
PATRON
FLYWEIGHT
DISEÑO DE SISTEMAS
START

CONTENIDO

INTRODUCCIÓN
¿Para que sirve?
01

ESTADOS
El patrón distingue entre dos estados que puede tener el objeto
02

ESTRUCTURA
El flyweight se relaciona con el patron factory
03

VENTAJAS
Puedes poner una breve descripción del tema aquí
04

DESVENTAJAS
Puedes poner una breve descripción del tema aquí
05

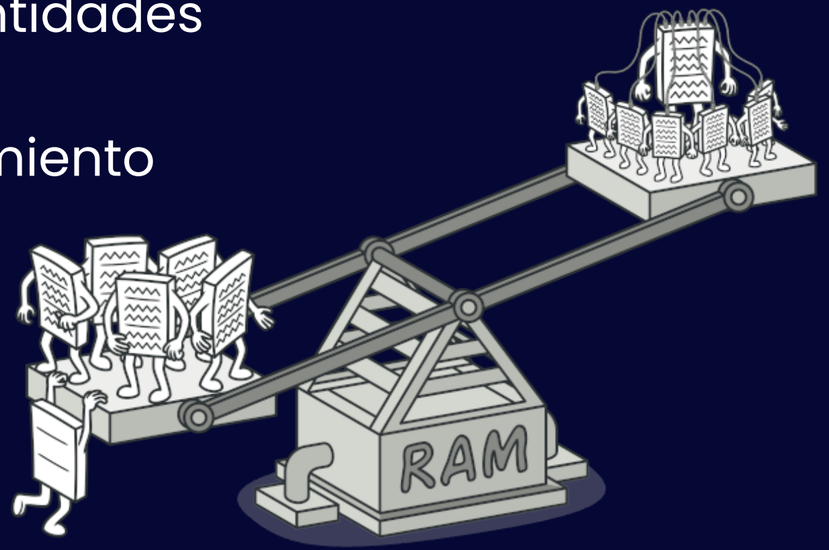
EJEMPLOS
Puedes poner una breve descripción del tema aquí
06

01
INTRODUCCIÓN

- Flyweight es un patrón de diseño estructural que te permite mantener más objetos dentro de la cantidad disponible de RAM compartiendo las partes comunes del estado entre varios objetos en lugar de mantener toda la información en cada objeto.

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

- Ayuda recudir las necesidades de almacenamiento



02
ESTADOS

ESTADO INTRÍNSECO

- SON LOS DATOS COMPARTIDOS POR TODOS LOS OBJETOS DE UN SUBTIPO DETERMINADO.
- HACEN REFERENCIA A LOS ESTADOS COMUNES QUE TIENE EL OBJETO O GRUPO DE OBJETO A REPLICAR

ESTADO EXTRÍNSECO

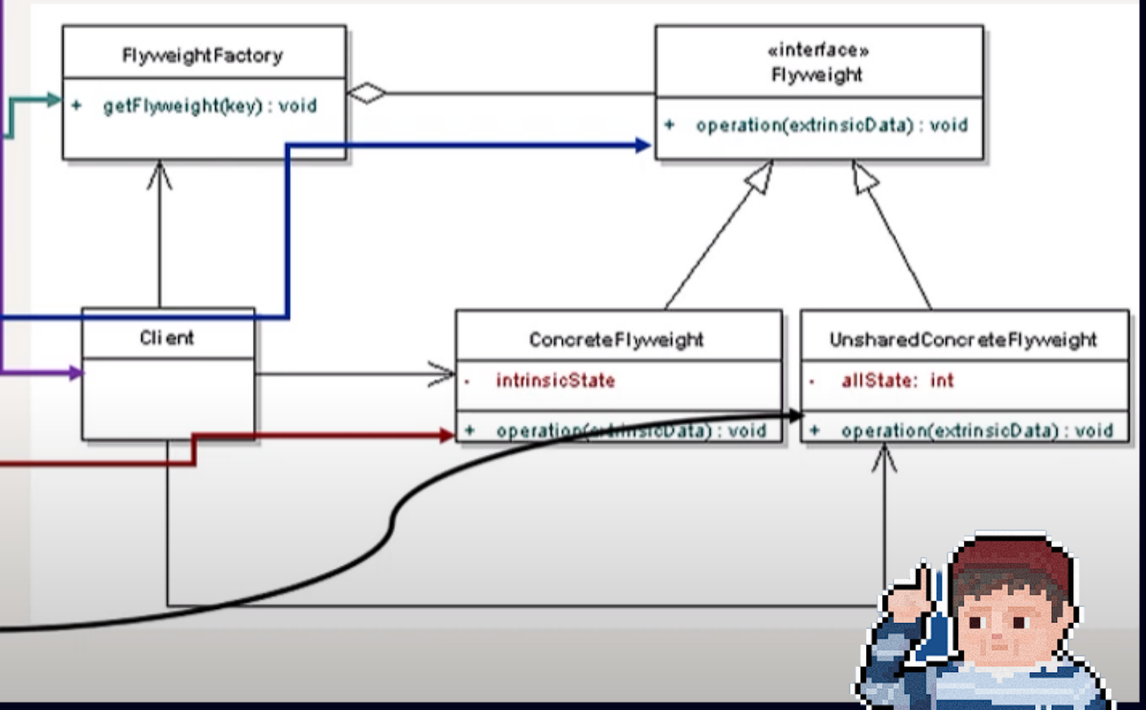
- ALUDEN A LAS CARACTERISTICAS PROPIAS DE LA INSTANCIA
- EL ESTADO SE CALCULA EN EL MOMENTO

