Il pattern Factory Method

a cura di **Angelo Furfaro**

da “Design Patterns”, Gamma et al.
“Patterns in Java”, Grand

Dipartimento di
Ingegneria Informatica, Elettronica, Modellistica e Sistemistica
Università della Calabria, 87036 Rende(CS) - Italy
Email: a.furfaro@unical.it
Web: http://angelo.furfaro.dimes.unical.it

Factory Method

Classificazione

- Scopo: creazionale
- Raggio d'azione: basato su classi

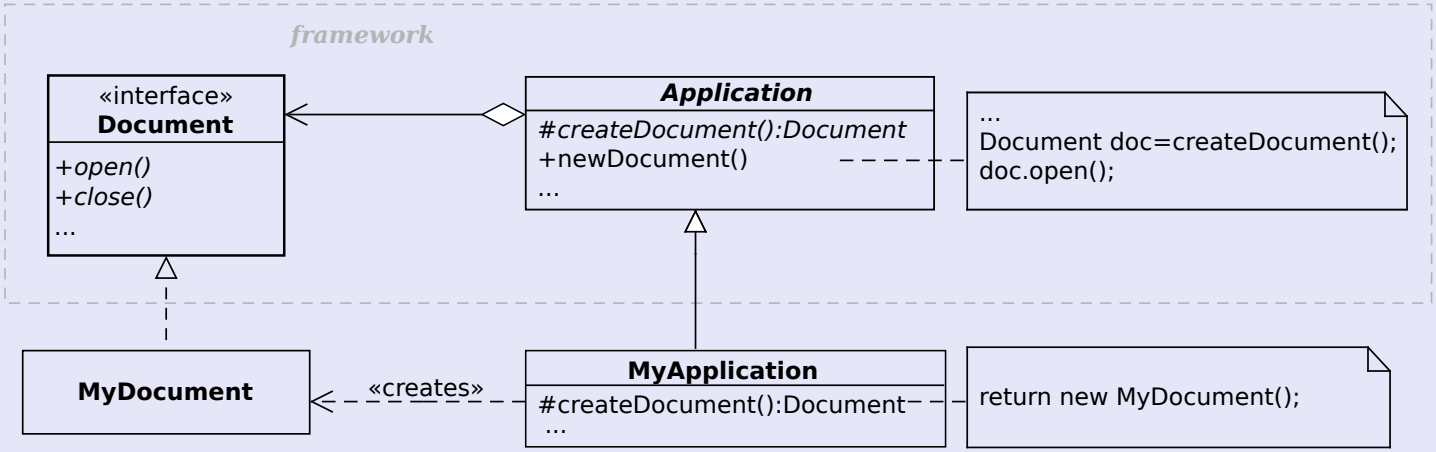
Altri nomi

Virtual Constructor

Scopo

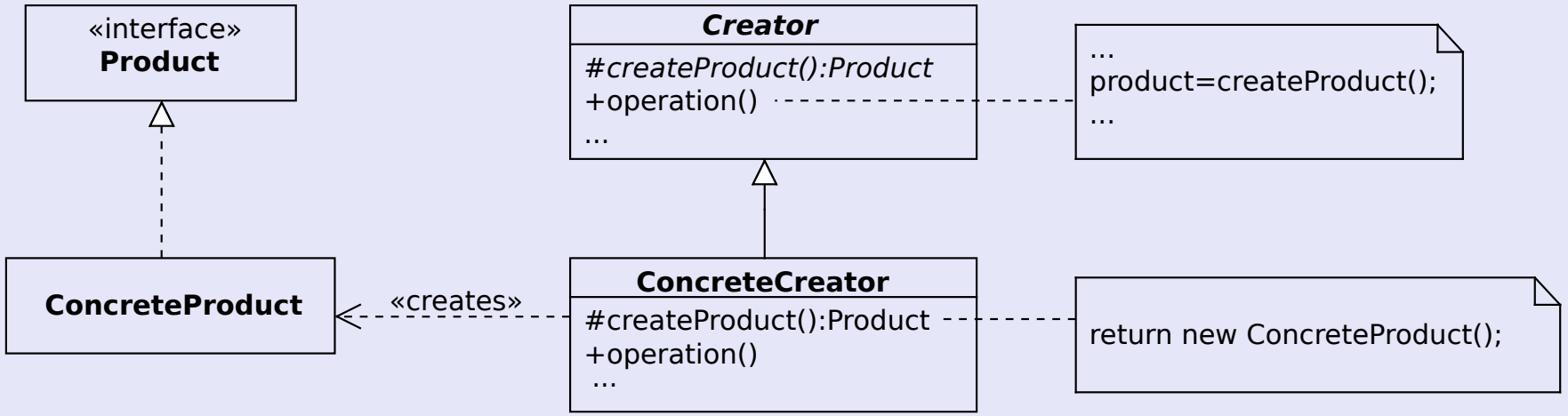
- Definisce un'interfaccia per la creazione di un oggetto, lasciando alle sottoclassi la decisione sulla classe concreta che sarà istanziata.
- Consente di deferire la creazione di un oggetto alle sottoclassi.

