## Structural pattern

**Definition:**

According to (Making, 2021), these design patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality.

* **Adapter:** Match interfaces of different classes.
* **Bridge:** Separates an object’s interface from its implementation.
* **Composite:** A tree structure of simple and composite objects.
* **Decorator:** Add responsibilities to objects dynamically.
* **Façade:** A single class that represents an entire subsystem.
* **Flyweight:** A fine-grained instance used for efficient sharing.
* **Private Class Data:** Restricts accessor/mutator access.
* **Proxy:** An object representing another object.
* Composite design pattern:
* Definition: The composite pattern is a partitioning design pattern that defines a collection of items that are processed as if they were a single instance of the same type of object. A composite's purpose is to "compose" things into tree structures in order to express part-whole hierarchies. It allows you to create a tree structure and assign tasks to each node in the tree structure (*GeeksforGeeks* 2017).

According to *Dofactory* (2021), a composite pattern consists of the following basic components:

Component

* Specifies the interface for the composition's objects.
* As applicable, implements default behavior for the interface common to all classes.
* Declares an access and management interface for its child components.
* (optional) specifies and implements an interface for accessing a component's parent in the recursive structure, if applicable.

Leaf

* Represents the composition's leaf objects. There are no children for a leaf.
* Determines the behavior of the composition's primitive items.

Composite

* Describes the behavior of components that have children.
* Child components are stored here.
* In the Component interface, it implements child-related operations.

Client

* Manipulates objects in the composition through the Component interface.

**When to use the Adapter Design Pattern:**

* When we want to create objects in tree structures to represent the classification system.
* When we want the client to be able to ignore the difference between the object constructors and the object itself. Clients affect each object and its components uniformly.

**Basic Structure:**

**Diagram

Description automatically generated**

**Figure 1: Basic Composite design pattern**

**Scenario:**

A product management system. Customer can add and delete products to buy and the system will summarize and give details of additional products, prices and total prices of all products. The Composite pattern will be used to specifically divide the work and properties of each component and object in the system.

**UML class diagram:**

**Diagram

Description automatically generated**

**UML explain:**

There are four classes in my class diagram. The relationship between Client and ProductProgram is an association relationship. SpecificProduct and CompositeProduct inherit ProductProgram. Besides, it’s an aggregation relationship between ProductProgram and CompositeProduct.

* Client: Customer is client in this UML.
* Component: Class ProductProgram is Component. It contains activities that other classes need to perform.
* Leaf: SpecificProduct is Leaf and it can only do already task(s) without making more functions, …
* Composite: is CompositeProduct with inheriting from Component and it can make more activities, …

# Bibliography

Making, S., 2021. *Design Patterns.* [Online]   
Available at: https://sourcemaking.com/design\_patterns  
[Accessed 20 June 2021].

Naik, K., 2021. *C-sharpcorner.* [Online]   
Available at: https://www.c-sharpcorner.com/UploadFile/bd5be5/design-patterns-in-net/  
[Accessed 22 6 2021].

*Composite Design Pattern - GeeksforGeeks* (2017). Available at: https://www.geeksforgeeks.org/composite-design-pattern/ (Accessed: 22 June 2021).

*Composite Design Pattern in C# .NET - Dofactory* (2021). Available at: https://www.dofactory.com/net/composite-design-pattern (Accessed: 22 June 2021).