PATRONES DE DISEÑO DE CREACIÓN

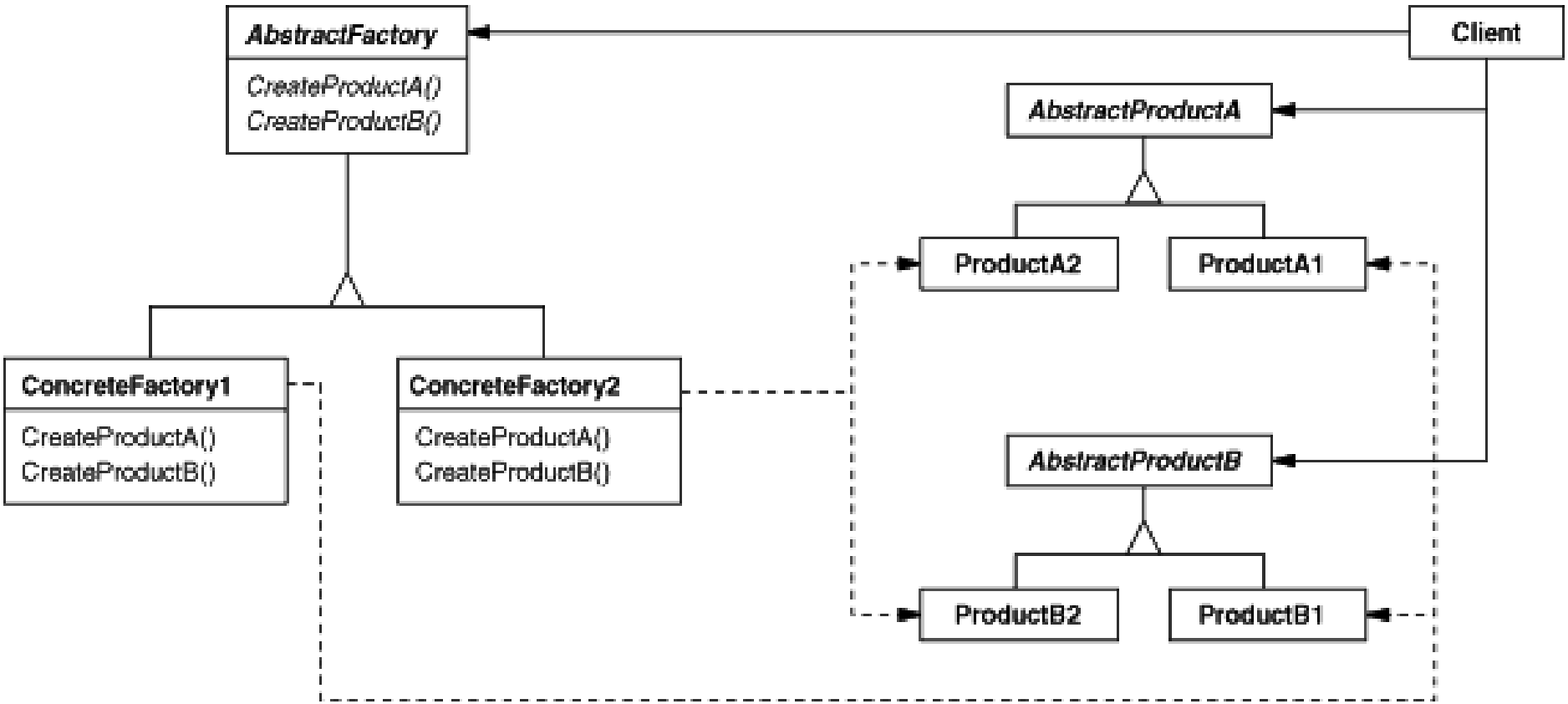
Abstract Factory
Builder
Factory Method
Prototype

Patrones de diseño de creación

- Abstraen el proceso de creación de instancias
- Encapsulan el conocimiento sobre las clases concretas que usa el sistema
- Ocultan como se crean y se asocian las instancias de estas clases
- El sistema conoce solamente las interfaces de los objetos, tal como las definen sus clases abstractas
- Permiten configurar un sistema con objetos que varían mucho en su estructura y funcionalidad
- La configuración puede ser estática (compilación) o dinámica (ejecución)

Abstract Factory

- Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas



Abstract Factory

Aplicabilidad

- Un sistema debe ser independiente de como se crean, componen y representan los productos
- Un sistema debe ser configurado con una familia de productos de entre varias
- Una familia de productos relacionados está diseñada para ser usada conjuntamente, y debe cumplirse esta restricción
- Se quiere proporcionar una biblioteca de clases de productos, y sólo se quiere revelar sus interfaces, no sus implementaciones