Motivazione



- I framework utilizzano classi astratte per definire e mantenere le relazioni tra gli oggetti e spesso sono anche responsabili della creazione di questi oggetti.
- Si consideri un framework per applicazioni in grado di presentare più documenti agli utenti.
- La classe `Application` è responsabile della gestione di oggetti conformi all'interfaccia `Document` e della loro creazione.
- `Application` sa solo *quando* dovrà creare un nuovo documento ma non di che tipo né come dovrà farlo.
- La responsabilità è spostata all'esterno del framework: le sottoclassi di `Application` devono fornire l'implementazione del metodo *factory* `createDocument()` in modo da restituire un'istanza della classe appropriata.

Struttura



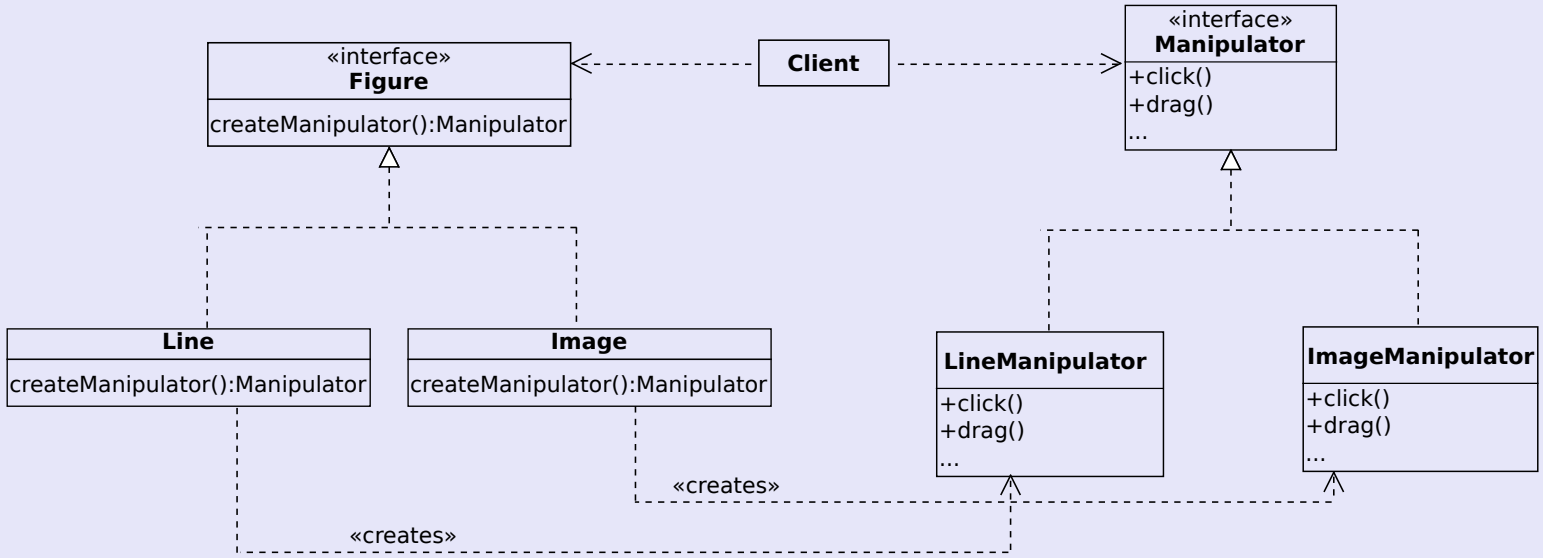
Partecipanti

- **Product** (`Document`):
  definisce l'interfaccia degli oggetti creati dal metodo factory.
- **ConcreteProduct** (`MyDocument`):
  implementa l'interfaccia `Product`.
- **Creator**(`Application`):
  dichiara il metodo factory (protetto nell'esempio), che restituisce un oggetto di tipo `Product`. Può invocare `createProduct()` per creare un prodotto.
- **ConcreteCreator** (`MyApplication`):
  sovrascrive `createProduct()` in modo da restituire una specifica istanza di `ConcreteProduct`.

