Object Oriented Programming

Lab Manual 06

Introduction

After a week of rigorous coding, Welcome back!

You have learned all about the Classes, Constructors, and member functions in the previous labs. Let's move on to the next, new, and interesting concept like Association between the classes.

What is Association?

If you refer back to your previous lab, we encountered an LMS problem wherein students are required to enroll in one degree program. In this code, we create or manage objects of other classes within the `Student` class. This concept is called association.
In simple words, the relationship between classes is called Association.

```
13 references
internal class Student
{
    public string Name;
    public int Age;
    public float EcatMarks;
    public float FscMarks;
    public float Merit;
    public List<DegreeProgram> PrefrencesList = new List<DegreeProgram>();
    public DegreeProgram EnrolledDegree;
    public List<Subject> RegisteredSubjectsList = new List<Subject>();


}
```

What is Aggregation? (Has-a Relationship)

When two objects have independent lifetime but one object has another object such type of association is called the Aggregation.
This is a "has-a" relationship, where one class (the whole) has a part-of relationship with another class (the part). The lifetime of the part is dependent on the lifetime of the whole. In simpler terms, if the whole object is destroyed, the part object is also destroyed.
Example: A car **has-a** engine. The car is the whole object, and the engine is the part. If the car is crushed in a junkyard and the engine stays.

```
5 references
public class Engine
{
    public string Type;
    1 reference
    public Engine(string type)
    {
        Type = type;
    }
}

3 references
public class Car
{
    public Engine Engine; // Aggregation relationship
    1 reference
    public Car(Engine engine)
    {
        Engine = engine;
    }

    1 reference
    public void Crush()
    {
        Console.WriteLine("Car crushed, including the engine ({0})", Engine.Type);
    }
}

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Engine engine = new Engine("V8");
        Car car = new Car(engine);
        car.Crush();
    }
}
```

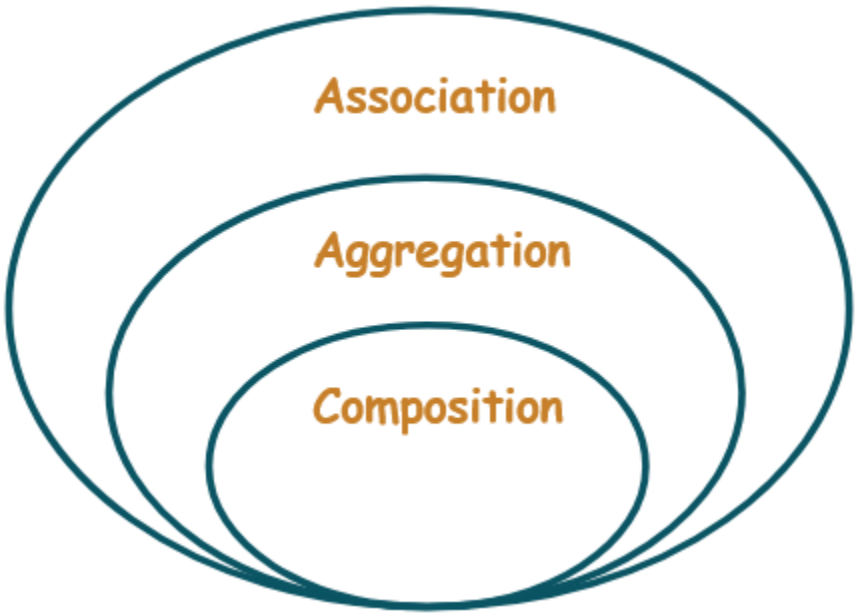
What is Composition? (Has-a Relationship)

When the time of an object depends on the lifetime of another object this type of Association is called Composition.

Example: While developing your writing application, you need to create a way to represent documents and their content. You decide to model a `Document` class that encapsulates the document's title and a list of `Paragraph` objects. Each paragraph holds its content as text.