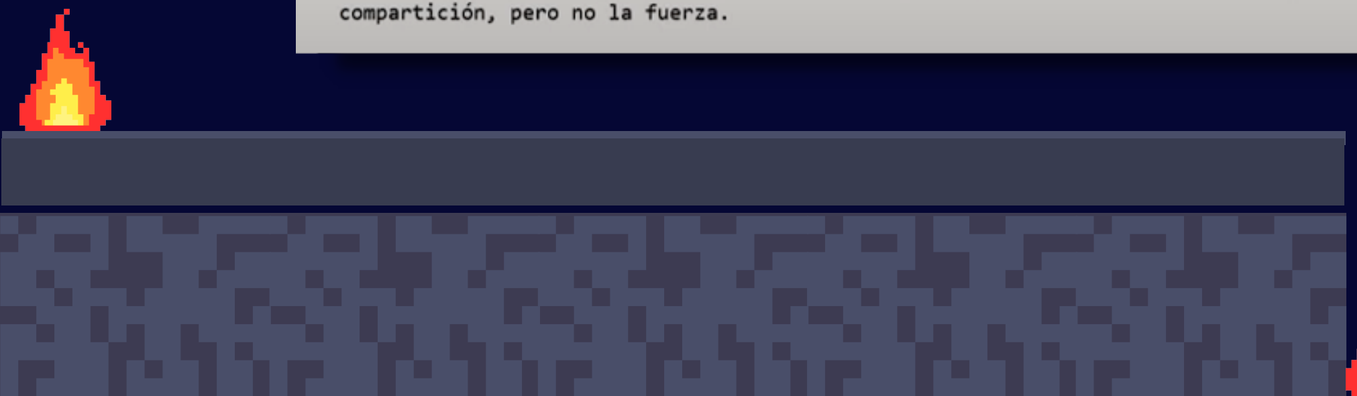
EL OBJETIVO DE ESTE PATRON ES ESTUDIAR LAS CARACTERISTICAS COMUNES DE LOS OBJETOS Y CREAR UNA CLASE EXTRINSECA. LUEGO A LA HORA DE REALIZAR LAS DIFERENTES INSTANCIAS, USA LA PARTE COMUN (INTRINSECA) Y LA COMPLEMENTA CON LOS DATOS ESPECIFICOS DE LA INSTANCIA

