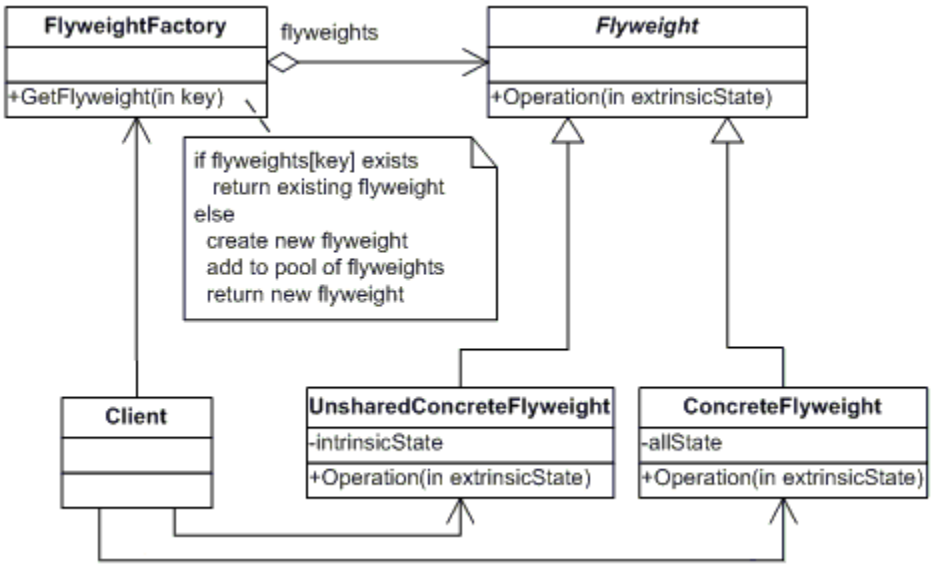
12 Flyweight Pattern [Gamma et al]

Flyweight pattern applies in situations when different identical objects have some common state that is shared among them. The state of flyweight objects is termed as intrinsic (common part) and extrinsic (rest part). Such type of scenario is usually implemented in games where different objects share a part of common representation. According to [1] flyweight factory is used to create and share flyweight objects from the pool. A pool object usually an array of a Dictionary object is maintained in the Factory Class implementation. Following are the variations in flyweight design pattern.

12.1 Constrainedly Shared Flyweight [80]

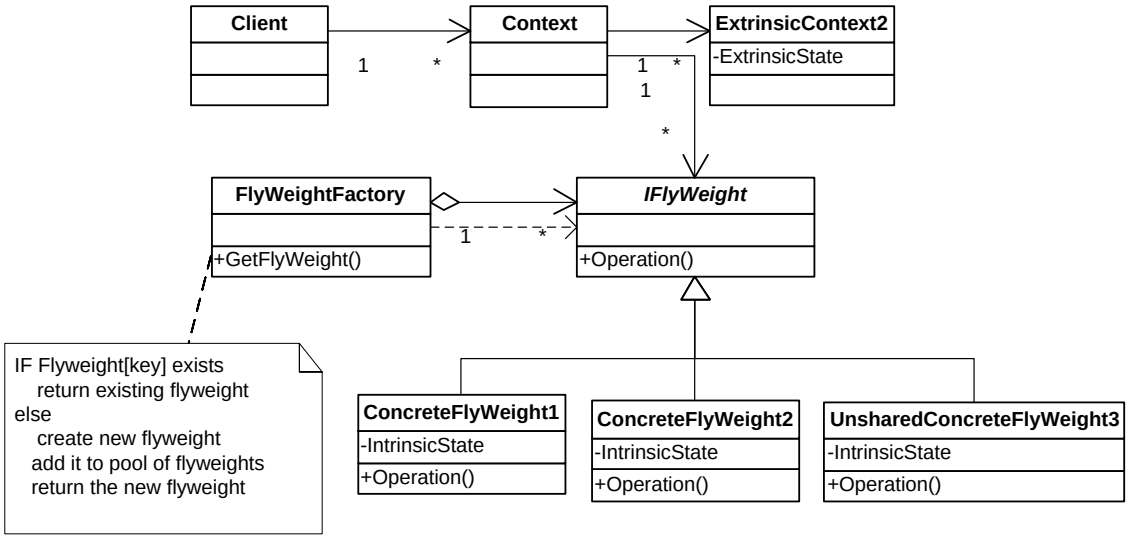
In this variant clients interact with flyweight objects through flyweight factory which is usually a Singleton object. An array or a hash map is maintained that checks for existance of a certain flyweight, if not found one is created and added to the pool. It can be best illustrated by drawing line flyweights, usually a set of lines shares common elements like color (intrinsic) and cordinates (extrinsic). The following variant describes that not all flyweight objects are shared.



12.1 UML Diagram of Constrainedly Shared Flyweight [80].

12.2 Externalizing Extrinsic State [80]

This variant describes external state and internal state separately and mixture of both these states in referenced in Context object. This implementation layers the flyweight intrinsic state and extrinsic state and combination of both is used by the client.



12.2 UML Diagram of Externalizing Extrinsic state of Flyweight [80].