Cartouche du document

Année : ING 1
Matière : UML
Activité : Travail dirigé

Objectifs

L'objectif de ce travail dirigé est d'aborder dans la partie conception, la notion de réutilisation et donc de pattern.

On va aborder :

- le principe général de l'interface pour découpler des classes
- le pattern singleton
- le pattern MVC

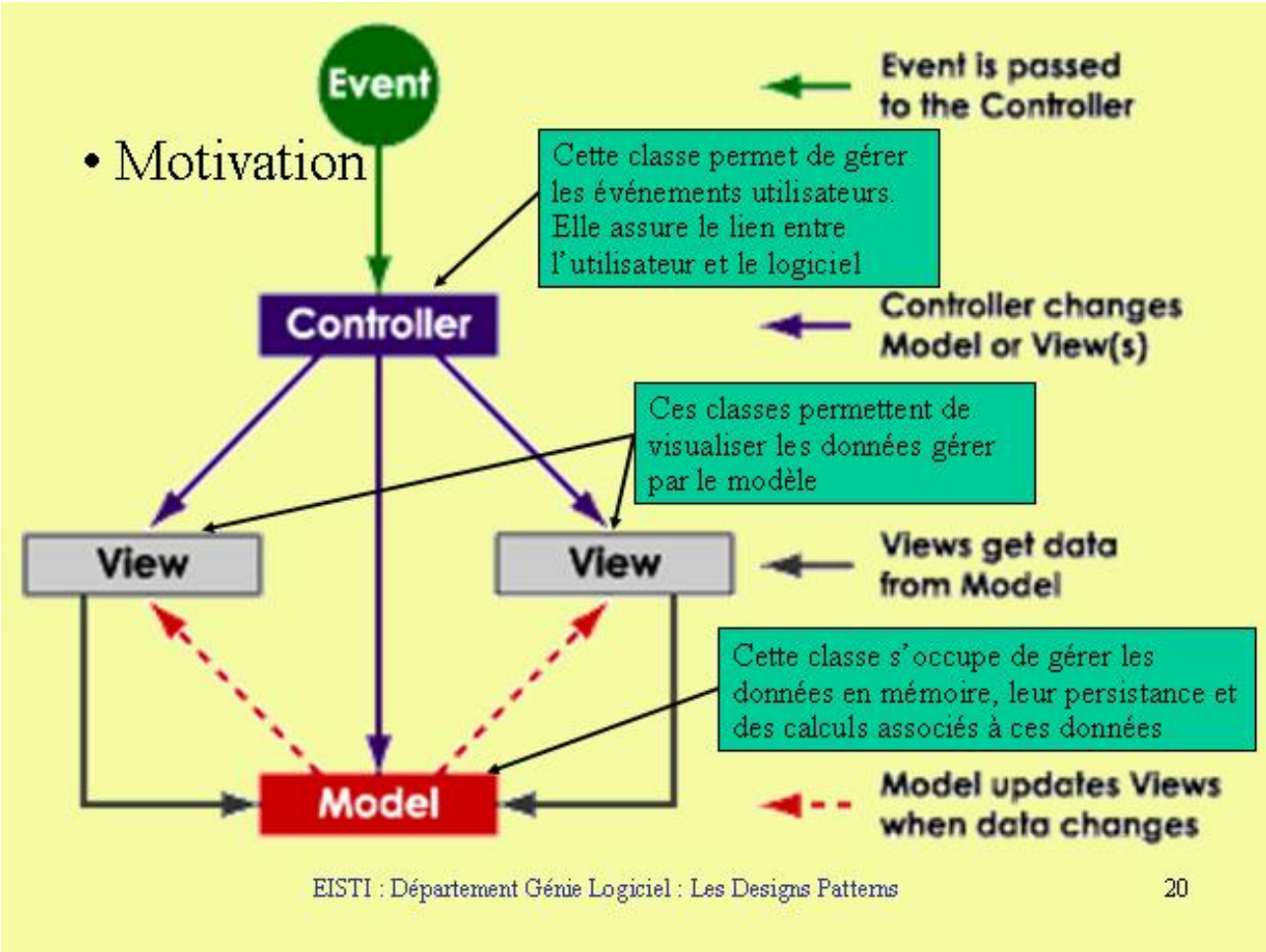
Le principe du Pattern Singleton est de construire une classe qui assure qu'une seule instance de cette peut-être créé. On donne ci-dessous le modèle de la classe à écrire.

```
class Singleton {
        private static Singleton refUnique = null;

        // le constructeur est protégé
        // cela interdit l'allocation directe d'un objet dans
        // une méthode d'une classe qui n'a pas de lien de parenté
        protected Singleton () {
                ...
        }

        // L'unique méthode qui permet d'allouer un objet
        public Singleton getRefUnique() {
                if(refUnique == null)
                        refUnique = new Singleton();
                return refUnique;
        }
}
```

