Aggregation & Composition modelisation and implementation

The UML (Unified Modeling Language) is a standard for modeling Object Oriented systems. In UML there are different types of relationships: Association, Aggregation, Composition, Dependency, and Inheritance.

In this post i will try to treat both: Aggregation & Composition and their implementation in C++.

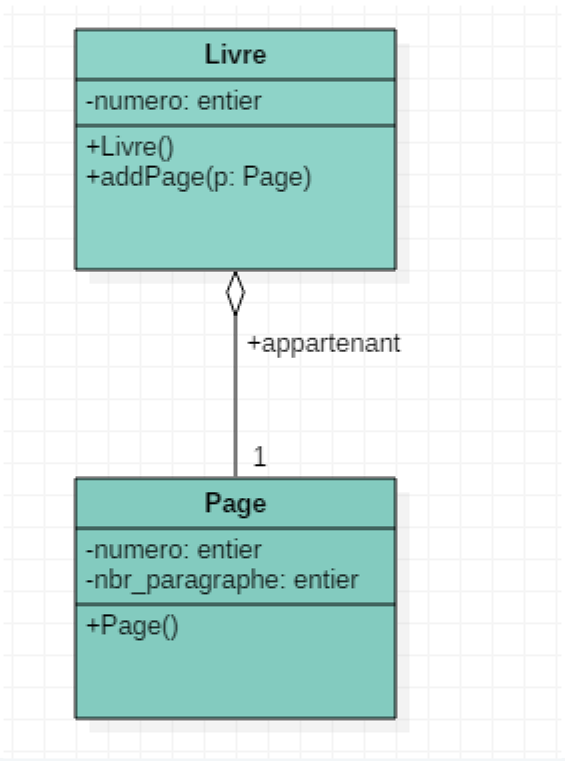
Association is a relationship between two objects. In other words, association defines the multiplicity between objects.

Aggregation:

The Aggregation is a special case of association. A directional association between objects. When an object "**has-a"** another object, then you have got an aggregation between them. Direction between them specified which object contains the other object. Aggregation is also called a "**Has-a"** relationship.

In C++ we can implement Aggregation by two different ways and the choice depends on the functional situation:

Using Pointers:



```
Class Livre{

      private:
             int numero;
             Page *appartenant_;
 };
 //------------------Constructeur --------------------------
 Livre::Livre(int num) : numero(num),appartenant_(NULL) // Can be NULL
 {}

 //------------------AddPage ------------------------------
 Livre::addPage(Page *p)
 {
       appartenant_ = p;
 }

 //------------------Fonction principale ------------------
 int main()
 {
       Livre monLivre(123456); // livre monLivre avec son
numéro :123456
       Page *p = new Page(12,8) // page numéro 12 contenant 8
paragraphes
       monLivre.addPage(p);
 }
```

2. Using References:

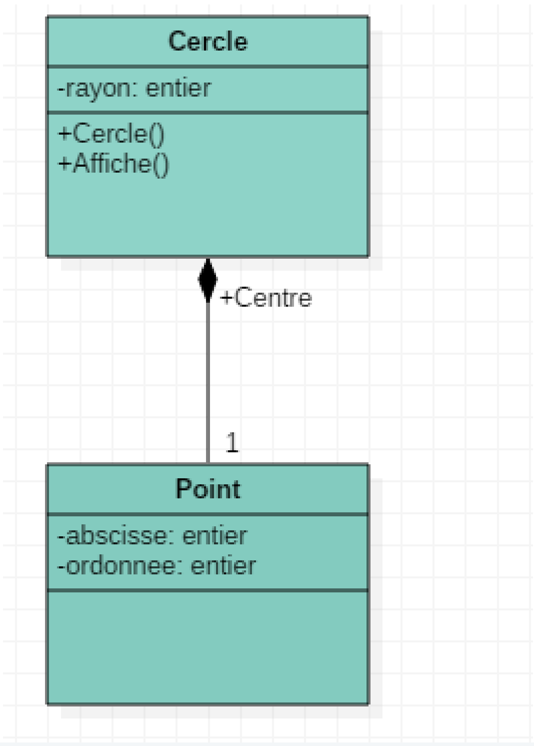
```
Class Livre{

       private:
              int numero;
              Page &appartenant_;
 };
 //------------------Constructeur --------------------------
 Livre::Livre(int num,Page &p): numero(num),appartenant_(p)) //can't be
NULL
 {}

 //------------------Fonction principale -------------------
 int main()
 {
       Page p(12,8) // page p avec numéro 12 contenant 8
paragraphes
       Livre monLivre(123456,p); // livre monLivre avec son
numéro :123456 et la page p
 }
```

Composition:

The composition is a special case of aggregation. In a more specific manner, a restricted aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition.



```
Class Cercle{

        private:
                int rayon;
                Point centre_;

        public:
                Cercle(...);
                affiche();

};
//------------------Constructeur ---------------------------
--------
Cercle::Cercle(int x, int y, int rayon_):centre_(x,y),rayon(rayon_)
{}
```