Flyweight

Definición:
Flyweight es un patrón que centra su atención en la construcción de objetos ligeros, mediante la abstracción de las partes reutilizables que pueden ser compartidas con otros objetos. Esto con el fin de que en lugar de crear objetos cada vez que sea requerido, podamos reutilizar objetos creados por otras instancias logrando con ello reducir en gran medida la capacidad de memoria requerida por la aplicación.

Propósito:
Esto con el fin de que en lugar de crear objetos cada vez que sea requerido, podamos reutilizar objetos creados por otras instancias logrando con ello reducir en gran medida la capacidad de memoria requerida por la aplicación.
Este patrón es utilizado cuando la optimización de los recursos es algo primordial ya que elimina la redundancia de objetos con propiedades idénticas.

¿Cómo lo hace?
Compartir de manera eficiente objetos comunes y separar las propiedades intrínsecas (compartidas) de las propiedades extrínsecas (variables) de dichos objetos. Los datos intrínsecos se almacenan en objetos Flyweight compartidos, mientras que los datos extrínsecos se mantienen separados.

Objetos Intrínsecos y Extrínsecos:
En el patrón Flyweight, los objetos se dividen en dos partes: intrínsecos y extrínsecos. Los objetos intrínsecos son aquellos que pueden ser compartidos y no cambian a lo largo de la vida de un objeto Flyweight. Los objetos extrínsecos contienen información que varía y no puede compartirse.

Componentes
Client: Objeto que dispara la ejecución.
FlyweightFactory: Fábrica que utilizaremos para crear los objetos Flyweight u objetos ligeros.
Flyweight: Corresponde a los objetos que deseamos reutilizar con el fin de que nuestros objetos sean más ligeros.

Diagrama de secuencia:
El cliente solicita al Factory la creación de un objeto Flyweight.
El Factory antes de crear el objeto, valida si ya existe un objeto idéntico al que se le está solicitando. De ser así, regresa el objeto existente; de no existir, crea el nuevo objeto y lo almacena en caché para ser reutilizado más adelante.
El objeto Flyweight se crea o es tomado del caché y es devuelto al cliente.

Aplicabilidad
Utiliza un gran número de objetos
Costes de almacenamiento elevados
El estado del objeto puede hacerse mayormente extrínseco
No depende de la identidad de un objeto

Ventajas:
El patrón Flyweight ofrece ventajas en términos de ahorro de memoria y rendimiento, especialmente en aplicaciones que trabajan con grandes cantidades de objetos similares. Reduce la duplicación de datos y mejora la eficiencia en la gestión de recursos.

Desventajas:
Implementar el patrón Flyweight puede complicar el diseño de una aplicación, ya que requiere identificar y separar los datos intrínsecos y extrínsecos de los objetos. Además, no es adecuado para todos los casos de uso, ya que se centra en la eficiencia de la memoria y puede ser excesivo en situaciones donde la memoria no es un recurso crítico.

Ejemplo:
El cliente, que es el documento, desea agregar letras al texto. Por ejemplo, el cliente quiere agregar "H" en la fuente "Arial".

El documento solicita la creación de una letra al FlyweightFactory, especificando el carácter ("H") y la fuente ("Arial").

La FlyweightFactory verifica si ya existe un objeto Flyweight con esas propiedades. Si existe, se devuelve el objeto Flyweight existente. Si no existe, se crea uno nuevo y lo almacena en la caché de la fábrica para futuras referencias y se devuelve al "Documento".

El objeto Flyweight (la letra "H" en "Arial") se almacena en el documento y se utiliza para representar esa letra en el texto.

Cuando se agrega otra letra idéntica ("l" en "Arial"), el proceso se repite, pero como la letra y la fuente son las mismas, se reutiliza el objeto Flyweight existente en lugar de crear uno nuevo.