

Modéliser avec le langage UML

Diagramme de Classes

Laure Gérard

Laure.gerard@grenoble-epsi.fr

Correction du devoir

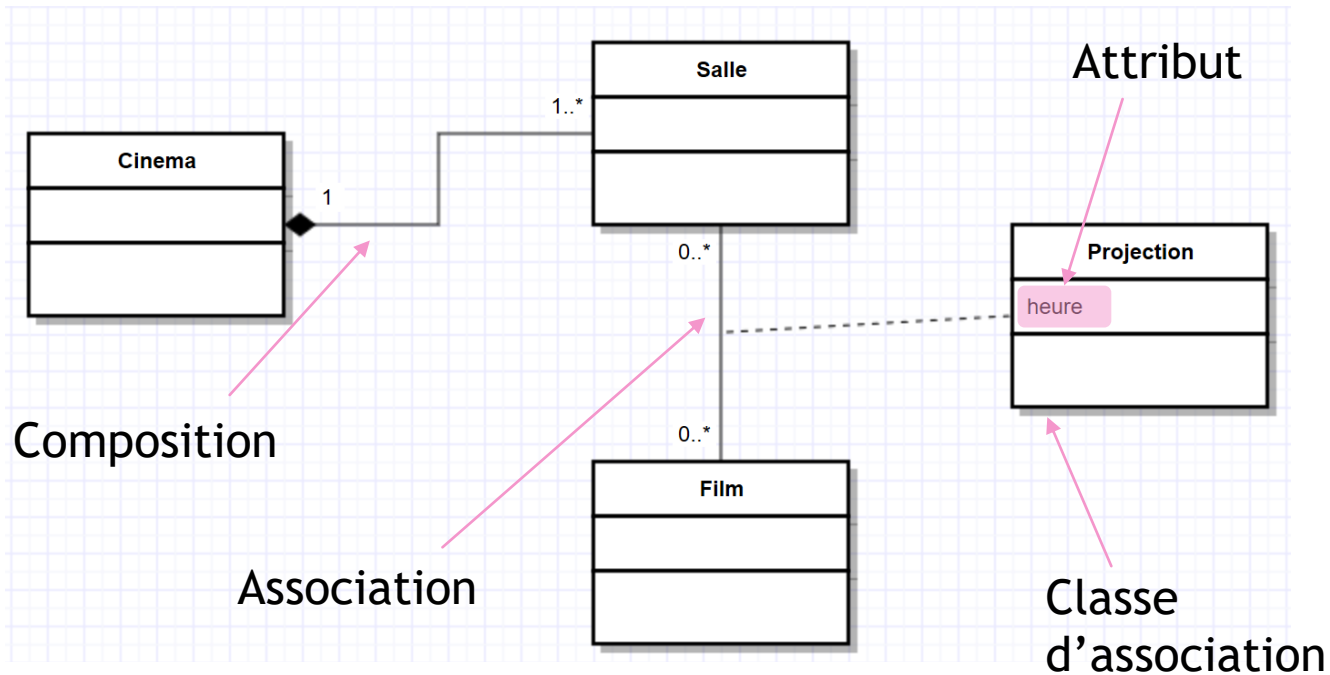
Exercice 1

- ▶ L'approche fonctionnelle est une approche intuitive qui vise à séparer les données de leur code de traitement. **VRAI**
- ▶ L'approche objet vise à regrouper un ensemble de propriétés (attributs) et le traitement associés (méthodes) en une entité autonome nommée objet. **VRAI**
- ▶ Un objet est :
 - ▶ Une entité définie par un type. **FAUX**
 - ▶ Un ensemble d'attributs caractérisant son état. **VRAI**
 - ▶ Un ensemble d'opérations définissant son comportement. **VRAI**
- ▶ Une classe est une instance d'objet. **FAUX**
- ▶ L'héritage est un mécanisme de transmission des attributs seulement vers une sous-classe. **FAUX**

Exercice 1

- ▶ Une classe peut être spécialisée en d'autres classes pour y ajouter des caractéristiques spécifiques. **VRAI**
- ▶ Plusieurs classes peuvent être généralisées en une classe qui les factorise pour regrouper les caractéristiques communes. **VRAI**
- ▶ L'association est une relation sémantique durable entre deux classes, nommée par un verbe et pouvant être enrichie des rôles et multiplicités. **VRAI**
- ▶ L'agrégation est une association symétrique exprimant une relation de contenance. **FAUX**
- ▶ La composition est une agrégation forte. Les cycles de vies des éléments ne sont pas liés. **FAUX**

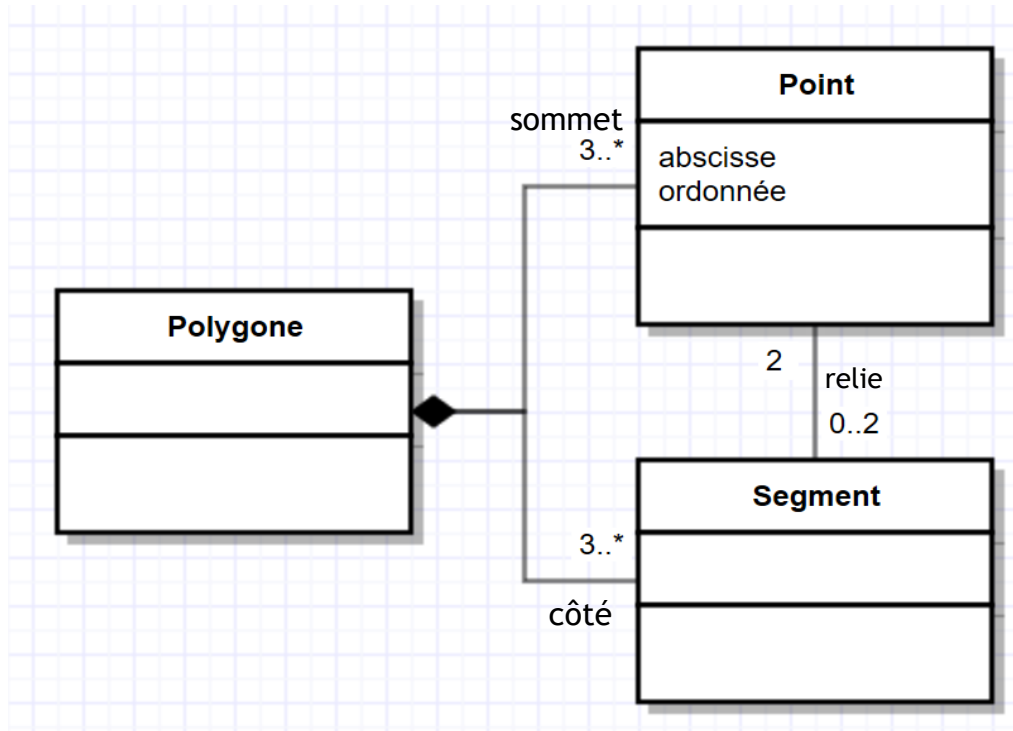
Exercice 2



=> Un cinéma est composé de plusieurs salles, dans lesquelles sont projetés des films. Les projections sont programmées à une heure déterminée.

