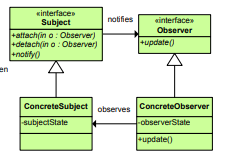
**Choix des Design Pattern**

* Observer:

Il s’agit d’un design pattern de comportement.

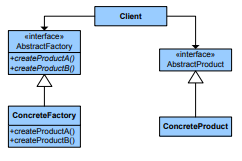
Permet d’établir une relation entre plusieurs objets, ainsi lorsqu’un objet change d’état, les autres sont avertis. Ce Design Pattern va nous être utile afin d’organiser les différents évènements au sein du restaurant comme : la disponibilité des plats sur le comptoir, si une table est disponible …



* Factory :

Il s’agit d’un design pattern de création.

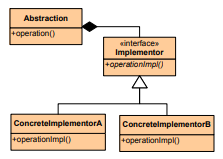
Permet de fournir une interface pour la création de familles d’objets ou d’objets sans avoir a spécifier la classe utilisée. Ce Design Pattern va nous être utile car on ne sera pas obligé de spécifier la classe à chaque création d’objet. Par exemple, cela peut être utile pour la création des clients.



* Bridge :

Il s’agit d’un design pattern de structure.

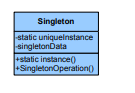
Permet de découpler une abstraction de son implémentation afin qu’ils puissent évoluer indépendamment.



* Singleton :

Il s’agit d’un design pattern de création.

Permet d’avoir une seule et unique instance d’une même classe dans un programme. Il nous permet de répondre aux principes SOLID, en effet si une classe est instancié une seule fois, elle n’aura qu’une seule responsabilité.



* Stratégie :

Il s’agit d’un design pattern de comportement.

Délègue une partie du comportement d’un objet à un autre. L’utilisation de ce Design Pattern est une alternative à l’utilisation excessive ou à la mauvaise utilisation de l’héritage d’une classe. Cela permet de déléguer seulement une partie du comportement d’un objet et pas la totalité, améliorant ainsi la qualité de code.

