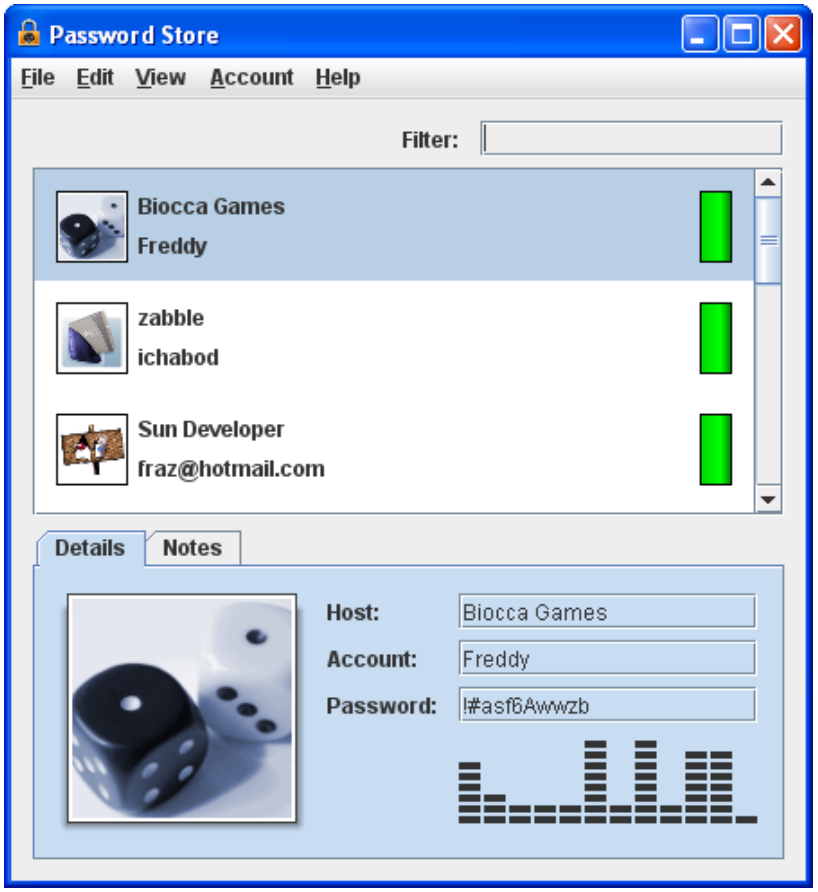
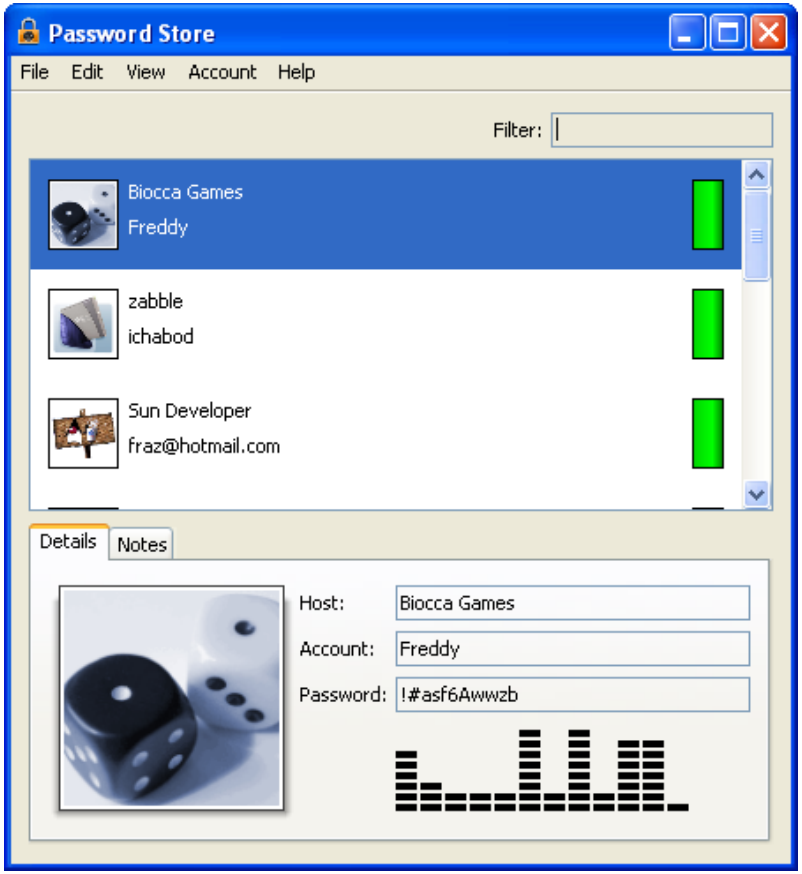
Abstract Factory – Esempio



