 **République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université Akli Mohand Oulhadj de Bouira**

**Faculté des Sciences et des Sciences Appliquées**

**Département d'Informatique**

**Compte Rendu de MTI**

***Spécialité*** *: GSI*

|  |
| --- |
| **TP n°6**  Design pattern :  -Stratégie pattern  -Singleton pattern  -Adaptateur pattern  -Façade pattern |

**Réalisé par**

-Boukezzoula Boualem

- Mehennaoui Abdelghani

2019/2020

**Compte rendu de TP :**

**Méthodologie de travail :**

Dans la 1er partie : de TP on a vu un autre model qui s’appel design pattern en français conception de modèle, ce dernier décrit un problème qui se produit très fréquemment, dans un environnement.

Et dans la 2éme : partie de TP c’est un travail a domicile, pour refaire tout le travail qui sera fait dans le TP en java.

Partie1 : TP

Le design pattern c’est l’architecture de la solution à ce problème de telle façon que l’on puisse utiliser cette façon des milliers de fois.

Le design pattern a décomposé à trois catégories :

**Création** : description de la manière dont un objet ou un ensemble d’objets peuvent être créés.

Parmi ses modèle : Abstract Factory, Boulder, Prototype, Singleton.

**Structure** : comme adapter (objet), Composite, Bridge…

**Comportement :** Description de comportement d’interaction entre objets.

Exemples : Stratégie, Observer, Médiator…

**Parti 1 :**

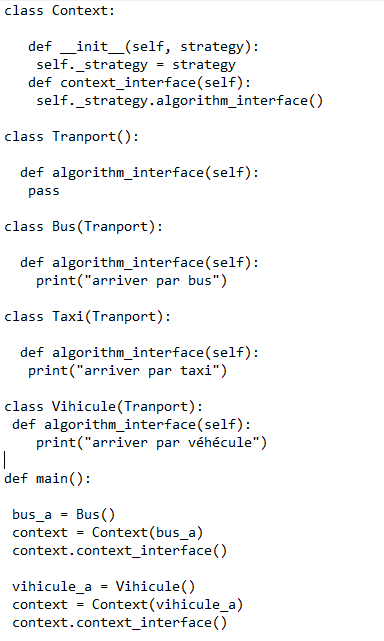
Le model qu’on a vu dans la première partie de ce TP c’est le model stratégie qui fait partir de catégorie comportement.

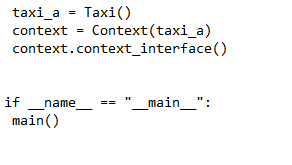
L’avantages de ce model est définir une famille d’algorithmes ,et encapsuler chacun et les rendre interchangeables tout en assurant que chaque algorithmes puisse évoluer indépendamment des clients qui l’utilisent, et la raison d’utiliser ce model que un objet pouvoir faire varier une partie de son algorithme dynamiquement.

Le diagramme de class qui correspond a une Stratégie

Dans cette partie on va reformuler le code et créer une stratégie pattern pour permettre au client de choisir un moyen de transport a l’aéroport d’Alger, soit avec un bus, un taxi, ou bien un véhicule personnel.

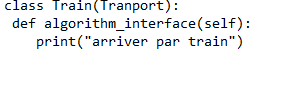
Voici le code qui correspond à cette stratégie :





Apres une construction de nouvelle gare de train on ava ajouter un nouveau moyen de transport, donc on va modifier notre stratégie on va implémenter une nouvelle classe train.

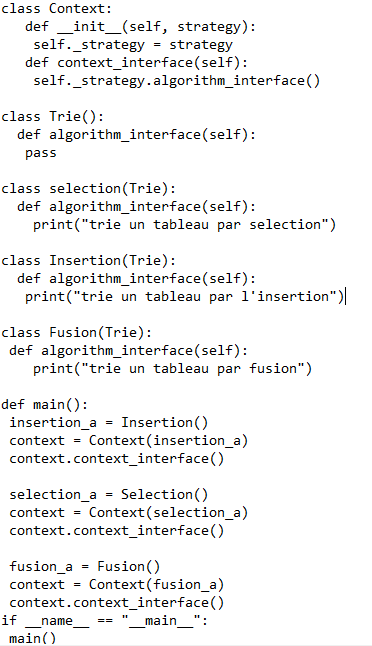
Voila comment :



La dernier partie de pattern stratégie on veut créer une application qui tri une liste en utilisant

Plusieurs algorithmes de tri.

