Patrones de Diseño: Patrones Estructurales. Tema 4-3: Bridge

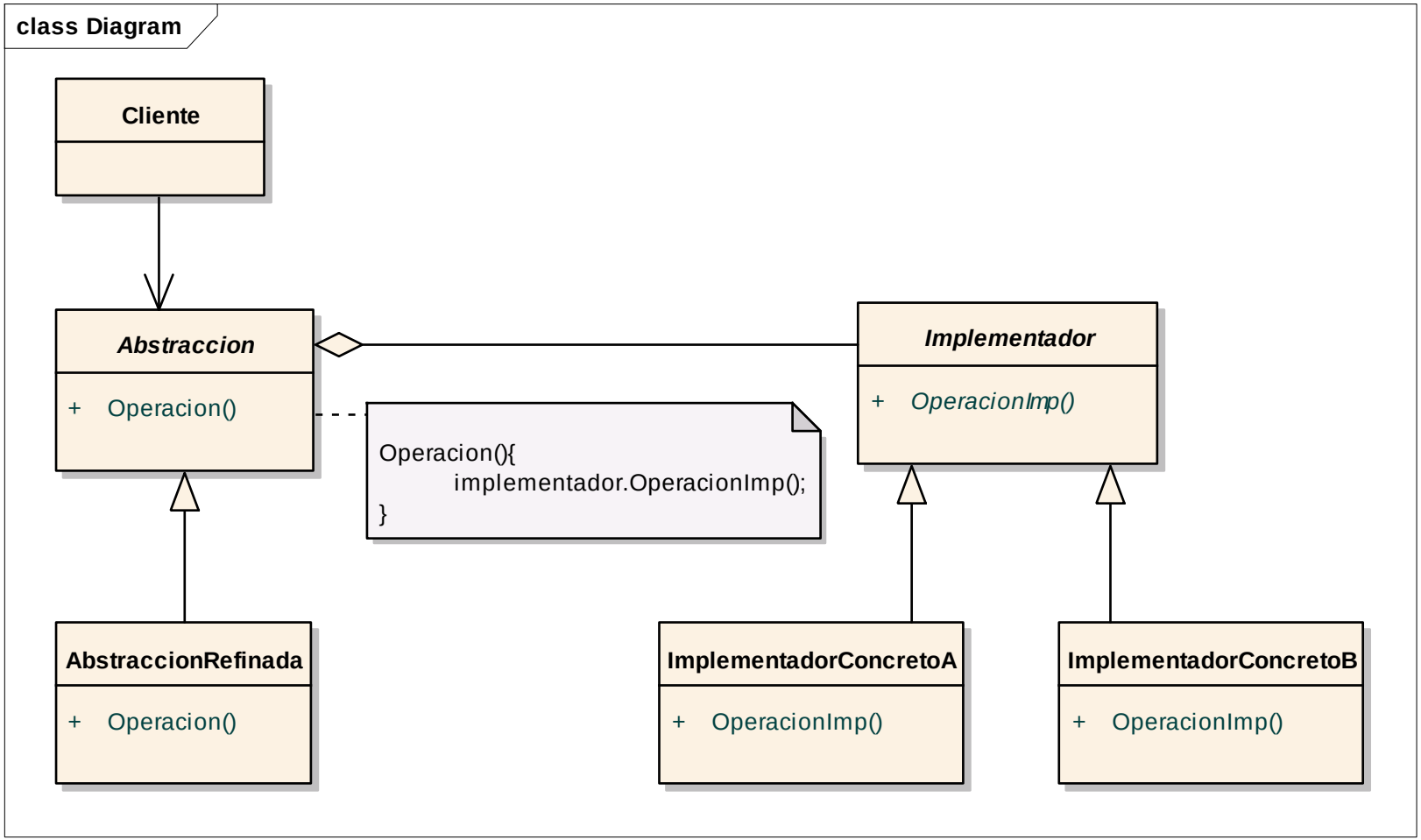
Descripción del patrón

- **Nombre:**
  - Puente
  - También conocido como Handle/Body
- **Propiedades:**
  - Tipo: estructural
  - Nivel: objeto, componente
- **Objetivo o Propósito:**
  - Desacopla un componente complejo en dos jerarquías relacionadas: la abstracción funcional de su implementación interna. De esta forma ambas partes pueden variar de forma independiente y el cambio de cualquier aspecto del componente se hace de manera más fácil.