Abstract Factory

Colaboraciones

- Normalmente sólo se crea una única instancia de una clase Fábrica Concreta
- La fábrica concreta crea objetos producto que tienen una determinada implementación
- Para crear diferentes objetos producto, los clientes deben usar una fábrica concreta diferente
- Fábrica Abstracta delega la creación de objetos producto en su subclase Fábrica Concreta

Abstract Factory

Consecuencias

- Aisla las clases concretas
- Facilita el intercambio de familias de productos
- Promueve la consistencia entre productos
- Es difícil el agregado de nuevos tipos de productos

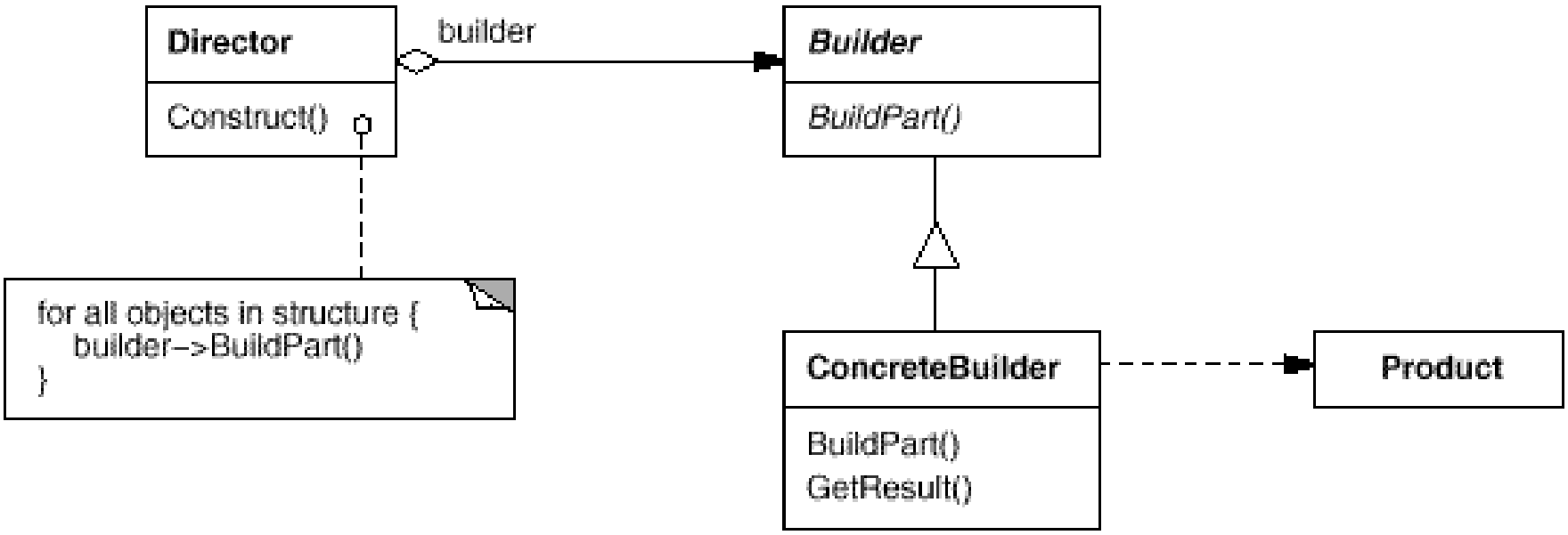
Abstract Factory

Implementación

- Fábricas concretas únicas por cada familia de objetos (Singleton)
- Un método de fabricación para cada producto en la Fábrica Concreta (Factory Method)
- En caso de tener muchas familias de productos, la fábrica concreta puede crear nuevos productos clonando su prototipo (Prototype)
- Fábricas extensibles, definiendo un único método de creación que reciba como parámetro el tipo de objeto a ser creado

