

# D'UML VERS C++

6

AGL – Chapitre

Mme. Lilia SFAXI  
Mme. Abir Gallas

# Génération de Code

- **Génération:** Production automatique d'artéfacts à partir d'un modèle
- *Exemple:* à partir d'un diagramme de classes, on peut générer:
  - ✓ Du code pour les classes du diagramme
  - ✓ Un document qui décrit les classes textuellement
  - ✓ Un rapport représentant des faits sur le modèle

# Rétro-Ingénierie

- Le contraire de la génération
- Analyse du code d'un programme pour produire des modèles qui représentent la même information de manière visuelle
- *Exemple:* faire de la rétro-ingénierie de classes C++ résulte en un diagramme de classes représentant les relations entre elles.

# Traduction UML-C++

- Diagramme de classes UML
  - ✓ Définit les composantes du système final
  - ✓ Ne définit pas le nombre et l'état des instances individuelles
- Un diagramme de classes proprement réalisé permet de:
  - ✓ Structurer le travail de développement de manière efficace
  - ✓ Séparer les composantes pour répartir le développement entre les membres d'un groupe
  - ✓ Construire le système de manière correcte

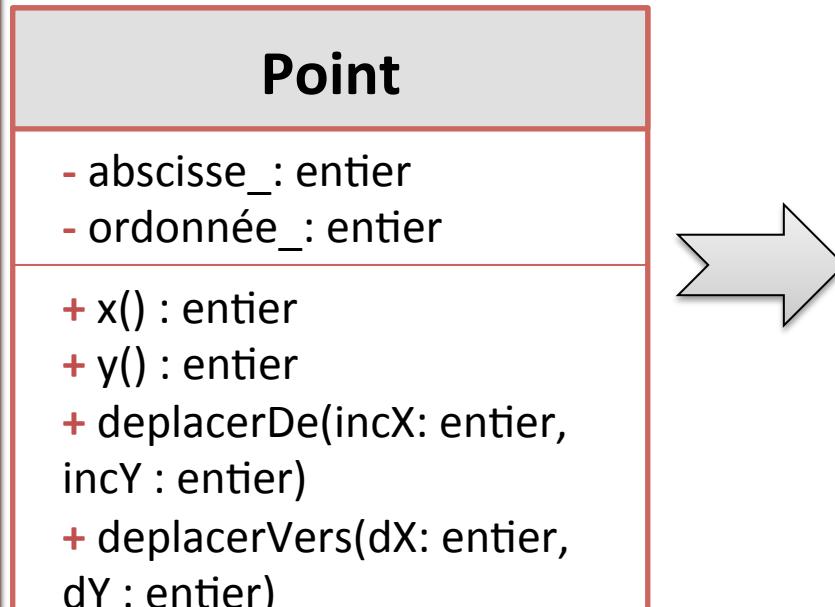
# D' UML vers C++ : Vocabulaire

- Le tableau suivant liste la conversion entre les éléments du modèle UML vers du code C++

Élément UML	Élément C++
Packetage	Répertoire / Namespace
Classe	Classe (fichiers .h et .cpp)
Relation de Généralisation	Héritage
Relation d'association/ agrégation/composition	Attribut
Attribut	Attribut
Méthode	Opération
Paramètre	Argument d'Opération

# Classes, Attributs et Méthodes

- Représentation d'une classe, avec des attributs et méthodes publics et privés.