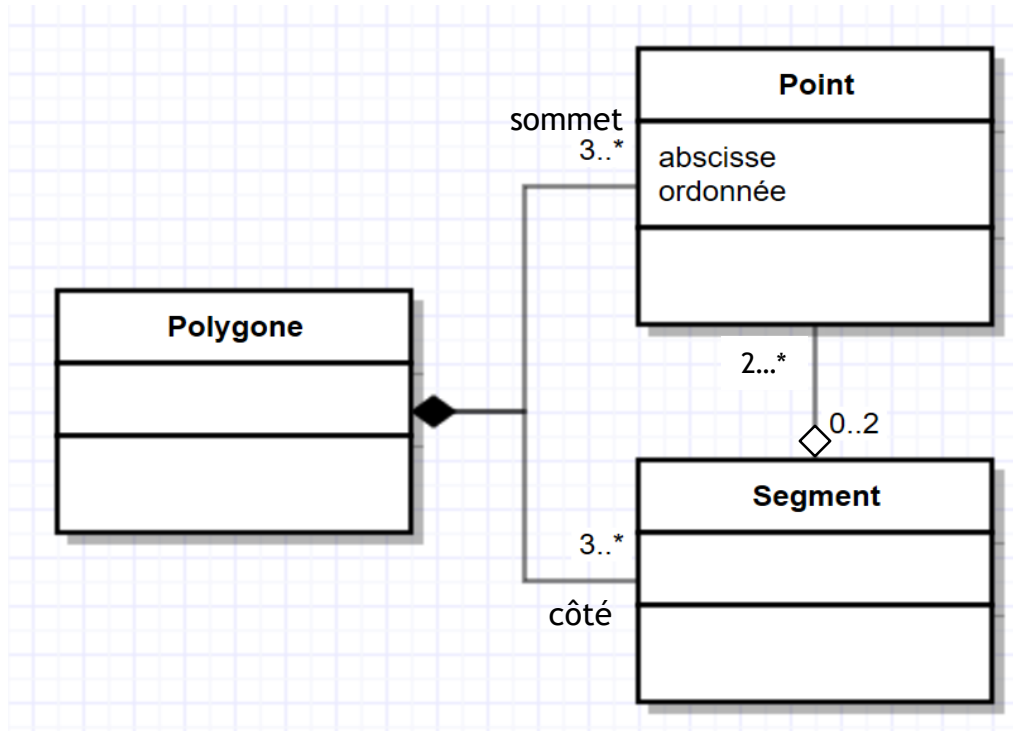
Exercice 3

Un polygone est constitué de points nommés sommets et de segments, nommés côtés, reliant deux points. Un point possède une abscisse et une ordonnée.



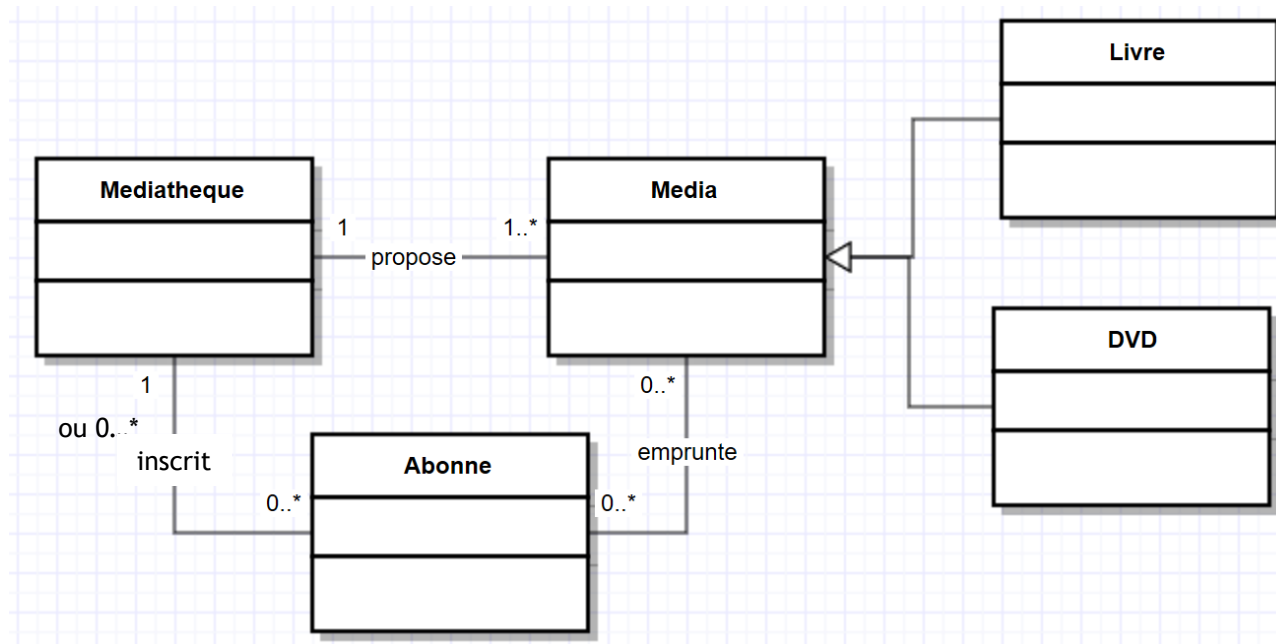
Exercice 3

Un polygone est constitué de points nommés sommets et de segments, nommés côtés, reliant deux points. Un point possède une abscisse et une ordonnée.



Exercice 3

Une médiathèque propose différents médias, tels que des livres et des DVD. Ces médias sont empruntables par les abonnés de la médiathèque.





