Clase 7: Patrones de Diseño

Rodrigo Arturo Saffie Kattan

Pontificia Universidad Católica de Chile

rasaffie@ing.puc.cl

23 de agosto de 2016

Contenidos

Repaso Clase Anterior

¿Qué es un patrón de diseño?

"[...] allows the software engineering community to capture design knowledge in a way that enables it to be reused." [Pressman, 2009]

Repaso Clase Anterior

¿Qué es un patrón de diseño?

Se caracteriza por:

- Un nombre
- Un problema
- Una solución
- Sus consecuencias

Repaso Clase Anterior

En este curso se estudiarán los 23 patrones definidos por [GoF, 1994]:

"The design patterns in this book are descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context."

Estos patrones son aplicados a OOP, pero existen **muchos** más

Repaso Clase Anterior

Existen 3 clasificaciones:

- Creacionales
- Estructurales
- De comportamiento

Patrones de Diseño

Creacionales

- Se centran en la creación, composición y representación de los objetos

Abstract Factory

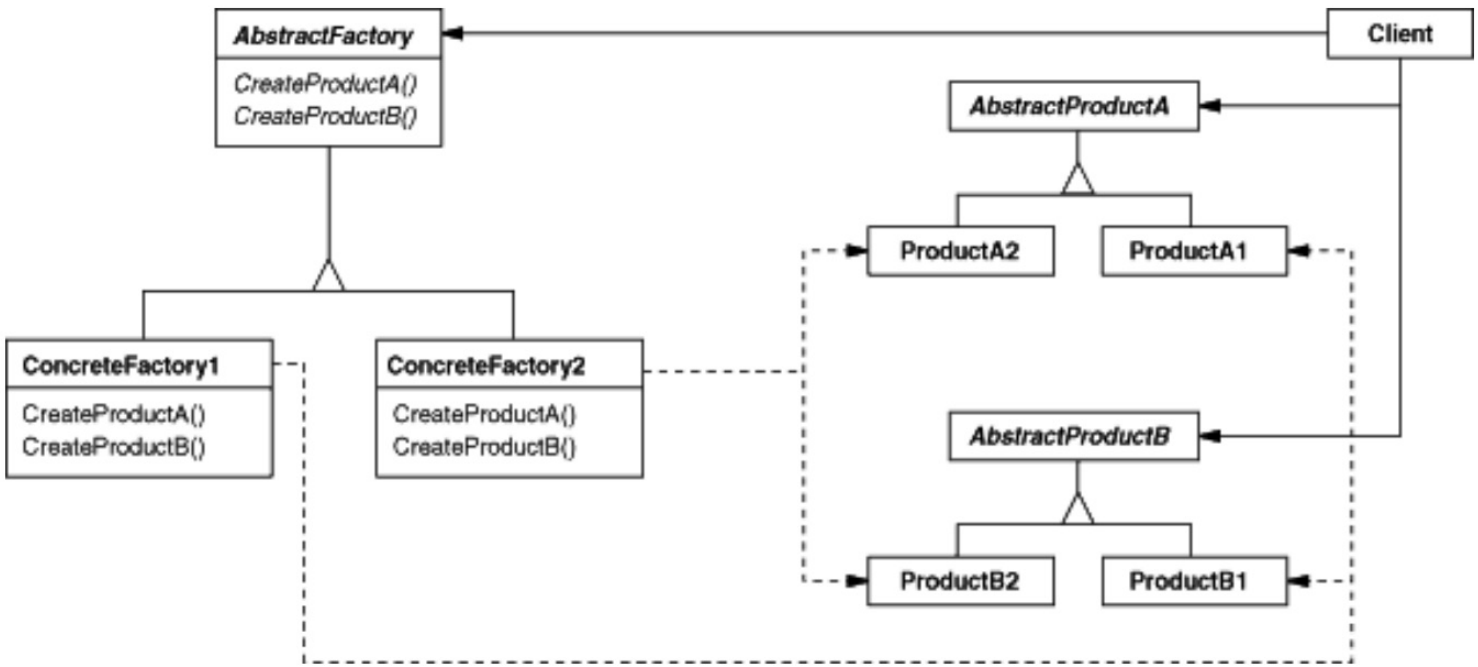
Provee una interfaz para la creación de familias de objetos relacionados o dependientes, sin especificar concretamente sus clases.

¿Cuándo se utiliza?

- Un sistema debe ser independiente de los productos que crea, compone y representa
- Un sistema debe ser configurado con una entre múltiples familias de productos
- Una familia de productos relacionados está diseñada para ser utilizada en conjunto, y se debe forzar esta regla
- Se desea proveer una librería de productos, pero ocultar su implementación

Abstract Factory

Provee una interfaz para la creación de familias de objetos relacionados o dependientes, sin especificar concretamente sus clases.



