Everything on Design Patterns

A design pattern is a general reusable solution to a commonly occurring problem in software design. It is not a finished design, it is a description or template for how to solve a problem, that can be used in many different solutions.

Object-oriented design patterns typically show relationships and interactions between classes or objects.

Main classification

Creational Patterns: Creating objects

Structural Patterns: Class/Object structure e.g. Composite

Behavioural Pattern: Interaction e.g. Publisher-Subscriber

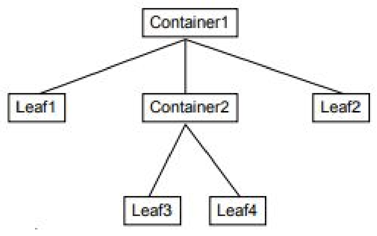
The Composite Pattern

Intent: It manages part-whole hierarchies. It uses tree structures.

Similar objects/components can be composed in a hierarchical structure.

Single instances and groups of similar instances can be dealt with uniformly.

Consider a Container object that contains an arbitrary number of Component objects, each of which might be a Container.



A container object is an aggregate that can contain uniform Component objects which are all instances of subclasses of the Component superclass.

Some of the subclasses are Leaf Components which are simple objects with no extension of the hierarchy. However some of the Component objects may themselves be Container objects; Container is a subclass of the class Component.

Example of a Composite is a file store Folder. It can contain different kinds of File objects but also contain Folder objects. Hence a folder is both a Container and a Component.

This pattern describes a not-so-natural structure in which the container or aggregate is a subclass of a component.

Publisher-Subscriber Pattern
Subscribers register with a Publisher. When the publisher has some new information all its subscribers are informed. The subscribers can then access the new information.

We have the abstract superclasses Publisher and Subscriber and the concrete subclasses ConcPublisher and ConcSubscriber