**Yonathan Nicolay Joya González 201311407**

**Patrón estructural Decore:**

Dotar de funcionalidades dinámicamente a objetos mediante composición. Es decir, vamos a decorar los objetos para darles más funcionalidad de la que tienen en un principio.

**Ventajas:**

Se puede añadir responsabilidades a un objeto de forma progresiva y dinámica. Más flexibilidad que con la herencia.

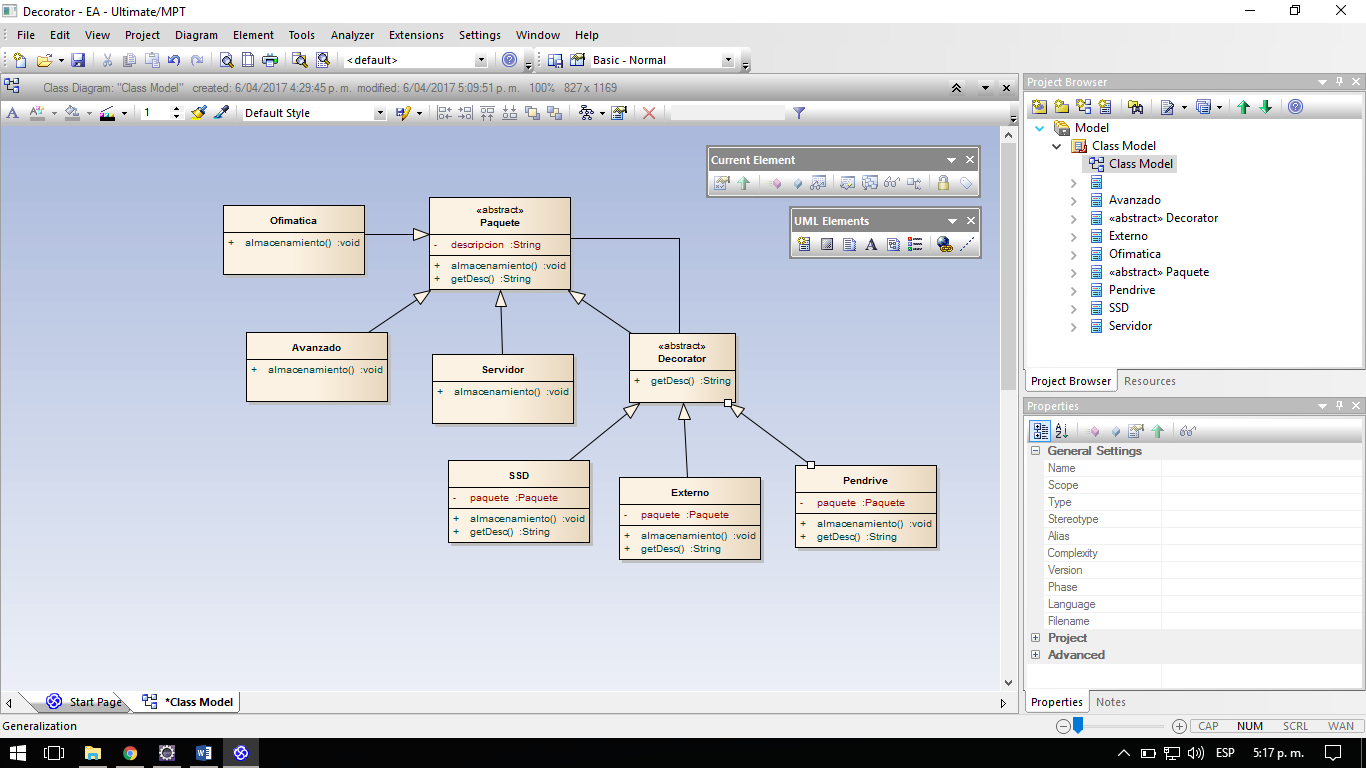
**Desventajas:**

Como resultado se pueden tener clases pequeñas, pero en gran cantidad de ellas.

**Ejemplo:**

Una fábrica de computadores vende sus productos de forma que el cliente puede especificar qué tamaño de almacenamiento además del estándar desean (SSD, Externo, Pendrive) desea adicionar a ciertos estándares ya establecidos (Ofimática, Avanzado, Servidor)

**Diagrama:**



**Patron Estructural Adapter/Facade**

Como se ilustra en la imagen que encabeza este artículo, el patrón Adapter sirve para hacer que dos interfaces, en principio diferentes, puedan comunicarse.