Ejemplo : Abstract Factory

Builder

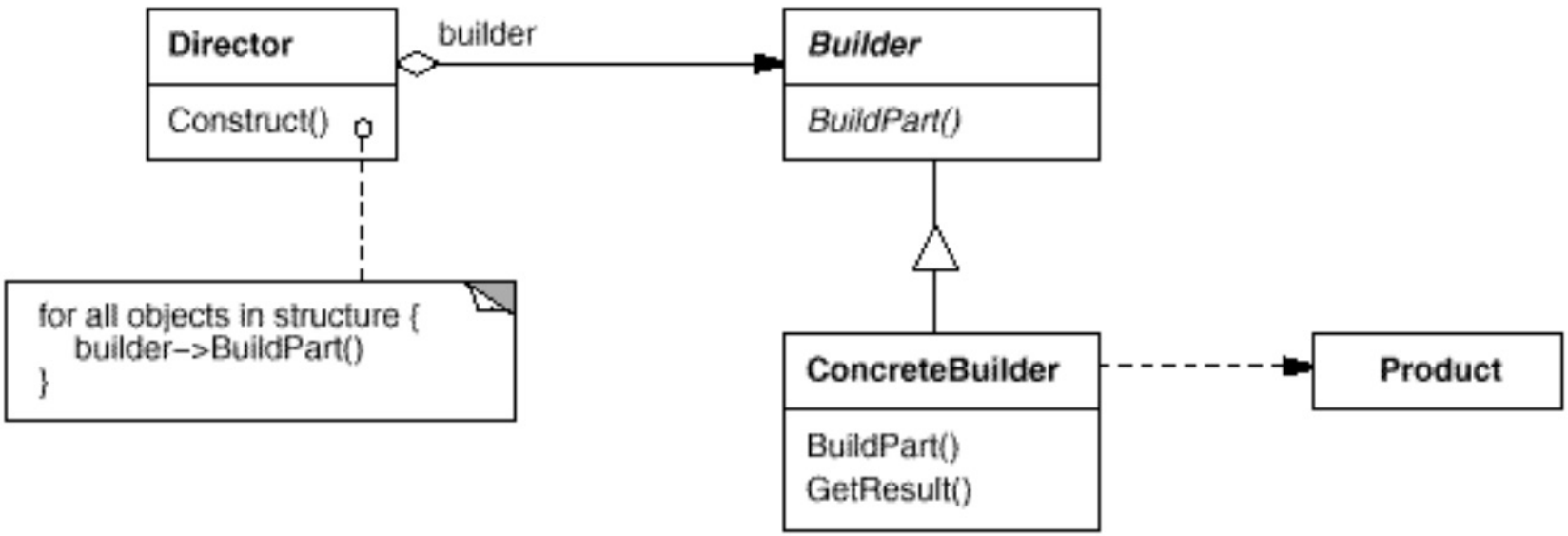
Separa la construcción de un objeto complejo de su representación, para que así el mismo proceso de construcción pueda crear diferentes representaciones.

¿Cuándo se utiliza?

- El algoritmo para crear objetos complejos debería ser independiente de las partes que componen el objeto, y de cómo se construyen
- El proceso de construcción debe permitir diferentes representaciones del objeto que se construye

Builder

Separa la construcción de un objeto complejo de su representación, para que así el mismo proceso de construcción pueda crear diferentes representaciones.



Ejemplo : Builder

Factory Method

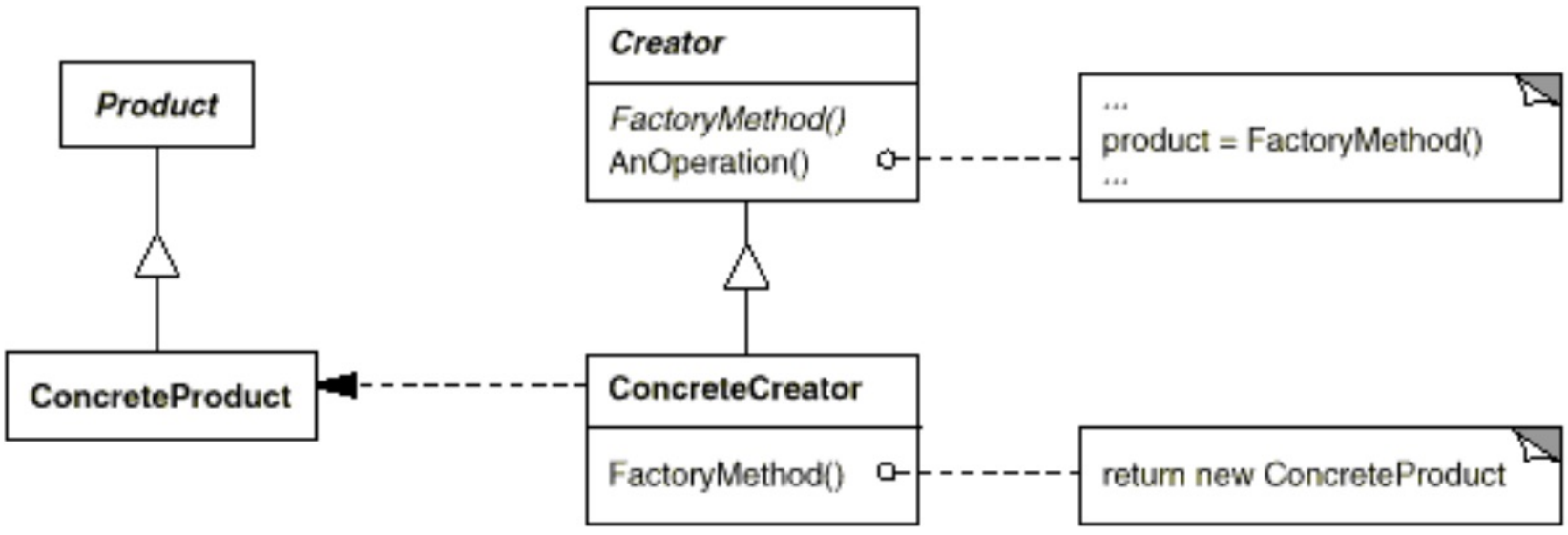
Define una interfaz para la creación de un objeto, pero delega a las subclases que decidan qué clase instanciar.