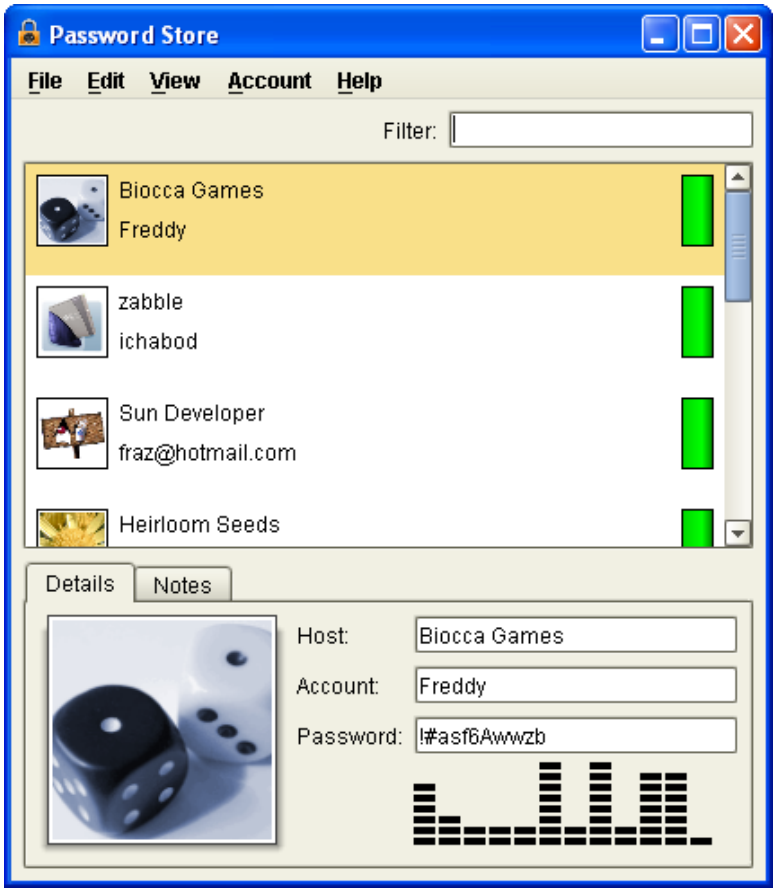
"Java" look and feel

"Windows" look and feel

Abstract Factory – Esempio

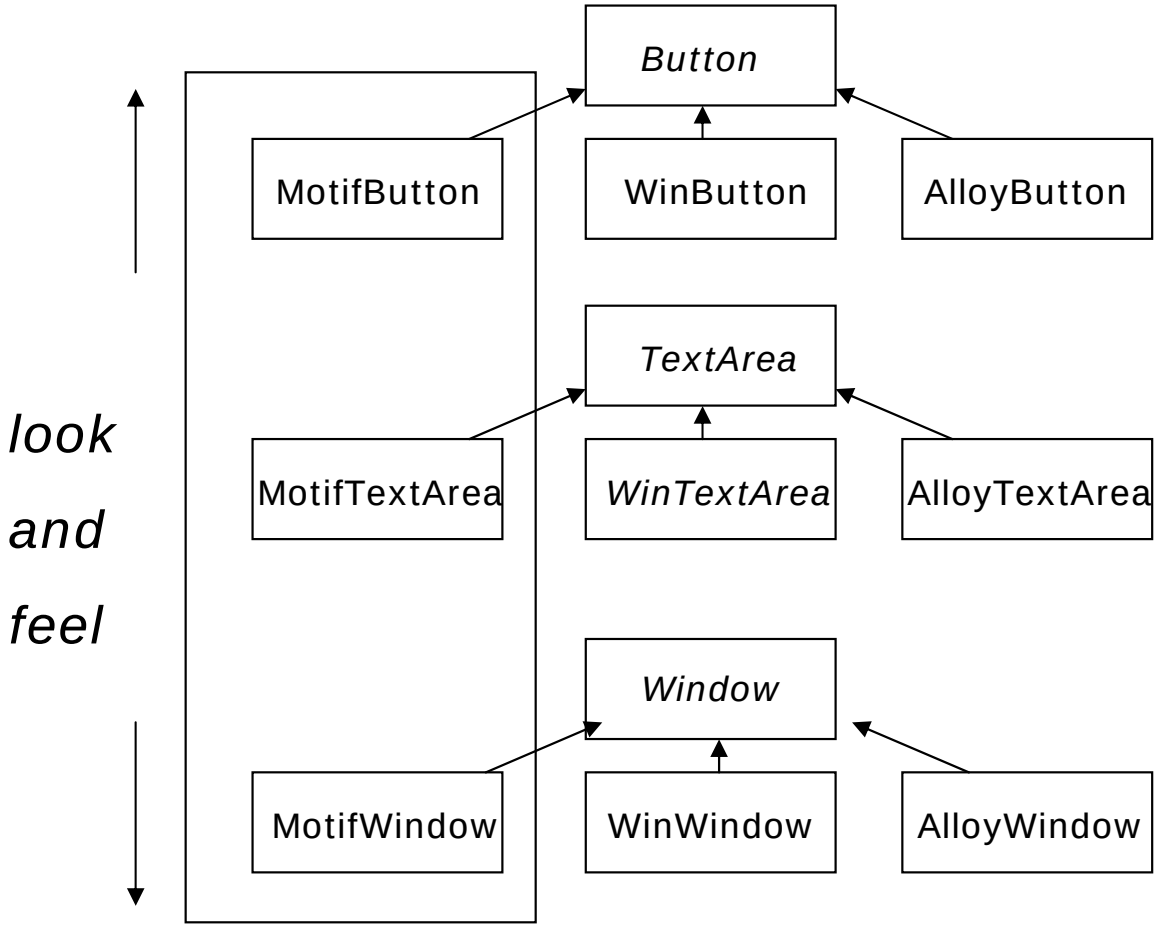


"Motif" look and feel



"Alloy" look and feel

Abstract Factory – Esempio



Abstract Factory – Esempio

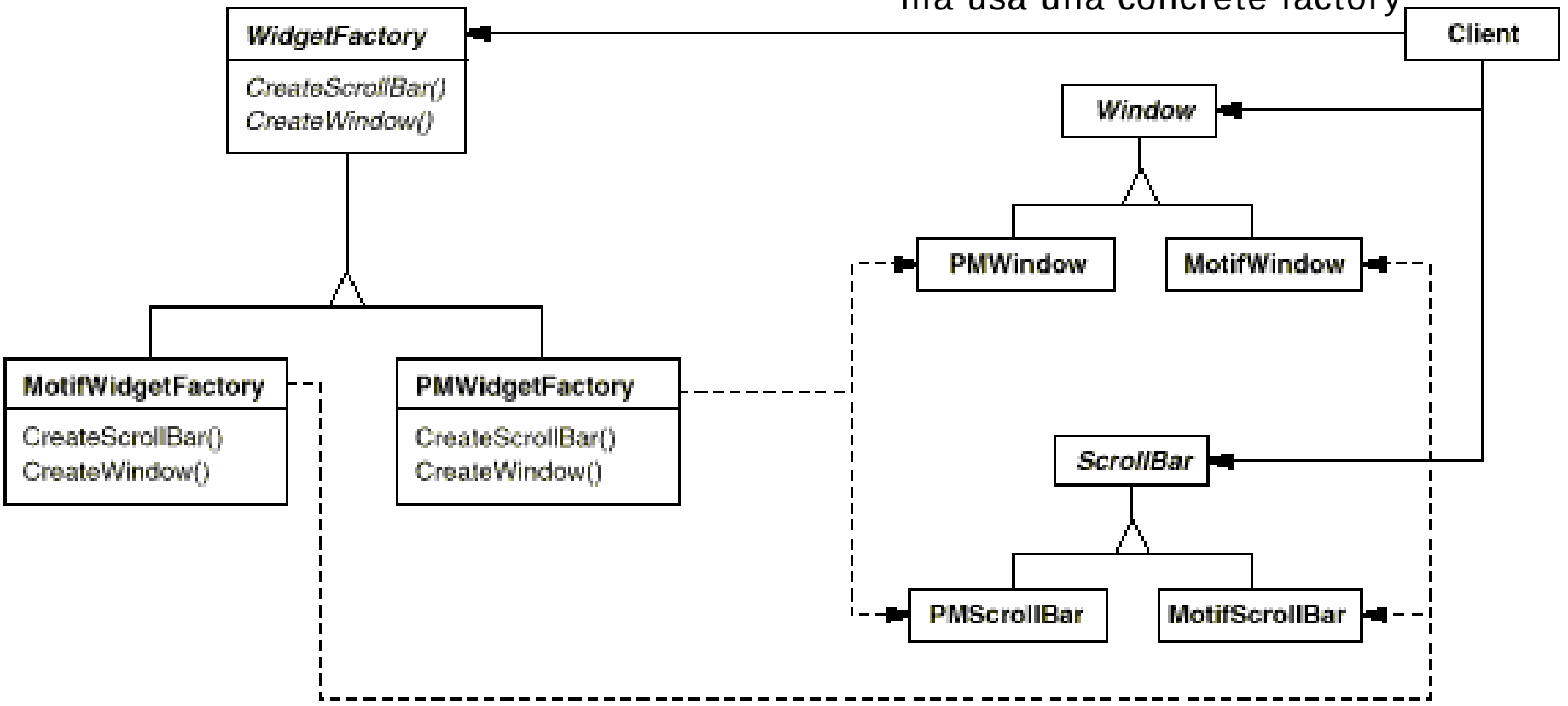
- Una GUI toolkit supporta diversi **“look-and-feel*”**, come Motif e Presentation Manager (le FAMIGLIE DI PRODOTTI)
- Diversi look-and-feel definiscono un diverso aspetto e diversi comportamenti per “widgets” (i PRODOTTI) come:
  - Scrollbars
  - Windows
  - Buttons
  - TextFields .. etc
- Ci aspettiamo che impostare un certo look and feel per la GUI imponga un vincolo sull’uniformità dello stile dei vari componenti (es. se i Button sono “stile” Motif allora la scrollbar deve essere “Motif” ...) vorremmo che l’aspetto dei vari componenti sia coerente e che controllare il look and feel dei singoli oggetti da creare non sia compito del programmatore)

- In generale vorremmo scegliere una volta per tutte la famiglia di prodotti da utilizzare e delegare il compito di instanziare volta per volta l’oggetto di tipo giusto

*ad es. forma dei componenti, tonalità, font etc

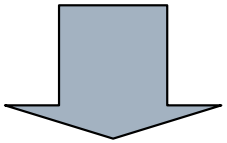
Abstract Factory – Esempio

non chiama i costruttori per i vari oggetti,
ma usa una concrete factory



Abstract Factory – Esempio

```
class GUI {
     //...
          void Create_gui(){
//...
button=new MotifButton();
textArea=new MotifTextArea(“text”);
window=new MotifWindow();
//..
};
```

