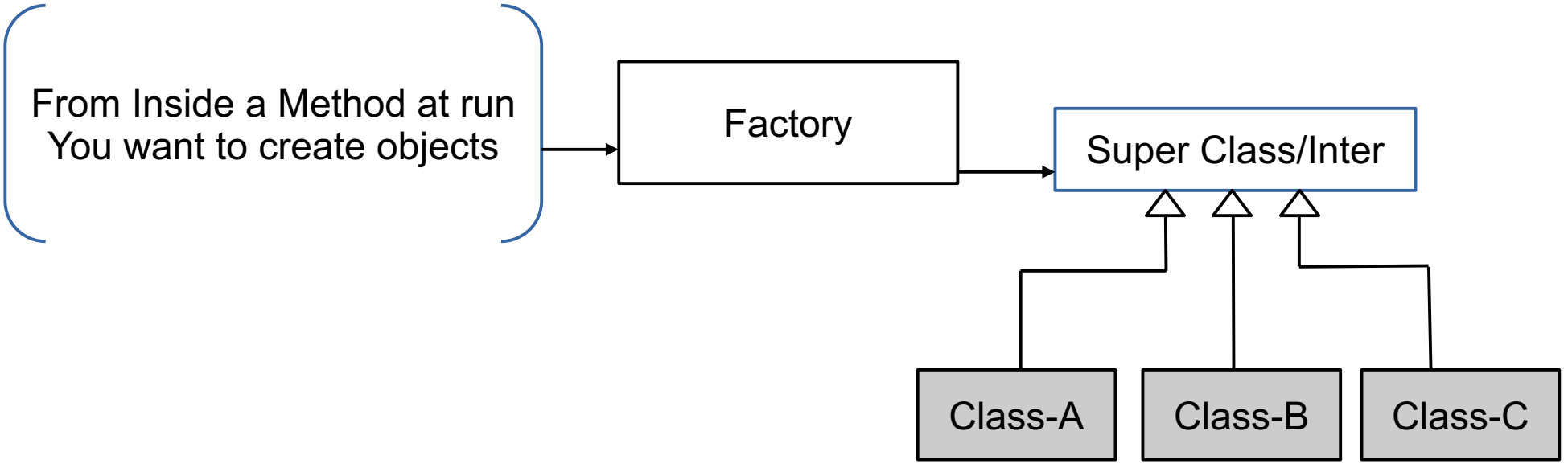
Abstract Factory Pattern

Problem – **Factory Method**

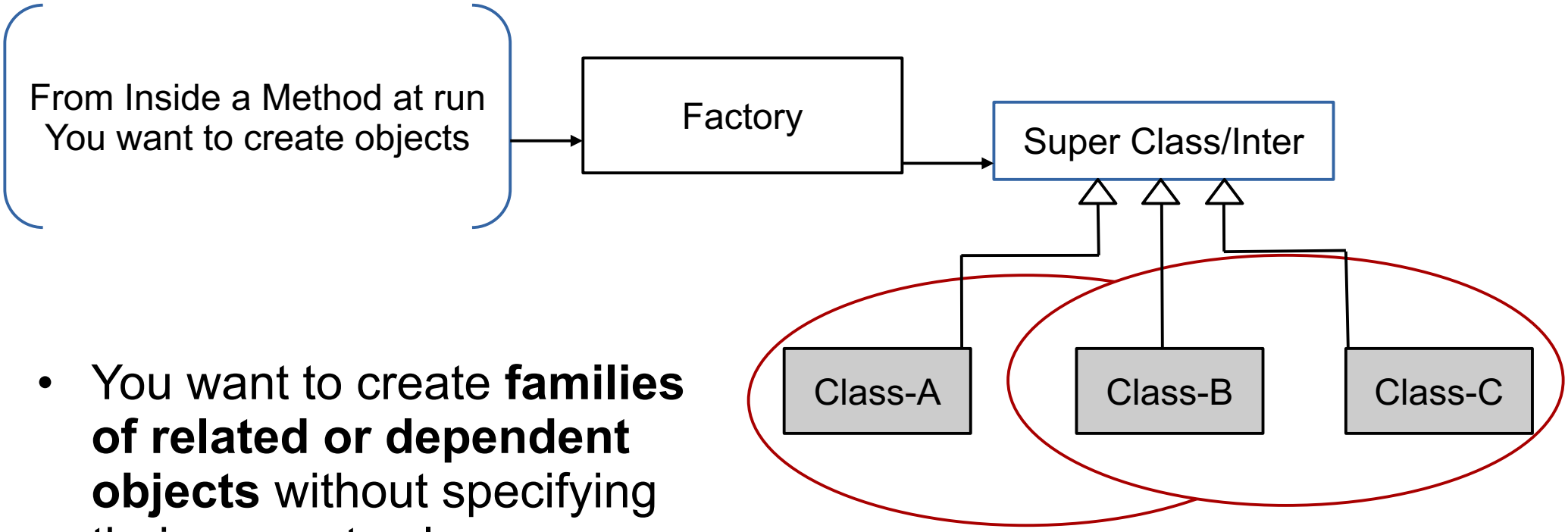


Factory Method

- Definition from Book: "*Factory method defines* ***an interface for creating an object****, but* ***let subclasses decide*** *which class to instantiate. Factory Method lets a class defer instantiation to subclasses.*"

- Factory pattern encapsulates object creation and delegates the responsibility to another class.

