Il pattern Abstract Factory

a cura di **Angelo Furfaro**

da “Design Patterns”, Gamma et al.

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

Abstract Factory

Classificazione

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

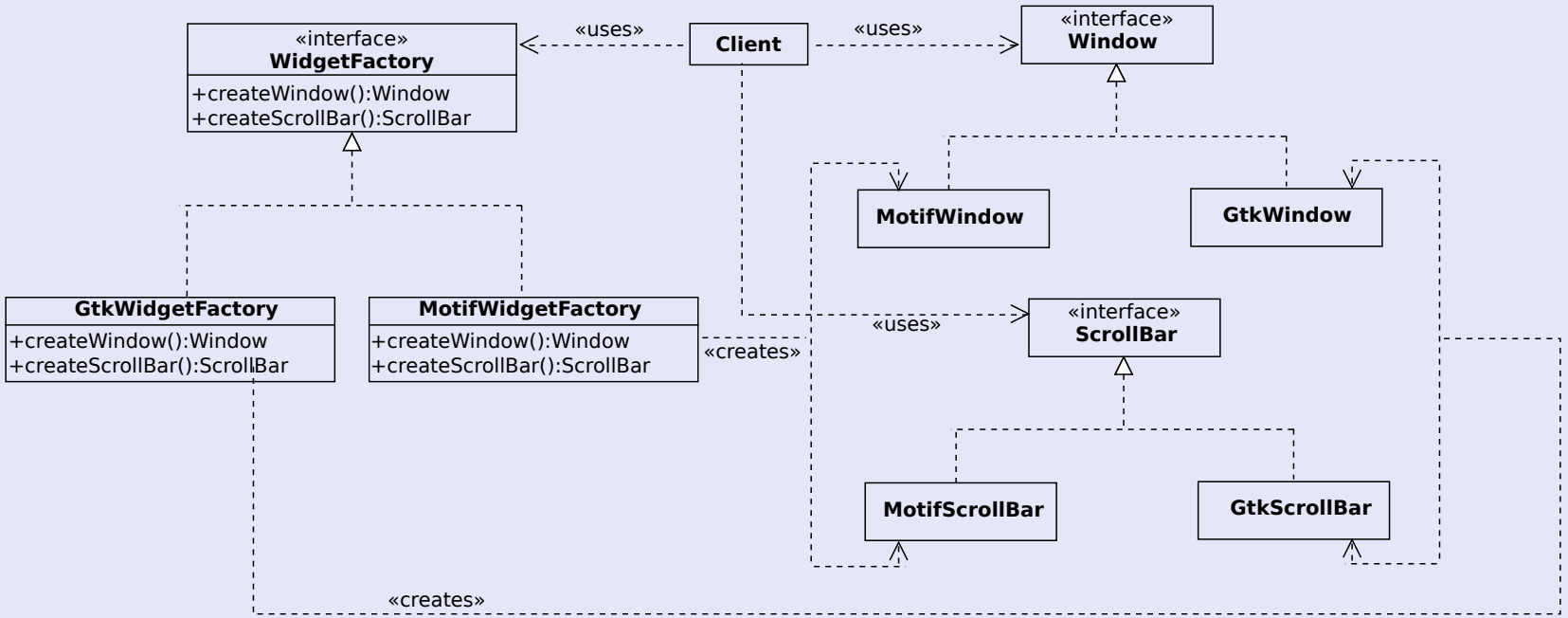
Altri nomi

Kit

Scopo

- Fornire un'interfaccia per la creazione di famiglie di oggetti correlati o dipendenti senza specificare quali siano le loro classi concrete

Motivazione



- Si consideri lo sviluppo di un toolkit per la realizzazione di GUI in grado di supportare diversi look-and-feel (L&F), per esempio Motif e Gtk, che definiscono differenti aspetti e modalità di comportamento per i vari componenti (*widget*).
- Affinché un'applicazione sia portabile su L&F diversi, il codice che la implementa non deve dipendere dal tipo specifico dei widget utilizzati quindi non può istanziarli direttamente.
- L'interfaccia *WidgetFactory* introduce un metodo per la creazione di ciascun tipo di base di widget definito a sua volta da un'oportuna interfaccia (es. *Window* e *ScrollBar*).
- I client invocano i metodi definiti da *WidgetFactory* per ottenere istanze di widget senza conoscere la classe concreta che utilizzano.
- Esiste una classe concreta che implementa *WidgetFactory* per ciascuno dei L&F considerati.