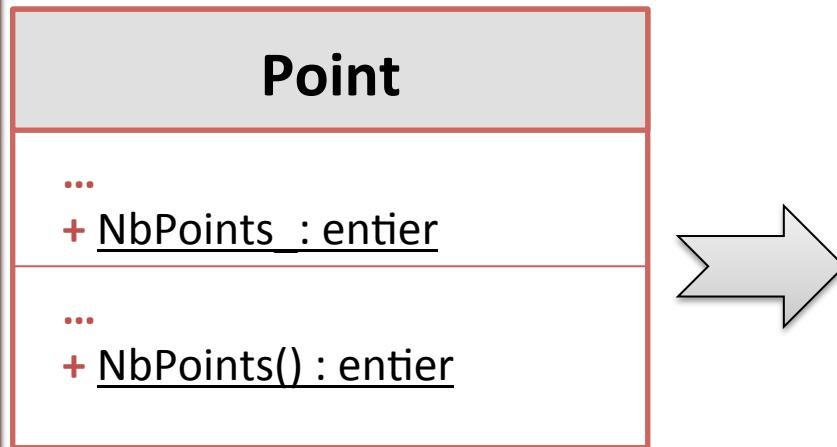


```
class Point{  
public:  
    Point( int abs, int ord)  
    int x(void) const;  
    int y(void) const;  
    void déplacerDe (int incX,  
                    int incY);  
    void déplacerVers (int dx,  
                      int dY);  
  
private:  
    int abscisse_;  
    int ordonnee_;  
};
```

# Attributs et Méthodes

## « de Classe »

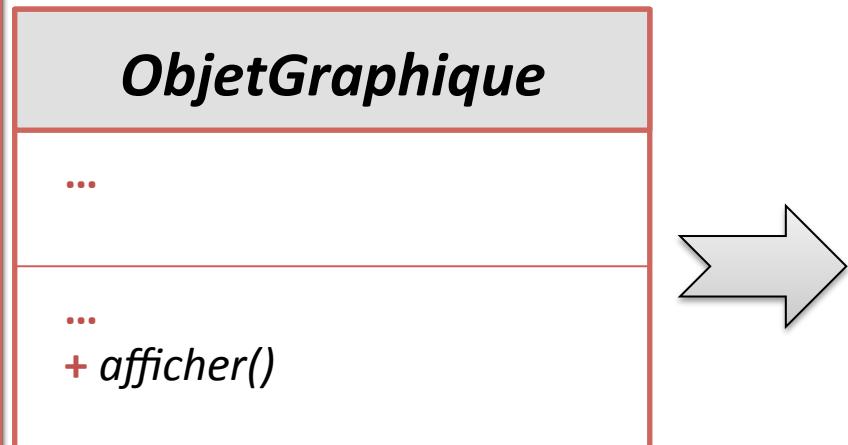
- Attribut/Méthode de classe:
  - ✓ Commun à tous les objets créés
  - ✓ Exemple: compteur du nombre d'instances créées.



```
class Point{  
public:  
    static int NbPoints(void);  
private:  
    static int NbPoints_;  
};
```

