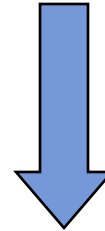


TEMA 1.- INTRODUCCIÓN A LOS PATRONES

1. Tipos de programas y sus grados de dificultades de desarrollo

- No orientados a objetos
- Orientados a objetos
- Orientados a objetos no reutilizables
- Orientados a objetos reutilizables



Incremento del
grado de
dificultad

- La solución para la programación orientada a objetos reutilizables
 - Utilizar los patrones de diseño

2. El concepto de patrones:

- Son resúmenes de experiencias en el diseño orientado a objetos en una forma de plantillas
- Son difíciles para los novatos aprender por sí mismo con su propia experiencia
- Cada patrón nombra, explica y evalúa un diseño recurrente e importante en el diseño de sistemas orientados a objetos.



3. Los cuatro elementos de un patrón: El nombre, el problema, la solución y la consecuencia

1) El nombre del patrón: describe un problema con una o dos palabras, sus soluciones y la consecuencia.

- Nombrar un patrón incrementa los vocabularios de diseño
- Nos permite diseñar con abstracción de un nivel más alta

2) El problema: describe cuándo se puede aplicar el patrón. Explica el problema y su contexto.

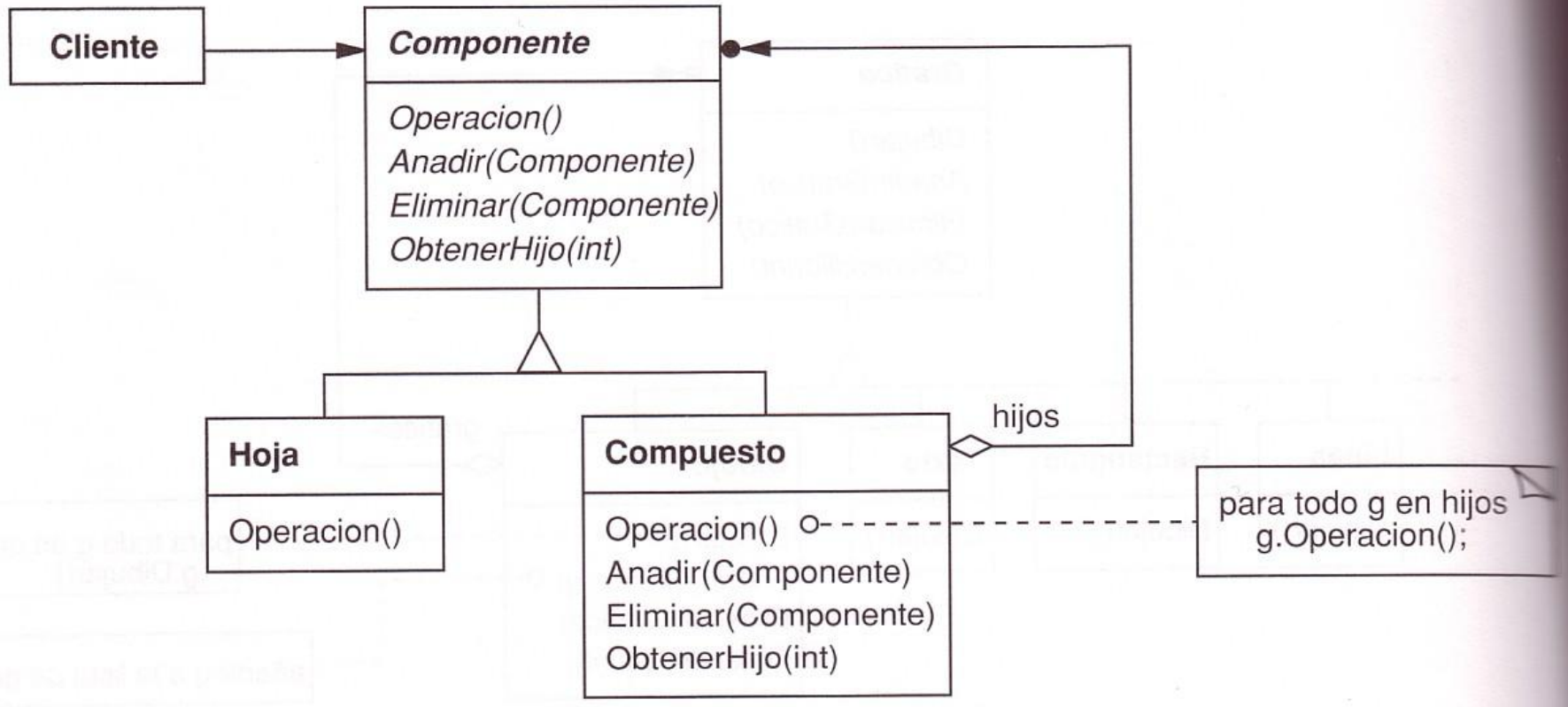
- Problemas específicos: tales como representar los algoritmos con objetos
- Clases o estructuras de objetos que son sintomáticas de un diseño inflexible
- Una lista de condiciones que se tienen que satisfacer antes de aplicar el patrón



