The Abstract Factory

Chapter Content

InterBit
Training & Consulting Ltd.

- The Abstract Factory Pattern - Overview
- The Abstract Factory Pattern – Example
- Abstract Factory UML Diagram
- Discussion
- Exercise

Overview

- One level of abstraction above the ordinary Factory.
- Define an **abstract Factory class** (or interface), and sub-class using concrete factories.

  Each concrete factory creates a family of objects with unique characteristics.
- The concrete factories are often obtained through a factory-of-factories.

Example

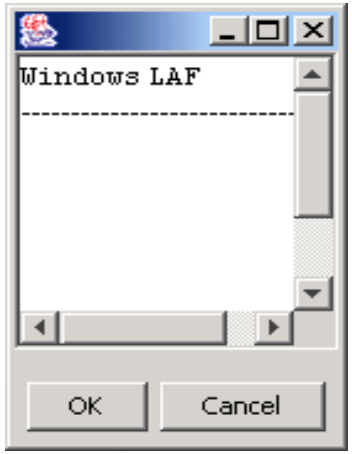
- Java Swing example:
  - **The products:** ComponentUI's classes
    - Responsible for painting GUI components.
  - **The abstract factory:** LookAndFeel classes
    - Each LookAndFeel creates its own family of ComponentUI's.
    - WindowsLookAndFeel creates buttons, lists, combo-boxes... with Windows appearance.
    - Similarly, we have Metal and Motif LAF's.
    - Java obtains ComponentUI's by calling

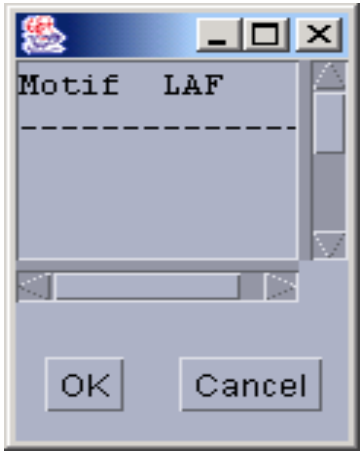

```
LookAndFeel.getDefaults().getUI(JComponent)
```

Example – cont.

- Thus, **a single line(!)** can switch the entire application appearance, simply by switching the “factory” (LAF):

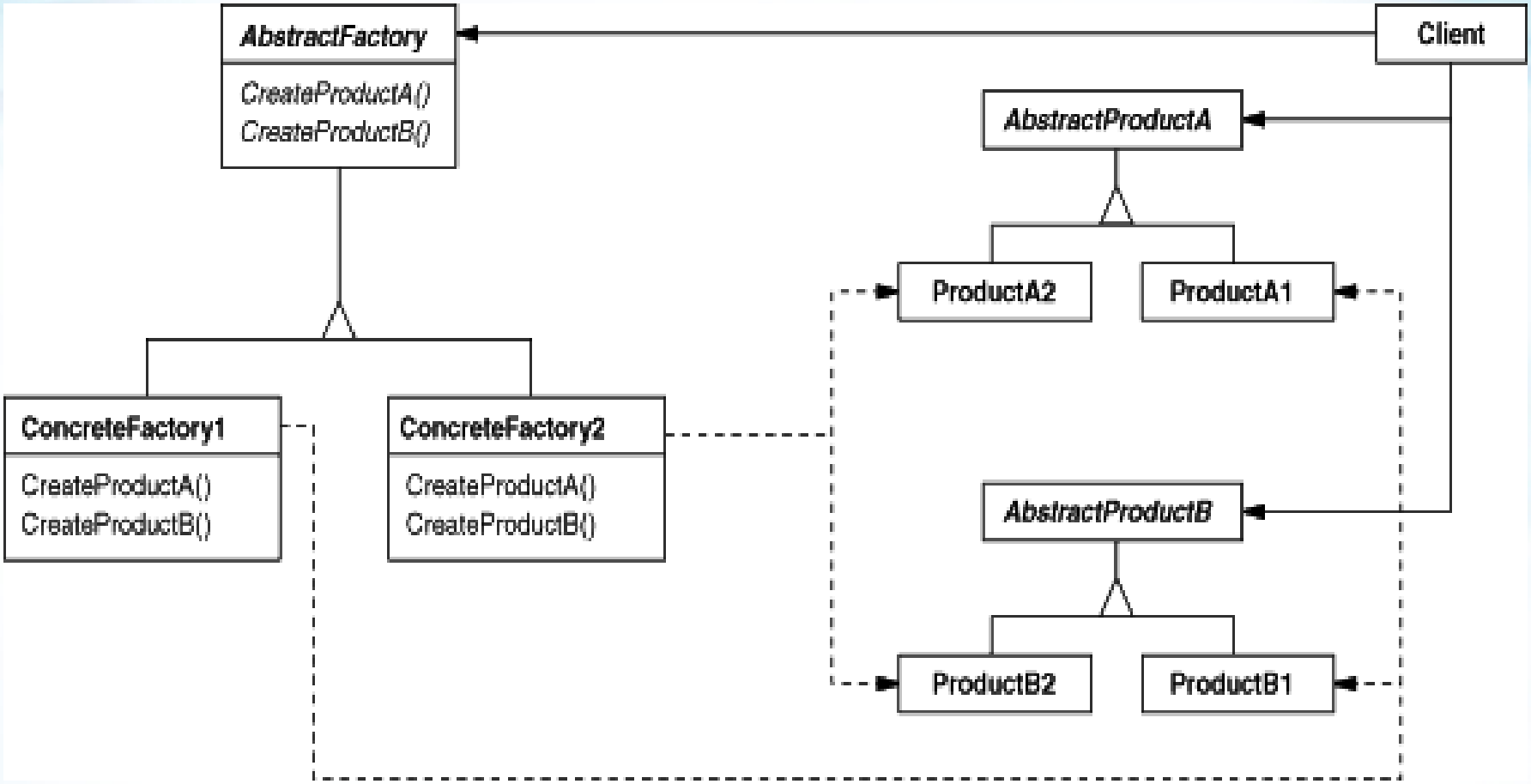
```
String laf = “com.sun.java.swing.plaf.motif.MofitLookAndFeel”;
UIManager.setLookAndFeel( laf );
```





