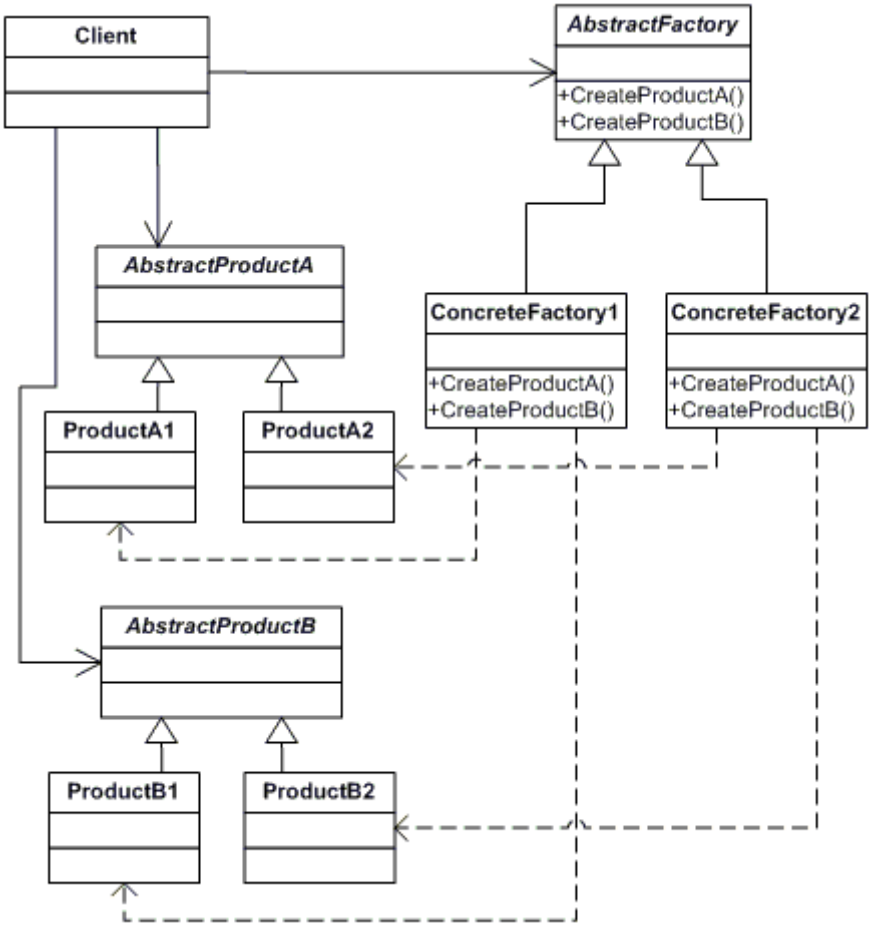
PATRONES DE DISEÑO CREACIONALES

- **Abstract Factory** (Frecuencia de uso: Alta)

  - Interfaz para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas (sin especificarse directamente).



Código Estructural

Salida

ProductB1 interacts with ProductA1
ProductB2 interacts with ProductA2

- Ejemplo

Participantes

Las clases que participan en este patrón son:

AbstractFactory (ContinentFactory)

- declara una interfaz para operaciones que crean productos abstractos

ConcreteFactory (AfricaFactory, AmericaFactory)

- implementa las operaciones para crear objetos de productos concretos.

AbstractProduct (Herbivore, Carnivore)

- declara una interfaz para un tipo de objeto de producto

Product (Wildebeest, Lion, Bison, Wolf)

- define un objeto producto a ser creado por la fábrica de concreto correspondiente
- implementa la interfaz de producto abstracto

Client (AnimalWorld)