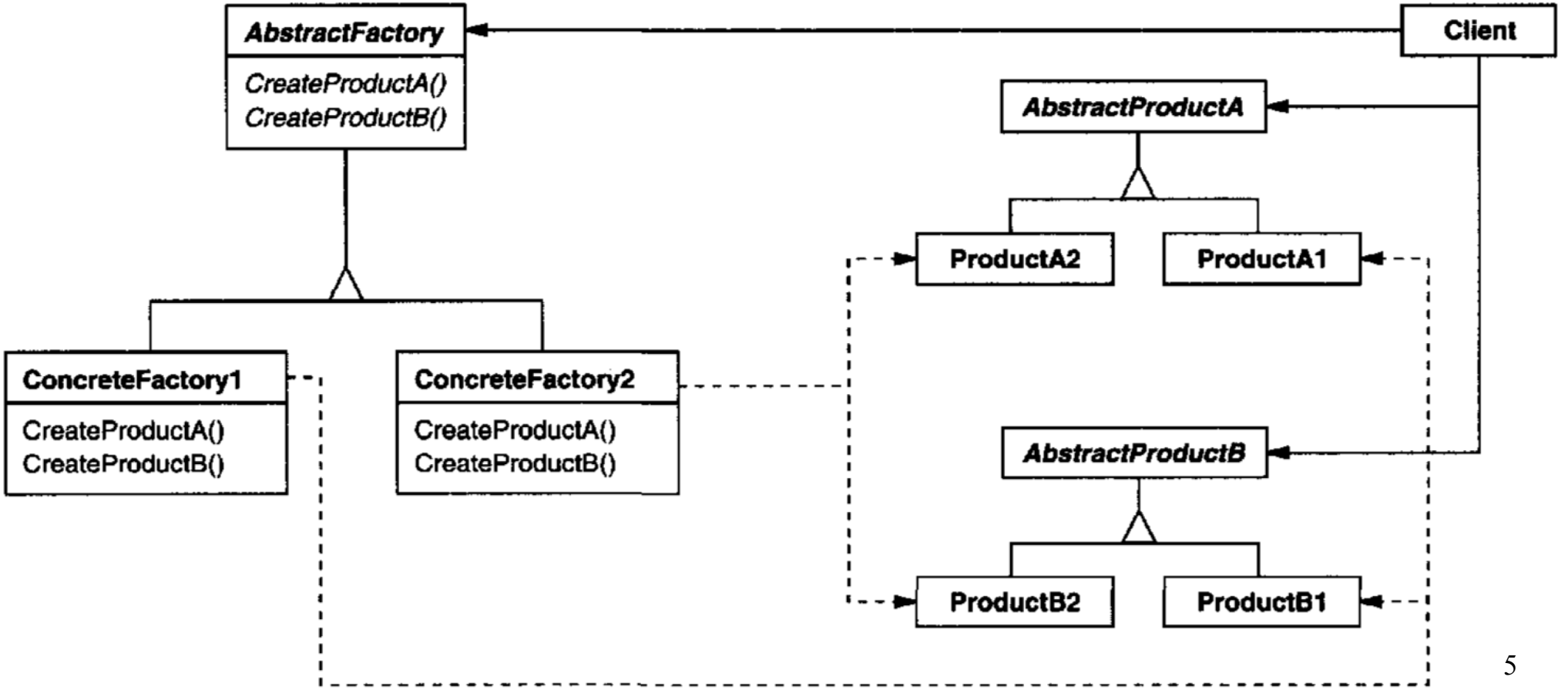
Problem – **Factory Method**



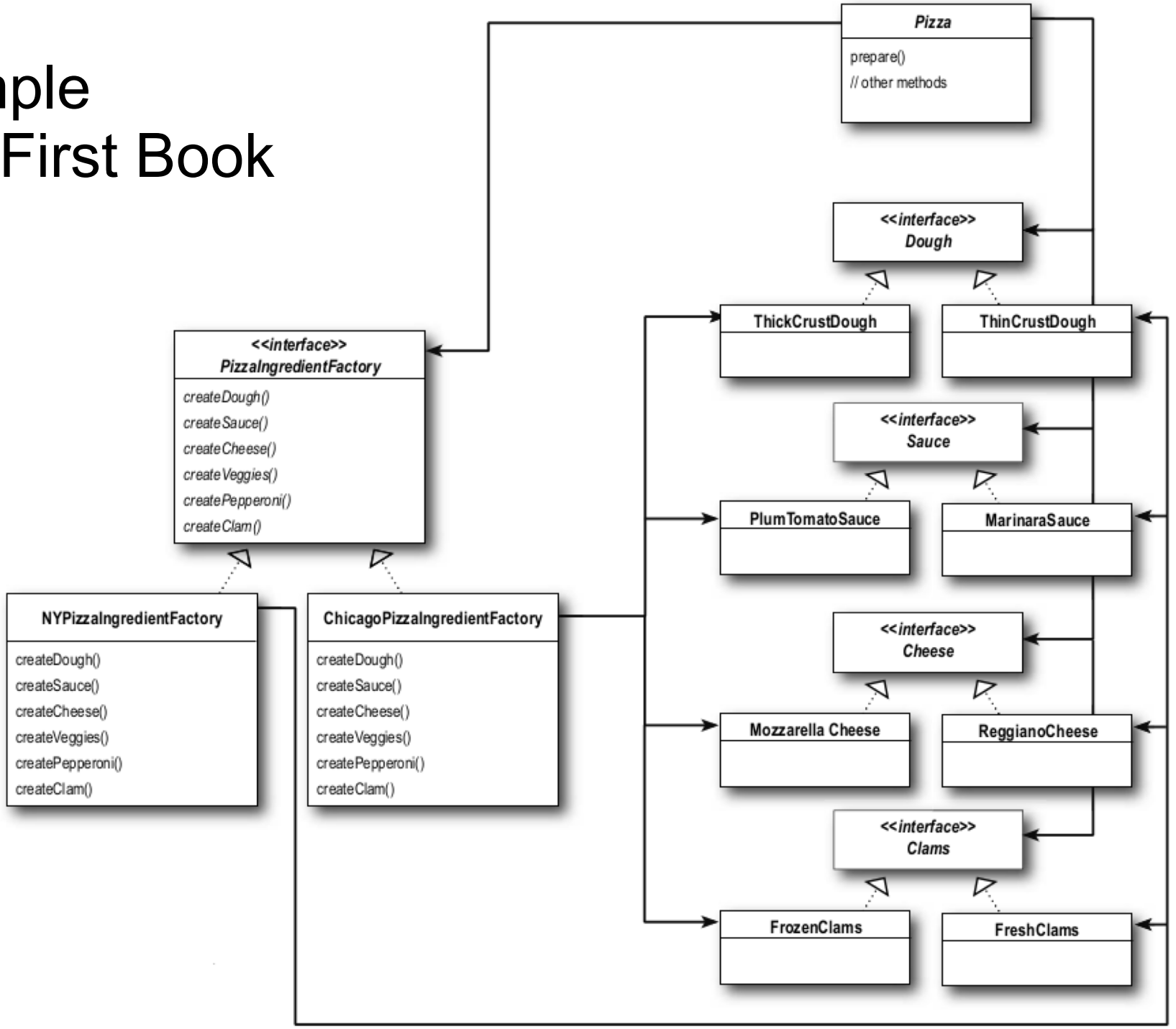
- You want to create **families of related or dependent objects** without specifying their concrete classes.
- They can be from different types and super types

Abstract Factory Method

- Provide an interface for creating families of related or dependent objects without specifying their concrete classes.
- ConcreteFactory1 creates ProducetA1 and ProductB1

