Flyweight design pattern:

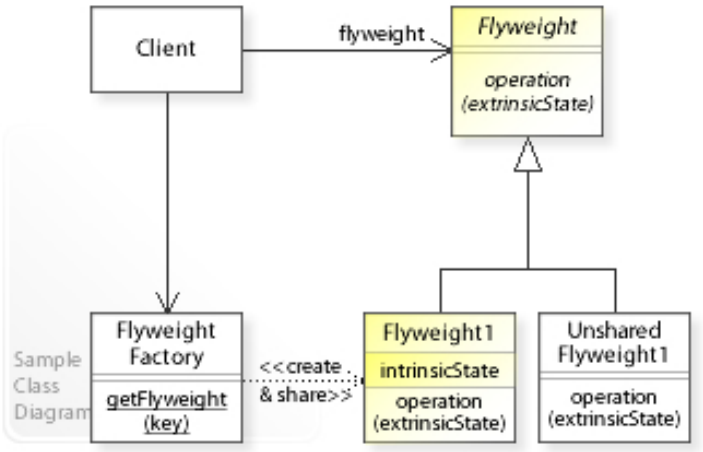
(Protyush Kumar Das)

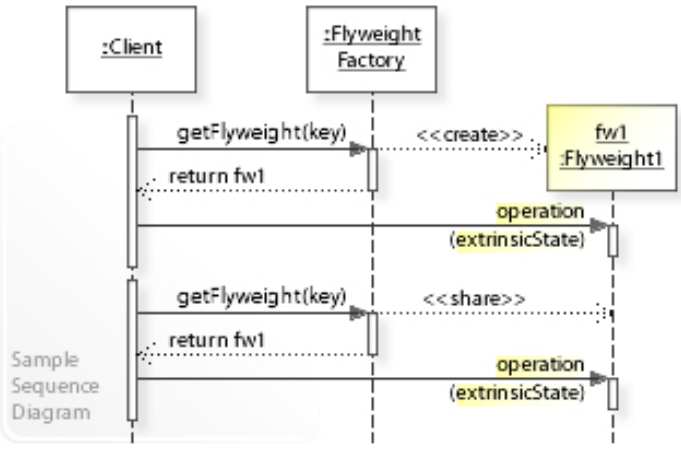
Description:

- A flyweight is an object that minimizes memory usage by sharing as much data as possible with other similar objects
- It is a way to use objects in large numbers when a simple repeated representation would use an unacceptable amount of memory.
- A classic example usage of the flyweight pattern is the data structures for graphical representation of characters in a word processor. It might be desirable to have, for each character in a document, a glyph object containing its font outline, font metrics, and other formatting data, but this would amount to hundreds or thousands of bytes for each character. Instead, for every character there might be a reference to a flyweight glyph object shared by every instance of the same character in the document; only the position of each character (in the document and/or the page) would need to be stored internally.

UML Diagram:

- In the below UML class diagram, the Client class refers (1) to the FlyweightFactory class to create/share Flyweight objects and (2) to the Flyweightinterface to perform an operation by passing in extrinsic (variant) state (flyweight.operation(extrinsicState)). The Flyweight1 class implements the Flyweight interface and stores intrinsic (invariant) state that can be shared.
  The sequence diagram shows the run-time interactions: The Client object calls getFlyweight(key) on the FlyweightFactory that creates and returns a Flyweight1 object. After calling operation(extrinsicState) on the returned Flyweight1 object, the Client again calls getFlyweight(key) on the FlyweightFactory, which now shares and returns the already existing Flyweight1 object.





References:

- https://en.wikipedia.org/wiki/Flyweight_pattern