Le principe du Pattern MVC est donné dans le schéma suivant :



Sommaire des exercices

Corps des exercices

1 - Simulateur de courses de voitures

Énoncé :

Il s'agit dans cet exercice de concevoir un simulateur qui permet à des utilisateurs de jouer avec des voitures.

Ce simulateur doit être conçu de telle manière qu'il ne doit connaître d'une voiture que son comportement. Si on applique ce principe, le simulateur n'a pas à connaître la structure interne d'une voiture et il peut donc être conçu pour n'importe quelle voiture.

Pour simplifier l'exercice, on supposera les choses suivantes :

- Le comportement d'une voiture se résume à démarrer, à accélerer, à tourner, à freiner et à s'arrêter.
- L'utilisation du simulateur se résume à un seul scénario :

- acquérir une voiture
- démarrer
- accélerer
- accélerer
- freiner
- tourner
- accélerer
- freiner
- freiner
- arrêter

Question 1)

Énoncé de la question

Identifier les différentes classes et interfaces de ce problème

Solution de la question

Les classes sont:

- Simulateur
- Ferrari (par exemple)
- SimulateurFerrari (par exemple)

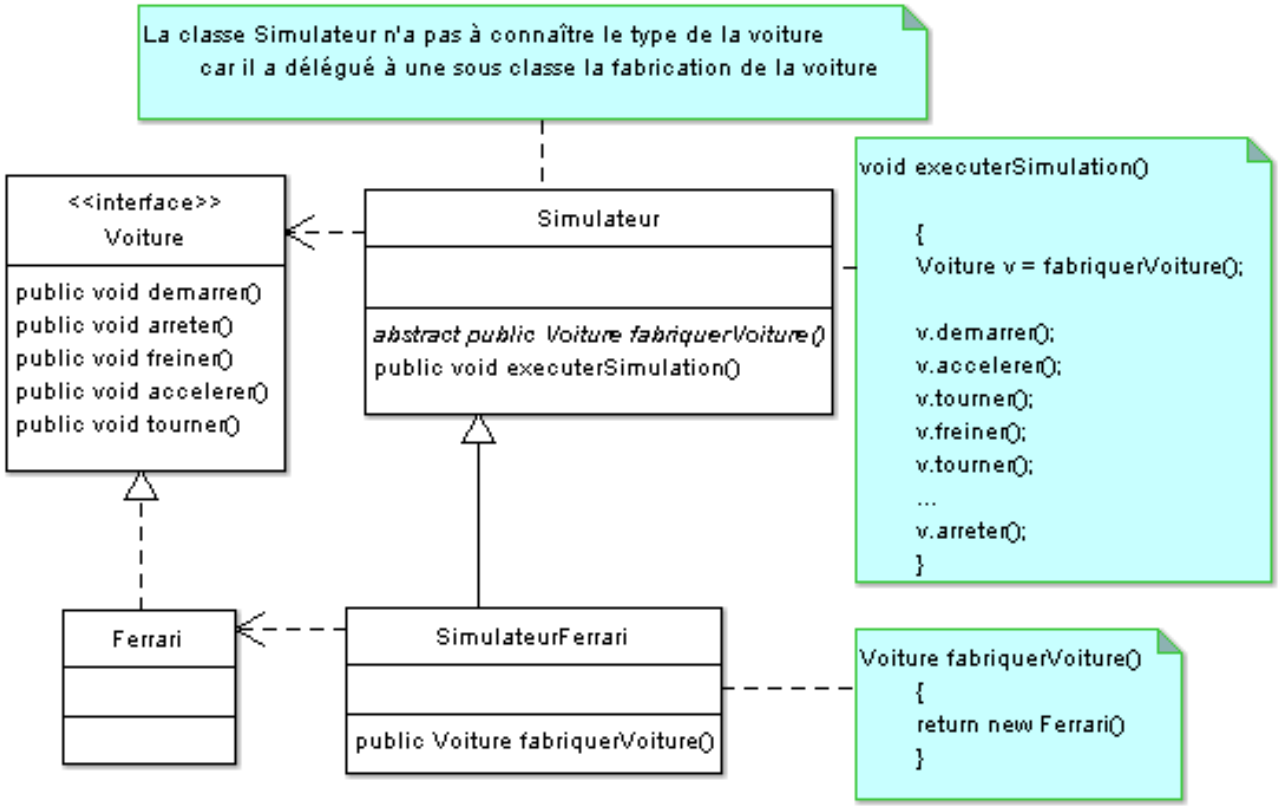
L'unique interface est Voiture.