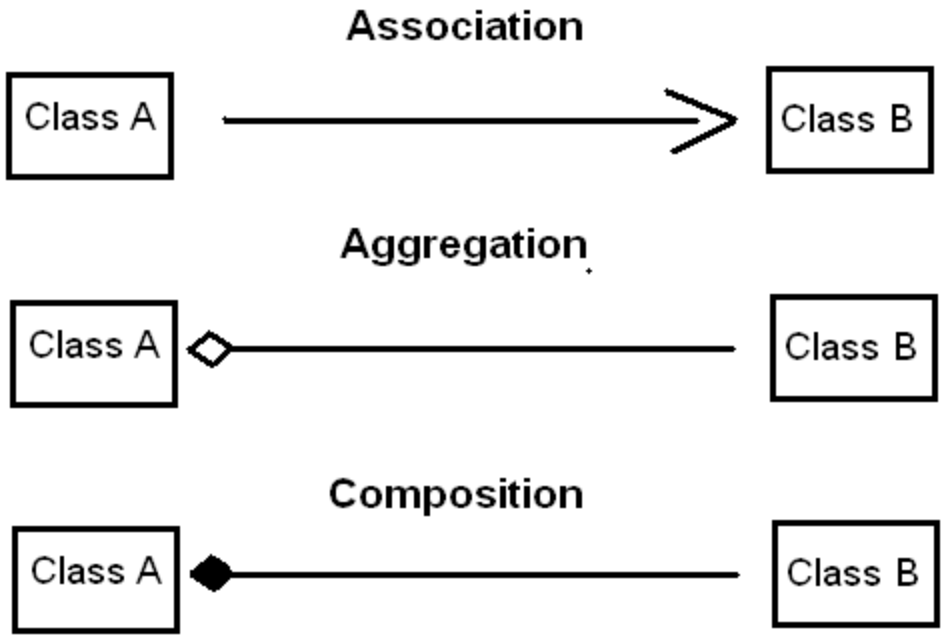
```
3 references
public class Document
{
    2 references
    private string Title
    3 references
    private List<Paragraph> Paragraphs
    1 reference
    public Document(string title)
    {
        Title = title;
        Paragraphs = new List<Paragraph>();
    }

    2 references
    public void AddParagraph(string content)
    {
        Paragraphs.Add(new Paragraph(content));
    }

    1 reference
    public void Print()
    {
        Console.WriteLine($"Document: {Title}");
        foreach (Paragraph paragraph in Paragraphs)
        {
            paragraph.Print();
        }
    }
}
```

```
5 references
public class Paragraph
{
    2 references
    private string Content
    1 reference
    public Paragraph(string content)
    {
        Content = content;
    }

    1 reference
    public void Print()
    {
        Console.WriteLine(Content);
    }
}

0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Document document = new Document("My Document");
        document.AddParagraph("This is the first paragraph.");
        document.AddParagraph("This is the second paragraph.");
        document.Print();
    }
}
```



Graphical representation of the relationships

Graphical representation of the relationships



Challenging Questions

Q1. Imagine you're tasked with developing a program for a library management system. The system needs to track books and their respective authors. However, before diving into the code, can you first determine the type of association between books and authors? Once identified, please proceed to write the code for implementing this association in your program.

Q2. Imagine you're creating a program to manage a music streaming service. One key feature is to organize playlists, which can contain multiple songs. Before proceeding to code, can you define the association between a playlist and its songs? Once you've determined this relationship, go ahead and implement it in your program.

Q3. Let's design a program for managing a fitness app. Users can create workout routines consisting of multiple exercises, and each exercise can have multiple sets with different repetitions and weights. Before coding, can you identify how composition and aggregation concepts apply here? Define the composition relationship between a workout routine and its exercises, as well as the aggregation relationship between exercises and sets. Once clarified, proceed to implement these relationships in your program.

Q4. Consider a social media platform where users can create posts, and each post can have multiple comments and likes. Additionally, users can follow other users, forming a network of followers and followers. Before proceeding to code, can you identify the composition and aggregation relationships within this platform? Define how posts are composed of comments and likes, and how users aggregate posts, comments, and likes. Also, consider the aggregation relationship between users and their followers/followers. Once you've clarified these relationships, proceed to implement them in your program, taking into account the complexities of social interactions and user networks on a social media platform