- 3) La solución: describe los elementos que forman el diseño, sus relaciones, responsabilidades, y cooperaciones
- No describe ningún diseño concreto
 - Un patrón es una plantilla para muchas situaciones diferentes
 - Un patrón dispone la descripción abstracta de un problema de diseño y cómo se resuelve con elementos (clases y objetos) generales
- 4) Las consecuencias: los buenos resultados e inconvenientes de aplicar el patrón.
- Para evaluar las soluciones del patrón y entender los costes y beneficios de aplicar el patrón.
 - El equilibrio entre el espacio y el tiempo
 - Puede tratar asuntos de lenguajes e implementaciones
 - El impacto sobre la flexibilidad, extensibilidad y portabilidad

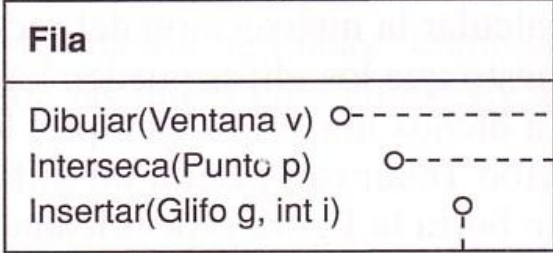
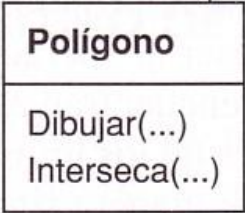
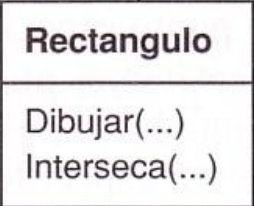
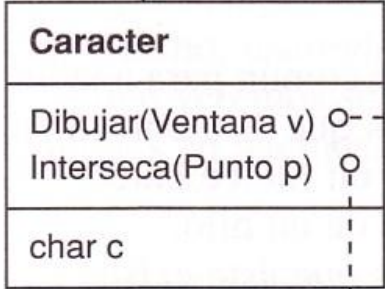
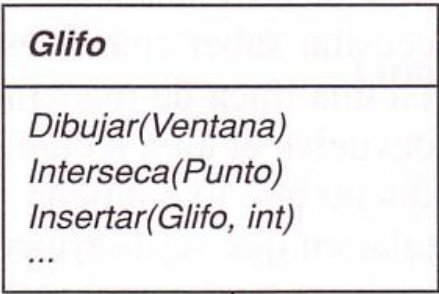


EJEMPLO: LA ESTRUCTURA DEL PATRÓN COMPOSIT



Una implementación: la clase glifo





hijos

devuelve verdadero si el punto p interseca a este carácter

v->DibujarCaracter(c)

insertar g en hijos, en la posición i

para todo c en hijos
if (c->Interseca(p)) return true

para todo c en hijos
comprobar que c está correctamente ubicado;
c->Dibujar(v)

- Un patrón del diseño nomina, abstrae y identifica los aspectos claves de un diseño común para crear un diseño orientado a objetos reutilizable.
- El patrón de diseño identifica
 - las clases e instancias participantes - objetos
 - sus papeles y colaboraciones
 - la distribución de responsabilidades
- Cada patrón enfoca a un solo problema de diseño orientado a objetos



4. Cómo expresar los patrones

- En su formato uniforme: 4 elementos
- Para demostrar cómo se puede implementar: se dispone códigos en C++
- En realidad, se realizan en lenguajes orientados a objetos tales como C++ y smallTalk
- No se suelen realizar en
 - Lenguajes procedimentales Pascal, C, Ada
 - Lenguajes orientados a objetos más dinámicos (Clos, Dylan, Self)