Builder

- Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones



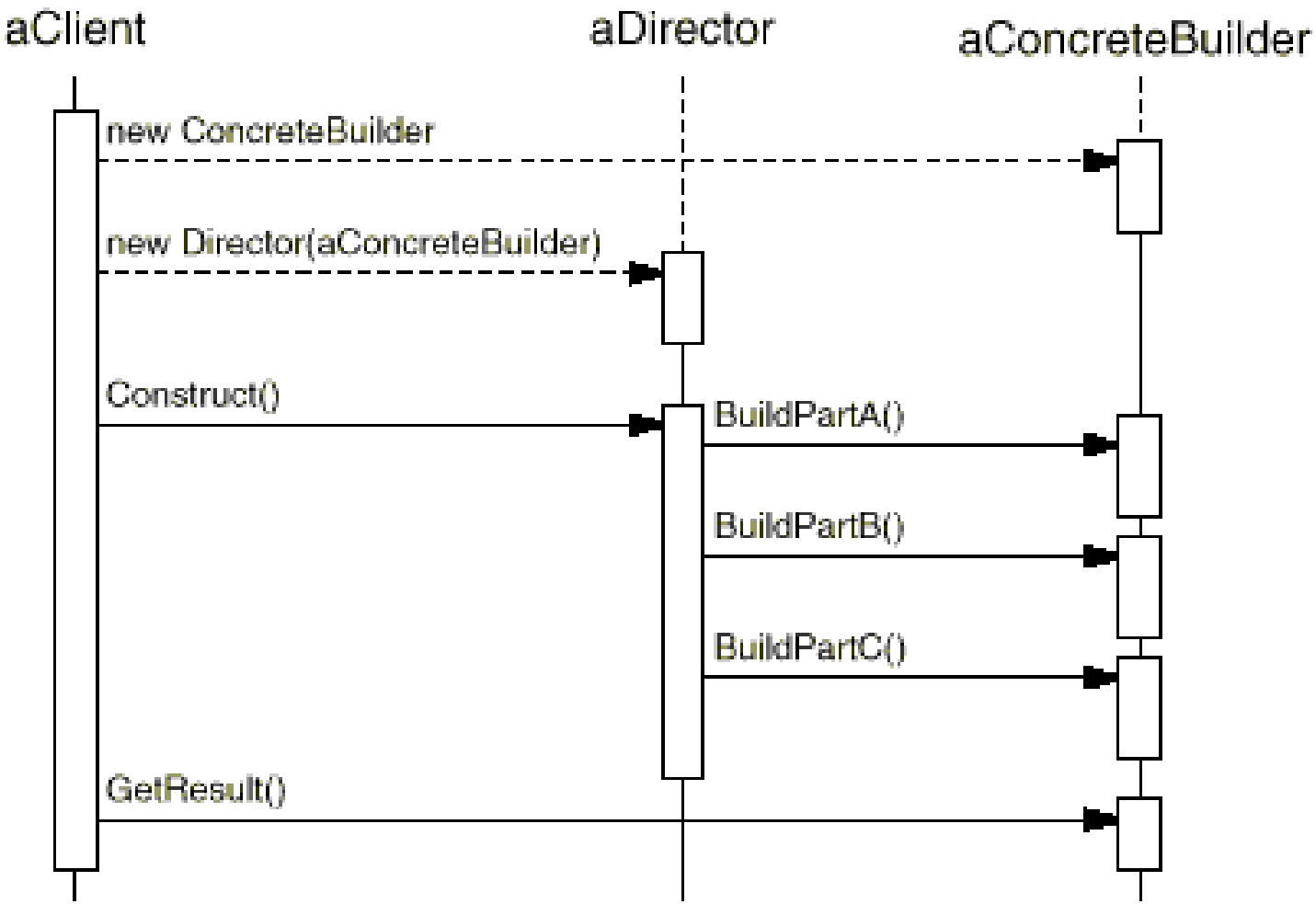
Builder

Aplicabilidad

- El algoritmo para crear un objeto complejo debiera ser independiente de las partes de que se compone dicho objeto y de cómo se ensamblan
- El proceso de construcción debe permitir diferentes representaciones del objeto que está siendo construido

Builder

Colaboraciones



Builder

Consecuencias

- Permite variar la representación interna de un producto
- Aísla el código de construcción de su representación
- Proporciona un control más fino sobre el proceso de construcción

