Patrón Flyweight:

-permite compartir información que se encuentra en pequeños objetos que existen en grandes cantidades.

- Ayuda a reducir las necesidades de almacenamiento.

Ejemplo del restaurante: Pensemos en una receta en particular, esta receta podría pertenecer a la sección de comida “italiana” y a la vez a la sección de “ensaladas” y al mismo tiempo pertenecer a “menú del día” o a la sección de comida “light”, etc. Tenemos una serie de objetos que pueden clasificarse de diferentes formas y pueden pertenecer a distintas colecciones. Si en este caso no utilizáramos este patrón correríamos el riesgo de tener objetos repetidos, teniendo una instancia de cada receta para cada una de las secciones donde se encuentre la misma. Por esto, este patrón nos permite tener un solo objeto y que podamos tener el mismo en distintas colecciones. nos permitirá compartir información que se encuentra en objetos pequeños y estos existen en grandes cantidades.

Ej: Restaurante -> tiene 100 recetas.

El patrón distingue entre dos estados que puede tener un objeto:

* + Intrínseco: se puede compartir a gran escala, disminuye el almacenamiento.
    - receta comparte: nombre, ingredientes, etc.
  + Extrínseco: el estado se calcula en el momento.
    - Calculo del costo de la receta.

El estado no compartido es el conjunto de grupos a los que va a pertenecer un objeto en particular. Estos múltiples objetos que se van a estar compartiendo en distintas secciones los llamamos “flyweights”. Por lo que vamos a tener una colección de flyweights.

Objetos del ejemplo dado:

Cliente: Es el que mantiene el estado no compartido y colecciones de flyweights.

IFlyweight: La interfaz por medio de la cual los Flyweights pueden recibir o actuar sobre un estado intrínseco.

FlyweightFactory: Administra los objetos Flyweight y garantiza que solo un valor es creado.

Flyweight: guarda el estado intrínseco que comparte a todos los objetos.