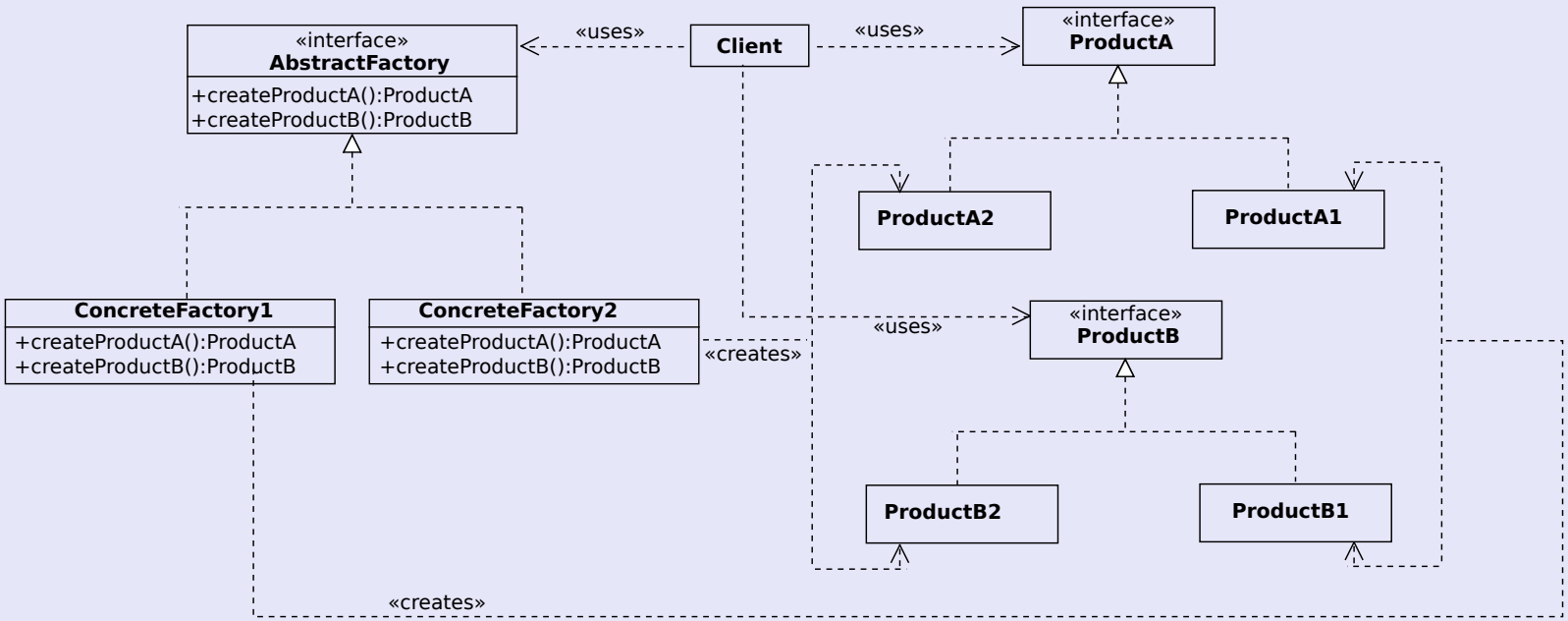
Motivazione

- I client richiedono la creazione degli elementi grafici solo per mezzo di *WidgetFactory* e non hanno alcuna conoscenza delle classi concrete relative a ciascun L&F.
- L'impiego di *WidgetFactory* rafforza le dipendenze tra le classi concrete che rappresentano i diversi oggetti grafici.
- Una scrollbar Motif deve essere utilizzata con una finestra Motif e con dei pulsanti Motif. L'uso di *MotifWidgetFactory* consente di soddisfare tale vincolo.

Applicabilità

- Un sistema deve essere indipendente dalle modalità di creazione, composizione e rappresentazione dei suoi prodotti.
- Un sistema deve poter essere configurato scegliendo tra più famiglie di prodotti.
- Esistono famiglie di oggetti correlati, progettati per essere usati insieme.
- Si intende fornire una libreria di classi, rivelando le interfacce e nascondendo le implementazioni.

Struttura



Partecipanti

- **AbstractFactory** (`WidgetFactory`): dichiara un'interfaccia per le operazioni di creazione di oggetti prodotto astratti.
- **ConcreteFactory** (`MotifWidgetFactory, GtkWidgetFactory`): implementa le operazioni degli oggetti prodotto concreti.
- **AbstractProduct**(`Window, ScrollBar`): dichiara un'interfaccia per un tipo di prodotti
- **ConcreteProduct**(`MotifWindow, GtkWindow`): implementa l'interfaccia `Product` definendo un oggetto prodotto creato dalla corrispondente factory concreta.

Conseguenze

- ☺ Isola le classi concrete. Poiché una classe factory incapsula il processo di creazione dei prodotti concreti rende i clienti indipendenti dalle classi utilizzate per la loro implementazione. I clienti conoscono solo le interfacce dei prodotti.
- ☺ Consente di cambiare in modo semplice la famiglia di prodotti utilizzata. È possibile ottenere una configurazione di oggetti diversi semplicemente cambiando il tipo di factory concreta utilizzata.
- ☺ Promuove la coerenza nell'utilizzo dei prodotti.
- ☹ L'aggiunta del supporto a nuove tipologie di prodotti è difficile. L'aggiunta di un nuovo tipo di prodotto comporta una modifica dell'interfaccia `AbstractFactory` e, di conseguenza, di tutte le classi che la implementano.

Implementazione

- *Factory come Singleton*
  Un'applicazione tipicamente richiede una singola istanza di una factory concreta per ciascuna famiglia di prodotti. Tale istanza è di solito ottenuta per mezzo del pattern Singleton.
- *Creazione dei prodotti*
  `AbstractFactory` definisce un'interfaccia per la creazione di prodotti mentre la responsabilità della creazione effettiva delle istanze compete alle classi `ConcreteProduct`. Ciò può essere realizzato in vari modi:
  - Introduzione di un metodo factory per ciascun prodotto.
  - Ricorso al pattern Prototype: In questo caso la factory concreta viene inizializzata con un'istanza prototipo di ciascun prodotto della famiglia. La creazione dei prodotti avviene per clonazione dei prototipi.