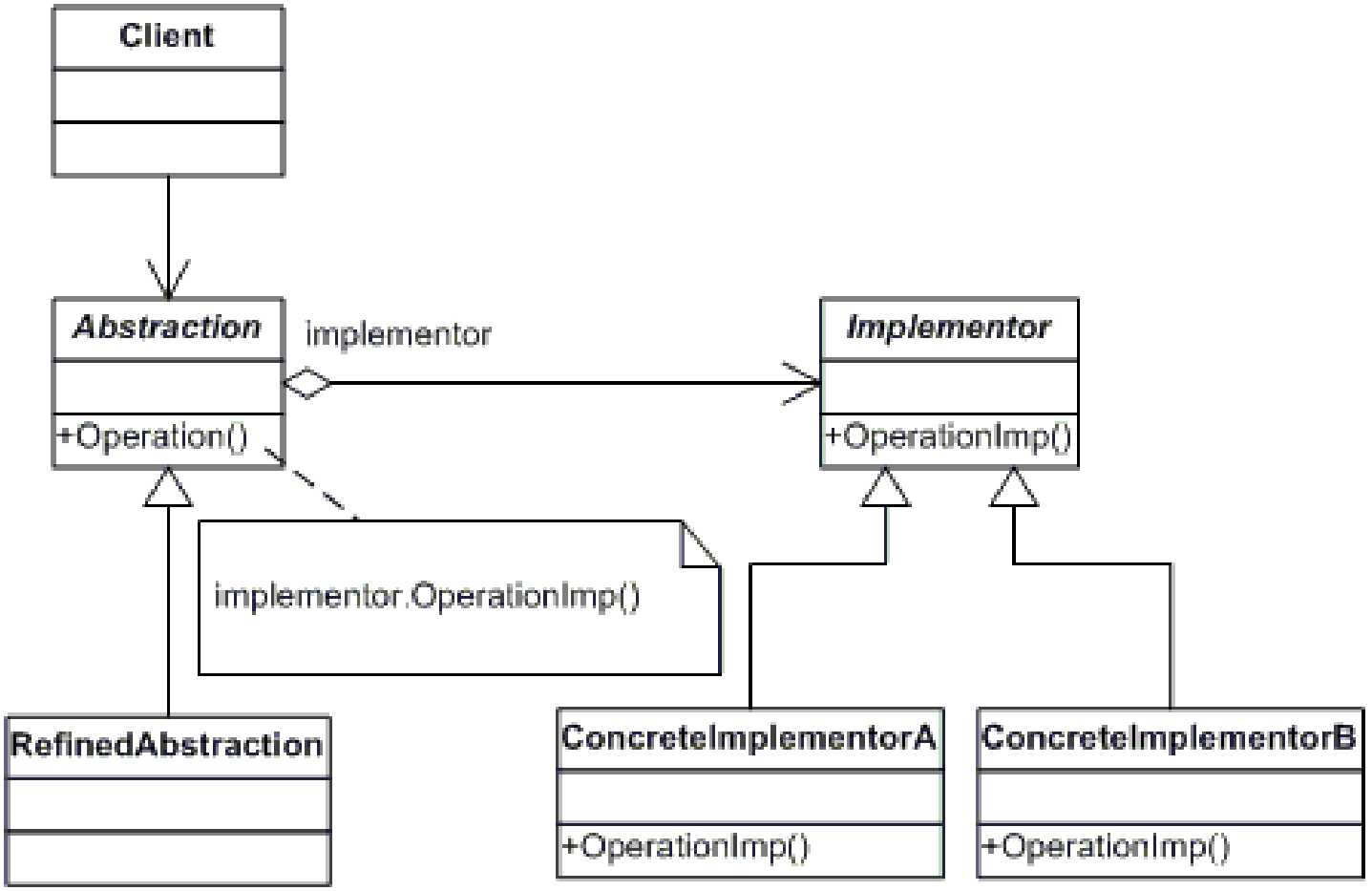
Aplicabilidad

- Use el patrón Bridge cuando:
  - Para evitar un enlace permanente entre una abstracción y su implementación, evitando una relación estática entre ambas, ya que la ligadura se produce en tiempo de ejecución.
  - Cuando tanto la abstracción como la implementación van a formar un árbol de clases derivadas. El patrón permite combinarlas y extenderlas de forma independiente.
  - Para evitar recompilar los clientes cuando cambien las implementaciones.
  - Los cambios en la implementación no deben ser visibles a los clientes.

Estructura


class Diagram
Cliente
Abstraccion
+ Operacion()
Implementador
+ OperacionImp()
Operacion(){
implementador.OperacionImp();
}
AbstraccionRefinada
+ Operacion()
ImplementadorConcretoA
+ OperacionImp()
ImplementadorConcretoB
+ OperacionImp()

Estructura


Client
Abstraction
implementor
+Operation()
Implementor
+OperationImp()
implementor.OperationImp()
RefinedAbstraction
ConcreteImplementorA
+OperationImp()
ConcreteImplementorB
+OperationImp()

Estructura. Participantes

- **Abstracción**: Define la interfaz de la abstracción y mantiene una referencia al objeto Implementador.
- **AbstracciónRefinada**: Extienden la interfaz definida por Abstracción.
- **Implementador**: Define la interfaz de las clases de implementación, que no se tiene que corresponder con la de Abstracción.
- **ImplementadorConcreto**: Implementa la interfaz Implementador y define su implementación concreta.

Estructura. Variaciones

- **Variaciones del patrón:**

- **Implementador único.** En ciertas ocasiones solo tendremos una clase implementador, que sirve a muchas clases de abstracción. Si hay una única clase implementador no es necesario definir una jerarquía de clases.

Consecuencias

- Se desacopla interfaz e implementación: ya que la implementación no está vinculada permanentemente a la interfaz, y se puede determinar en tiempo de ejecución (incluso cambiar). Además se eliminan dependencias de compilación y se consigue una arquitectura más estructurada en niveles.
- Se mejora la extensibilidad: ya que las jerarquías de abstracción y de implementación pueden evolucionar independientemente.
- Se esconden detalles de implementación a los clientes.

