Cartouche du document

Année : ING 1
Matière : UML
Activité : Travail dirigé

Objectifs

L'objectif principal de ce travail dirigé est d'aborder le passage de l'analyse à la conception.

Nous imposons comme choix de conception de générer du code Java.

Nous étudierons des règles de passages ou de Mapping des classes d'analyse.

- **vers des classes Java.**

Les régles concernent
- La correspondance entre classe d'analyse et classe Java (attributs et opérations)
- Les méthodes get et set pour récupérer et mettre à jour les attributs d'un objet
- L'information contenue dans les associations
  - - La navigabilité : qui connait qui ?
  - - Les cardinalités : en quelles quantités ?
  - - Les liens d'existence entre les objets associés : qui dépend de qui ?
- L'information contenue dans les automates d'états des objets

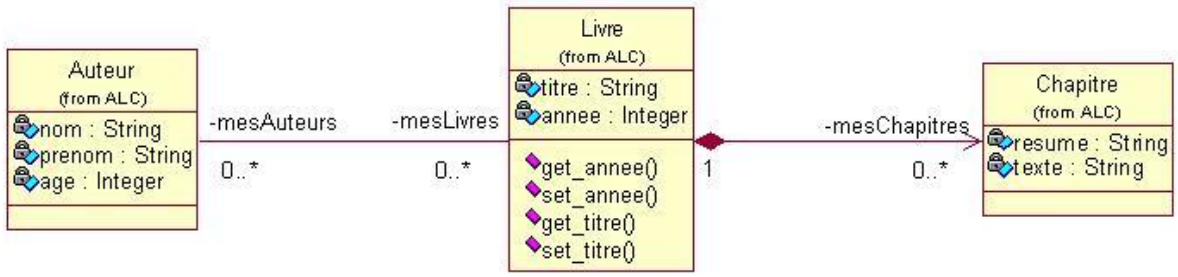
Sommaire des exercices

Corps des exercices

1 - UML vers Java : Association, composition et agrégation

Énoncé :

Dans le cadre d'une gestion de bibliothèque, on vous demande de traduire le schéma d'analyse ci-dessous en classe Java.



On vous demande de traduire en Java, ce diagramme de classes

Question 1)

Énoncé de la question

Que déduisez de ce schéma pour les cardinalités ?

Question 2)

Énoncé de la question

Que déduisez-vous de ce schéma pour la vie des objets ?

Question 3)

Énoncé de la question

Que déduisez-vous de ce schéma pour la navigabilité ?

Question 4)

Énoncé de la question

Faire le mapping des classes d'analyse en Java ?

On vous donne ci-dessous quelques éléments de code Java à compléter :

- Le fichier

```
/**
 Le fichier auteur.java
*/
public class Auteur1 {
  private String nom;
  private String prenom;
  private int age;

  // Pour gérer les différents livres d'un objet Auteur


  public Auteur1(String pnom, String pprenom, int page) {
    nom = pnom;
    prenom = pprenom;
    age = page;
```

```
    }

    public int getAge() { return age; }

    String getNom() { return nom; }
    String getPrenom() { return prenom; }

    // je dois pouvoir associer des livres à un auteur

    // je dois pouvoir dissocier des livres d'un auteur

  }
```

- Le fichier

```
  /* Le fichier Livre1.java */
  public class Livre1 {
    private String titre;
    private int annee;

    /* Pour gérer les différents auteurs d'un objet livre */


    /* Pour gérer les différents chapitres d'un objet livre */

    public Livre1(String ptitre, int pannee) {
      titre = ptitre;
      annee = pannee;
    }

    public int getAnnee() { return annee; }
    String getTitre() { return titre; }

    // Il faut pouvoir associer des auteurs à un objet Livre

    // Il faut pouvoir créer des objets Chapitre à un objet Livre

    // Il faut pouvoir supprimer des chapitres à un objet Livre

    // Il faut pouvoir dissocier des auteurs à un objet Livre

  }
```

- Le fichier

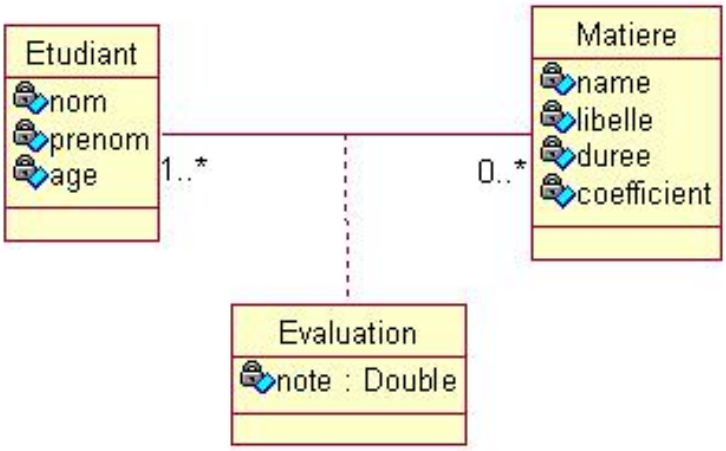
```
  /* Le fichier Chapitre1.java */
  public class Chapitre1 {
```

```
    private String resume;
    private String texte;

    public Chapitre1(String presume, String  ptexte) {
     resume = presume;
     texte = ptexte;
    }

    String  getResume() { return resume; }
    String  getTexte() { return texte; }
  }
```

