Flyweight

Propósito

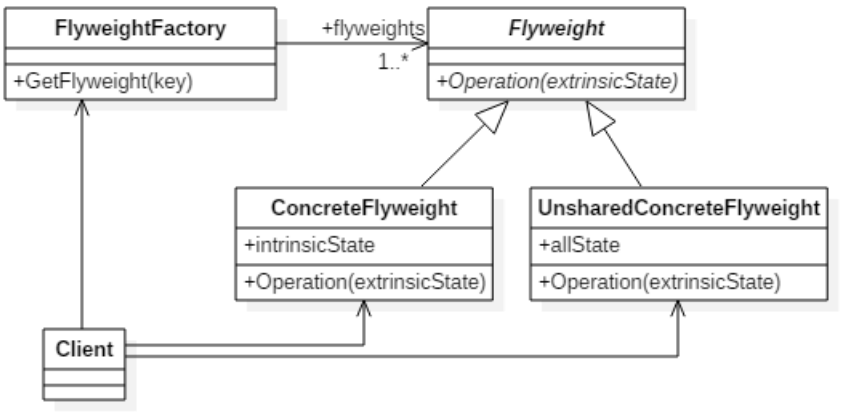
Utiliza compartir para soportar de manera eficiente un gran número de objetos pequeños.

Aplicabilidad

La efectividad del patrón Flyweight depende de cómo y dónde se utilice. El patrón Flyweight se debe aplicar cuando todas las siguientes condiciones se cumplan:

- Una aplicación utiliza un gran número de objetos
- El coste de almacenamiento es alto debido a la cantidad de objetos
- La mayor parte del estado del objeto se puede hacer extrínseco
- Muchos grupos de objetos se pueden reemplazar por unos pocos una vez eliminado el estado extrínseco
- La aplicación no depende de la identidad de objetos. Como los objetos flyweight se pueden compartir, los test de identidad devolverán true para objetos conceptualmente diferentes

Estructura



Participantes

- Flyweight: declara una interface mediante la cual los flyweights pueden recibir y actuar sobre el estado extrínseco
- ConcreteFlyweight: implementa la interface Flyweight y agrega almacenamiento para su estado intrínseco, si existe. Un objeto ConcreteFlyweight debe ser poder compartir. Cualquier estado que almacene debe ser intrínseco; esto es, debe ser independiente del contexto del objeto ConcreteFlyweight.
- UnsharedConcreteFlyweight: no todas las subclases de Flyweight necesitan ser compartidas. La interface Flyweight permite compartir; no obliga. Es común que los objetos UnsharedConcreteFlyweight tengan objetos ConcreteFlyweight como hijos en algún nivel de la estructura de objetos flyweight
- FlyweightFactory:
    - crea y gestiona los objetos flyweight

    - Asegura que los flyweights se comparten. Cuando un cliente solicita un flyweight, la factoría proporciona una instancia existente o crea una si no existe
- Client:
    - mantiene una referencia los flyweights
    - calcula o almacena el estado extrínseco de los flyweights

Colaboraciones

- El estado que necesita un flyweight debe ser caracterizado como intrínseco o extrínseco. El estado intrínseco se almacena en el objeto ConcreteFlyweight; el estado extrínseco se almacena o calcula en los objetos Client
- Los clientes no deben instanciar directamente ConcreteFlyweight sino que los deben obtener de la FlyweightFactory, para asegurarnos así que se comparten adecuadamente

Consecuencias

Los Flyweights pueden tener costes de tiempo de ejecución al buscar y/o calcular el estado extrínseco, especialmente si se había almacenado previamente como estado intrínseco. Sin embargo, tal coste se puede balancear con el ahorro de espacio, que aumente a medida que hay más flyweights.

El ahorro de almacenamiento depende de varios factores:

- La reducción del número total de instancias consecuencia de compartir
- La cantidad de estado intrínseco por objeto
- Si el estado extrínseco es almacenado o calculado

Cuantos más flyweights se compartan, mayor es el ahorro de almacenamiento. El ahorro aumenta con la cantidad de estado compartido. El mayor ahorro ocurre cuando los objetos utilizan cantidades sustanciales de ambos estados, intrínseco y extrínseco, y el extrínseco se calcula en vez de almacenarlo.