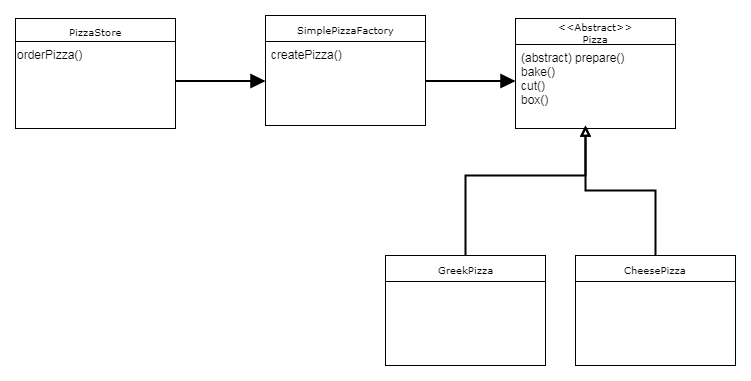
**Design Pattern**

**Creational**

**Singleton**

The singleton pattern restricts the initialization of a class to ensure that only one instance of the class can be created.

**Method Factory**

The factory pattern takes out the responsibility of instantiating an object from the class to a Factory class. (Without knowing the logic inside of the factory)

**Builder**

Creating an object step by step and a method to finally get the object instance.

**Prototype**

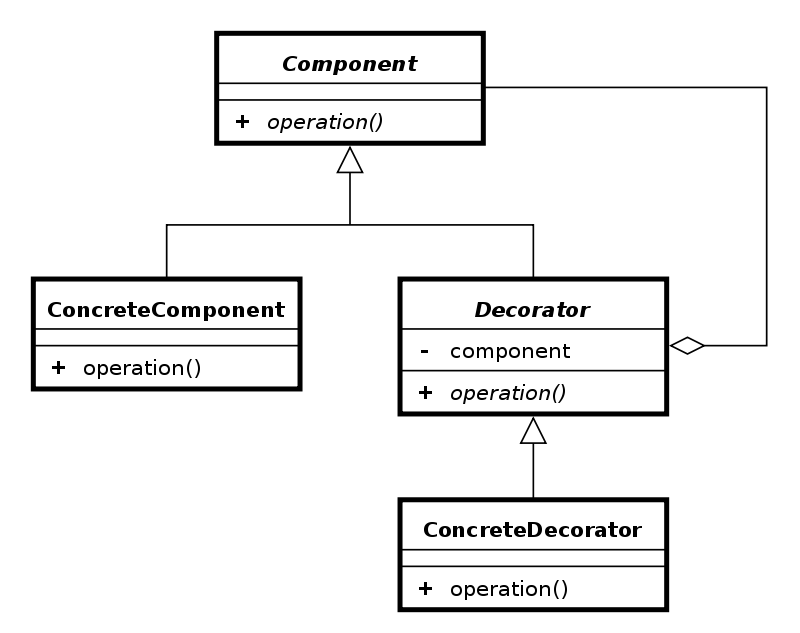
Creating a new object instance from another similar instance and then modify according to our requirements.

**Adapter**

Provides an interface between two unrelated entities so that they can work together.

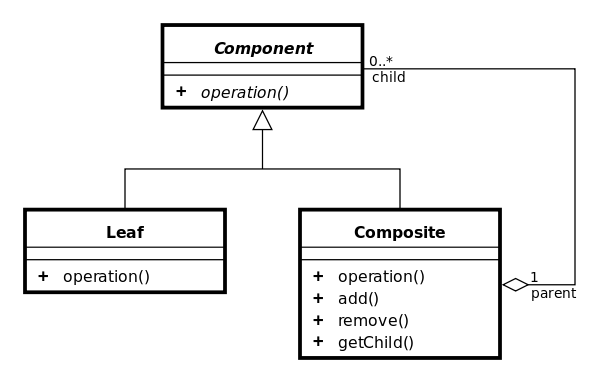
**Decorator**

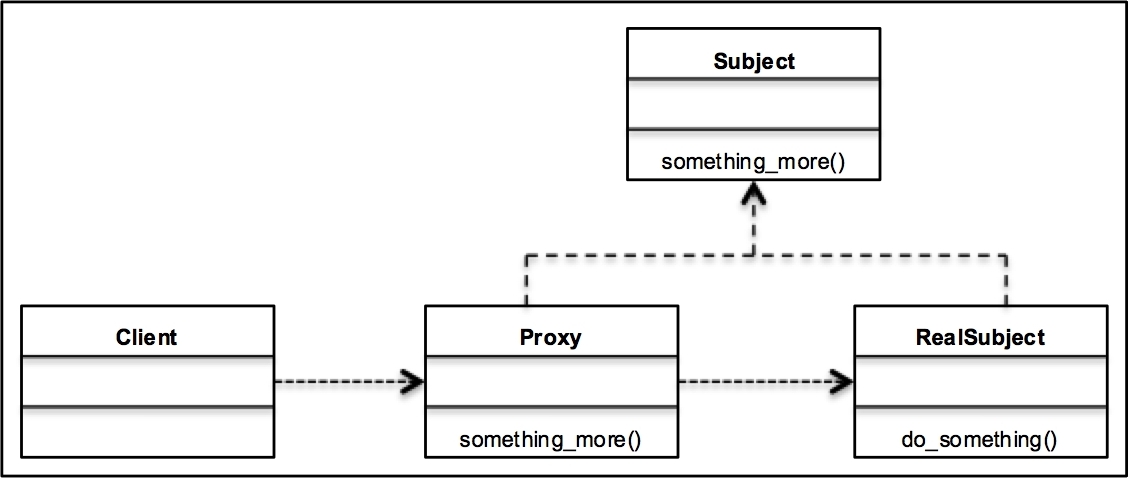
The decorator design pattern is used to modify the functionality of an object at runtime.



**Composite**

Used when we have to implement a part-whole hierarchy. For example, a diagram made of other pieces such as circle, square, triangle, etc.

**Proxy**

Provide a surrogate or placeholder for another object to control access to it.