2 - UML vers Java : Classe d'association : Des étudiants, des matières et des notes

Énoncé :

Dans le cadre d'une gestion de la scolarité, on s'intéresse au diagramme suivant :



Question 1)

Énoncé de la question

Faire le mapping des classes d'analyse en Java ?

3 - UML vers Java : Diagramme d'états d'un objet adhérent

Énoncé :

Dans cet exercice, on étudie comment intégrer le diagramme d'état dans la conception d'une classe.

Dans le cadre d'une gestion de bibliothèque, on s'intéresse à la classe Adherent. L'analyse nous

fournit les diagrammes suivants :

- Un premier diagramme de classe sommaire

Ebauche de la classe Adherent

- Le diagramme d'états/transitions de la classe Adherent

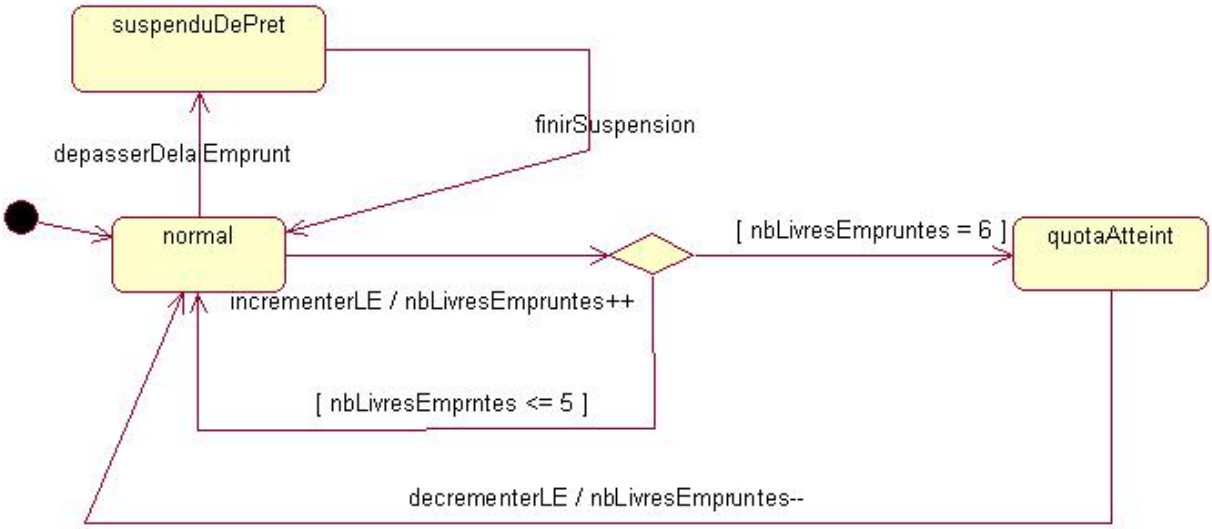


Diagramme d'états d'un objet Adherent

Question 1)

Énoncé de la question

Faire le mapping de la classe Adherent en tenant compte du diagramme d'état ? On définira dans la classe Adherent en plus des attributs pour le nom et le nombre de livres empruntés, un autre attribut pour gérer l'état de l'objet. Cet attribut sera de type int.

Dans cette même classe, on définira autant de constantes qu'il y a d'états possibles.

Question 2)

Énoncé de la question

Dans la solution précédente, on a un code truffé de structures conditionnelles portant sur l'état de l'objet. Si on veut faire évoluer cette classe, on sera probablement obligé de tout recoder.

Il existe une technique plus élégante et donc plus facilement modifiable. L'attribut d'état est une référence sur une interface. Cette interface est constituée de tous les prototypages de la classe Adherent. Le code de chaque méthode f1 de la classe Adherent consiste entre autre à demander à l'objet d'état d'exécuter sa propre méthode f1. Enfin, la classe Adherent aura une méthode qui permettra de changer l'attribut d'état.

Il faut coder autant de classes qu'il y a d'états possibles. Chacune de ces classes implémente l'interface et doit avoir un attribut qui référence l'objet Adherent. Grâce à cette référence, l'objet d'état lors de l'exécution d'une de ces méthodes pourra mettre à jour l'etat courant de l'objet Adherent.

Il est évident que dans un premier temps, cette solution paraît plus lourde mais le gain évident se trouve (comme souvent) dans l'évolution de la classe.

Faire le mapping de la classe Adherent en tenant compte du diagramme d'état avec cette nouvelle technique ?