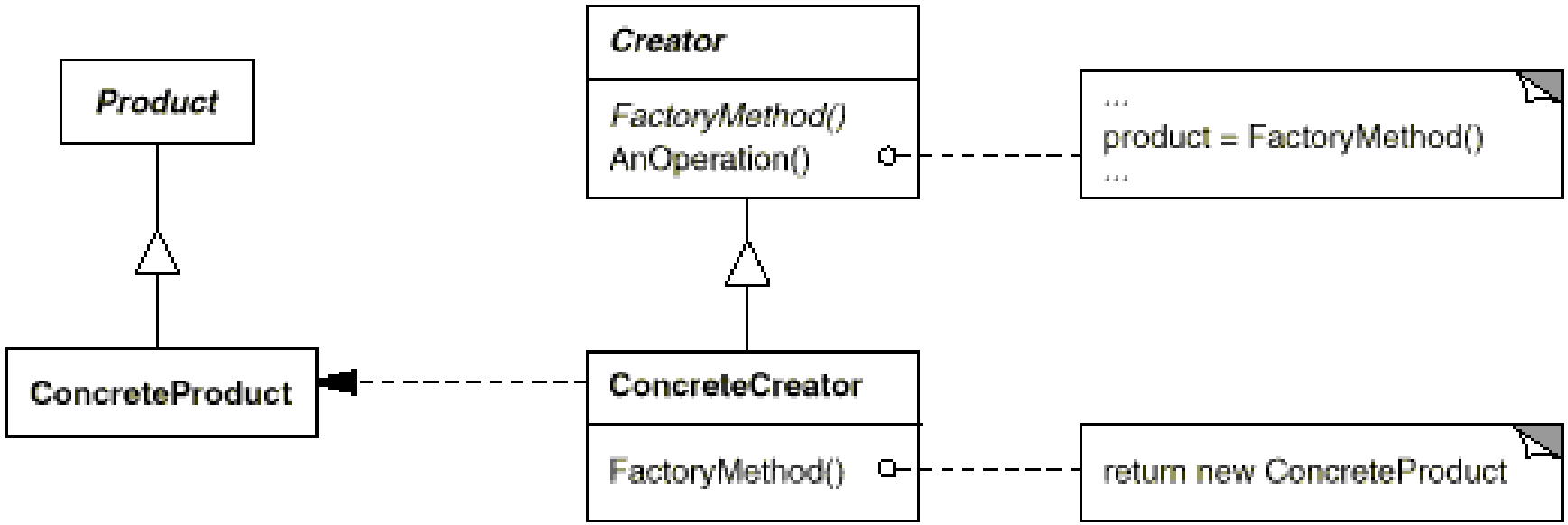
Builder

Implementación

- Normalmente hay una clase Constructor Abstracto que define una operación para cada componente que puede ser creado
- La implementación por defecto de estas operaciones no hace nada
- Una clase Constructor Concreto redefine las operaciones para los componentes que está interesado en crear
- La interfaz de la clase Constructor debe permitir construir productos por parte de todos los tipos de Constructores Concretos

Factory Method

- Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar.
- Permite que una clase delegue en sus subclases la creación de objetos



Factory Method

Aplicabilidad

- Una clase no puede prever la clase de objetos que debe crear
- Una clase quiere que sean sus subclases quienes especifiquen los objetos que ésta crea
- Las clases delegan la responsabilidad en una de entre varias clases auxiliares, y queremos localizar qué subclase de auxiliar concreta es en la que se delega

Factory Method

Colaboraciones

- El Creador se apoya en sus subclases para definir el método de fabricación de manera que éste devuelva una instancia del Producto Concreto apropiado

Factory Method

Consecuencias

- Elimina la necesidad de ligar clases específicas de la aplicación a nuestro código. El código sólo trata con la interfaz Producto
- Proporciona enganches para las subclases
- Conecta jerarquías de clases paralelas
- Los clientes pueden tener que heredar de la clase Creador simplemente para crear un determinado objeto Producto Concreto