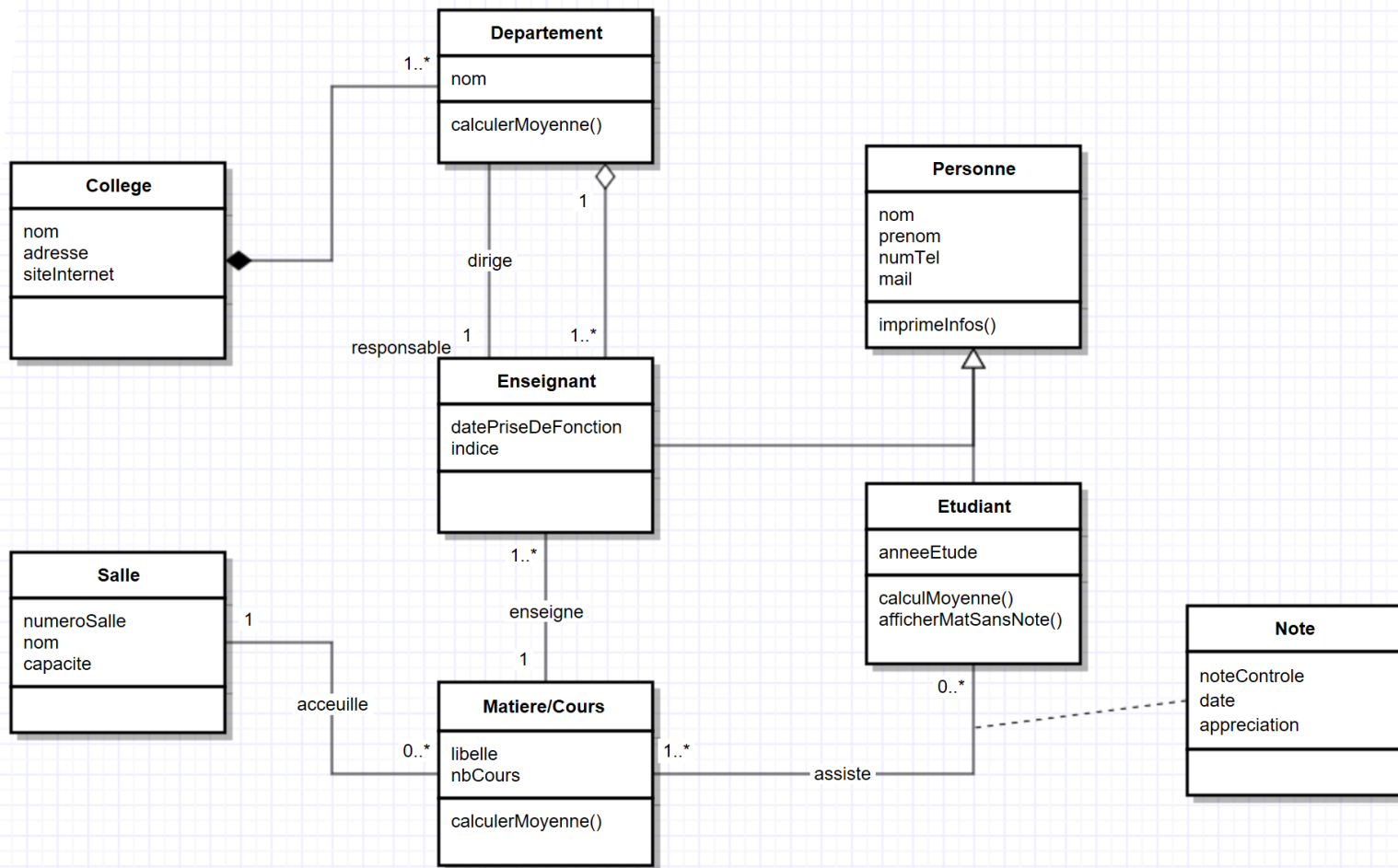
Cas d'étude 5

Gestion des cours

Gestion des cours

Une académie souhaite gérer les cours dispensés dans plusieurs collèges. Pour cela, les renseignements suivants sont disponibles :

- ▶ Chaque collège est caractérisé par un nom, une adresse et un site internet.
- ▶ Chaque collège est structuré en départements, qui regroupent plusieurs enseignants. Parmi ces enseignants, l'un d'eux est responsable du département.
- ▶ Un enseignant se définit par son nom, prénom, numéro de téléphone, mail, date de prise de fonction et son indice.
- ▶ Chaque enseignant ne dispense qu'une seule matière.
- ▶ Les étudiants suivent quant à eux plusieurs matières et reçoivent une note et une appréciation pour chacune d'elle.
- ▶ Pour chaque étudiant, on veut gérer son nom, prénom, numéro de téléphone, mail, ainsi que son année d'entrée au collège.
- ▶ Une matière peut être enseignée par plusieurs enseignants, mais a toujours lieu dans la même salle de cours (chacune ayant un nombre de places déterminées).
- ▶ On désire pouvoir calculer la moyenne par matière ainsi que par département
- ▶ On veut également calculer la moyenne générale d'un élève et pouvoir afficher les matières dans lesquelles il n'a pas été noté
- ▶ Enfin, on doit pouvoir imprimer la fiche signalétique (prénom, nom, numéro de téléphone, mail) d'un enseignant ou d'un élève.



Gestion des cours

Contraintes en UML

Contraintes formalisées via UML

- ▶ Contraintes structurelles : attribut de classes, type de relation en les classes, multiplicité et navigabilité des propriétés structurelles, etc.
- ▶ Contraintes de types : typage des propriétés (attributs et méthodes), etc.
- ▶ Autres contraintes : visibilité, méthode ou classes abstraites, etc.

Oui, mais les autres ?

- ▶ Une contrainte est une **condition** ou une **restriction sémantique** qui doit être appliquée par une implémentation correcte du système.

- ▶ Syntaxe : {contrainte}

Corps de la contrainte dans un langage qui peut être naturel, dédié (OCL) ou issu d'un langage de programmation.

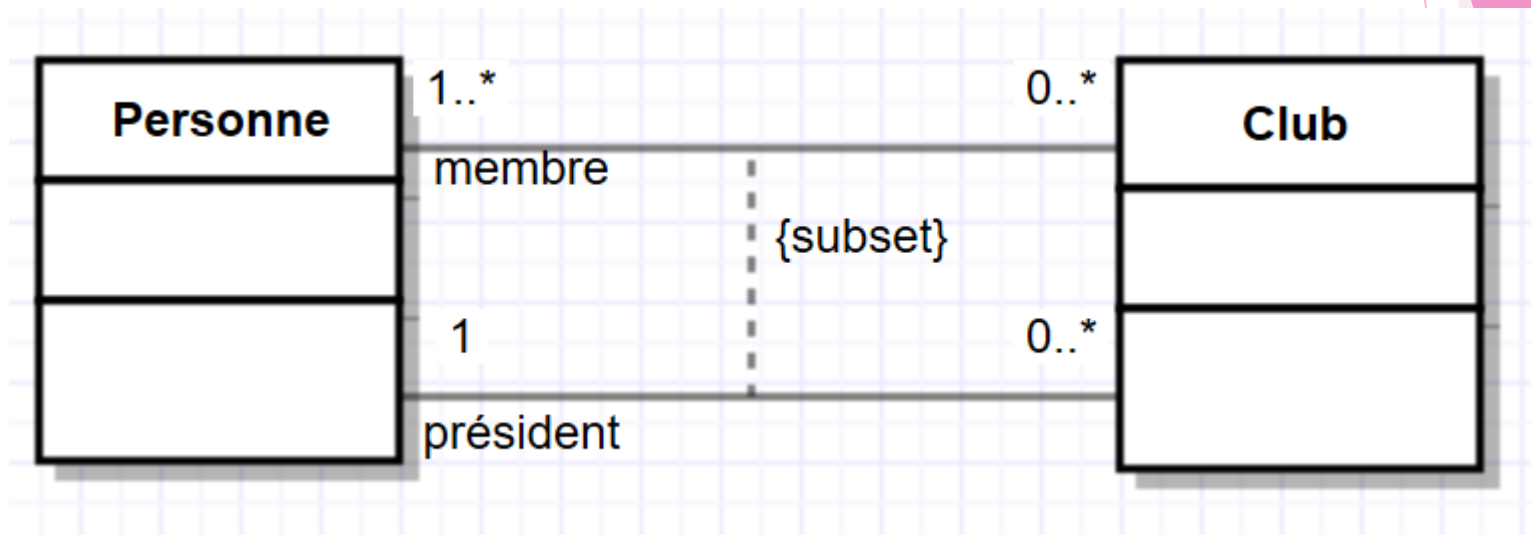
- ▶ Une contrainte peut avoir un nom.

