Design Pattern Report

Flyweight Pattern: (Structural Pattern)

**Motivation:**

Some programs require a large number of objects that have some shared state among them.

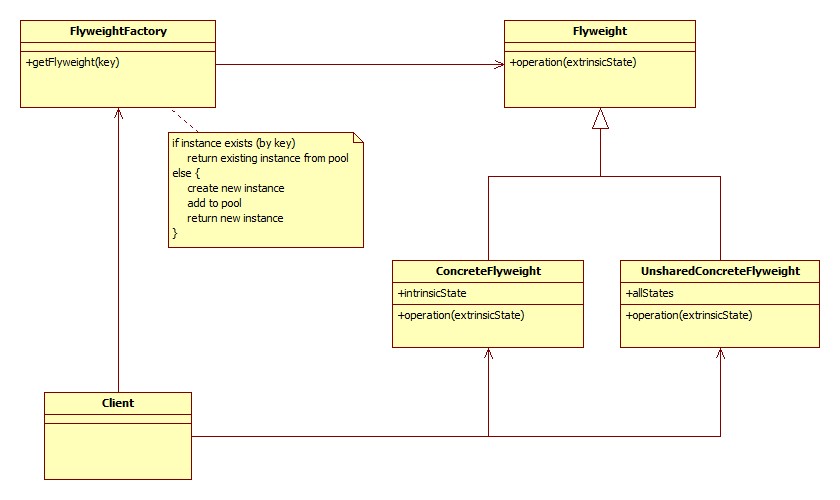
**Intent:**

* The intent of this pattern is to use sharing to support a large number of objects that have part of their internal state in common where the other external state can vary.

**Description:**

A client needs a flyweight object; it calls the factory to get the flyweight object. The factory checks a pool of flyweights to determine if a flyweight object of the requested type is in the pool, if there is, the reference to that object is returned. If there is no object of the required type, the factory creates a flyweight of the requested type, adds it to the pool, and returns a reference to the flyweight. The flyweight maintains intrinsic state (state that is shared among the large number of objects that we have created the flyweight for) and provides methods to manipulate external state (State that vary from object to object and is not common among the objects we have created the flyweight for).

**Diagram:**

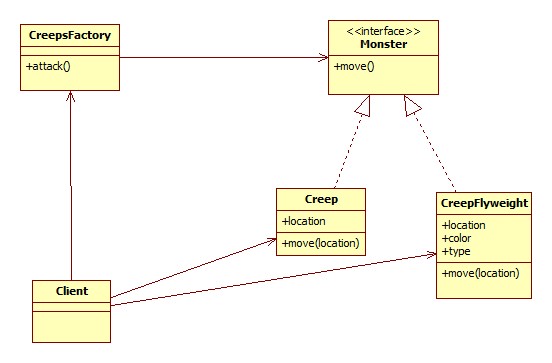


**Advantages / Disadvantages:**

Flyweight pattern saves memory by sharing flyweight objects among clients. The amount of memory saved generally depends on the number of flyweight categories saved

Dis: adds complexity, which makes it harder to debug and maintain. When an error occurs, it can come from the factory, the client or the flyweight.

**Example:**



**Related Pattern:**

**Factory : use to implement the factory of the flyweight**

Composite: is often used with flyweight, create more flexible design