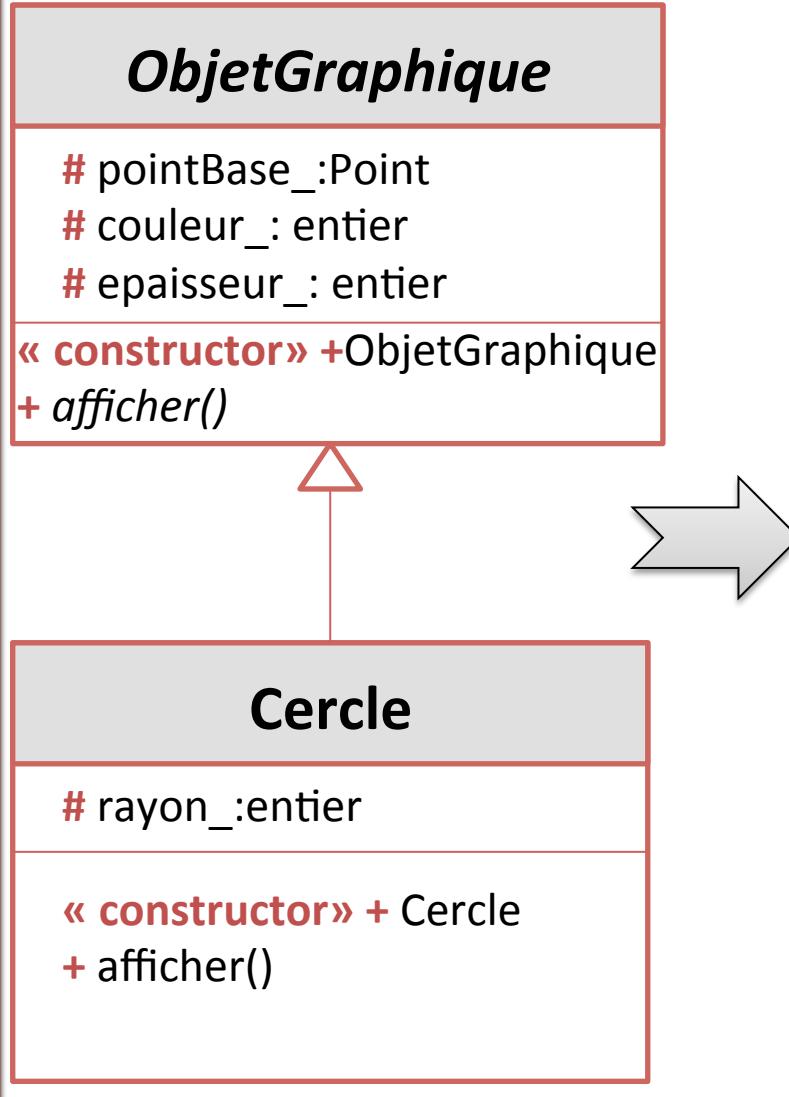
# Classes et Méthodes Abstraites

- Méthode Abstraite
  - ✓ Méthode déclarée sans d'implémentation
- Classe Abstraite
  - ✓ Classe contenant au moins une méthode abstraite

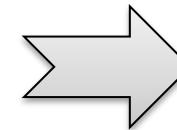
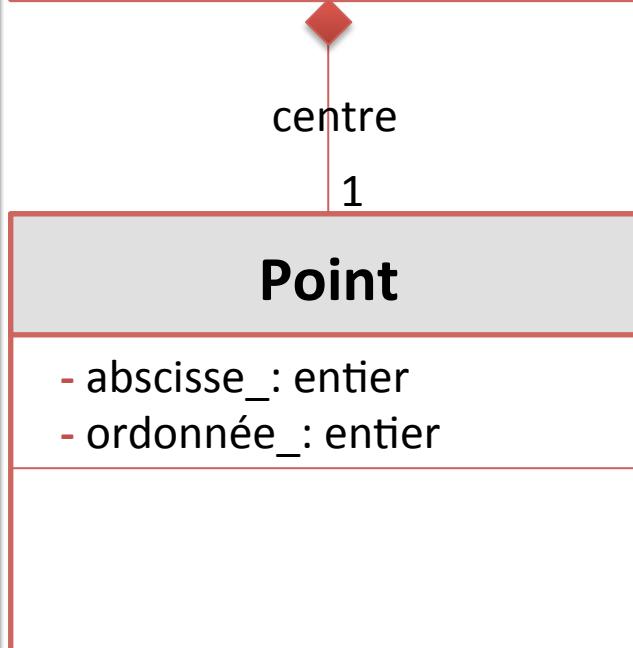
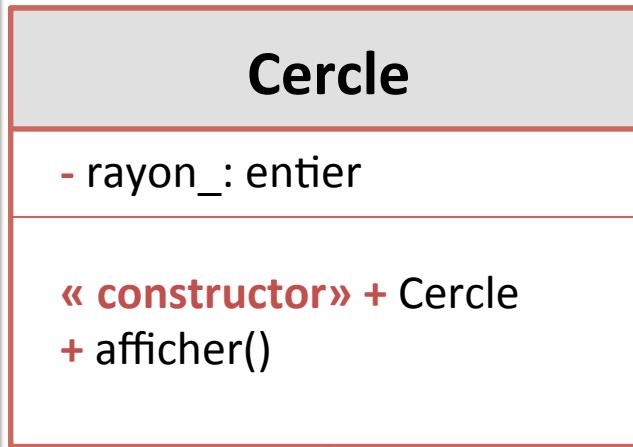


```
class ObjetGraphique{
public:
    virtual void afficher(void) = 0;
    ...
};
```