Contraintes OCL

- ▶ OCL (Object Constraint Language) permet la formalisation des contraintes sur la plupart des diagrammes UML.
- ▶ Spécifier des contraintes sur l'état d'un objet ou d'une ensemble d'objets comme
 - ▶ Des invariants sur des classes
 - ▶ Des préconditions/postconditions à l'exécution d'opérations
 - ▶ Des attributs dérivés...

Contraintes prédéfinies

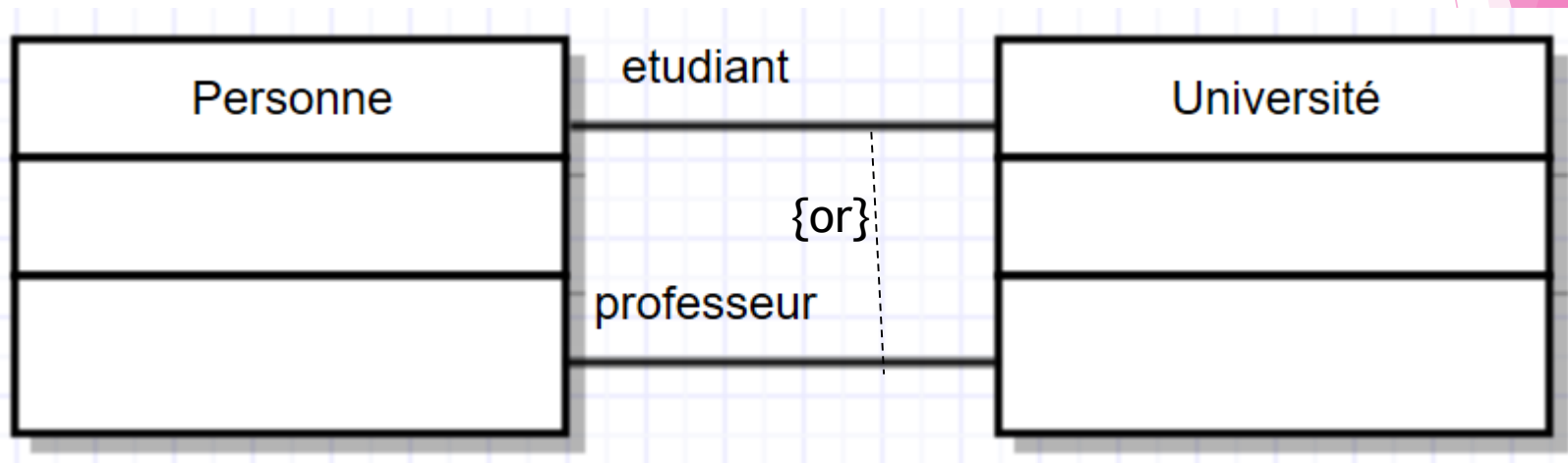
- Inclusion : {subset}



=> Ici, une personne peut être membre d'un club et être président de ce même club.

Contraintes prédéfinies

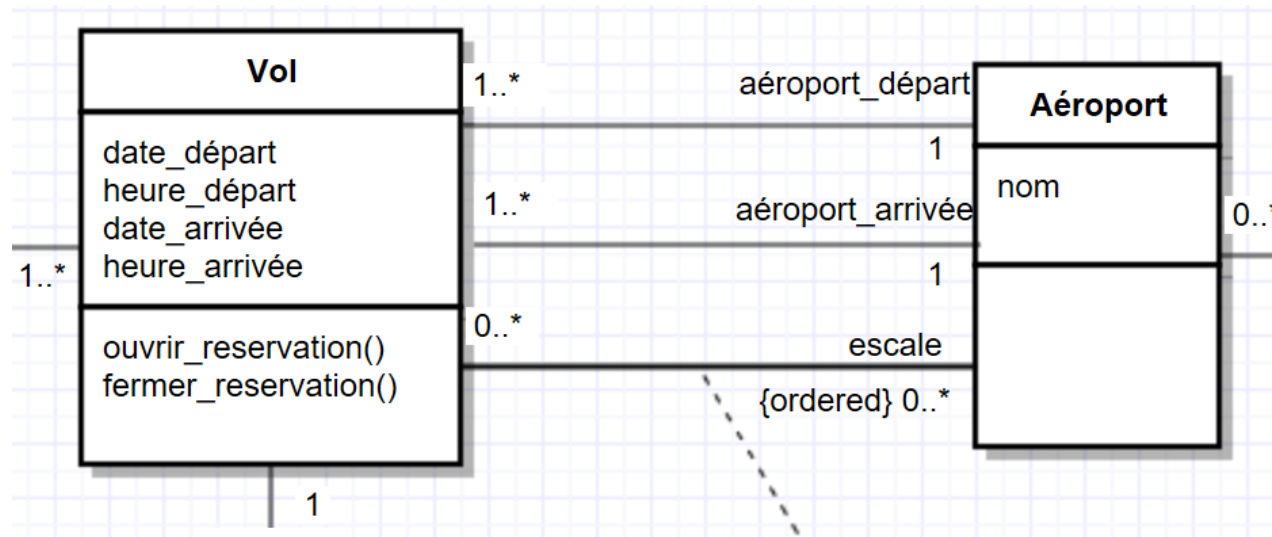
- Exclusion : {or}



=> Ici, une personne a soit le rôle d'étudiant soit le rôle de professeur au sein d'une université.

Contraintes prédéfinies

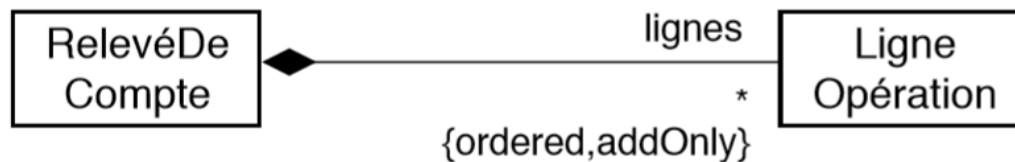
► Ordre : {ordered}



=> Ici, les différentes escales d'un vol sont ordonnées.