Conseguenze

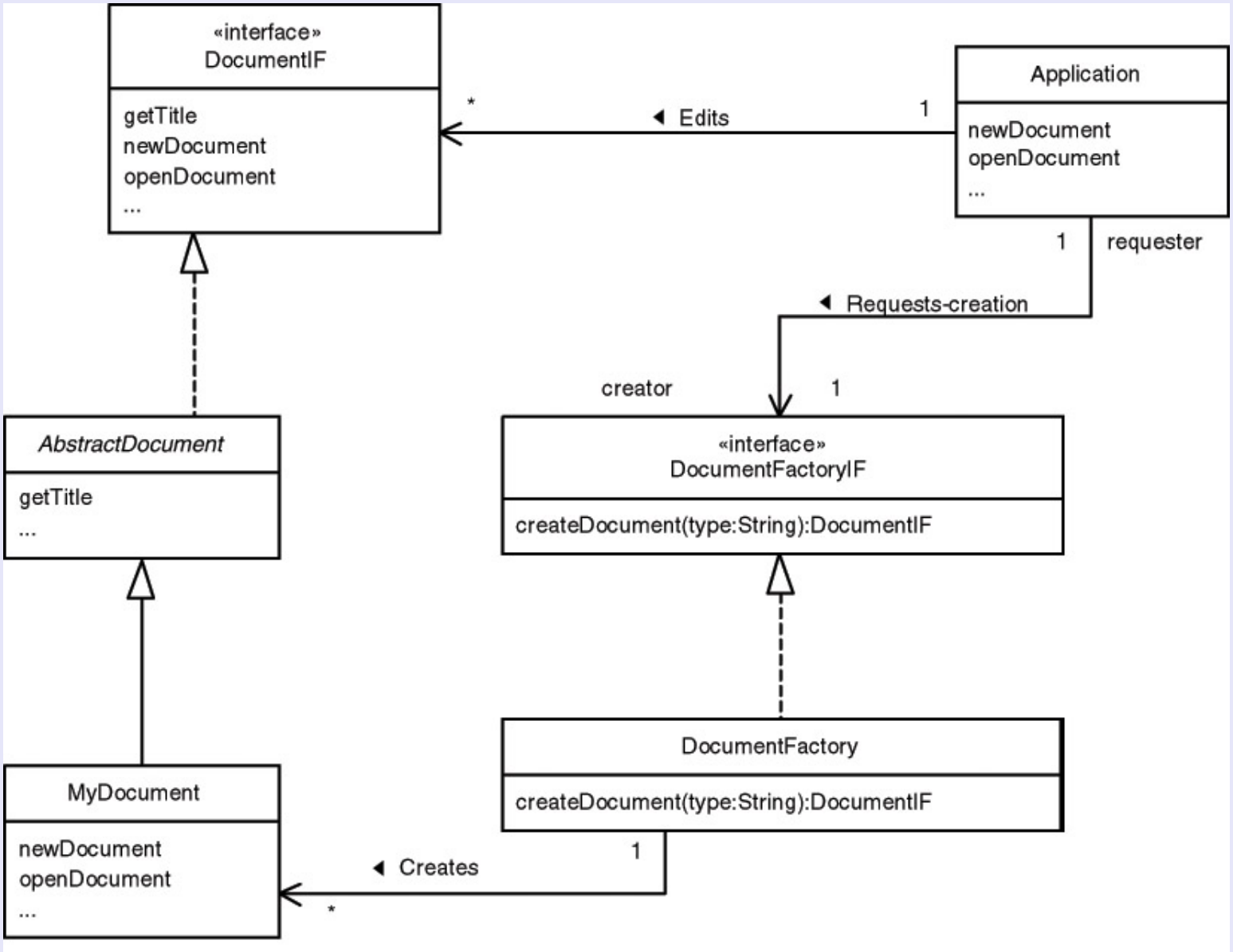
- ☺ Elimina la necessità di riferirsi a classi dipendenti dall'applicazione all'interno del codice del framework.
- ☹ Gli utilizzatori potrebbero essere costretti a definire sottoclassi di `Creator` per creare un particolare oggetto `ConcreteProduct`.
- ☺ Fornisce un punto d'aggancio per le sottoclassi per la produzione di una versione specializzata di un prodotto.
- ☺ Connette gerarchie di classi parallele. Il metodo factory potrebbe invocato da oggetti diversi dal `Creator`. Si veda esempio seguente.

Gerarchie di classi parallele



- `Figure` è il tipo di base di figure geometriche manipolabili graficamente.
- La logica di manipolazione può essere complicata e diversa da figura a figura.
- Si può attribuire la responsabilità di implementare una logica di manipolazione ad oggetti separati introducendo il tipo `Manipulator`
- Si ottiene una gerarchia parallela a quella delle figure. Ogni classe che implementa `Figure` ha il proprio metodo factory per creare l'apposito `Manipulator`

Factory Method parametrico



Esempio: framework Polinomi

Pattern correlati

- Abstract Factory