# Héritage

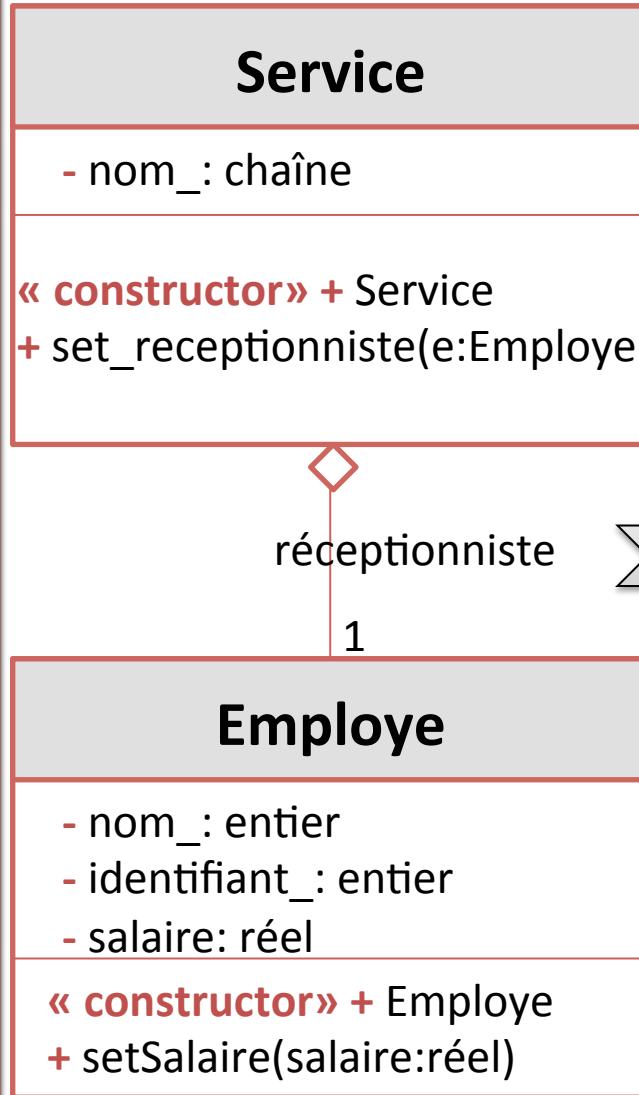


# Composition

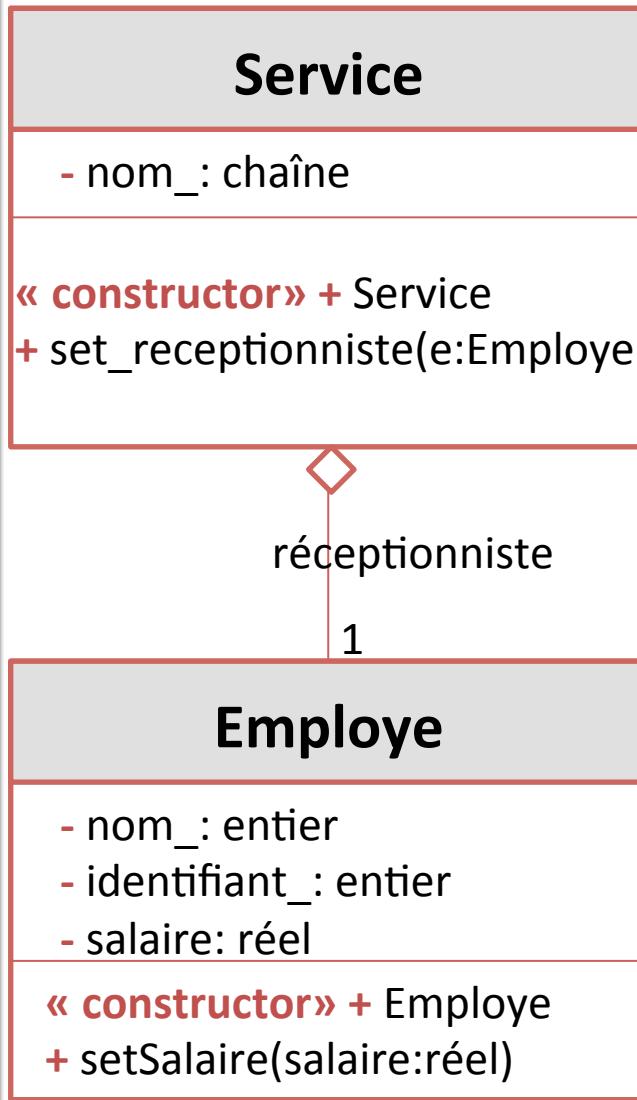


```
class Cercle{  
public:  
    Cercle(...);  
private:  
    Point centre_;  
    int rayon_;  
};  
//constructeur  
Cercle::Cercle (int x, int y, int rayon)  
: centre_(x,y), rayon_(rayon)  
{}
```

# Agrégation (par pointeur)



# Agrégation (par référence)

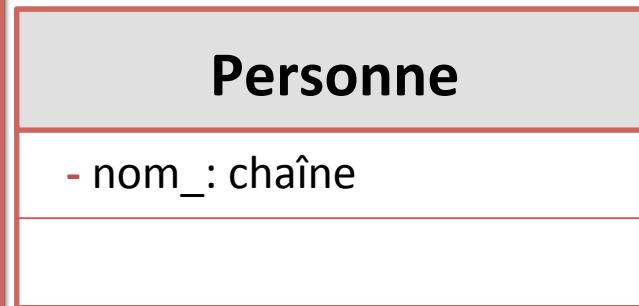


```
class Service{
    private:
        string nom_;
        Employe& receptionniste_;
    };
    //constructeur
    Service::Service (string nom,
                      Employe& employe)
    : nom_(nom),
      receptionniste_(employe)
    {}
    //fonction principale
    int main()
    {
        Employe employe(" Michel", 12568);
        Service expedition("Expéditions",
                           employe);
        employe.set_salaire(500);
    }
}
```