Voila notre stratégie pattern qui permet de choisir un algorithme pendant l’exécution :



**Parti 2 :**

Dans cette deuxième partie on verra un autre model de design pattern que on l’appel singleton pattern qui fait partie de catégorie création, son principe est de créer une seule instance dans une classe.

Ces avantages :

-Le singleton permet de créer une seule instance unique d’une application.

-Il est très exploité dans les applications qui s’exécutent dans un environnement multithreads.

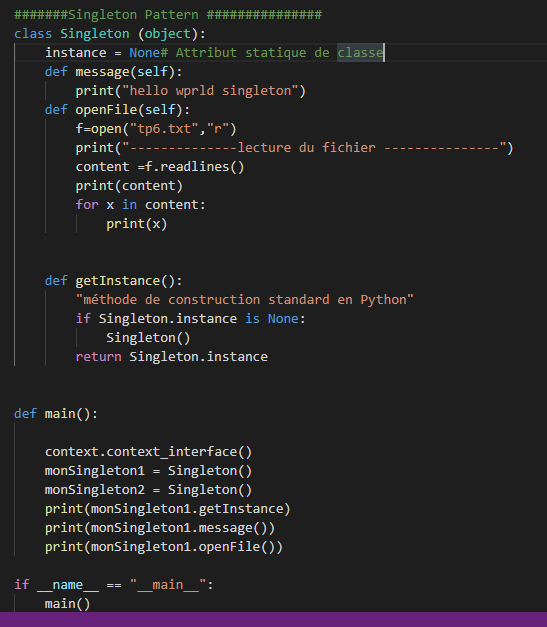
Le diagramme de classe général de singleton pattern :

|  |
| --- |
| **Singleton** |
| - singleton: Singleton |
| -Singleton ()  + getIstance (): Singleton |

Dans cette partie on va créer un fichier csv et lire les données de ce fichier, et pour les mesures de sécurités on va limiter l’accès au fichier ca veut dire on aura le droit de l’ouvrir une seul fois.

Et pour faire ca on va créer une classe qui permet d’ouvrir un fichier CSV et lire les données puis on va l’implémenter pour qu’elle puisse créer une seule instance pour la manipulation de fichier.

Voila le code :

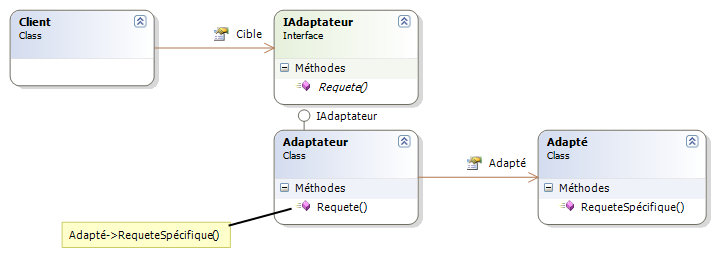


**Parti 3 :**

Dans cette partie on a vu un autre pattern qui s’appelle adapter.

Adaptateur est un patron de conception (*design pattern*) de type structure (*structural*). Il permet de convertir l'interface d'une classe en une autre interface que le client attend.

