraient être définies ailleurs car elles dépendent de chaque jeu en particulier.

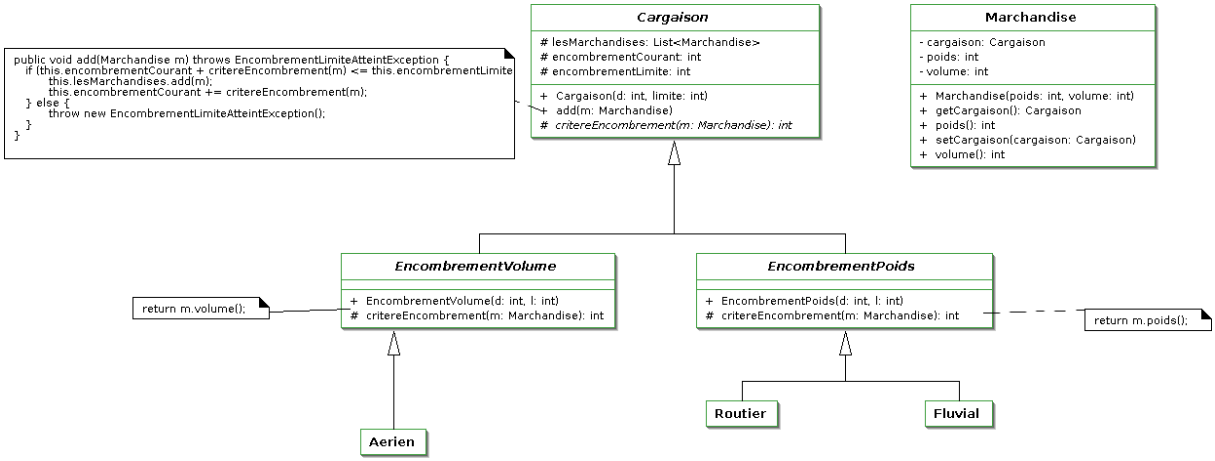
```
public void play(Situation starting, Player player1, Player player2) {
   this.setCurrentSituation(starting);
   this.player1 = player1;
   this.player2 = player2;
   this.currentPlayer = player1;

   while (!this.isFinalSituation(this.currentSituation,this.currentPlayer)) {
      this.display();
      Situation playerChoice = this.currentPlayer.play(this.legalPlays());
      this.setCurrentSituation(playerChoice);
      this.changePlayer();
   }

   this.display ();
   Player winner = this.getWinner(currentPlayer);
   System.out.println("*************** Winner is "+winner);
}
```

Cargaison : gestion de l'encombrement

Gestion de l'encombrement : la méthode `add` s'appuie sur la méthode abstraite `critereEncombrement`



Epreuves : `ajouteResultat`

On peut considérer que la méthode `ajouteResultat` satisfait ce design pattern car elle dépend de la méthode abstraite `creeResultat`.