**Doit être initialisé**

# Associations

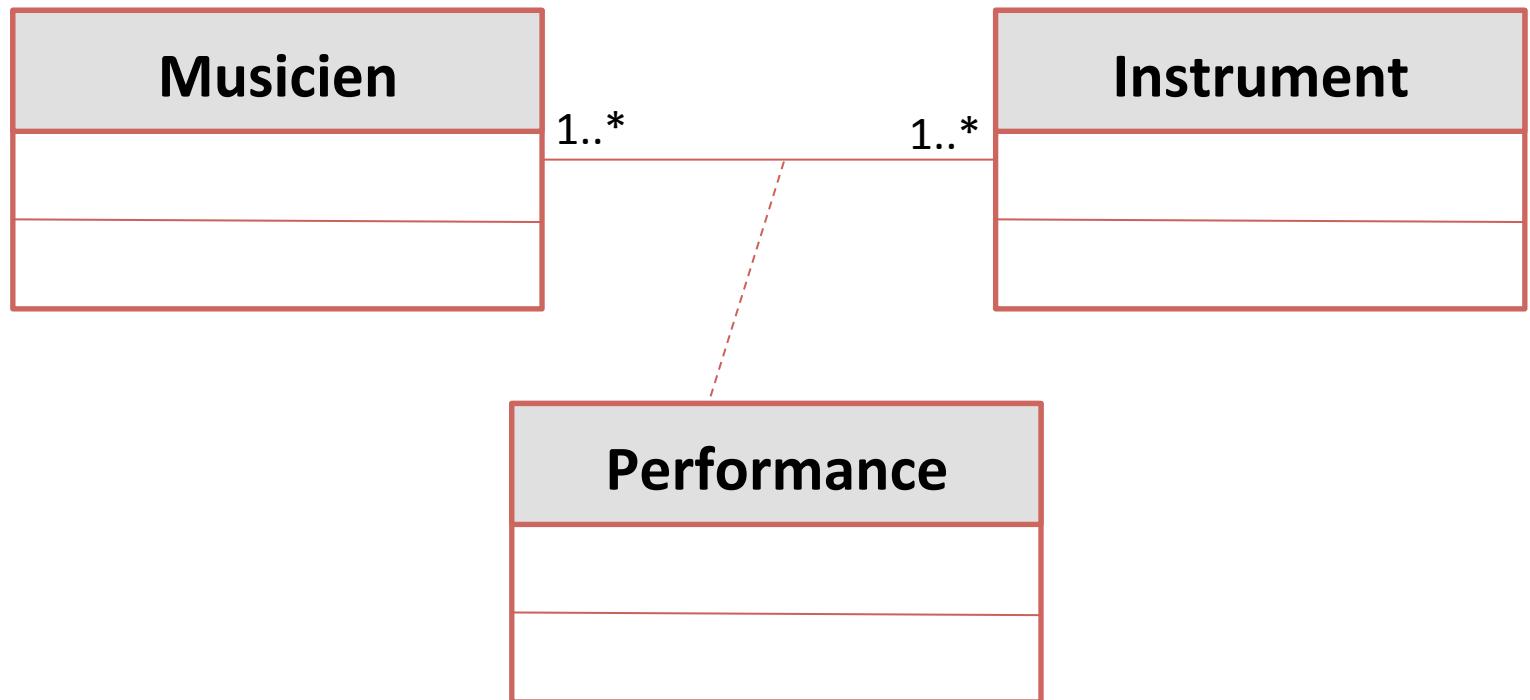
- Les associations, comme les agrégations, sont représentées par des attributs
  - ✓ La différence se voit essentiellement dans le sens



```
class Personne{
    private:
        string nom_;
        Liste<Voiture>* voitures_;
};
```

```
class Voiture{
    private:
        string matricule_;
        string marque_;
        Personne* proprietaire_;
};
```

