Cartouche du document

Année : ING 1
Matière : UML
Activité : Travail dirigé

Objectifs

Il s'agit dans ce travail dirigé d'introduire les Design Patterns à travers

- Fabrique abstraite
- Fabrique
- Singleton
- Itérateur

Sommaire des exercices

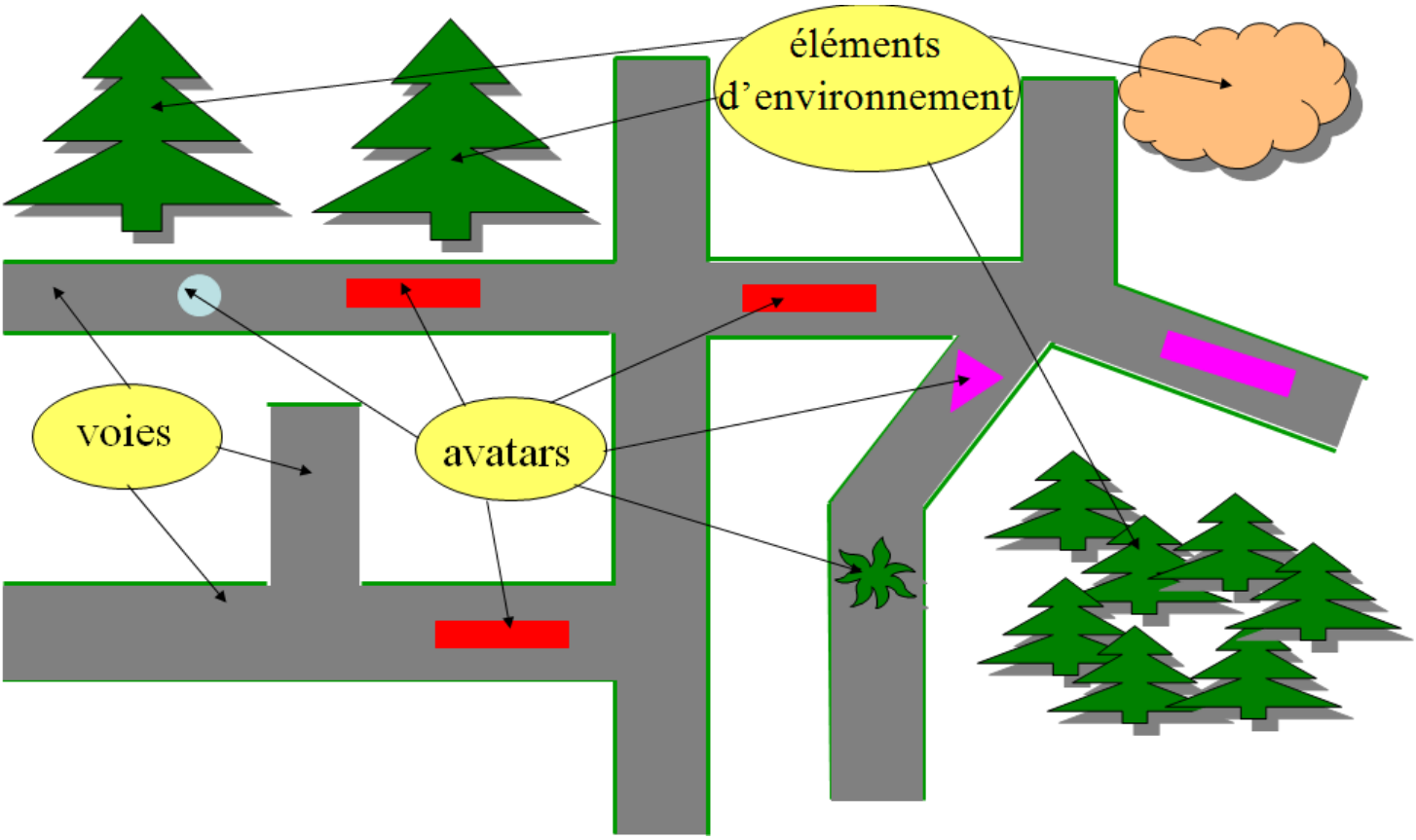
Corps des exercices

1 - Un logiciel d'environnement

Énoncé :

Dans cet exercice on s'intéresse à un logiciel de fabrication d'environnement. Dans un environnement, on trouve :

- des voies de circulations
- des avatars qui circulent dans les voies : animaux, cyclistes, piétons, ...
- des éléments d'environnement : arbres, rivières, nuages, ...



Un exemple d'environnement

Question 1)

Énoncé de la question

Donner en UML une description générique d'un environnement en utilisant le pattern de Fabrique Abstraite.

Question 2)

Énoncé de la question

On s'intéresse à un cas spécifique d'environnement : des parc municipaux.

Dans cet environnement :

- les voies sont des chemins de terre ou des pistes cyclables;
- les éléments d'environnement sont des arbres, des tables pour piqueniquer
- les avatars sont des piétons ou des cyclistes

Enrichir la description UML de la question précédente pour tenir compte de ces nouveaux éléments.

Question 3)

Énoncé de la question

Dans le code de l'objet client qui va manipuler des environnements, on désire pouvoir à tout moment disposer

- du nombre de voies utilisés
- du nombre d'éléments d'environnement utilisés
- du nombre d'avatars utilisés

- Quel est la classe la plus adéquate qui doit gérer ces traitements ?
- Que préconisez-vous pour cette classe pour être assuré que les traitements soient centralisés ?

2 - Pattern Iterateur : les arbres k-aire

Énoncé :

Dans cet exercice, on s'intéresse au type abstraite ArbreKaire :

```
TYPE ABSTRAIT ArbreKAire
Début du concept
Un objet de ce type permet de gérer des arbres d'arité k.
Opérations de base
Constructeur ArbreKAire : arbreVide() : ArbreKAire
Constructeur ArbreKAire : creerArbre(Element e,Entier k) :
ArbreKAire
Transformateur ArbreKAire : modifierRacine (Element) : ArbreKAire
Transformateur ArbreKAire : affFils (Entier, ArbreKAire):
ArbreKAire
```

```
Transformateur ArbreKAire : supprimerFeuille(Entier) :ArbreKAire
Observateur ArbreKAire : estVide () :  Booleen
Observateur ArbreKAire : recRacine() :  Element
Observateur ArbreKAire : recFils(Entier) :  ArbreKAire
Observateur ArbreKAire : recParent() :  ArbreKAire
Observateur ArbreKAire : recArite() :  Entier
Fin des opérations de base
```

Nous avons vu en algorithmique plusieurs façons de parcourir un arbre. Il paraît donc intéressant d'implémenter ce type abstrait en s'appuyant sur le pattern Itérateur.

Question 1)

Énoncé de la question

Donner en UML la traduction en classes de ce type abstrait en s'inspirant du pattern Itérateur.

Question 2)

Énoncé de la question

En reprenant les classes précédentes, on vous demande d'écrire le diagramme de séquences qui modélise une instance du cas d'utilisation "suivant".

Question 3)

Énoncé de la question

Une question plus conséquente pour d'une part faire le lien entre la conception et l'implémentation et d'autre part s'entraîner en Java..

Implémenter en Java les classes précédentes en créant les deux itérateurs concrets : parcours en profondeur et parcours en largeur.