Section	**B3**
Date	**22/05/2024**
Enseignant	**M.MALDONADO**
Matière	**UML**

ATELIER METIER UML

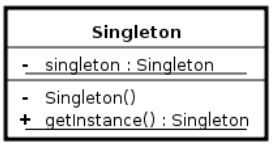
Ces ateliers sont à mener par groupe de 3 élèves. Le thème choisi est le jeu Monopoly. Vous pourrez chercher sur internet les informations dont vous aurez besoin au fur et à mesure des questions.
Nous nous intéressons à la traduction de design patterns dans le contexte du jeu.

Vous choisirez le langage cible de votre choisi parmi les suivant : C#, Java, ou PHP.

Atelier 1 : Design pattern SINGLETON

https://fr.wikipedia.org/wiki/Singleton_(patron_de_conception)

Objectif : n'autoriser la création que d'une seule instance d'une classe



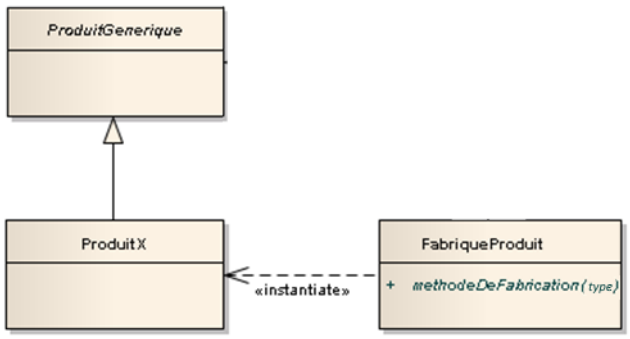
Question 1 :
Appliquez ce design pattern à la classe Banque :
- attribut : *int cash*
- méthodes : *get/set*

Programme de test :
1- création d'un premier objet b1 de type Banque, cash=1000
2- affichage du cash de b1
3- création d'un second objet b2 de type Banque, cash=500
4- affichage du cash de b2
5- affichage du cash de b1

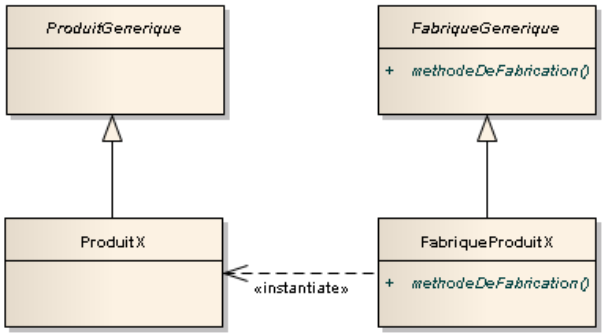
Atelier 2 : Design pattern FACTORY *https://fr.wikipedia.org/wiki/Fabrique_(patron_de_conception)*

Objectif : éliminer le couplage entre programme client et les produits créés, factorisation de la logique de création des produits créés (dans la factory)

V1 : simple factory



V2 : pattern factory method



Vous choisirez d'implémenter la version 1 ou la version 2.

Question 2 *(V1)*:
Appliquez ce design pattern aux types de propriétés du jeu (Terrain, Gare, CompagnieEE) :
- ProduitGénérique : *Propriété*
- ProduitX : *Terrain, Gare, CompagnieEE*
  attribut : *int prix, string nom*
  méthodes : *get/set, afficher*
- FabriqueProduit : *ProprieteFactory*
  méthode : creer

Programme de test :
1- création des terrains Rue de la Paix - 400€, Rue de Courcelles – 100€, et affichage
2- création de la gare Montparnasse – 200€, et affichage

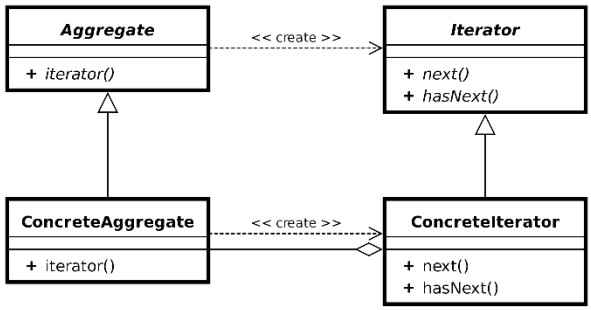
Question 2 *(V2)*:
Appliquez ce design pattern aux types de propriétés du jeu (Terrain, Gare, CompagnieEE) :
- ProduitGénérique : *Propriété*
- ProduitX : *Terrain, Gare, CompagnieEE*
-
- attribut : *int prix, string nom*
- méthodes : *get/set*

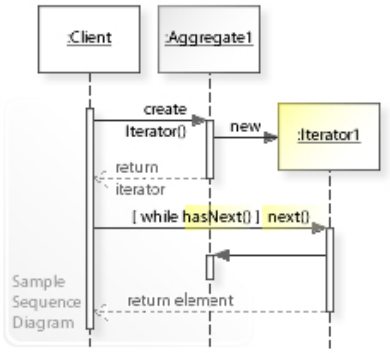
Programme de test :
1- création d'une factory terrainFactory et d'une factory gareFactory
2- création des objets terrain et gare de la V1, et affichage

Atelier 3 : Design pattern ITERATOR *https://en.wikipedia.org/wiki/Iterator_pattern*

Objectif : parcourir un ensemble/collecton d'éléments, en découplant le code de parcours et la structure de l'ensemble.