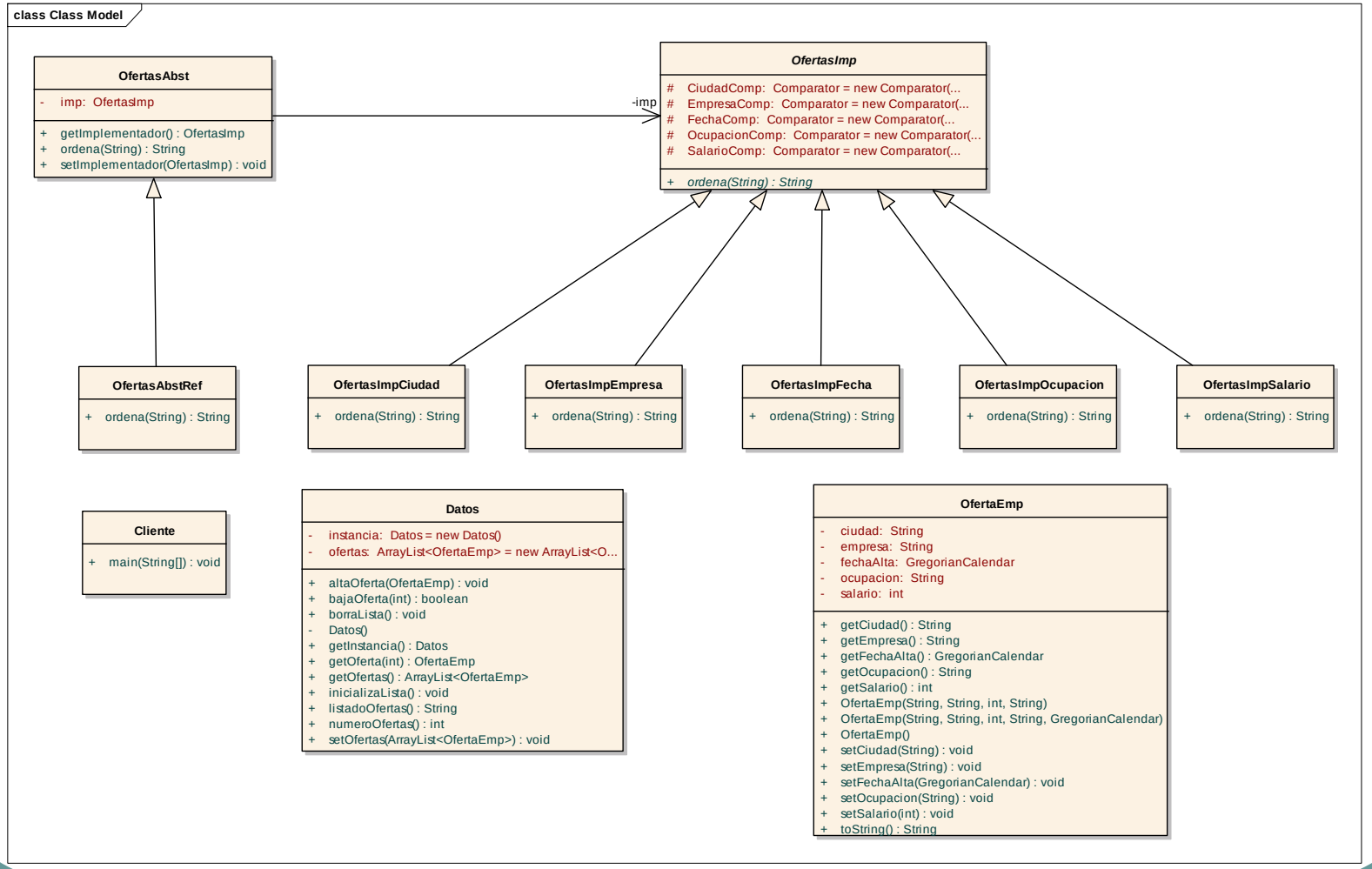
Patrones relacionados

- **Abstrac Factory**: El patrón factoría abstracta puede ser usado por el patrón Bridge para enlazar al objeto de la clase abstracción una implementación concreta.
- **Adapter**: El patrón Adaptador se suele utilizar entre clases ya diseñadas, el patrón Bridge se suele utilizar al comienzo del proyecto, cuando tanto las abstracciones como las implementaciones están en evolución.
- **Singleton**: Utilizado si tenemos un implementador único y es compartido por muchas clases abstracción.

Código de ejemplo

OFERTAS DE EMPLEO

Código de ejemplo



Código de ejemplo

- Identificamos a continuación los elementos del patrón:
  - Abstracción: OfertasAbst.
  - AbstracciónRefinada: OfertasAbstRef.
  - Implementador: OfertasImp.
  - ImplementadorConcreto: OfertasImpCiudad, OfertasImpEmpresa, OfertasImpFecha, OfertasImpOcupacion, OfertasImpSalario.