Abstract Factory UML Diagram

InterBit
Training & Consulting Ltd.



Discussion

InterBit
Training & Consulting Ltd.

The Differences between Abstract & Normal Factory:

- We shall demonstrate for LookAndFeel.

Note: the following example does not quote the Swing code, but rather demonstrate the Factory/AbstractFactory idea in general.

Discussion – cont.

InterBit Training & Consulting Ltd.

Normal factory:

```
public class  NoramlGuiFactory {
     private int currentLAF;

     public Button createButton(){
          if (currentLAF==WINDOWS)   return new WinButton();
          else if (currentLAF==MOTIF) return new MotifButton();
          ...
     }

     public ComboBox createCombo(){
          if (currentLAF==WINDOWS)   return new WinComboBox();
          else if (currentLAF==MOTIF) return new MotifComboBox();
          ...
     }

}
```

Discussion – cont.

InterBit Training & Consulting Ltd.

Abstract Factory:

```
public interface  GuiFactory {
     public abstract Button   createButton();
     public abstract ComboBox createCombo();
}
public class WinGuiFactory implements GuiFactory {
     public Button createButton()     { return new WinButton();  }
     public ComboBox createCombo()    { return new WinComboBox();}
}
public class MotifGuiFactory implements GuiFactory {
     public Button createButton()     { return new MotifButton();  }
     public ComboBox createCombo()    { return new MotifComboBox();}
}
public class GuiManager {
    private GuiFactory currentFactory;
    public void setLAF(String classname){
        currentFactory = (GuiFactory)Class.forName(classname).newInstance();
    }
    public GuiFactory getCurrentFactory(){
        return currentFactory;
    }
}
```

Discussion – cont.

What is the advantage of the latter approach over the former one?

AbstractFactory Advantages

InterBit Training & Consulting Ltd.

It isolates concrete classes from the client.

- o You use the Abstract Factory to control the classes of objects the client creates.
- o Product names are isolated in the implementation of the Concrete Factory, clients use the instances through their abstract interfaces.

Exchanging product families is easy

- o None of the client code breaks because the abstract interfaces don’t change frequently.
- o Because the abstract factory creates a complete family of products, the whole product family changes when the concrete factory is changed.

It promotes consistency among products

- o It is the concrete factory’s job to make sure that the right products are used together.

AbstractFactory Disadvantages

Adding a new product requires extending the abstract interface which implies that all of its derived concrete classes also must change:

- New abstract product class is added
- New product implementation is added
- Abstract factory interface is extended
- Derived concrete factories must implement the extensions
- Client has to be extended to use the new product

Disadvantage Overcoming

InterBit
Training & Consulting Ltd.

We can use the following approach:

We can add a parameter to the operation that creates objects. This parameter specifies the kind of object to be created. It could be a class identifier, an integer, a string, or anything else that identifies the kind of product.

In fact, with this approach, Abstract Factory only needs a single "Make" operation with a parameter indicating the kind of object to create.