Contraintes prédéfinies

- Etat de l'objet : {addOnly} , {frozen} (invariable)



=> Ici, un relevé de compte se compose de différentes lignes d'opération qui sont ordonnées et impossibles à supprimer.

Exercice 1 : Gestion comptes bancaires

- ▶ Un compte bancaire est identifié par un numéro de compte. Ce numéro de compte est un entier positif, attribué par la banque à l'ouverture du compte et ne peut être modifié par la suite.
- ▶ Un compte est associé à une personne (civile ou morale) titulaire du compte, cette personne étant caractérisée par son nom, son prénom et son adresse. Une fois le compte créé, le titulaire du compte ne peut plus être modifié. Une même personne peut être titulaire de plusieurs comptes.
- ▶ La somme d'argent disponible sur un compte est exprimée en float. Cette somme est désignée sous le terme de solde du compte. Ce solde est un nombre décimal qui peut être positif, nul ou négatif.
- ▶ Dans le cas où le solde est négatif, on dit que le compte est à découvert. Le découvert d'un compte est nul si le solde du compte est positif ou nul, il est égal à la valeur absolue du solde si ce dernier est négatif. Un découvert par défaut est défini lors de l'ouverture du compte, mais peut évoluer en fonction des revenus du propriétaire du compte.
- ▶ Créditer un compte consiste à ajouter un montant positif au solde du compte.
- ▶ Débitier un compte consiste à retirer un montant positif au solde du compte. Le solde résultant ne doit en aucun cas être inférieur au découvert maximal autorisé pour ce compte. Lors d'une opération de retrait, un compte ne peut être débité d'un montant supérieur à une valeur (débit max autorisé). Un débit max par défaut est défini lors de l'ouverture du compte, mais peut évoluer en fonction des revenus du propriétaire du compte.
- ▶ Effectuer un virement consiste à débiter un compte au profit d'un autre compte qui sera crédité du montant du débit.
- ▶ Lors de la création d'un compte seule l'identité du titulaire du compte est indispensable. En l'absence de dépôt initial le solde est fixé à 0. Les valeurs par défaut pour le découvert max et le débit max sont respectivement de 800 € et 1000 €. Il est éventuellement possible d'attribuer d'autres valeurs à ces caractéristiques du compte lors de sa création.
- ▶ Toutes les informations concernant un compte peuvent être consultées : numéro du compte, identité du titulaire, montant du découvert maximal autorisé, montant du débit maximal autorisé, situation du compte (est-il à découvert ?), montant du débit autorisé (en fonction du solde courant et du débit maximal autorisé).

Exercice 2 : Facturation et garantie

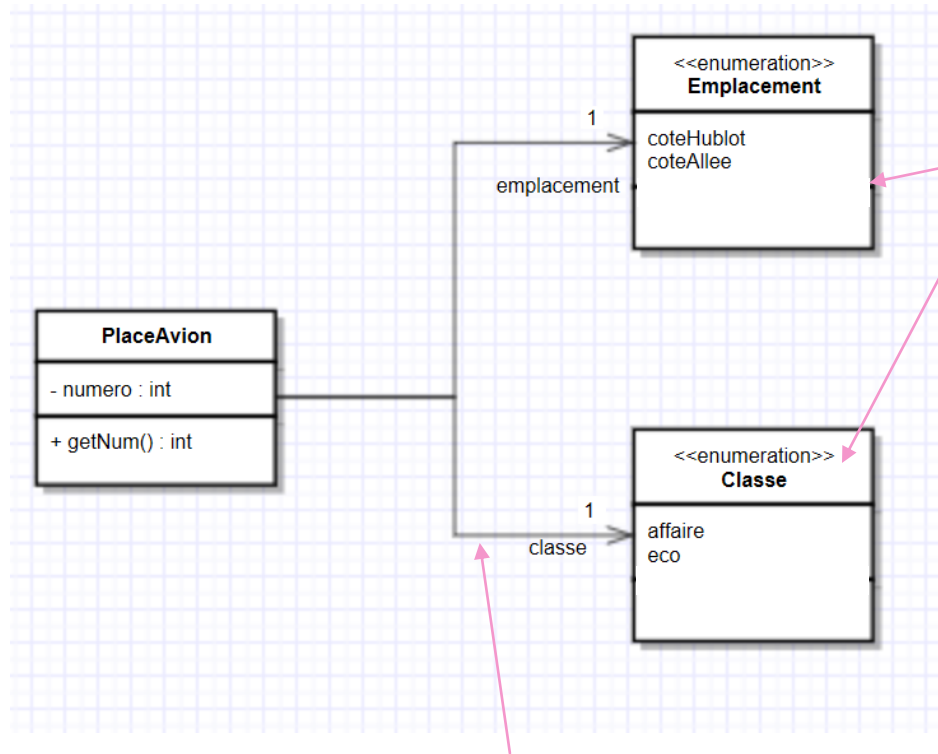
- ▶ Un véhicule est caractérisé par une marque, un modèle et une immatriculation.
- ▶ Une réparation est toujours relative à un véhicule. Elle est caractérisée par un numéro et une date de réception, une date de restitution du véhicule et une facture (chaîne de caractères)
- ▶ La facture est envoyée au propriétaire (qui est toujours un client) du véhicule ou à une compagnie d'assurance en cas d'accident. En cas de réparation prise en charge par la garantie, aucune facture n'est envoyée.
- ▶ Une compagnie d'assurance est un client pour le garage.
- ▶ Un client est caractérisé par un numéro, un nom, prénom, numéro de téléphone et une adresse.
- ▶ Pour bénéficier d'une réparation sous garantie, le client doit amener son véhicule à l'atelier avant l'expiration du délai de garantie.
- ▶ La garantie est de trois ans à partir de la date d'achat du véhicule. En fin de période de garantie, il se peut qu'il soit difficile de réaliser la réparation avant la date d'expiration. Pour résoudre ce dilemme et éviter toute réclamation, lorsqu'un client prend un rendez-vous pour effectuer une réparation en garantie, une fiche de réparation "garantie" est créée indiquant la date de la demande de rendez-vous du client, en plus des 2 dates de réception et restitution du véhicule pour la réparation.

Enumération

Comment modéliser un attribut de classes dont les valeurs possibles sont limitées?

- ▶ Pour modéliser la localisation d'une place assise dans un avion deux informations doivent être prises en compte
 - ▶ Emplacement : coté hublot ou coté allée
 - ▶ Classe : Affaire et Eco
- ▶ On va définir deux énumérations qui seront considérées comme deux types de données pour les attributs emplacement et confort.

Comment modéliser un attribut de classes dont les valeurs possibles sont limitées?



Représentée comme une classe, une énumération est reconnaissable par le stéréotype «enumeration». Elle ne contient que des attributs dont le nom correspond à chacune des valeurs possible.