- usa interfaces declaradas por las clases AbstractFactory y AbstractProduct

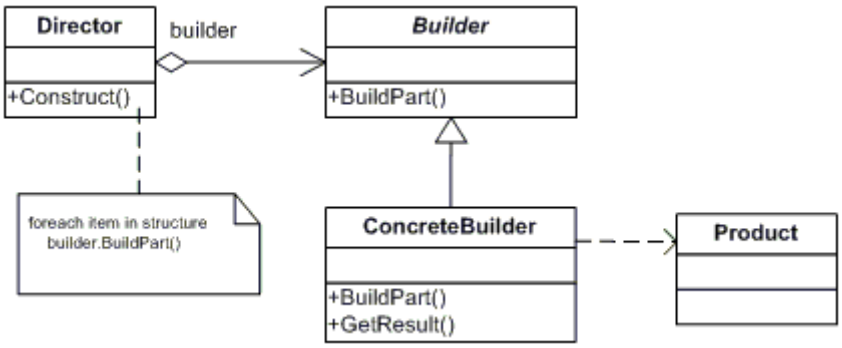
Salida

Lion eats Wildebeest
Wolf eats Bison

Creación de jerarquía paralela de objetos y abstracción de la creación de objetos- No se crean todas las clases dentro de Cliente.

- **Builder** (Frecuencia de uso: Media baja)

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



Código Estructural

Salida

Product Parts ———-
PartA
PartB

Product Parts ———-
PartX
PartY

- Ejemplo

Participantes

Las clases y objetos que participan en este patrón son:

Builder (VehicleBuilder)

- especifica una interfaz abstracta para crear partes de un objeto Product

ConcreteBuilder (MotorCycleBuilder, CarBuilder, ScooterBuilder)

- construye y ensambla partes del producto implementando la interfaz Builder
- define y realiza un seguimiento de la representación que crea
- proporciona una interfaz para recuperar el producto

Director (Shop)

- construye un objeto usando la interfaz de Builder

Product (Vehicle)

- representa el objeto complejo en construcción. ConcreteBuilder construye la representación interna del producto y define el proceso mediante el cual se ensambla
- incluye clases que definen las partes constituyentes, incluidas las interfaces para ensamblar las partes en el resultado final

Salida

Vehicle Type: Scooter
Frame : Scooter Frame
Engine : none

Wheels: 2

Doors : 0

Vehicle Type: Car
Frame : Car Frame
Engine : 2500 cc

Wheels: 4

Doors : 4

Vehicle Type: MotorCycle
Frame : MotorCycle Frame
Engine : 500 cc

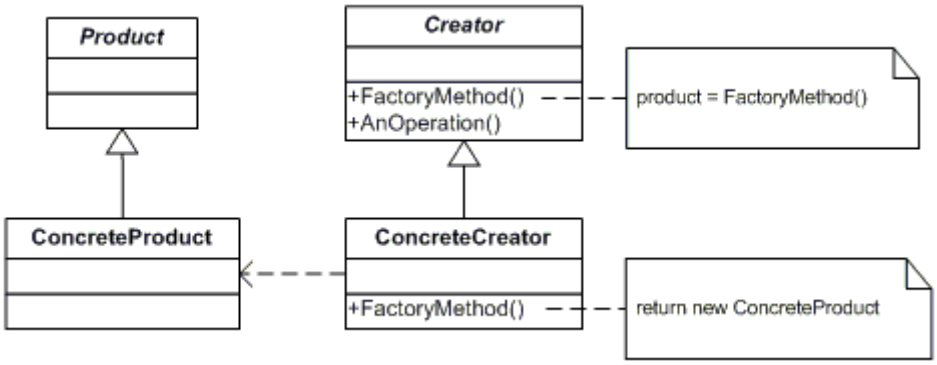
Wheels: 2

Doors : 0

- **Factory Method** (Frecuencia de uso: Alta)

- Define una interfaz para crear un objeto, pero deja que las subclases decidan de qué clase crear una instancia. Factory Method permite que una clase difiera la creación de instancias a subclases.

- Flexibilidad para crear diferentes objetos. Va a haber una clase abstracta que puede proporcionar un obj predeterminado, pero cada subclase va a crear una instancia de una versión extendida del obj.



