

Composite

The composite pattern allows combining objects into a larger structure, representing a hierarchy of components. For instance, the composite pattern can be employed to depict an organizational hierarchy, a menu hierarchy, or a file hierarchy. It enables clients to treat individual objects and compositions of objects in the same manner. This can simplify client code and make it more generic. The participants in this pattern are:

The component: Declares a common interface for all objects. Defines default behavior for all classes.

Leaf: This represents the leaf, implementing elementary behavior.

Composite: Defines the behavior of components with children, stores the children, and implements the necessary operations for their management.

The client: If the receiver is a leaf, the request is directly handled. Otherwise, the composite forwards it to all its children.

1. Initial

This first implementation represents the general model for the composite Class. It is as follows:

Model