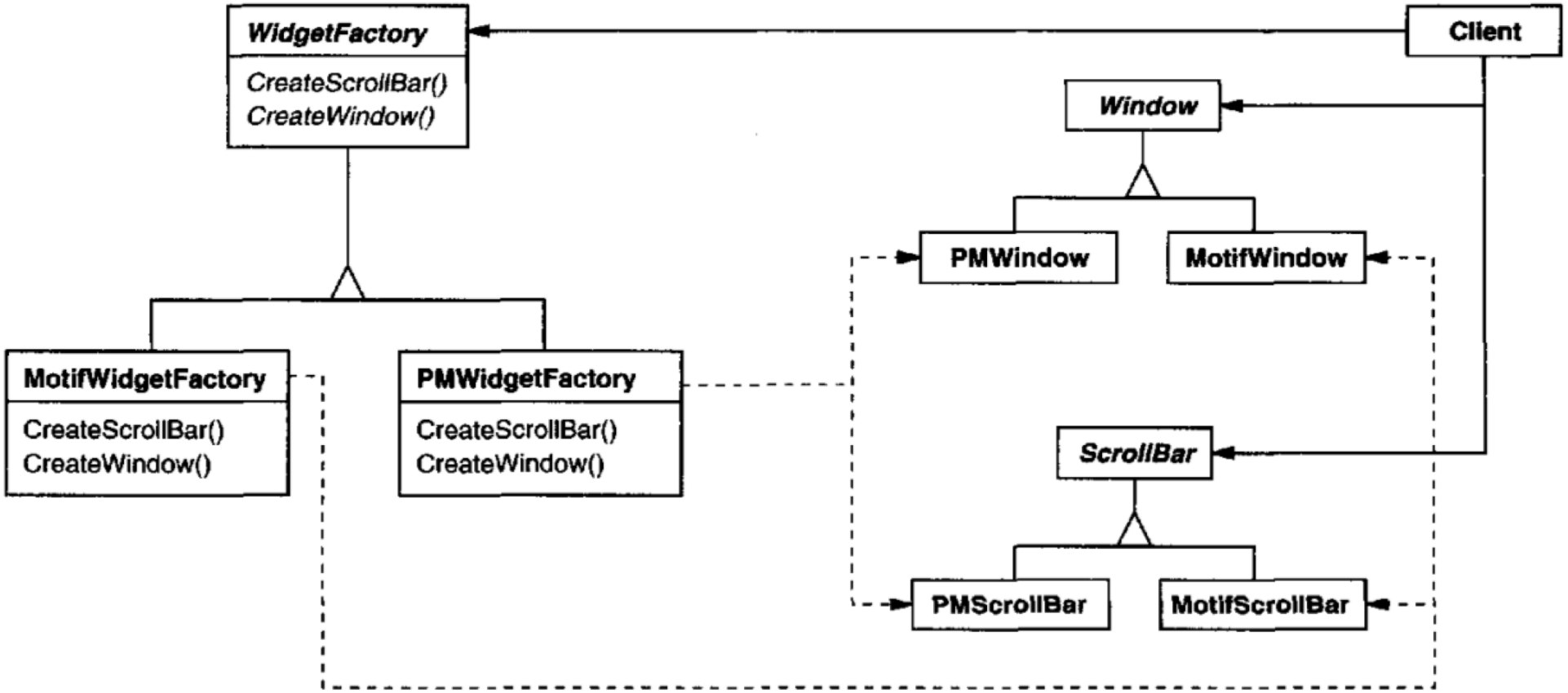


Example
HeadFirst Book



Example – 2 : Implement this on your own

- Consider a user interface toolkit that supports multiple look-and-feel standards, such as **Motif** and **Presentation** Manager.
- Different look-and-feels define different appearances and behaviors for user interface "**widgets**" like scroll bars, windows, and buttons.



When use the Abstract Factory Pattern

- Use Abstract Factory Pattern when
  - your application should be **independent of how its products are created**, composed, and represented.
  - your system should be **configured with one of multiple families of** products.
  - a family of related product objects is **designed to be used together**, and you need to enforce this constraint.
  - you want **to provide a class library of products**, and you want to reveal just their interfaces, not their implementations.

Consequences

+ It isolates concrete classes.

- It helps to control the classes of objects that an application creates. Factory encapsulates the responsibility and the process of creating product objects and isolates clients from implementations.

+ It makes exchanging product families easy.

- It is easier to change the concrete factory an application uses.
- The concrete factory appears only once in your application, where it's instantiated.

Consequences

+ It promotes consistency among products.

- When objects in a family/group are designed to work together. It is important that an application use objects from **only one family at a time**.

- Supporting new kinds of products is difficult.

- Extending “abstract factories” to produce new kinds of Products isn't easy. That's because the Abstract Factory interface **fixes the set of products that can be created**. (Think about solutions for this!)

Summary

- The Factory Method pattern is one of the straightforward patterns.

Abstract Factory

- Abstract Factory classes are often implemented with factory methods (Factory Method), but they can also be implemented using **Prototype Pattern** (We will learn that later) .
- We've certainly seen that Abstract Factory allows a client to use an abstract interface to create a set of related products without knowing about the concrete products that are actually produced.