V1 : simple factory





Question 3 :
Appliquez ce design pattern à la classe Plateau (contenant un ensemble de cases)
- ConcreteAgregate : *Plateau*
- ConcreteIterator : *PlateauIterator*

Plateau :
attributs : *case[] cases*
méthodes :*ajouterCase(Case c), getCase(int i), nbCases()*

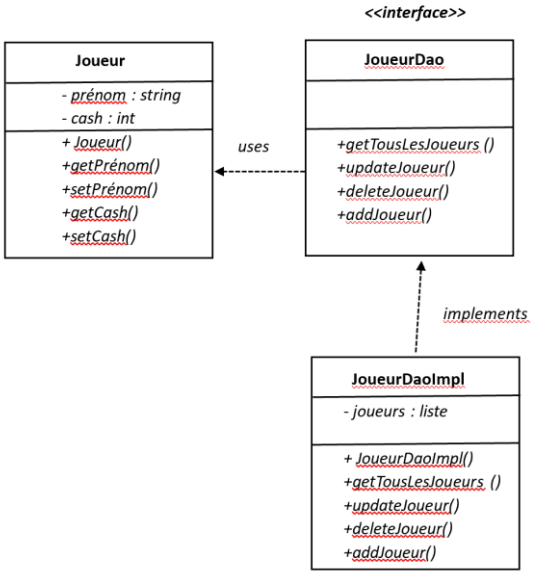
Case :
attribut : *int numero, string nom*
méthodes *afficher()*

Programme de test :
1- création d'un objet plateau
2- création des 10 premières cases du jeu et ajout dans le plateau
3- boucle de parcours sur le plateau et affichage de chaque case

Atelier 4 : Design pattern DAO

https://fr.wikipedia.org/wiki/Objet_d accès_aux_données

Objectif : découplage des objets métiers et de leur stockage



La base de données pourra être simulée :
- la méthode *getTousLesJoueurs* renverra une liste en mémoire
- les méthodes *addJoueur*, *updateJoueur*, *deleteJoueur* travailleront sur cette liste

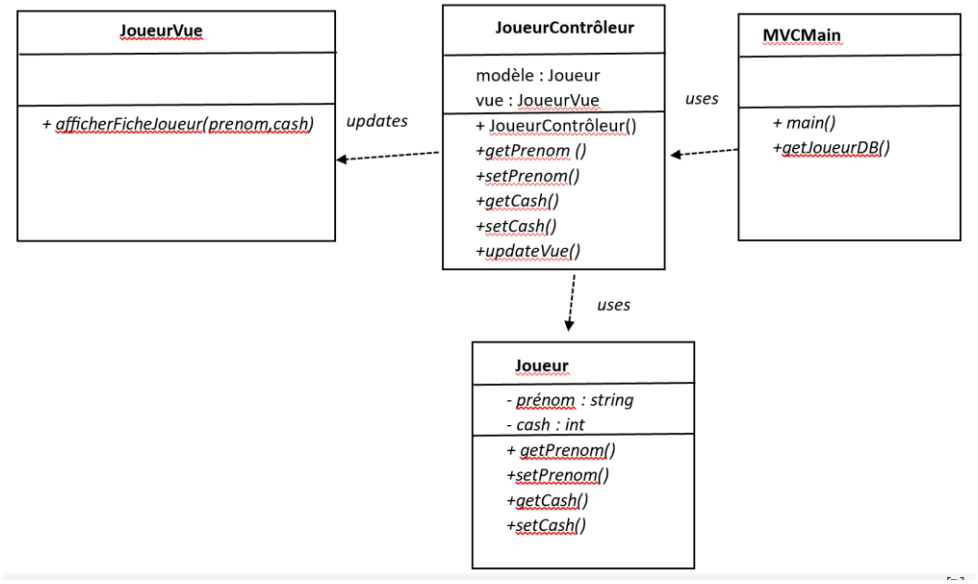
Question 4 :
Appliquez ce design pattern à la persistance des objets *Joueur*

Programme de test :
1- récupération de la liste des joueurs (liste de 3 joueurs préenregistrés)
2- ajout de 100€ à chaque joueurs
3- mise à jour des joueurs
4- suppression d'un joueur
5- récupération de la liste des joueurs pour affichage (boucle)

Atelier 5 : Design pattern MVC

https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur

Objectif : séparation des responsabilités
- modèle : classes métiers (données et logique métier)
- vue : présentation (interface)
- contrôleur : traitement des actions de l'utilisateur (modification modèle et actualisation vue)



Question 5 :
Appliquez ce design pattern pour afficher la fiche d'un jour (prénom, cash)

Programme de test :
1- récupération d'un joueur de la base de données (simulé)
2- création d'une vue Joueur
3- création d'un controleur Joueur et affichage (de la vue)
4- ajout de 100€ au cash du joueur
5- mise à jour de la vue