C++ Aggregation (HAS-A Relationship)

In C++, aggregation is a process in which one class defines another class as any entity reference. It is another way to reuse the class. It is a form of association that represents HAS-A relationship.

C++ Aggregation Example

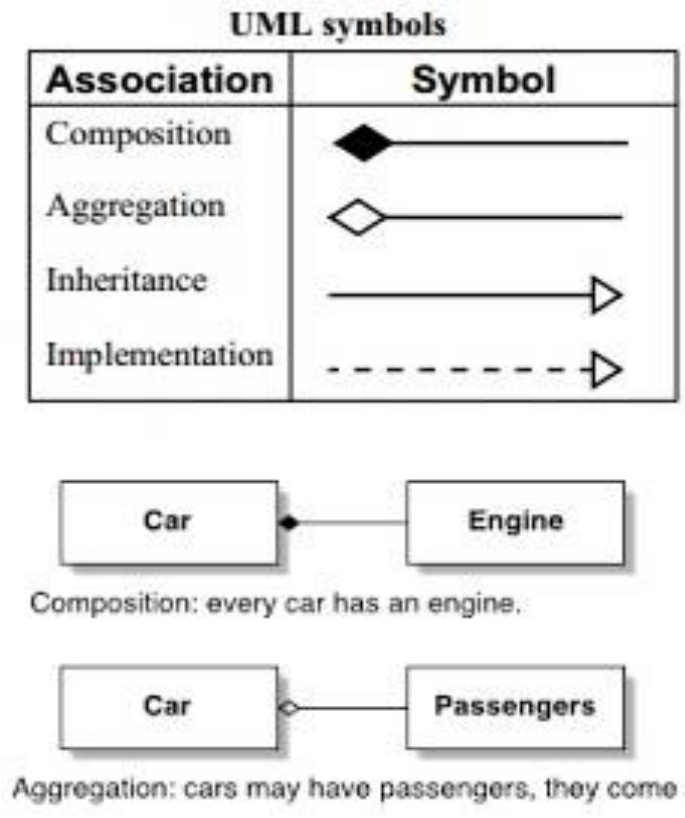
Let's see an example of aggregation where Employee class has the reference of Address class as data member. In such way, it can reuse the members of Address class.

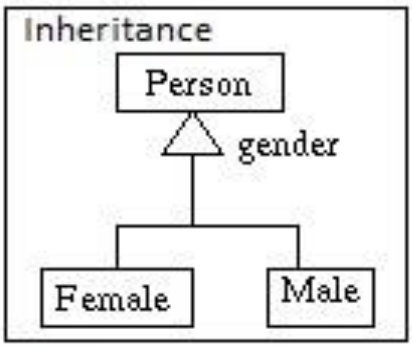
```
1. #include <iostream>
2. using namespace std;
3. class Address {
4.     public:
5.    string addressLine, city, state;
6.       Address(string addressLine, string city, string state)
7.     {
8.         this->addressLine = addressLine;
9.         this->city = city;
10.        this->state = state;
11.    }
12. };
13. class Employee
14.    {
15.        private:
16.        Address* address;  //Employee HAS-A Address
17.        public:
18.        int id;
19.        string name;
20.        Employee(int id, string name, Address* address)
21.       {
22.           this->id = id;
23.           this->name = name;
```

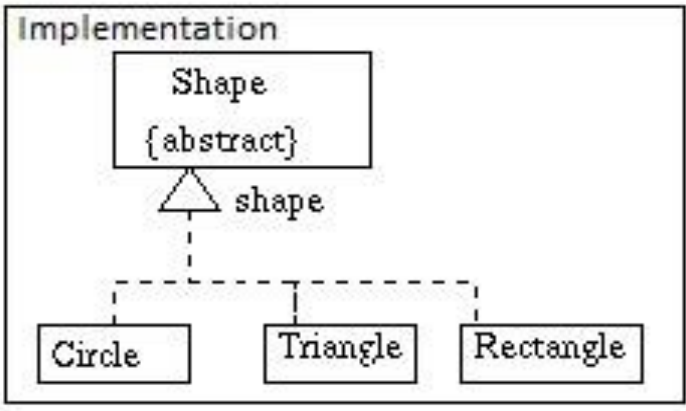
```
24.          this->address = address;
25.      }
26.     void display()
27.      {
28.          cout<<id <<" "<<name<< " "<<
29.            address->addressLine<< " "<< address->city<< " "<<address->state<<endl;
30.      }
31.  };
32. int main(void) {
33.    Address a1= Address("C-146, Sec-15","Noida","UP");
34.    Employee e1 = Employee(101,"Nakul",&a1);
35.           e1.display();
36.   return 0;
37. }
```

Output:

```
101 Nakul C-146, Sec-15 Noida UP
```







Association:-

Association is a relationship between two objects. It's denoted by "**has-a**" relationship. In this relationship all objects have their own lifecycle and there is no owner. Let's take an example of Teacher and Student. Multiple students can associate with single teacher and single student can associate with multiple teachers, but there is no ownership between the objects and both have their own lifecycle. Both can create and delete independently.

Aggregation:-

Aggregation is a another type of specialised form of Association where all objects have their own lifecycle, but there is ownership and child objects can not belong to another parent object. Let's take an example of Department and teacher. A single teacher can not belong to multiple departments, but if we delete the department teacher object will *not* be destroyed. We can think about it as a "has-a" relationship.

Composition:-

Composition is again specialised form of Aggregation and we can call this as a "death" relationship. *It is a strong type of Aggregation*. Child object does not have its lifecycle and if parent object is deleted, all child objects will also be deleted. Let's take again an example of relationship between House and Rooms. House can contain multiple rooms – there is no independent life of room and any room can not belong to two different houses. If we delete the house – room will automatically be deleted. Let's take another example relationship between Questions and Options. Single questions can have multiple options and option can not belong to multiple questions. If we delete questions options will automatically be deleted. Let's take another example of relationship between Car and Moter, Moter is a vital part of Car, if you remove moter then Car become useless.

Hence, we can take lots of example of composition in real world that represent a strong relationship classes .

Inheritance :-

Inheritance is an another type of relationship and it's denoted by **"is-a"** relationship. Inheritance is the inclusion of behaviour (i.e. methods) and state (i.e. variables) of a base class in a derived. for example- Circle **is a** shape, Rectangle **is a** Shape.