Question 2)

Énoncé de la question

Ecrire le diagramme de classe de la solution. On illustrera en commentaire l'unique scénario.

Solution de la question



Question 3)

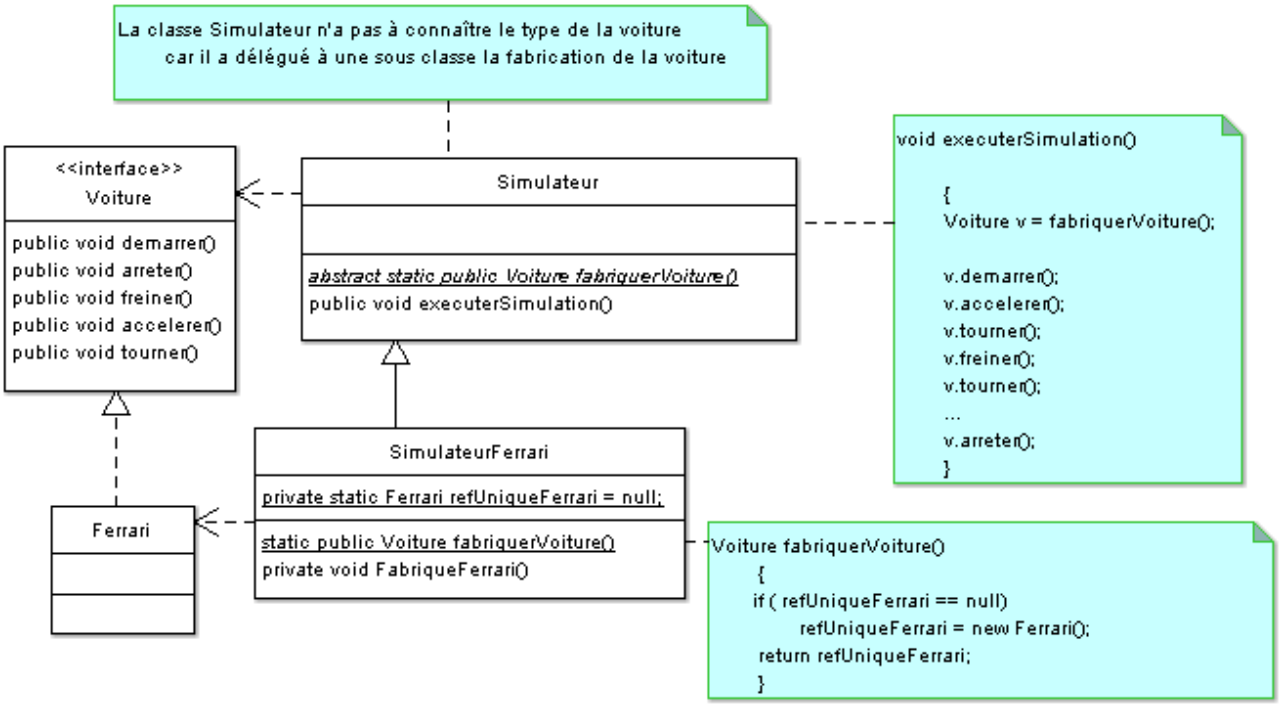
Énoncé de la question

On suppose maintenant que le simulateur est plus compliqué. Il implémente plusieurs scénarii. Dans chaque scénario, on a besoin de récupérer une voiture.