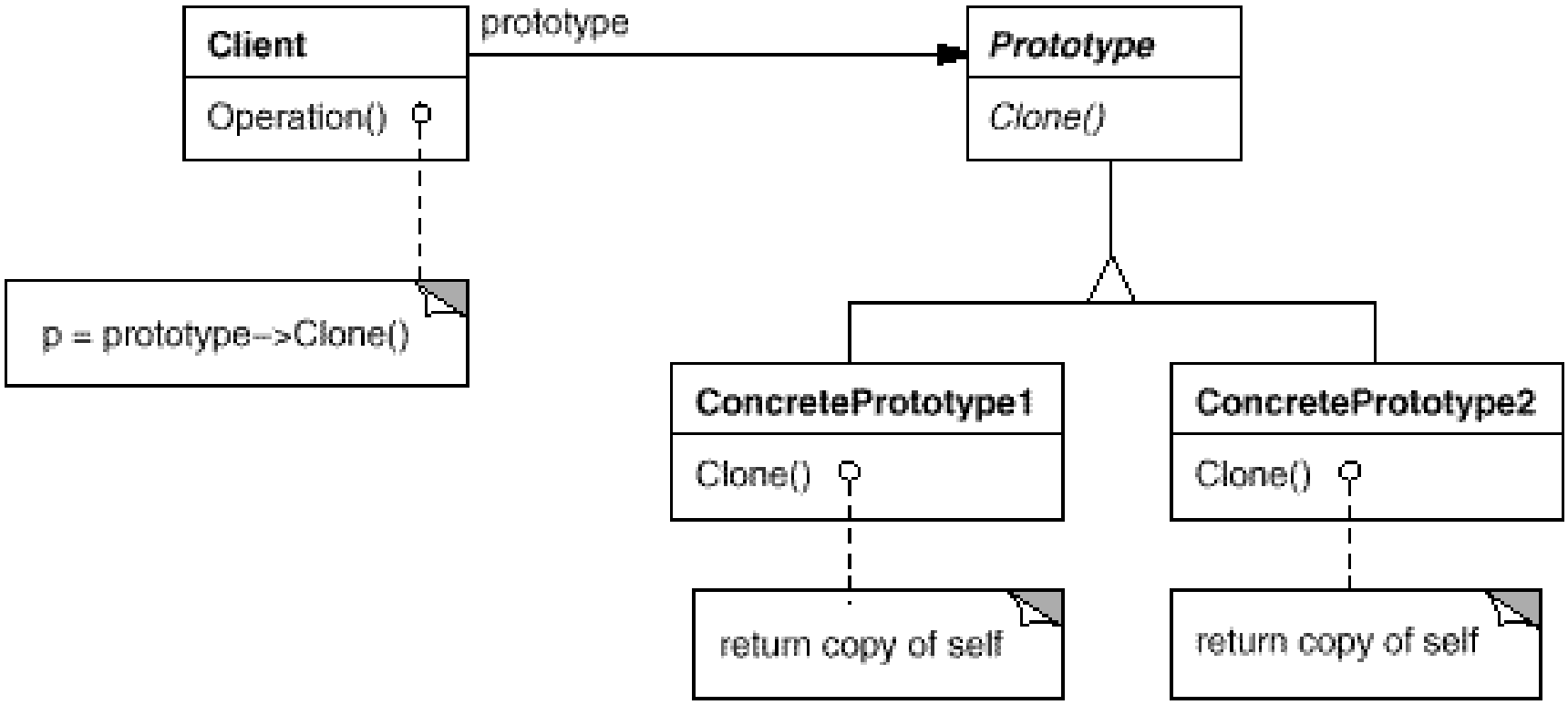
Factory Method

Implementación (Variantes)

- La clase Creador es una clase abstracta y no proporciona una implementación para el método de fabricación que declara
- El Creador es una clase concreta y proporciona una implementación predeterminada del método de fabricación
- Métodos de fabricación parametrizados
- Inicialización lazy del producto en el Creador
- Usar templates (generics) para evitar la herencia
- Convenciones de nombres que dejen en claro que estamos usando métodos de fabricación

Prototype

- Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crea nuevos objetos copiando dicho prototipo



Prototype

Aplicabilidad

- Un sistema debe ser independiente de como se crean, componen y representan los productos
- Las clases a instanciar sean especificadas en tiempo de ejecución
- Para evitar construir una jerarquía de clases de fábricas paralela a la jerarquía de clases de los productos
- Las instancias de una clase puedan tener uno de entre sólo unos pocos estados diferentes

Prototype

Colaboraciones

- Un cliente le pide a un prototipo que se clone

Prototype

Consecuencias

- Oculta al cliente las clases producto concretas
- Permite añadir y eliminar productos en tiempo de ejecución
- Permite especificar nuevos objetos modificando valores
- Reduce la herencia
- Permite configurar dinámicamente una aplicación con clases
- Cada subclase de Prototipo debe implementar la operación «clonar», lo cual puede ser difícil

Prototype

Implementación

- Es útil en con lenguajes estáticos como C++, donde las clases no son objetos, y en los que poca o ninguna información de tipos está disponible en tiempo de ejecución
- Usar un gestor de prototipos
- Implementar la operación Clonar
  - Tener en cuenta las referencias circulares
  - La copia superficial es simple y a menudo suficiente
- Inicializar los clones

Patrones de diseño de creación

Factory Method

Abstract Factory
Builder
Prototype

Herencia

Composición

Flexibilidad

Complejidad

Flexibilidad

Complejidad

Bibliografía

- Design Patterns

  Elements of Reusable Object-Oriented Software

  Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides

FIN