03

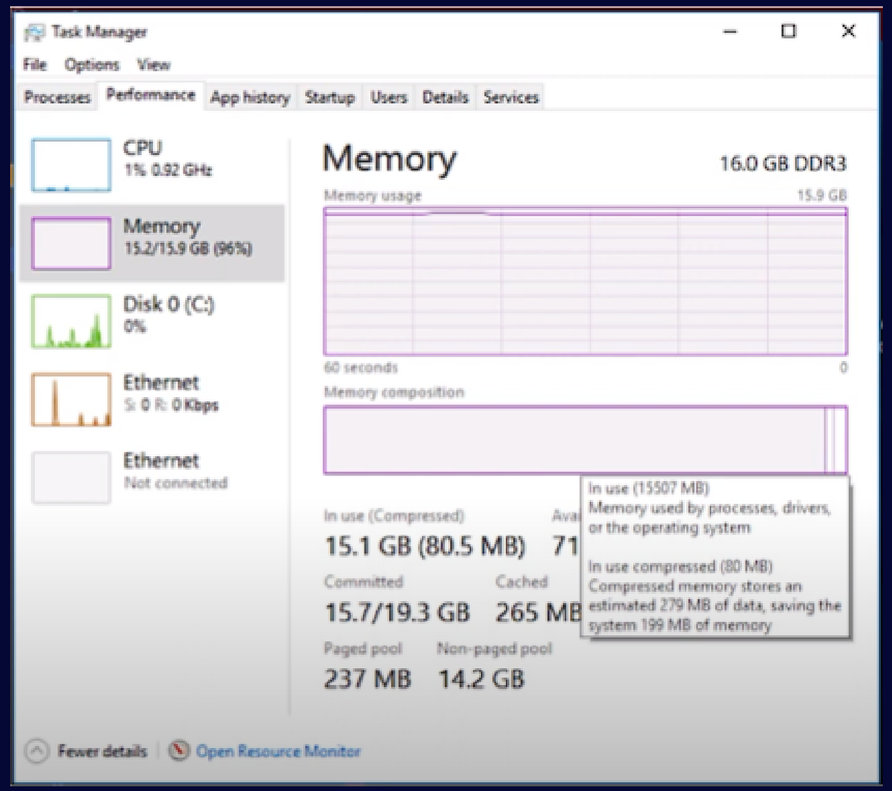
ESTRUCTURA

- Client: Trabaja con una referencia a un Flyweight. Establece los estados extrínsecos de los objEtos.
- FlyweightFactory: Crea y maneja objetos flyweight asegurándose que se compartan adecuadamente. Cuando el cliente solicita un flyweight, FlyweightFactory proporciona una instancia existente y si no existe la crea.
- Flyweight: Interfaz que contiene métodos para el establecimiento, acceso y modificación de las propiedades extrínsecas. Deberá ser implementada por las instancias del Flyweight.
- ConcreteFlyweight: Implementa la interfaz Flyweight y añade el almacenamiento de las características intrínsecas (si las hay) y deben poder ser compartidos.
- UnsharedConcreteFlyweight: Todos los Flyweight no tienen por qué ser compartidos, la implementación de la interfaz habilita la compartición, pero no la fuerza.