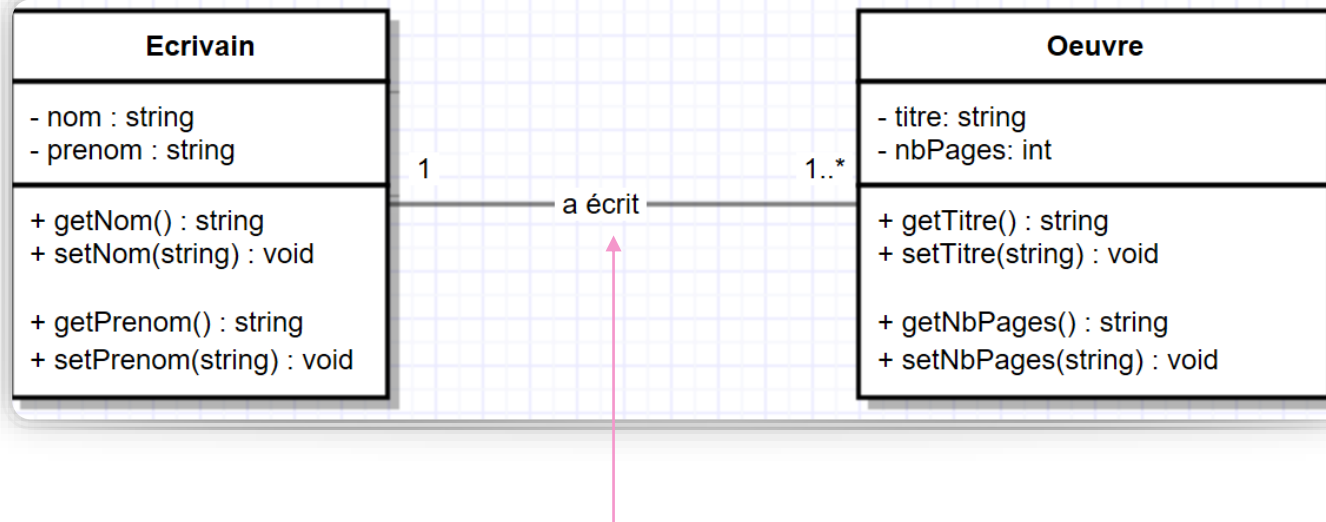
Reliée à la classe via une association orientée vers l'énumération qui peut être complétée par une multiplicité et un rôle.

Exercice 3 : Calendrier

- ▶ Une date est caractérisée par une
 - ▶ Journée : lundi, mardi, mercredi, jeudi, vendredi, samedi ou dimanche
 - ▶ Numéro (entier ≤ 31)
 - ▶ Mois : janvier, février, mars, avril, mai, juin, juillet, août, septembre, octobre, novembre, décembre.
 - ▶ Année (entier)
- ▶ Modéliser le concept de date.

Comment développer à partir
d'un diagramme UML?

L'association



Une association est une relation symétrique.

⇒ Un attribut de type `List<Œuvre>` est défini dans la classe **Ecrivain** représentant l'ensemble de ces œuvres.

⇒ Un attribut de type **Ecrivain** est défini dans la classe **Œuvre** représentant l'auteur de cette œuvre.

```

class Oeuvre
{
public:
    Oeuvre();
    ~Oeuvre();

    string getTitre();
    string getNbPages();
    Ecrivain getAuteur();

    void setTitre(string titre);
    void setNbPages(int nbPages);
    void setAuteur(Ecrivain auteur);

private :
    string m_titre;
    int m_nbPages;
    Ecrivain m_auteur;
};

```

Une œuvre a un auteur.

```

class Ecrivain
{
public:
    Ecrivain(string nom, string prenom);
    ~Ecrivain();

    string getNom();
    string getPrenom();
    vector<Oeuvre> getOeuvres();

    void setNom(string nom);
    void setPrenom(string prenom);

    void addOeuvre(Oeuvre oeuvre);

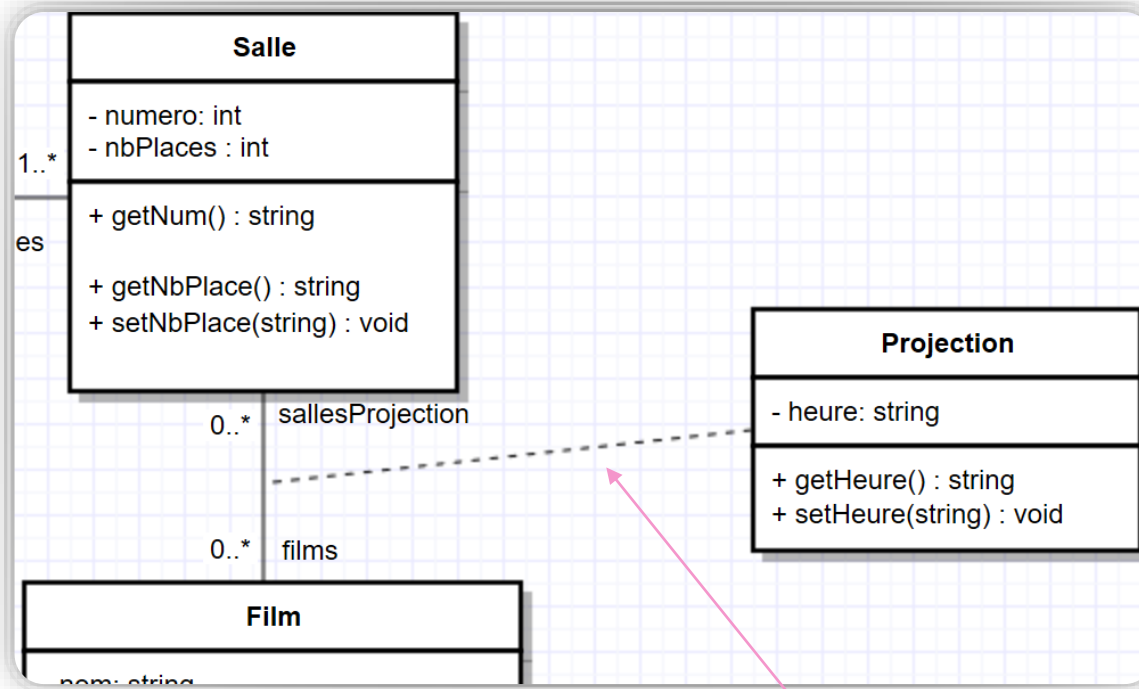
private :
    string m_nom;
    string m_prenom;
    vector<Oeuvre> m_listeOeuvres;
};

```

Un auteur a une liste d'œuvres.

L'association

La classe d'association



Une classe d'association apporte des informations supplémentaires sur la relation entre une Salle et un Film.

