Apart from programming, a lot of my spare time sat at the computer is spent reading group, blog postings, etc from other developers. One particular posting that caught my eye recently provoked a lot of response and mixed answers to a question posed by a poster. This question was, 'What is the difference between composition and aggregation and how would I express it in my programs?'

Phil Curnow Sep 29 2012

8 23 434.6k

Introduction:

Apart from programming, a lot of my spare time sat at the computer is spent reading group, blog postings, etc from other developers. One particular posting that caught my eye recently provoked a lot of response and mixed answers to a question posed by a poster. This question was, 'What is the difference between composition and aggregation and how would I express it in my programs'?

Reading the responses to the post, I had a mixed reaction, many of the responses reflected my understanding of the difference, others turned my understanding right around and explained composition as my understanding of aggregation.

This short article will put forward my understanding of composition and aggregation and how I would express it in C# code.

Composition:

As we know, inheritance gives us an 'is-a' relationship. To make the understanding of composition easier, we can say that composition gives us a 'part-of' relationship. Composition is shown on a UML diagram as a filled diamond (see Figure 1).

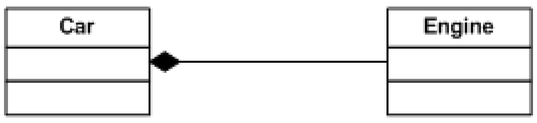


Figure 1 - Composition

If we were going to model a car, it would make sense to say that an engine is part-of a car. Within composition, the lifetime of the part (Engine) is managed by the whole (Car), in other words, when Car is destroyed, Engine is destroyed along with it. So how do we express this in C#?

```
public class Engine
{
```

```
{
   Engine e = new Engine();
   .......
}
```

As you can see in the example code above, Car manages the lifetime of Engine.

Aggregation:

If inheritance gives us 'is-a' and composition gives us 'part-of', we could argue that aggregation gives us a 'has-a' relationship. Within aggregation, the lifetime of the part is not managed by the whole. To make this clearer, we need an example. For the past 12+ months I have been involved with the implementation of a CRM system, so I am going to use part of this as an example.

The CRM system has a database of customers and a separate database that holds all addresses within a geographic area. Aggregation would make sense in this situation, as a Customer 'has-a' Address. It wouldn't make sense to say that an Address is 'part-of' the Customer, because it isn't. Consider it this way, if the customer ceases to exist, does the address? I would argue that it does not cease to exist. Aggregation is shown on a UML diagram as an unfilled diamond (see Figure 2).

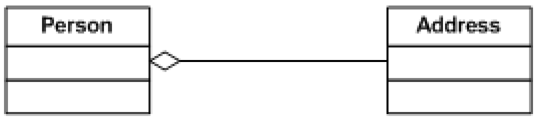


Figure 2 - Aggregation

So how do we express the concept of aggregation in C#? Well, it's a little different to composition. Consider the following code:

```
public class Address
{
 . . .
}

public class Person
{
    private Address address;
    public Person(Address address)
    {
       this.address = address;
    }
    . . .
}
```

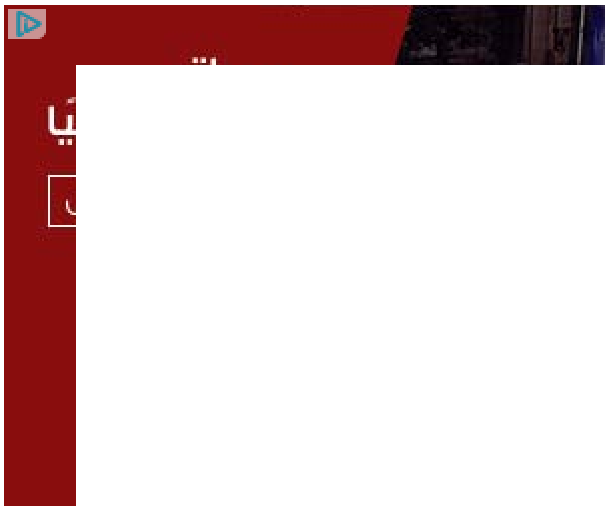
Person would then be used as follows:

```
Address address = new Address();
Person person = new Person(address);
```

or

Conclusion:

As I said at the beginning of the article, this is my take on composition and aggregation. Making the decision on whether to use composition or aggregation should not be a tricky. When object modelling, it should be a matter of saying is this 'part-of' or does it 'have-a'?



aggregation | aggregation relationship | composition | CRM system

Phil Curnow
I work as a consultant in the UK with a large software and services company. I have been in the IT industry as a consultant and developer for over 20 years. I have worked with many programming languages over the years bu... Read more

http://www.curnow.biz

1098 1m

View Previous Comments
8 23

Type your comment here and press Enter Key (Minimum 10 characters)

Is is possible to also that the constructor method of Person, doesn't have Address as Parameter, but (some way) there is still an aggregation relation? So another way to do it? Asking for a particular project...
Joren Wouters Oct 20, 2017
1748 2 0 1 0 Reply

You did a great job. Simply explained a very complex concept to me. Bravo
Zahid Mehmood Aug 17, 2017
1748 2 0 1 0 Reply