Abstract Factory

Presented by John Collins
February 11th, 2011

Overview of Abstract Factory

- Provide an interface for creating families of related or dependent objects without specifying their concrete classes.
- Clients only know and understand the interface and do not have to know everything about the underlying concrete class.
- An Abstract Factory separates the implementation details of objects from their general usage.

Applicability of Abstract Factory Pattern

The Abstract Factory Pattern Should be used:

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

- When a system should be configured with one of multiple families of products or a developer needs to enforce a family of related objects to be used together.

- When a developer wants to provide just the interfaces of a library of products.

Abstract Factory Analogy

An abstract factory is similar to a factory that can make train cars. The train cars have all different types of specifications. For example, a sleeper car is different from a diner car. The main program running the factory does not need to keep track of the differences between train cars, it only needs to know how to make a train car. The different specifications between train cars are stored in concrete factories. The abstract factory is the interface that calls the concrete factories.

Positive Consequences of Abstract Factories

1) An Abstract Factory isolates concrete classes by encapsulating the process and responsibility of creating objects.

2) Clients are only able to manipulate instances of objects through the interface.

3) Abstract Factories limit hardware platform dependencies because of the abstract interface.

4) Abstract Factories force consistency among products because all object instances are accessed through the same interface.

Negative Consequences of Abstract Factories

1) Supporting new kinds of products is difficult because it requires the extension of the entire interface. In other words, the train factory now has to make boats, which have different process of being made.

2) All of the product objects are returned to the client with the same abstract interface as given by the return type. This means that the client will not be able to access subclass-specific operations through the abstract interface.

UML Layout explaination

AbstractFactory(Train Factory)
- delcares an interface for operations that create abstract product objects.

ConcreteFactory (SleeperCarFactory and DinerCarFactory)
- implements the operations to create concrete product objects.

AbstractProduct (TrainCar)
- declares an interface for a type of product object.

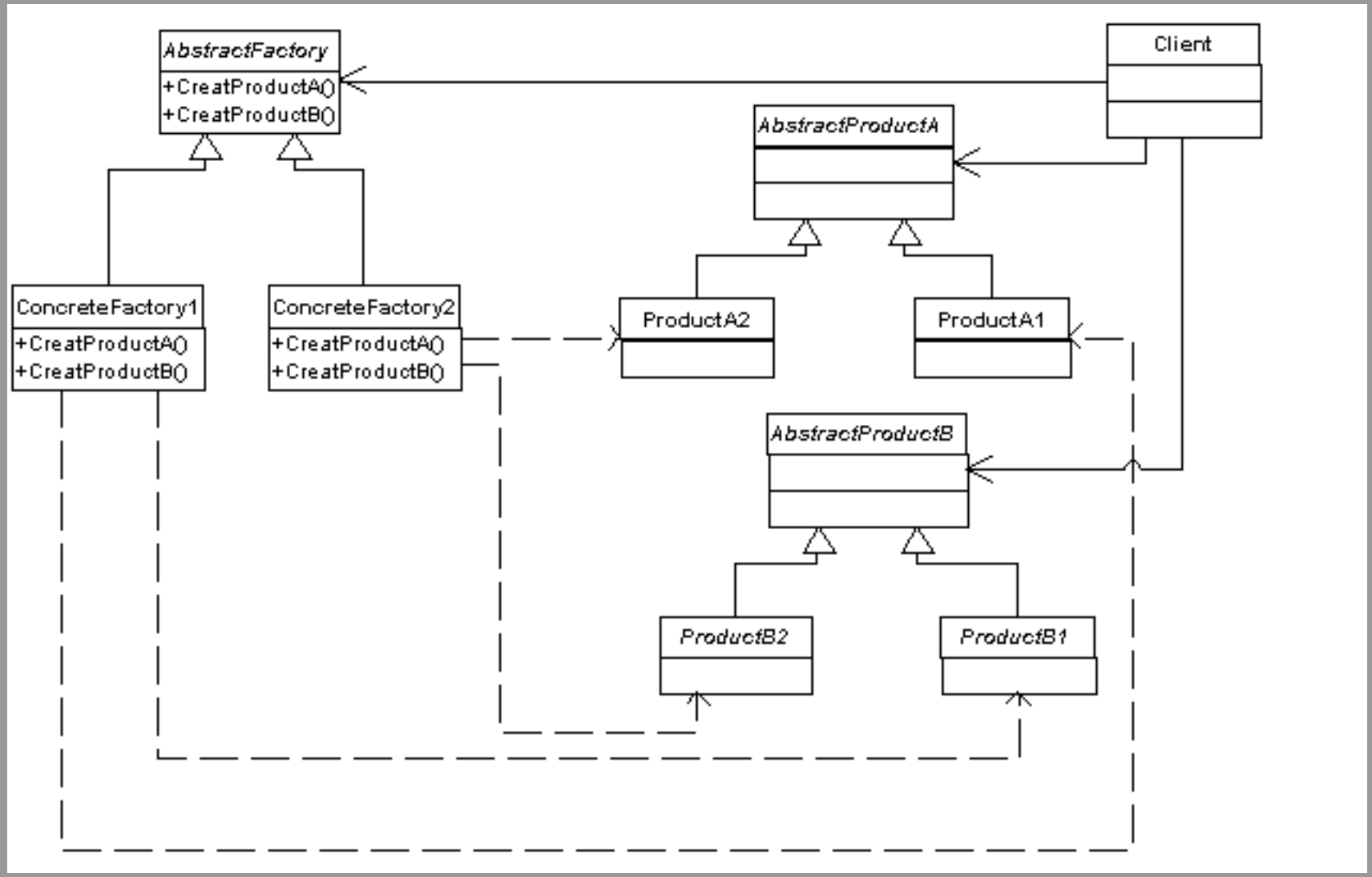
UML Layout

ConcreteProduct (Sleeper Car and Diner Car)
- defines a product object to be created by the corresponding concrete factory.
-implements the AbstractProduct interface.

Client - uses only interfaces declared by Abstract Factory and AbstractProduct Classes

Abstract Factory UML

UML Diagram



Abstract Factory Java Pseudocode

```
public abstract class TrainFactory
{
    public abstract TrainCar createCar();
}

public class SleeperCarFactory extends TrainFactory
{
    public TrainCar createCar()
    {
    return new CreateSleeperCar();
    }
}
```

Continued

```
public class DinerCarFactory extends TrainFactory
{
    public TrainCar createCar()
    {
        return new CreateDinerCar();
    }
}

 public abstract class TrainCar
    {
    abstract void buildCar();
    }
}
```

Continued

```
public class SleeperCar extends TrainCar
{
    public void buildCar()
      { System.out.println(“I'm a SleeperCar.”); }
}

public class DinerCar extends TrainCar
{
    public void buildCar()
       { System.out.println(“I'm a DinerCar.”); }
}
```

Client Code

```
// The Client side main would look something like this.
public static void main(String[] args)
{
// This dictates which Concrete Factory is constructed.
TrainFactory = new SleeperCarFactory();
}
```

References:

Design Patterns -Elements of Reusable Object-Oriented Software
by Gamma, Helm, Johnson and Vlissides

http://en.wikipedia.org/wiki/Abstract_factory_pattern