La classe d'association


```
class Projection
{
public:
    Projection(Film film, Salle salle, string date);
    ~Projection();

    Film getFilm();
    Salle getSalle();
    string getHeure();

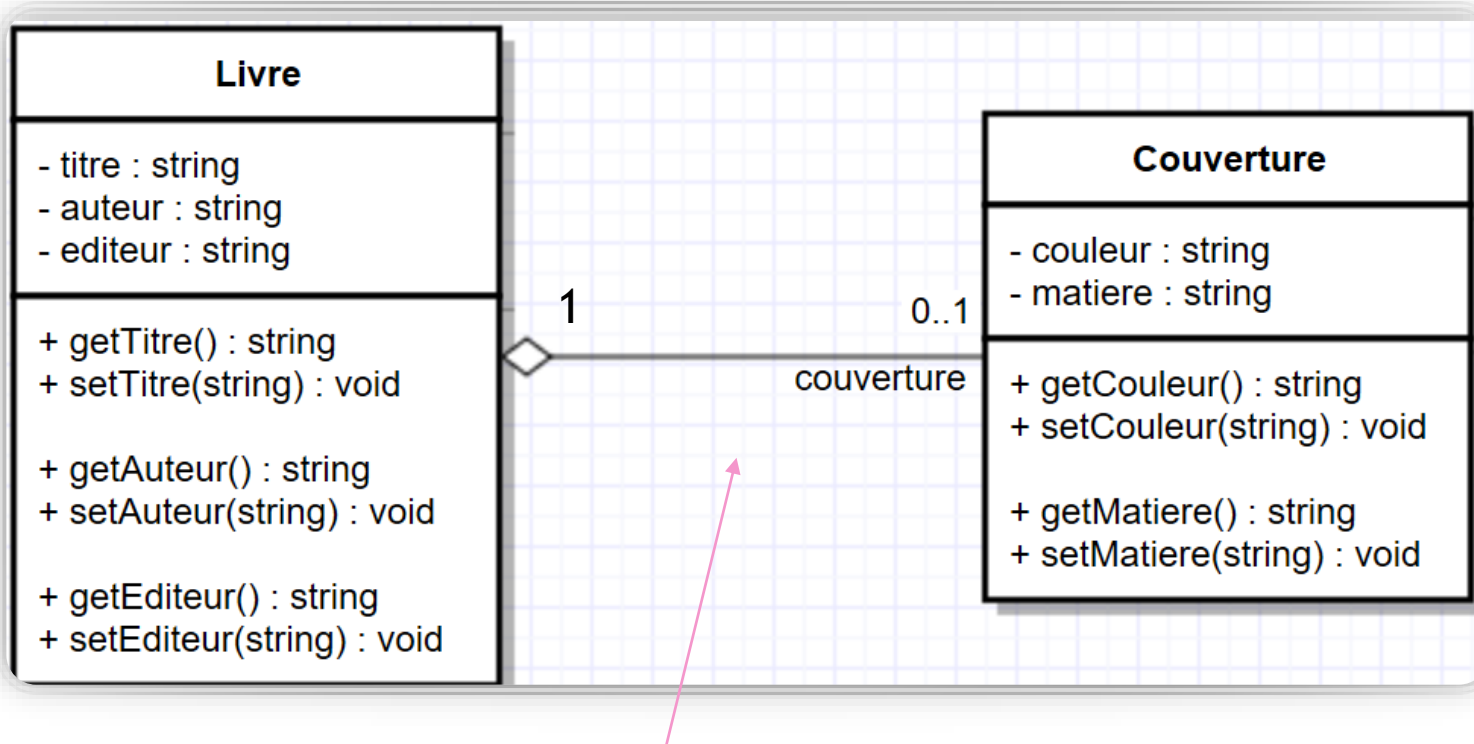
    void setHeure(string heure);

private :
    Film m_film;
    Salle m_salle;
    string m_heure;
};
```

Pour représenter cette relation, on ajoute un attribut de type Salle et un attribut de type Film.



L'agrégation



Attribut couverture à ajouter dans la classe
Livre

```
private Couverture couverture;
```


L'agrégation

Ajout des fonctionnalités permettant de manipuler cet attribut :

```
// Constructeur
public Livre(Couverture couverture)
{
    this.auteur = " ";
    this.editeur = " ";
    this.titre = " ";
    this.couverture = couverture;
}

// Méthodes de classe
public Couverture GetCouverture () { return this.couverture; }
public void SetCouverture(Couverture c) { this.couverture = c; }
```

Les cycles de vie n'étant pas liés, l'objet Couverture doit être instancié avant d'être associé à l'objet Livre.