Código Estructural

Salida

Created ConcreteProductA
Created ConcreteProductB

- Ejemplo

Participantes

Las clases y objetos que participan en este patrón son:

Product (Page)

- define la interfaz de los objetos que crea el Factory method

ConcreteProduct (SkillsPage, EducationPage, ExperiencePage)

- implementa la interfaz del Product

Creator (Document)

- declara el Factory method, que devuelve un objeto de tipo Product. El Creator también puede definir una implementación predeterminada del Factory method que devuelve un objeto ConcreteProduct predeterminado.

ConcreteCreator (Report, Resume)

- Hace override del Factory method para devolver una instancia de ConcreteProduct.

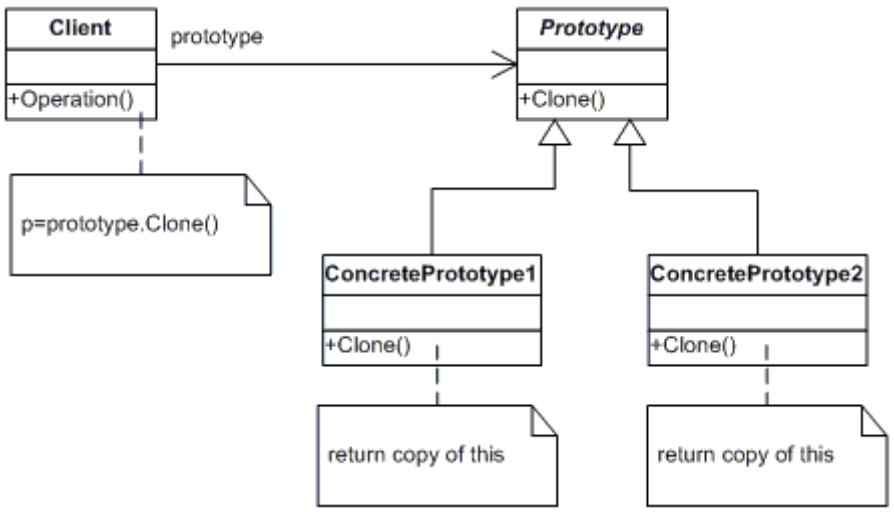
Salida

Resume ———-
SkillsPage
EducationPage
ExperiencePage

Report ———-
IntroductionPage
ResultsPage
ConclusionPage
SummaryPage
BibliographyPage

- **Prototype** (Frecuencia de uso: Media)

- Especifica el tipo de objetos para crear utilizando una instancia prototípica (obj existentes del mismo tipo) y crea nuevos objetos copiando este prototipo.



Código estructural

Output

Cloned: I
Cloned: II

- Ejemplo

Participantes

Las clases y objetos que participan en este patrón son:

Prototype (ColorPrototype)

- declara una interfaz para clonación propia

ConcretePrototype (Color)

- implementa una operación de clonación propia.

Client (ColorManager)

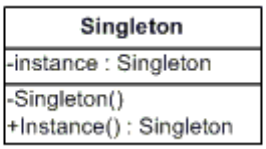
- crea un nuevo objeto pidiendo a un prototipo que se clone a sí mismo

Salida

Cloning color RGB: 255, 0, 0
Cloning color RGB: 128,211,128
Cloning color RGB: 211, 34, 20

- **Singleton** (Frecuencia de uso: Media alta)

- Se asegura de que una clase tenga solo una instancia y proporciona un punto global de acceso a ella.



Código estructural

Salida

Objects are the same instance

- Ejemplo

Participantes

Las clases y objetos que participan en este patrón son:

Singleton (LoadBalancer)

- Define una operación de instancia que permite a los clientes acceder a su instancia única. La instancia es una operación de clase.
- Responsable de crear y mantener su propia instancia única.

Output

Same instance

ServerIII
ServerII
ServerI
ServerII
ServerI
ServerIII
ServerI
ServerIII
ServerIV
ServerII
ServerII
ServerIII
ServerIV
ServerII
ServerIV