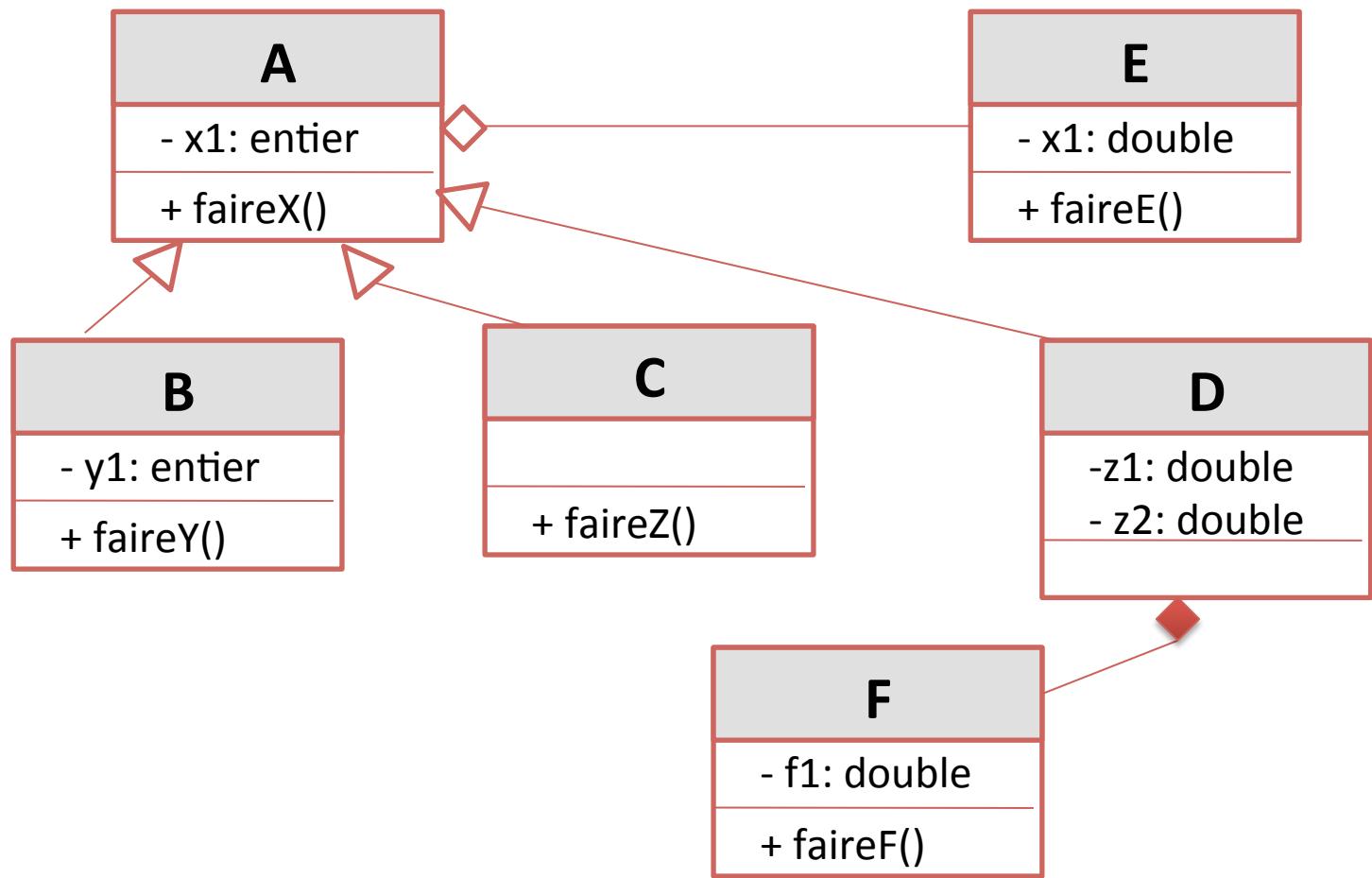
# Classe d'Association



```
class Performance{
    private:
        Musicien unMusicien;
        Instrument unInstrument;
};
```

# Exercice 1 : Forward Engineering

- Réaliser en C++ le squelette du programme qui accompagne ce diagramme



# Exercice 2 : Reverse Engineering

- Représenter le diagramme de classes UML correspondant au squelette de code suivant:

```
class Department {  
    private : char* name_p;  
    public:  
        Department (char *dName) {  
            name_p = new char(sizeof(strlen(dName)));  
            name_p = dName;  
        }  
        char* dName();  
};  
  
class Student {  
    private : char* name_p;  
    public:  
        Student (char *sName) {  
            name_p = new char (sizeof(strlen(sName)));  
            name_p = sName;  
        }  
        char* sName();  
};
```

```
class Course {  
    private:  
        Student * std_p;  
        Department * dept_p;  
        char * courseName_p;  
        static int index;  
        static Course *courseList[4];  
    public:  
        Course (char* crseName, Student* student,  
                Department* dept):  
            courseName_p(0), std_p(student),  
            dept_p(dept) { };  
        static char* findStudent (char *crseName,  
                                char* deptName);  
        char * getStdName() ;  
        char * getDeptName{};  
        char * getCourseName();  
};
```

# Exercice 2: Correction

**Course class Associates Student and Department classes**