Por lo general se utilizan en sistemas fuertemente acoplados que se repiten interacciones con frecuencias.

**Ventajas:**

hace que dos interfaces incompatibles, sean compatibles. Puede servir para encapsular clases que no controlamos, y que pueden cambiar.

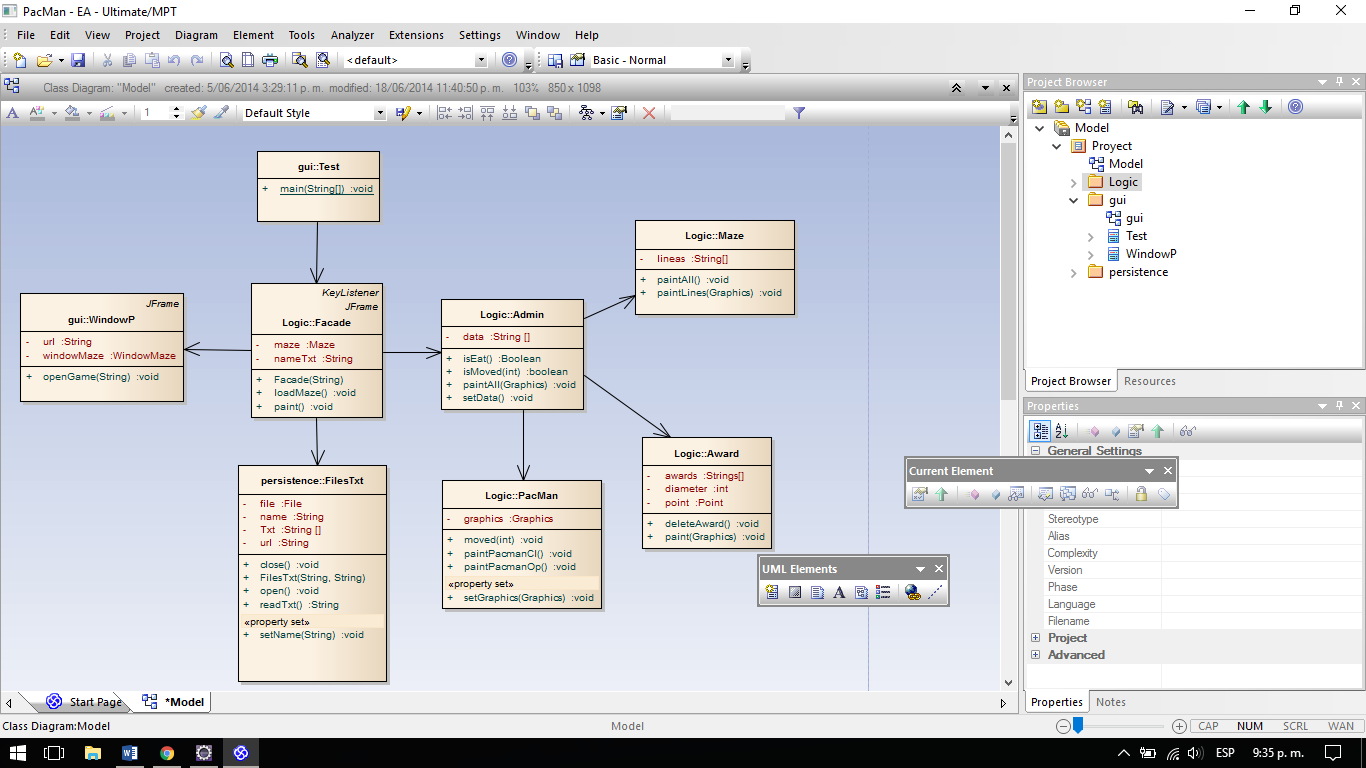
**Desventajas:**

Hace un poco más complejo el diseño

**Ejemplo:**

Se desea desarrollar un videojuego de PACMAN, sin embargo, se debe desacoplar el manejo de interfaz, de procesamiento de datos y de persistencia que almacenan los niveles.

**Diagrama:**



**Patrón Creacional Abstract Factory:**

Este patrón es utilizado para crear instancias mediante interfaces de objetos que en su forma más primitiva son de la misma clase, sin embargo, se comportan de formas diferentes.

**Pros:**

Brinda flexibilidad al aislar a las clases concretas.