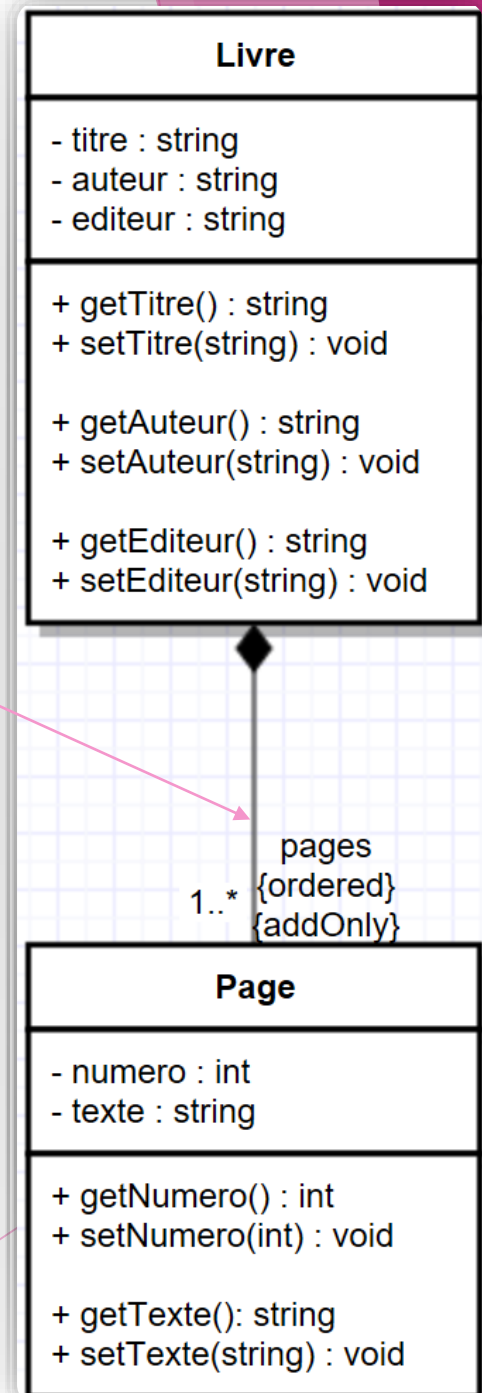
La composition

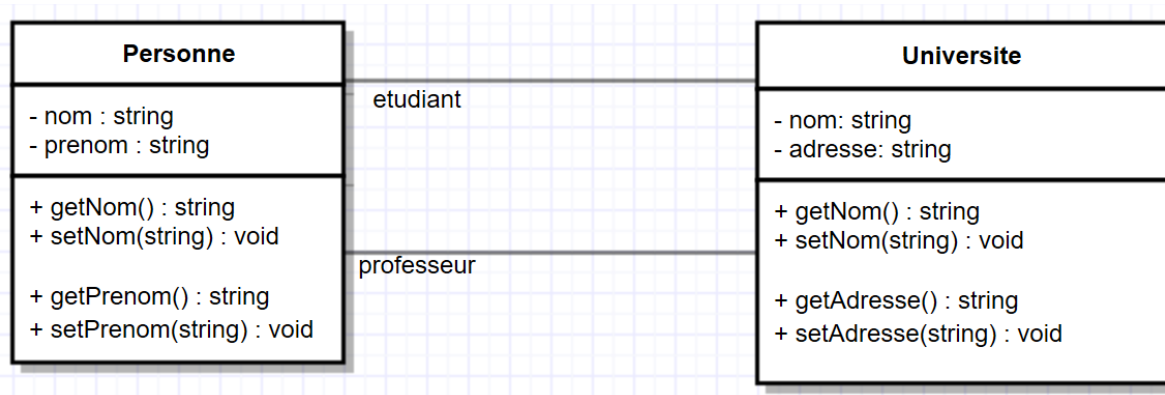
Attribut pages de type tableau de Page à ajouter dans la classe Livre.

Dans le constructeur de Livre :
Initialisation de la liste de pages
contenant un objet Page.

Ajout des fonctionnalités permettant de
manipuler cet attribut :

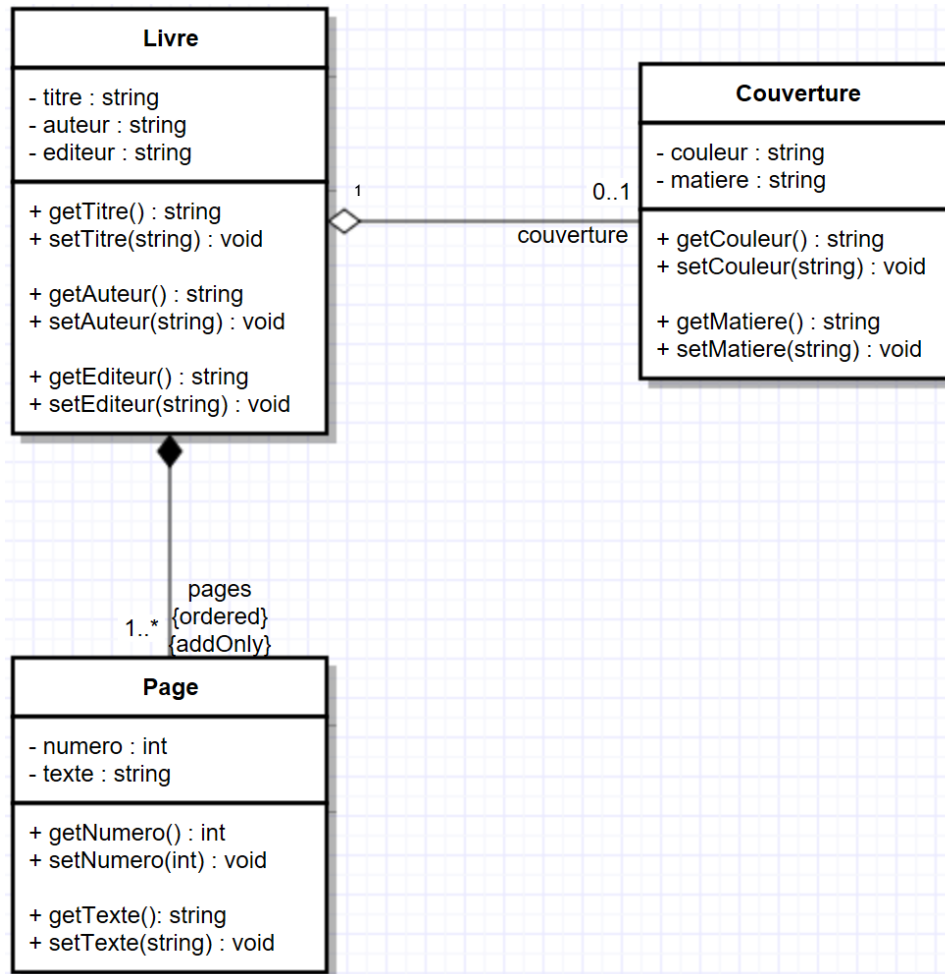
```
vector<Page> getPages();  
void addPage(string texte);
```





Exercice

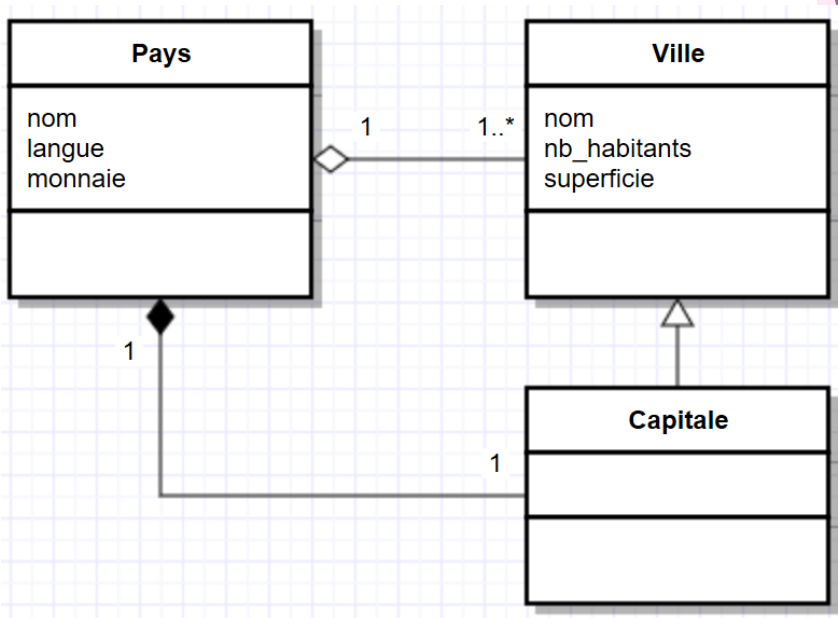
Développer le modèle ci-dessus.



Exercice

Développer le modèle ci-dessus.

Exercice



- Compléter le diagramme de classe avec les informations suivantes :
 - Une capitale comporte plusieurs monuments historiques et plusieurs institutions.
 - Une ville a une mairie (string) et une équipe municipale dont l'un des membres est le maire.
- Réaliser le programme correspondant.