04 VENTAJAS

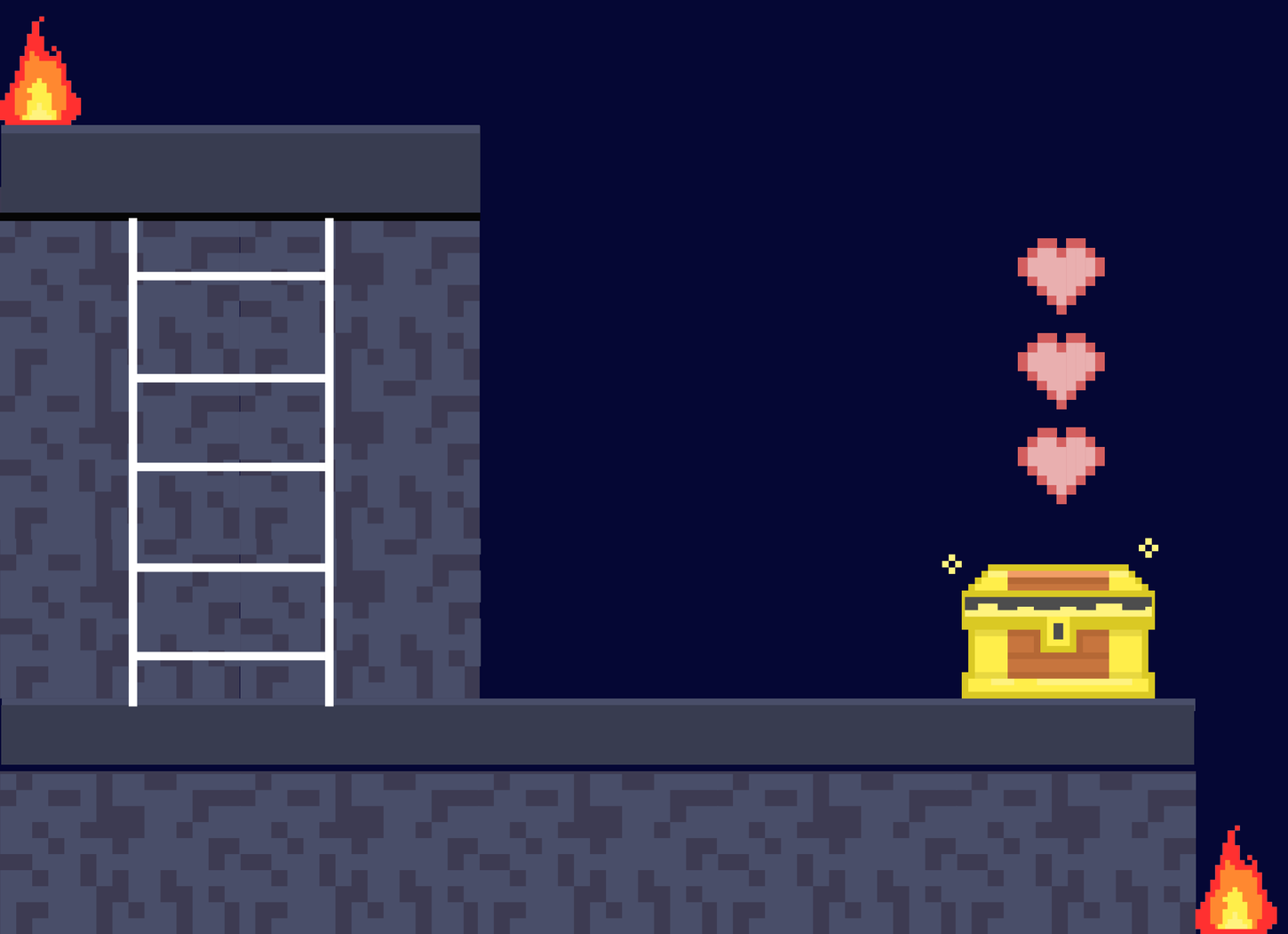


- Menos uso de memoria derivado de la reutilizacion de objetos ya existentes que comparten mismos estados intrinsecos

- Mejora en la eficiencia, sobre todo en relación con el ahorro en el almacenamiento, que aumenta paralelamente con la compartición de objetos y es función de:

(a) la reducción en el número total de instancias.
(b) la cantidad de estado intrínseco por objeto.
(c) si el estado extrínseco es computado (calculado) o almacenado.

05
DESVENTAJAS



ANTECEDENTE 01

- Se introducen costos en tiempo de ejecución asociados a las transferencias, búsquedas y/o computación del estado extrínseco (y su almacenamiento).

ANTECEDENTE 02

- En algunos casos, al mover el estado afuera del objeto puede quebrar la encapsulacion y puede ser menos eficiente que mantener el objeto completamente intrinsico

06
EJEMPLOS DE USO

Marca:	Seat
Modelo:	Ibiza
Color:	Amarillo
NIF:	12345678A
Matrícula:	1234-CAA
Fecha Mat:	24/03/2014