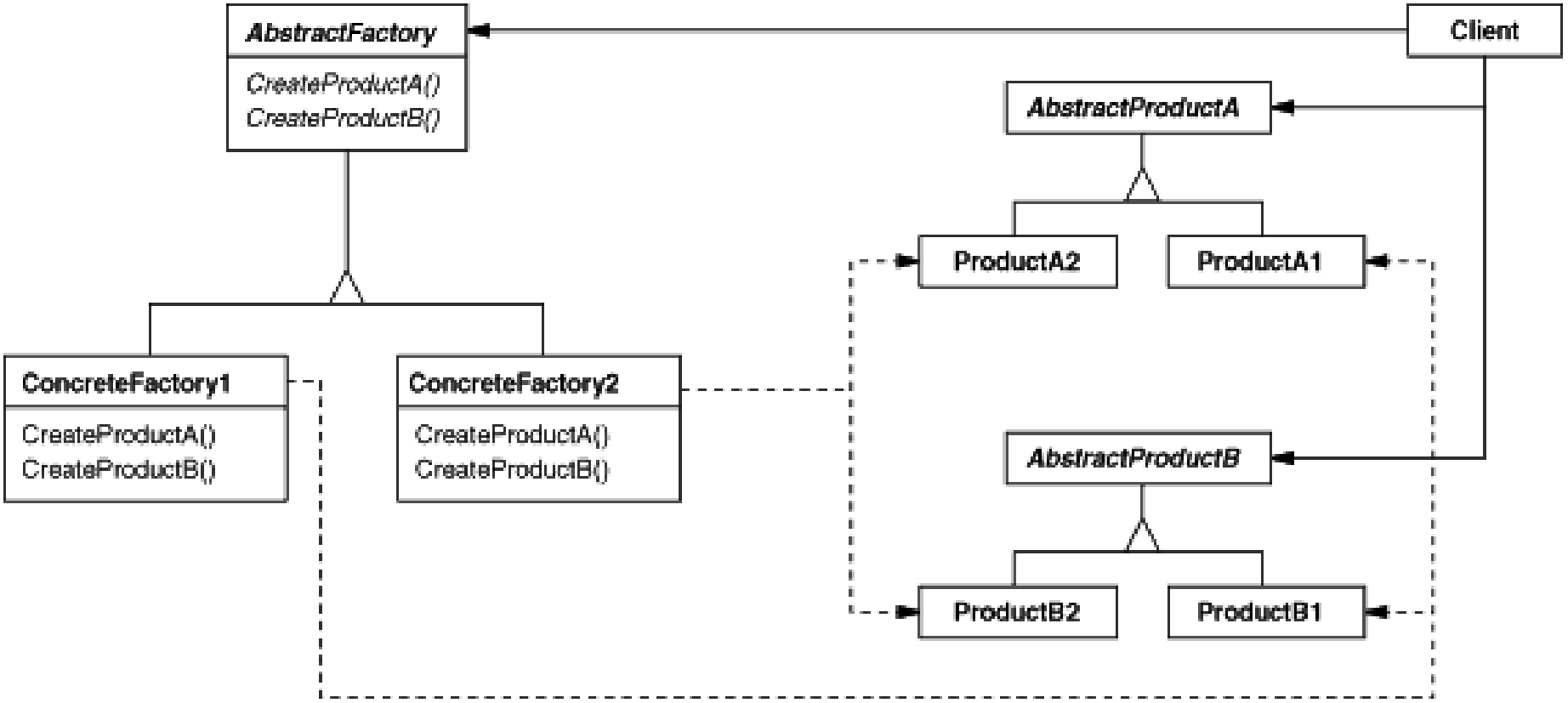


```
class GUI {
     //...
     WidgetFactory* factory;
     void Create_gui(){
//...
button=factory->createButton();
textArea=factory->createTextArea(“text”);
window=factory->createWindow();
//..
};
```

