Índice

1. Birdge

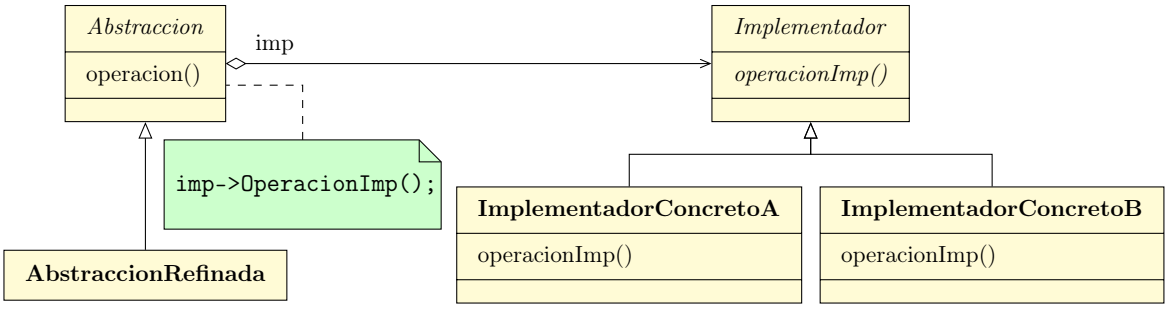
1.1. Propósito

Desacopla una abstracción de una implementación de manera tal que ambas puedan variar de forma independiente.

1.2. Aplicabilidad

- Evitar un enlace permanente entre una abstracción y su implementación. Por ejemplo, cuando debe seleccionarse o cambiarse la implementación en tiempo de ejecución.
- Evitar que los cambios en la implementación de una abstracción impacten en el código cliente; es decir, su código no tendría que ser recompilado.
- Extender de forma independiente las diferentes abstracciones y sus implementaciones.
- Compartir una implementación entre varios objetos.

1.3. Estructura



1.4. Participantes

Abstracción

- Define la interfaz de abstracción.
- Mantiene una referencia a un objeto de tipo Implementador.

Abstracción Refinada

- Extiende la interfaz definida por Abstracción.

Implementador

- Define la interfaz de las clases de implementación. Esta interfaz no tiene por que corresponderse exactamente con la de Abstracción; de hecho, ambas interfaces pueden ser muy distintas. Normalmente la interfaz Implementador solo proporciona operaciones primitivas, y Abstracción define operaciones de mas alto nivel basadas en dichas primitivas.

Implementador Concreto

- Implementa la interfaz Implementador y define su implementación concreta.

1.5. Colaboraciones

Abstracción redirige las peticiones del cliente a su objeto Implementador.

1.6. Consecuencias

- *Desacopla la interfaz y la implementación.* No une permanentemente una implementación a una interfaz, sino que la implementación puede configurarse en tiempo de ejecución. Incluso es posible que un objeto cambie su implementación en tiempo de ejecución.

  Además, este desacoplamiento potencia una división en capas que puede dar lugar a sistemas mejor estructurados. La parte de alto nivel de un sistema sólo tiene que conocer a Abstracción y a Implementador.

- *Mejora la extensibilidad.* Podemos extraer las jerarquías de Abstracción y de Implementador de forma independiente.
- *Oculta detalles de implementación a los clientes.* Podemos aislar a los clientes de los detalles de implementación, como el compartimiento de objetos implementadores y el correspondiente mecanismo de conteo de referencias (si es que hay alguno).

1.7. Patrones relacionados

- El patrón Abstract Factory puede crear y configurar el Bridge.
- El patrón Adapter esta orientado a conseguir que trabajen juntas clases que no están relacionadas. Normalmente se aplica a sistemas que ya han sido diseñados. El patrón Bridge, por otro lado, se usa al comenzar un diseño para permitir que abstracciones e implementaciones varíen independientemente unas de otras.