Voici le diagramme de classe générale qui correspondant a ce design pattern :



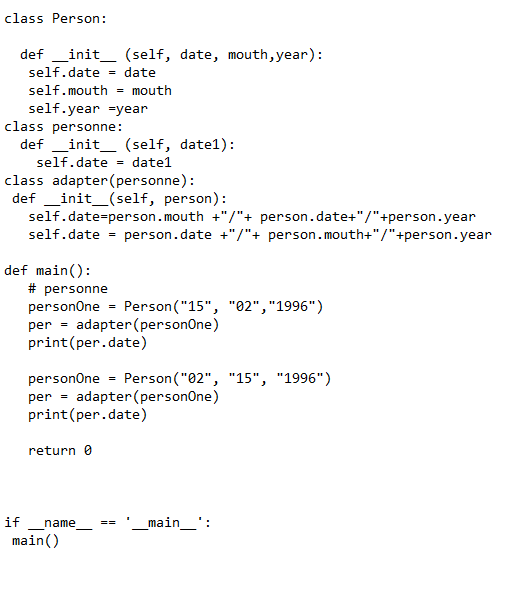
Les avantages de l’adaptateur :

Permettre a des classes de fonctionner ensemble.

L’objectif de l’adaptateur :

On peut également utiliser un adaptateur lorsque l'on ne veut pas développer toutes les méthodes d'une certaine interface.

Code1 :



Adaptateur qui permet d’afficher la date en plusieurs styles

Parti4 :

Dans cette partie on a fait un autre pattern qui fait partie des patterns structure, façade est utilisée pour simplifier l'utilisation

Le diagramme de classe générale :

Ces avantages :

-Simplifier l’utilisation de l'API.

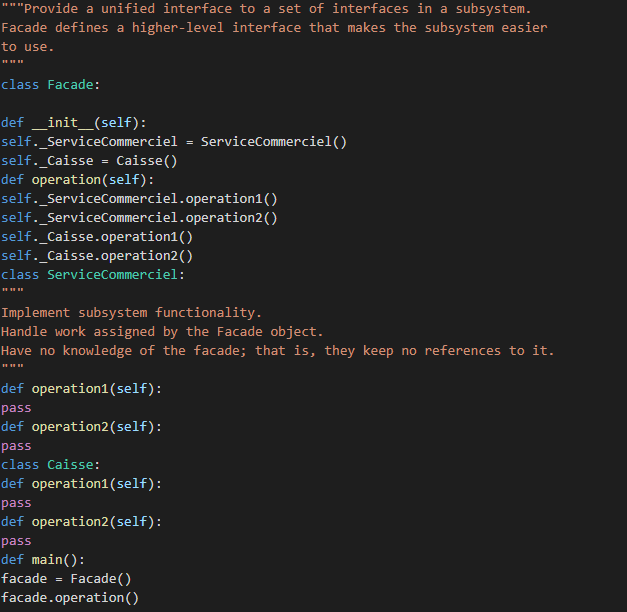
-La façade permet de simplifier cette complexité en fournissant une interface simple du sous-système.

-La façade encapsule la complexité des interactions entre les objets métier participant à un workflow (la représentation d'une suite de tâches ou opérations effectuées par une personne, un groupe de personnes).

 -Façade a pour but de cacher une conception et une interface complexe difficile à comprendre.

Dans cette partie on va formuler un code et créer une façade pattern pour permettre au client à utiliser un guichet unique pour payer une facture de téléphone ou d’électricité.

Voila la solution proposée :

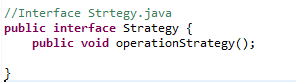


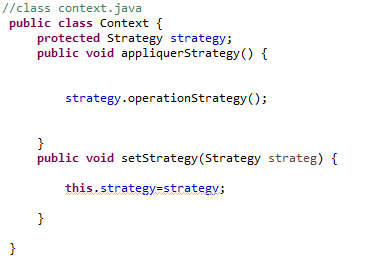
Partie2 : Travail a domicile

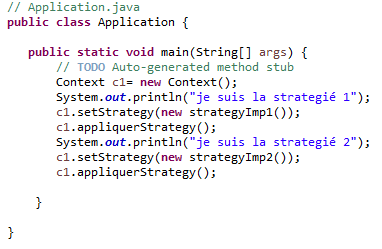
TP en java :

Le travail demandé dans cette partie si de refaire tout le travail précédant en langage java.