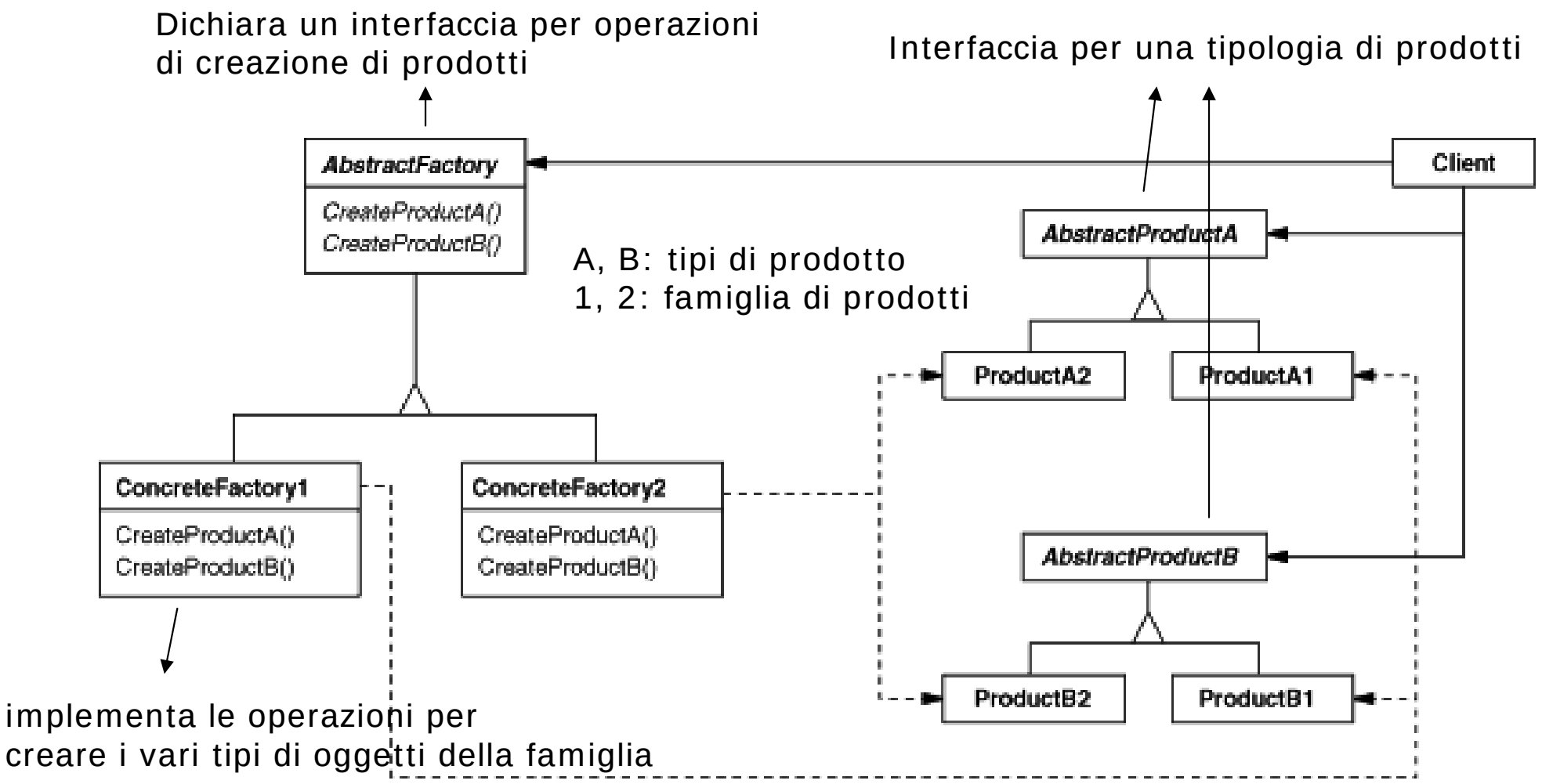
Abstract Factory (object / creational)

- Fornisce un'interfaccia per creare famiglie di oggetti correlati/dipendenti senza specificare le loro classi concrete
- Quando usare Abstract Factory:
  - Un sistema deve essere indipendente da come i suoi prodotti sono creati, composti e rappresentati
  - Un sistema deve essere configurato per **una** di diverse famiglie di prodotti disponibili
  - Una famiglia di oggetti relativi a prodotti correlati è progettata per essere usata nella sua totalità
  - Si vuole fornire una libreria di classi di prodotti, e si vogliono rivelare solo le interfacce, non le loro implementazioni

Abstract Factory (ii)



Abstract Factory (iii)



Abstract Factory: Pros

- Isola le classi concrete
  - I client non devono sapere niente delle classi concrete che useranno, neanche al momento dell'istanziazione degli oggetti
- E' facile cambiare famiglia di prodotto
  - Basta cambiare 1 linea di codice che riguarda la creazione della factory
- Promuove la consistenza tra i prodotti
  - I prodotti sono organizzati in famiglie. I prodotti di una famiglia sono coordinati per lavorare insieme

Abstract Factory: Cons

- Supportare l'inserimento di un nuovo prodotto in una famiglia è difficile
  - Può richiedere cambiamenti all'interfaccia dell'Abstract Factory e alle sue sottoclassi
  - Richiede l'aggiunta di una nuova classe ConcreteFactory e di nuovi AbstractProducts e Products
- La creazione di oggetti non avviene nel modo standard
  - I client devono sapere che devono usare la factory invece del costruttore per istanziare nuovi oggetti
  - Altrimenti la correttezza dell'implementazione non è garantita