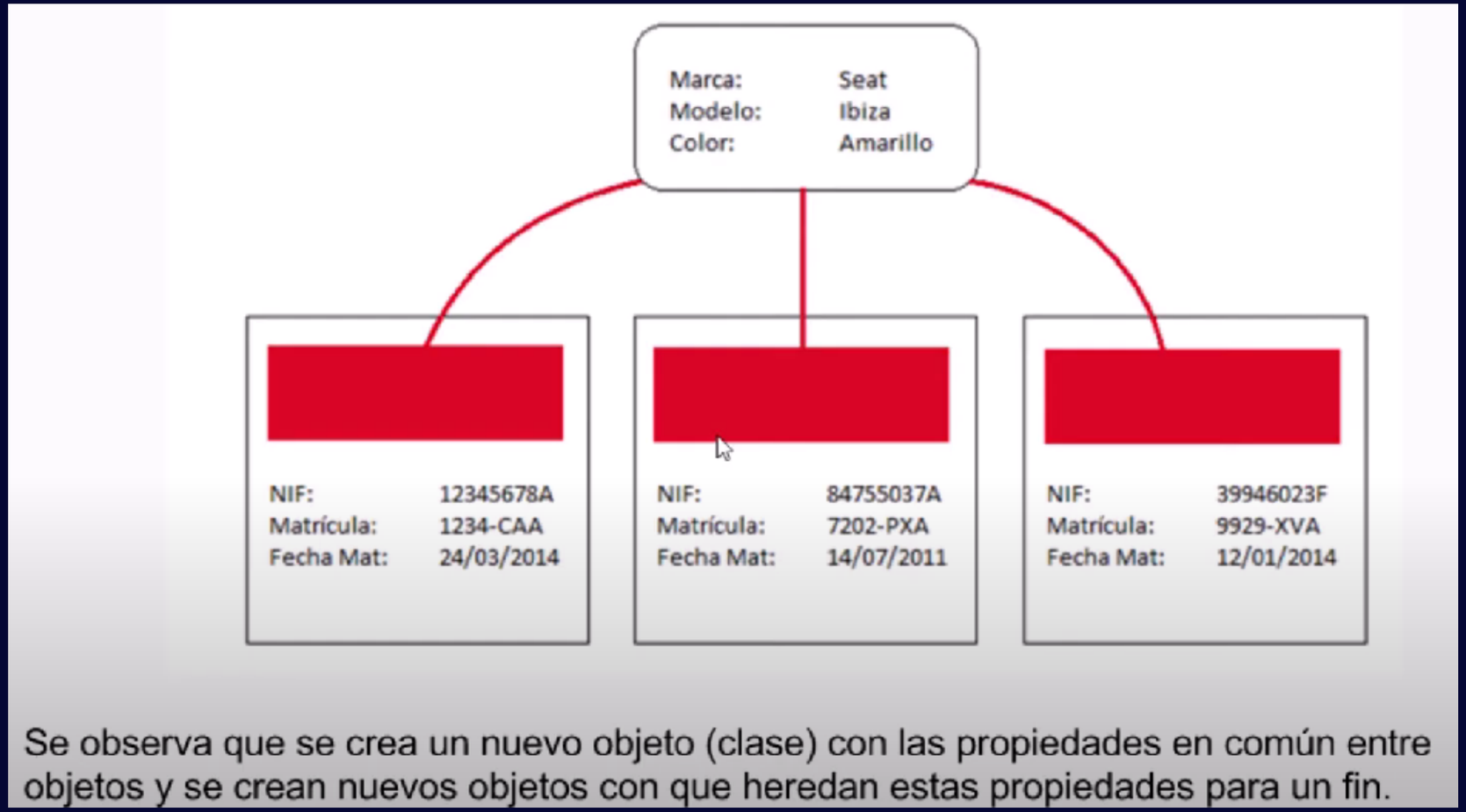
Marca:	Seat
Modelo:	Ibiza
Color:	Amarillo
NIF:	84755037A
Matrícula:	7202-PXA
Fecha Mat:	14/07/2011

Marca:	Seat
Modelo:	Ibiza
Color:	Amarillo
NIF:	39946023F
Matrícula:	9929-XVA
Fecha Mat:	12/01/2014

Los elementos en común de los objetos de la imagen son:

- Marca
- Modelo
- Color

EJEMPLOS DE USO



Se observa que se crea un nuevo objeto (clase) con las propiedades en común entre objetos y se crean nuevos objetos con que heredan estas propiedades para un fin.

PASOS A SEGUIR PARA LA IMPLEMENTACION DEL PATRÓN

- Divida el objeto principal en 2 estados: Estado Intrínseco (elementos que se puedan compartir o son comunes) y Estado Extrínseco (elementos particulares a cada tipo).
- Retire los elementos con estado extrínseco de los atributos de la clase, y añádele mas bien una llamada a métodos.
- Crear una fabrica que pueda almacenar y reutilizar las instancias existentes de clases
- El cliente debe usar la fabrica en vez de utilizar el operador new si requiere de creacion de objetos
- El cliente (o un tercero) debe revisar los estados extrínsecos, y reemplazar esos a métodos de la clase.