Abstract Factory

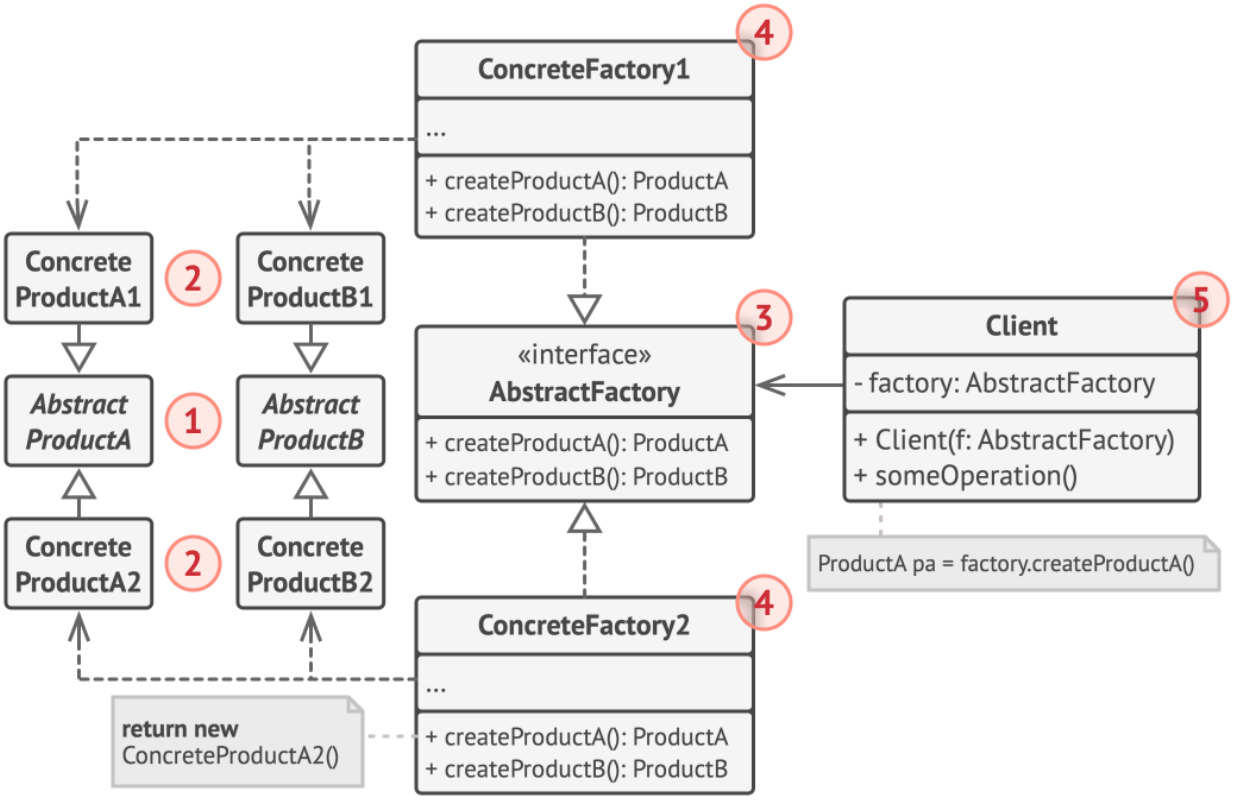
Abstract Factory è un pattern creazionale che consente di produrre famiglie di oggetti correlati senza specificarne le classi concrete.

Tale pattern suggerisce di creare le interfacce per ogni prodotto distinto della famiglia. Poi, è possibile fare in modo che tutte le varianti dei prodotti seguano tali interfacce. Ad esempio tutte le varianti di sedia possono implementare l'interfaccia sedia.

Successivamente, dichiariamo l'Abstract Factory, un'interfaccia con un elenco di metodi di creazione per tutti i prodotti che fanno parte della famiglia di prodotti (createChair, createSofa, createTable ecc...). Questi metodi resituiscono tutti i tipi di prodotti astratti rappresentati dalle interfacce che identifica ogni prodotto (ad esempio chair, sofa, table ecc...).

Adesso per ogni tipologia di famiglia di prodotti creiamo una classe factory separata basata sull'interfaccia Abstract Factory.

Struttura



- I prodotti astratti dichiarano le interfacce per un insieme di prodotti distinti ma correlati che formano una famiglia di oggetti;
- I prodotti concreti sono varie implementazioni delle interfacce, raggruppate per varianti. Ogni prodotto astratto (sedia/divano/tavolo ecc...) deve essere implementato in tutte le varie varianti;
- L'interfaccia Abstract Factory dichiara un set di metodi per creare ciscuno dei prodotti astratti;
- Le concrete factory implementano i metodi di creazione della Abstract Factory. Ogni concrete factory corrisponde a una specifica variante di prodotto e crea solo quella determinata variante;

Applicabilità

- Usare Abstract Factory quando il codice deve funzionare con diverse famiglie di oggetti correlati, ma non devono dipendere dalle classi concrete di tali prodotti;

- Si potrebbe passare da un Factory Method ad un Abstract Factory quando abbiamo una classe con un set di metodi factory.

Vantaggi

- I prodotti di una stessa famiglia sono compatibili tra loro;
- Si evita un accoppiamento stretto tra prodotti concreti a client;
- Principio di singola responsabilità;
- Principio open/closed: si può introdurre nuove varianti di oggetti senza compromettere il codice client esistente;

Svantaggi

- Il codice potrebbe diventare complesso poichè insieme al pattern vengono introdotte numerose nuove interfacce e classi.

Relazioni con altri modelli

- Molti progetti iniziano con Factory method e si evolvono in Abstract Factory, Prototype o Builder;
- Builder si concentra sulla costruzione di oggetti complesso passo dopo passo e permette di eseguire alcuni passaggi di costruzione aggiuntivi prima di recuperare l'oggetto. Abstract Factory è specializzato nella creazione di famiglie di oggetti correlato e restituisce il prodotto immediatamente;
- le Abstract Factory sono spesso basate su un set di Factory Method, ma è anche possibile usare Prototype per comporre i metodi su queste classi;
- Abstract Factory può fungere da alternativa a Facade quando si desidera semplicemente nascondere il modo in cui vengono creati gli oggetti del sottosistema dal codice client;
- E' possibile usare Abstract Factory insieme a Bridge quando alcune astrazioni definite da Bridge possono funzionare solo con implementazioni specifiche. In

questo caso, Abstract Factory può incapsulare queste relazioni e nascondere la complessità al codice client;

- Abstract Factory, Builder e Prototype possono essere tutti implementati come Singleton.