1.8. Documentación

Pattern based on	**Nombre en del diseño** Bridge		
because	Fundamentación de la elección del patrón en términos de: ▪ Los cambios que este admite y los cambios probables anticipados en el diseño concreto. ▪ Las necesidades funcionales de alguna parte del sistema. ▪ Las restricciones de diseño que se deseen imponer.		
where	*Abstraccion*	**is**	Elemento del diseño
	operacion()	**is**	Elemento del diseño
	imp	**is**	Elemento del diseño
	AbstraccionRefinada	**is**	Elemento del diseño
	Implementador	**is**	Elemento del diseño
	operacionImp()	**is**	Elemento del diseño
	ImplementadorConcretoA	**is**	Elemento del diseño
	ImplementadorConcretoB	**is**	Elemento del diseño
comments	Explicación coloquial de la relación entre los elementos del patrón y los elementos del diseño concreto; otros comentarios adicionales que ayuden a entender cómo se aplica el patrón de diseño		

2. Composite

2.1. Propósito

2.2. Aplicabilidad

2.3. Estructura

2.4. Participantes

2.5. Colaboraciones

2.6. Consecuencias

2.7. Patrones relacionados

2.8. Documentación

Pattern	**Nombre**	
based on	Composite	
because		
where	*Componente*	**is**
	operacion()	**is**
	añadir(Componente)	**is**
	eliminar(Componente)	**is**
	obtenerHijo(int)	**is**
	Compuesto	**is**
	Hoja	**is**
	hijos	**is**
comments		

3. Command

3.1. Propósito

3.2. Aplicabilidad

3.3. Estructura

3.4. Participantes

3.5. Colaboraciones

3.6. Consecuencias

3.7. Patrones relacionados

3.8. Documentación

Pattern	**Nombre**	
based on	Command	
because		
where	Invocador	**is**
	Orden	**is**
	ejecutar()	**is**
	Receptor	**is**
	Accion	**is**
	OrdenConcreta	**is**
	estado	**is**
	receptor	**is**
comments		

4. Abstract Factory

4.1. Propósito

4.2. Aplicabilidad

4.3. Estructura

4.4. Participantes

4.5. Colaboraciones

4.6. Consecuencias

4.7. Patrones relacionados

4.8. Documentación

Pattern	**Nombre**	
based on	Abstract Factory	
because		
where	*FabricaAbstracta*	**is**
	crearProductoA()	**is**
	crearProductoB()	**is**
	FabricaConcreta1	**is**
	FabricaConcreta2	**is**
	ProductoAbstractoA	**is**
	ProductoA1	**is**
	ProductoA2	**is**
	ProductoAbstractoB	**is**
	ProductoB1	**is**
	ProductoB2	**is**
comments		

5. Strategy

5.1. Propósito

5.2. Aplicabilidad

5.3. Estructura

5.4. Participantes

5.5. Colaboraciones

5.6. Consecuencias

5.7. Patrones relacionados

5.8. Documentación

Pattern	**Nombre**	
based on	Strategy	
because		
where	Contexto	**is**
	interfazContexto()	**is**
	estrategia	**is**
	Estrategia	**is**
	interfazAlgoritmo()	**is**
	EstrategiaConcretaA	**is**
	EstrategiaConcretaB	**is**
comments		

6. Iterator

6.1. Propósito

6.2. Aplicabilidad

6.3. Estructura

6.4. Participantes

6.5. Colaboraciones

6.6. Consecuencias

6.7. Patrones relacionados

6.8. Documentación

Pattern	**Nombre**	
based on	Iterator	
because		
where	*Agregado*	**is**
	crearIterador()	**is**
	AgregadoConcreto	**is**
	Iterador	**is**
	primero()	**is**
	siguente()	**is**
	haTerminado()	**is**
	elementoActual()	**is**
	IteradorConcreto	**is**
comments		

7. Visitor

7.1. Propósito

7.2. Aplicabilidad

7.3. Estructura

7.4. Participantes

7.5. Colaboraciones

7.6. Consecuencias

7.7. Patrones relacionados

7.8. Documentación

Pattern	**Nombre**	
based on	Visitor	
because		
where	*Visitante*	**is**
	visitanteConcretoElementoA(ElementoConcretoA)	**is**
	visitanteConcretoElementoB(ElementoConcretoB)	**is**
	VisitanteConcreto1	**is**
	VisitanteConcreto2	**is**
	EstructuraDeObjetos	**is**
	Elemento	**is**
	aceptar(Visitante)	**is**
	ElementoConcretoA	**is**
	operacionA()	**is**
	ElementoConcretoB	**is**
	operacionB()	**is**
comments		

8. Wrapper

8.1. Propósito

8.2. Aplicabilidad

8.3. Estructura

8.4. Participantes

8.5. Colaboraciones

8.6. Consecuencias

8.7. Patrones relacionados

8.8. Documentación

Pattern	**Nombre**	
based on	Wrapper	
because		
where	*Objetivo*	**is**
	peticion()	**is**
	Adaptable	**is**
	peticionConcreta()	**is**
	Adaptador	**is**
	implementacion	**is**
comments		