Que faut-il changer dans l'une des classes pour être assuré de travailler avec la même voiture dans tous les scénarii ?

Solution de la question

Il faut changer la classe FabriqueFerrari en s'inspirant du pattern Singleton.



2 - Programme de calcul matriciel

Énoncé :

Il s'agit dans cet exercice de concevoir un logiciel de calcul matriciel.

Globalement ce programme doit pouvoir saisir des matrices, faire des calculs sur ces matrices et afficher des résultats.

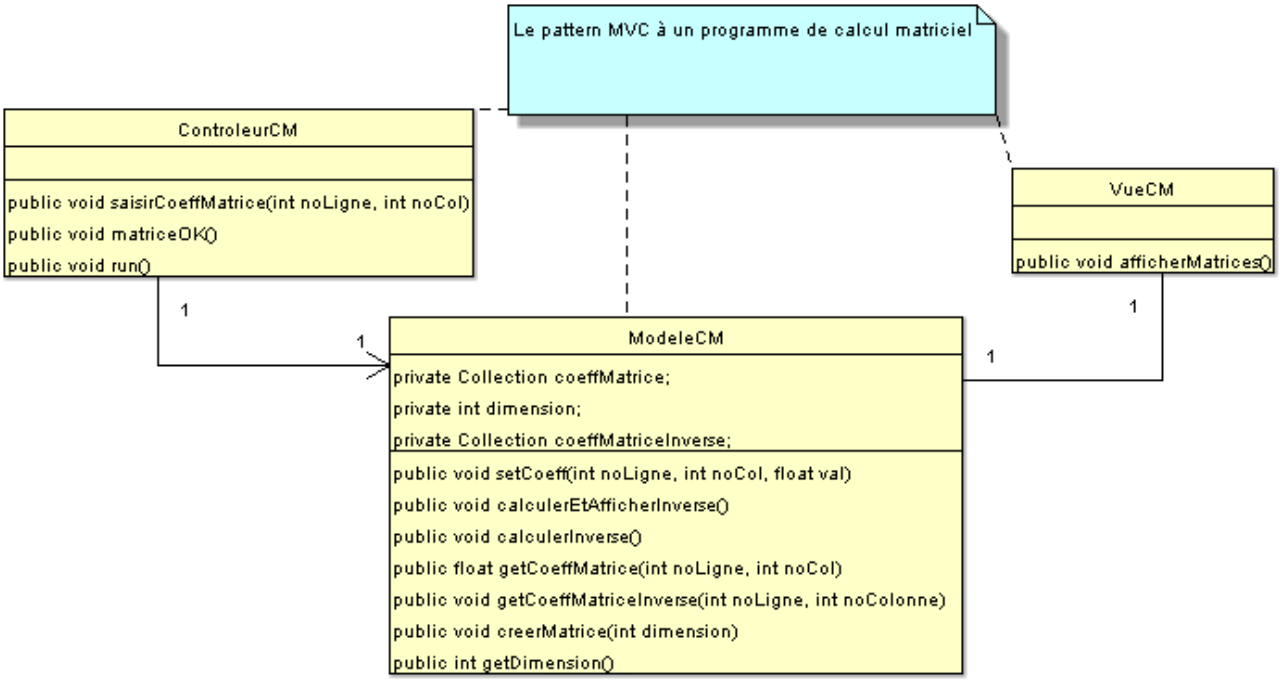
Pour simplifier l'exercice, on supposera les choses suivantes :
- Les matrices sont carrées et de dimension 2.
- On doit pouvoir saisir une matrice et modifier tout coefficient d'une matrice déjà saisie
- Le seul calcul possible est l'inverse de la matrice
- On affiche la matrice et son inverse l'une à côté de l'autre

Question 1)

Énoncé de la question

En appliquant le pattern MVC, on vous demande de définir les différentes classes en utilisant le langage UML.

Solution de la question



Question 2)

Énoncé de la question

On suppose maintenant que l'on désire afficher la matrice et son inverse l'une en dessous de l'autre.

Que faut-il changer dans le code précédent ?

Solution de la question

Avec le principe du pattern MVC, il suffit simplement de changer la classe VueCM. Dans cette classe, il faut recoder la méthode afficherMatrices.