1. Le design pattern stratégie en java :

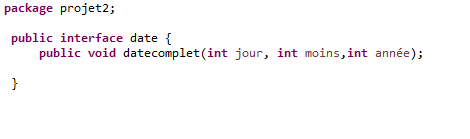




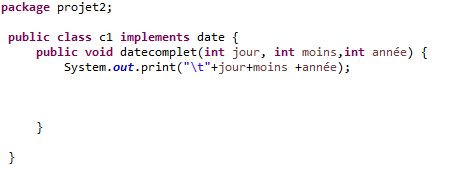


1. le design pattern adaptateur en java :

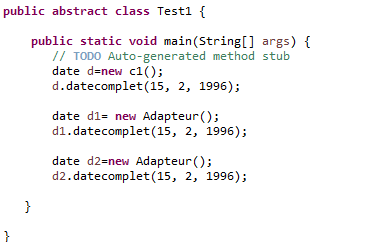
/-----------------------------------------L’interface -----------------------------------------------/



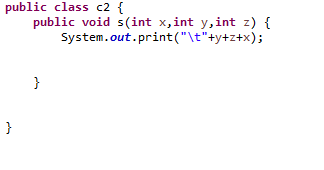
/-----------------------La classe c1 qui implémente l’interface date ------------------------------/



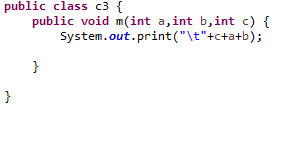
/-----------------------------------------------La classe Test1-------------------------------------------------/



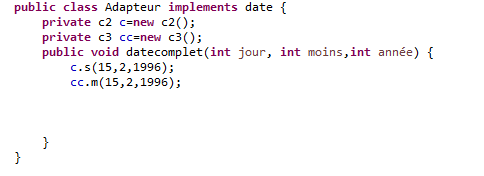
/-------------------------------------------------------la classe c2---------------------------------------------------------/



/--------------------------------------------------la classe c3------------------------------------------------------/

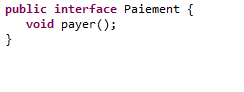


/--------------------------------------------------Adaptateur---------------------------------------------------------/



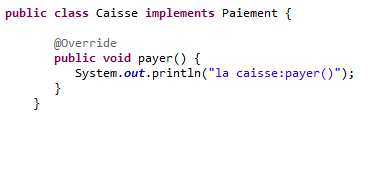
1. le design pattern façade en java :

1) une interface :

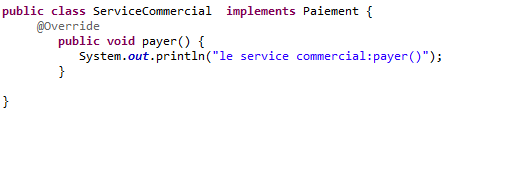


2) créer des classes qui implémentent l’interface :

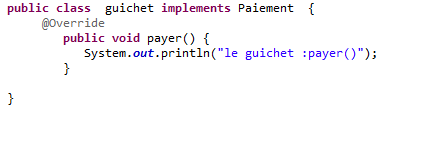
/----------------------Caisse.java--------------------------/



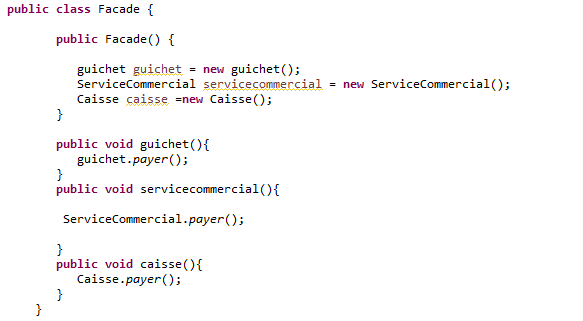
/----------- ServiceCommercial.java---------/



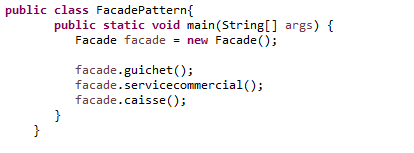
/----------- guichet.java--------/



3) Créer la classe façade :



1. utiliser la classe façade pour pouvoir accéder a tout les services de paiement :



1. vérifier les sorties :

