Université Nice Sophia Antipolis

Patrons de Conception
-
Abstract Factory

Simon Urli
urli@i3s.unice.fr

Master 1 MIAGE
2014-2015

Objectifs

“ Fournir une interface pour créer des familles d’objets dépendants ou associés sans connaître leur classe réelle. ”

Classification : patron de création

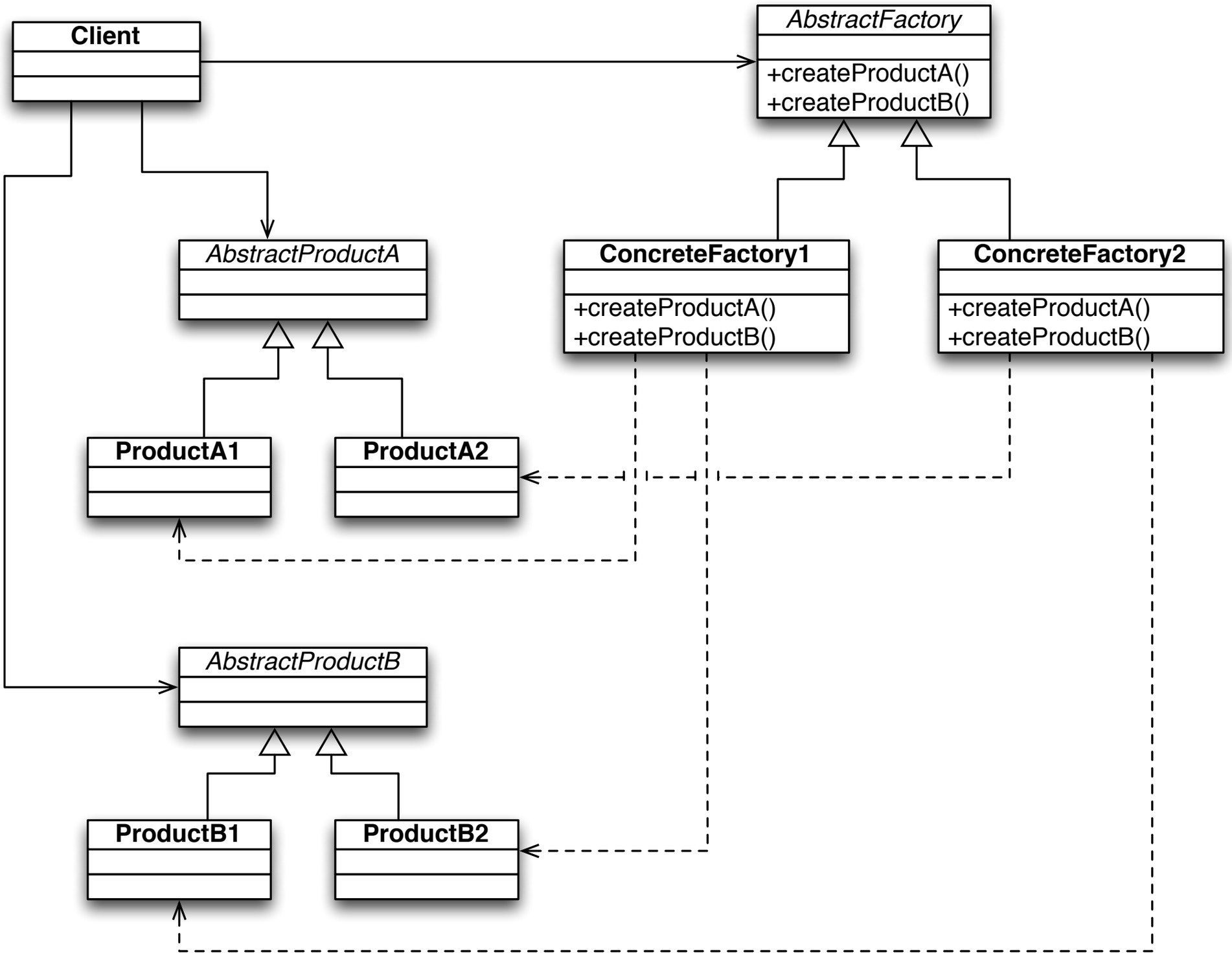
Exemple


12:35
12
35
Ulverston
Partly Sunny
Thu, Dec 10
7° H: 9° L: 6°
My Life Flashed Before My Eyes - The Cribs
Messages
Google M...
Browser
Twidroid
Phone

Applications

- Indépendance de comment les objets sont créés
- Gestion de familles de produits
- Bibliothèque fournie avec seulement leurs interfaces, pas leurs implémentations

Structure


Client
AbstractFactory
+createProductA()
+createProductB()
AbstractProductA
ConcreteFactory1
+createProductA()
+createProductB()
ConcreteFactory2
+createProductA()
+createProductB()
ProductA1
ProductA2
AbstractProductB
ProductB1
ProductB2

Implémentation

- **AbstractFactory** : déclare l'interface pour les opérations qui créent des objets abstraits
- **ConcreteFactory** : implémente les opérations qui créent les objets concrets
- **AbstractProduct** : déclare une interface pour un type d'objet
- **ConcreteProduct** : définit un objet qui doit être créé par la fabrique concrète correspondante et implémente l'interface AbstractProduct
- **Client** : utilise seulement les interfaces déclarées par AbstractFactory et par les classes AbstractProduct

Exemple 1/3

```
public class ClockWindows extends WindowWidget implements IWidget
{
     private int decalage_utc;
     public ClockWindows(Point2D position, int d) {
        super(position);
        this.decalage_utc = d;
     }
     ...
}
public class ClockFactoryWindows extends ClockWidgetFactory {
     public IWidget createWidget(Point2D position) {
        return new ClockWindows(position, 0);
     }
}
```

Exemple 2/3

```
public class ClockOSX extends OSXWidget implements IWidget {
     private int decalage_utc;
     public ClockOSX(Point2D position, int d) {
        super(position);
        this.decalage_utc = d;
     }
     ...
}
public class ClockFactoryOSX extends ClockWidgetFactory {
     public IWidget createWidget(Point2D position) {
        return new ClockOSX(position, 0);
     }
}
```

Exemple 3/3

```
public abstract class ClockWidgetFactory implements IWidgetFactory {
     public static IWidgetFactory getFactory() {
        if (System.os.equals(‘‘Windows’’)) {
           return new ClockFactoryWindows();
        } else if (System.os.equals(‘‘OSX’’)) {
           return new ClockFactoryOSX();
        } else throw new RuntimeException(‘‘Unsupported OS’’);
     }
     public abstract IWidget createWidget(Point2D position);
}
public class Client {
     private IWidgetFactory clockFactory =
     ClockWidgetFactory.getFactory();
     ...
}
```

Exercice

- Un grand distributeur de boisson vend des cannettes, des bouteilles de 33 cl, et des fûts de 5L de ses différentes boissons anonymisées (A, B et C). Modélisez son système d'information.