Facilita cambiar las familias de productos.

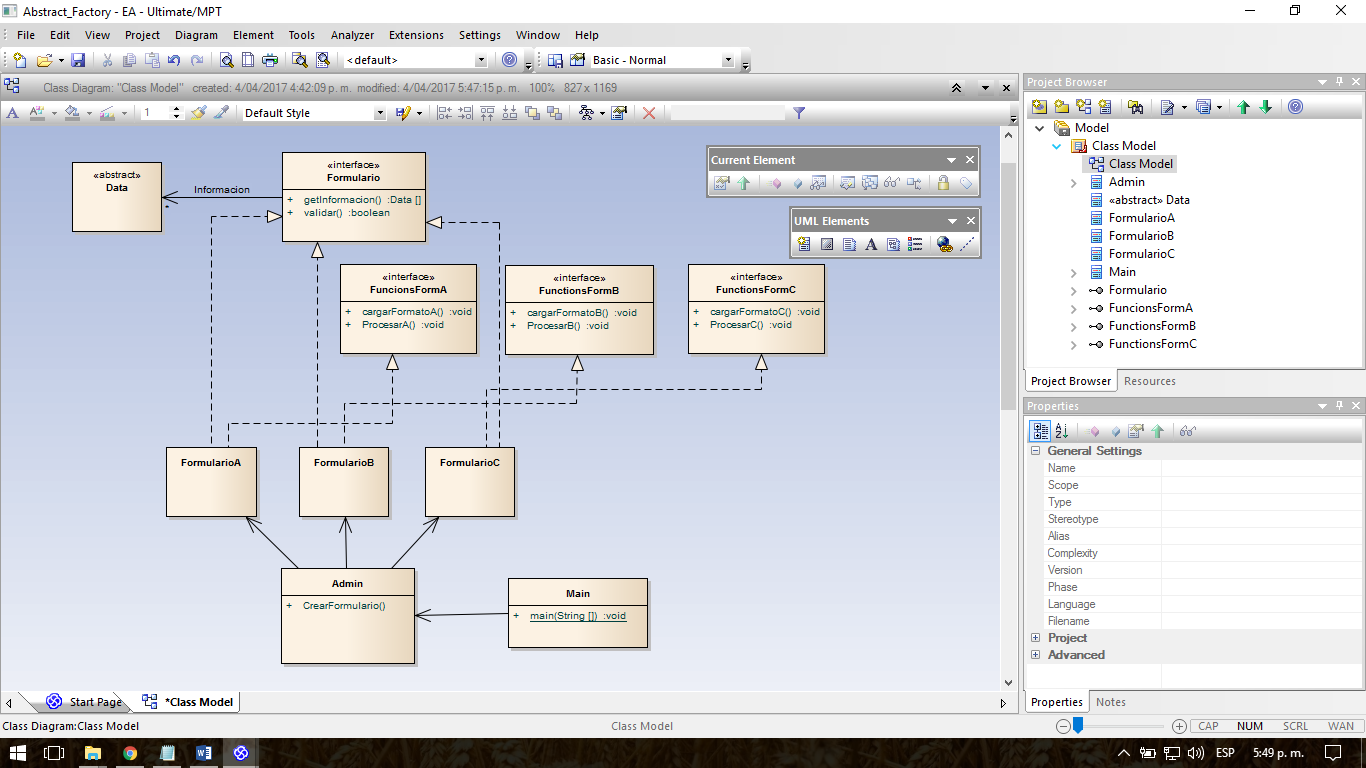
**Contras:**

Para agregar nuevos productos se deben modificar tanto las fabricas abstractas como las concretas.

**Ejemplo:**

Un sistema de información que administra el registro de usuarios de varios subsistemas que necesita el llenado de diferentes formularios con diferentes formatos, cada subsistema tiene necesidades específicas y deben ser procesados de diferentes formas, el objetivo es realizar una interfaz que describa el funcionamiento de un formulario genérico y que para cada formulario hijo se implemente una interfaz con sus funciones específicas.

**Diagrama de clases**



**Patrón Creacional Singleton:**

Este patrón es utilizado para crear solo una instancia global de un objeto, y con esto evitar duplicidad.

**Ejemplo:**

Un sistema distribuido en topología en malla que debe conectarse a través de sockets y todos los Hots deben acceder al procesamiento de una variable específica para mostrarla en la interfaz de cada uno.

**Diagrama de clases**

