Abstract Factory Design Pattern

Creational Design Patterns

Abstract the instantiation process.

Help make a system independent of how its objects are created, composed, and represented.

Creational Design Patterns

They all encapsulate knowledge about which concrete classes the system uses.

They hide how instances of these classes are created and put together.

- System knows only about the interfaces as defined by abstract classes.

Creational Design Patterns

Abstract Factory	Creates an instance of several families of classes
Factory Method	Creates an instance of several derived classes
Builder	Separates object construction from its representation
Singleton	A class of which only a single instance can exist
Prototype	A fully initialized instance to be copied or cloned

Abstract Factory

Intent

Provide an interface for creating families of related or dependent objects without specifying their concrete classes.

A hierarchy that encapsulates *platforms*, and the construction of a suite of *products*.

Abstract Factory

Motivation

Support three variations of services with different quality of service parameters

- Basic
- Moderate
- Premium

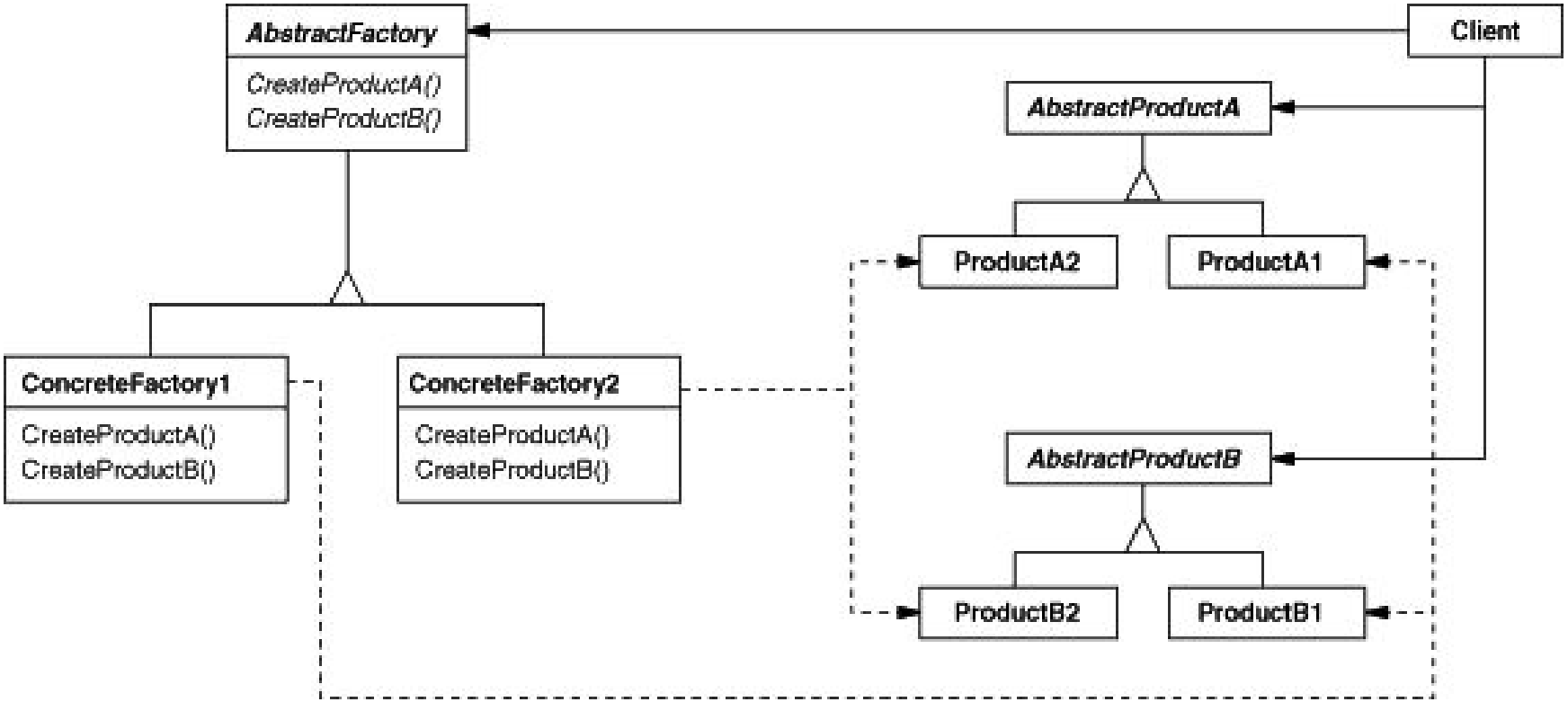
Abstract Factory

Applicability

Use the Abstract Factory pattern when

- a system should be independent of how its products are created, composed, and represented.
- a 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.

Abstract Factory - Structure



Abstract Factory - Participants

AbstractFactory (AbstractResourceFactory)

ConcreteFactory (ModerateResourceFactory, PremiumResourceFactory)

AbstractProduct (Messenger, AudioPlayer)

ConcreteProduct (PremiumMessenger, BasicAudioPlayer)

Client

uses only interfaces declared by AbstractFactory and AbstractProduct classes.

Abstract Factory - Participants

AbstractFactory (AbstractResourceFactory)
declares an interface for operations that create abstract product objects.

ConcreteFactory (ModerateResourceFactory, PremiumResourceFactory)
implements the operations to create concrete product objects.

AbstractProduct (Messenger, AudioPlayer)
declares an interface for a type of product object.

ConcreteProduct (PremiumMessenger, BasicAudioPlayer)
implements the AbstractProduct interface.
defines a product object to be created by the corresponding concrete factory.

Client
uses only interfaces declared by AbstractFactory and AbstractProduct classes.

Abstract Factory

Collaborations

Normally a single instance of a ConcreteFactory class is created at run-time.

This concrete factory creates product objects having a particular implementation.

To create different product objects, clients should use a different concrete factory.

Abstract Factory

Consequences

It isolates concrete classes.

It makes exchanging product families easy.

It promotes consistency among products.

Supporting new kinds of products is difficult.

> That's because the AbstractFactory interface fixes the set of products that can be created.

Abstract Factory

Implementation

Study the code in the *src* directory.

Known Uses.

Related Patterns

- Singleton
- Factory Method
- Prototype