¿Cuándo se utiliza?

- Una clase no puede anticipar las clases de los objetos que debe crear
- Una clase necesita permitir a sus subclases especificar la creación de objetos
- Una clase delega la responsabilidad de crear objetos a sus subclases, para así encapsular lógica

Factory Method

Define una interfaz para la creación de un objeto, pero delega a las subclases que decidan qué clase instanciar.



Ejemplo : Factory Method

Prototype

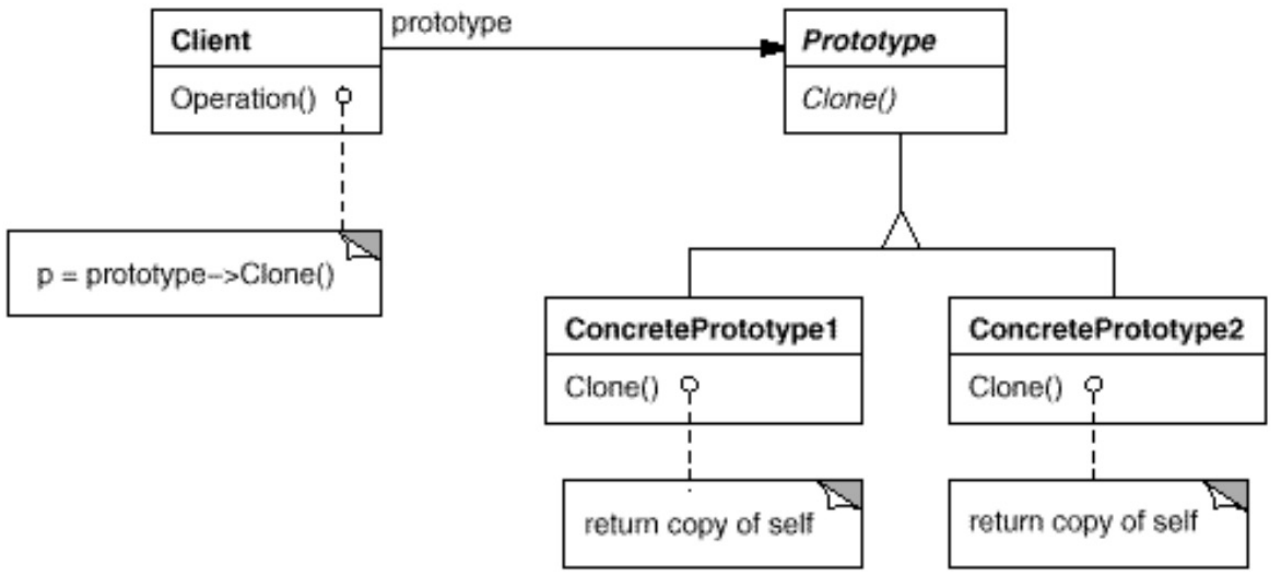
Especifica los tipos de objetos a crear utilizando prototipos, y crea objetos nuevos a través de copias de estos prototipos.

¿Cuándo se utiliza?

- Las clases a crear se especifican en tiempo de ejecución
- Los objetos a crear son similares, y se pueden desprender de un objeto existente
- Instancias de una clase tienen una cantidad limitada de estados. Puede ser más conveniente instanciar una sola vez estos estados, para luego entregar copias.

Prototype

Especifica los tipos de objetos a crear utilizando prototipos, y crea objetos nuevos a través de copias de estos prototipos.



Ejemplo : Prototype

Referencias

Pressman, R. S. (2009)

Software Engineering: A Practitioner's Approach

7th ed., *McGraw-Hill Education*

Gamma, E., Helm. R., Johnson R., Vlissides, J. (1994)

Design Patterns: Elements of Reusable Object-Oriented Software

1st ed., *Addison-Wesley Professional*

dofactory.com

http://www.dofactory.com/net/design-patterns

Fin