Abstract Factory Pattern Tutorial
Written Date: September 28, 2009

This tutorial is aimed to guide the definition and application of Gang of Four (GoF) abstract factory design pattern. By reading this tutorial, you will know how to develop a model for the abstract factory pattern, and how to apply it in practice.

Modeling Design Pattern with Class Diagram
1. Create a new project Design Patterns.
2. Create a class diagram Abstract Factory.
3. Select Class from diagram toolbar. Click on the diagram to create a class. Name it as AbstractFactory.
4. Select the **AbstractFactory** abstract by right clicking on **AbstractFactory** and select **Model Element Properties > Abstract** from the popup menu.

5. Create the abstract product classes **AbstractProductA** and **AbstractProductB**. Set them as abstract. Up to now, the diagram should look like this:

6. Right click on **AbstractFactory** and select **Add > Operation** from the popup menu.

7. Name it as **CreateProductA()**, and make **AbstractProductA** as return type.

8. Add also operation **CreateProductB()**, and make **AbstractProductB** as return type.
9. Set both operations as abstract by right clicking on the operation and selecting **Model Element Properties > Abstract** from the popup menu.

![Diagram showing set both operations as abstract]


![Diagram showing creation of subclasses]

11. Similarly, create subclasses **ProductB1** and **ProductB2** from AbstractProductB. Up to now, the diagram should look like this:

![Diagram showing creation of subclasses for Product A and B]
12. Create subclasses `ConcreteFactory1` and `ConcreteFactory2` from `AbstractFactory`.

```
AbstractFactory
  + CreateProductA() : AbstractProductA
  + CreateProductB() : AbstractProductB

ConcreteFactory1
ConcreteFactory2
```

13. Inherit operations from `AbstractFactory` by right clicking on `ConcreteFactory1` and selecting `Related Elements > Realize all Interfaces` from the popup menu.
14. Repeat step 13 on ConcreteFactory2. Up to now, the diagram should look like this:

![Diagram showing AbstractFactory and ConcreteFactories with ProductA and ProductB]

15. Link up the Factory and Product hierarchies by visualizing their dependencies. Right click on AbstractFactory's operation CreateProductA and select Show Dependencies from the popup menu.

![Diagram showing visualized dependencies between AbstractFactory and products]

16. Repeat step 15 on operation CreateProductB. Up to now, the diagram should look like this:

![Diagram showing visualized dependencies between AbstractFactory and products]

https://www.visual-paradigm.com/tutorials/abstractfactorydesignpattern.jsp
17. Finally, create the **Client** class.

---

**Defining Pattern**

1. Select all classes on the class diagram.

2. Right click on the selection and select **Define Design Pattern...** from the popup menu.
3. In the **Define Design Pattern** dialog box, specify the pattern name *Abstract Factory*. Keep the file name as is. Click **OK** to proceed.

**Applying Design Pattern on Class Diagram**

In this section, we are going to apply the abstract factory pattern in modeling a restaurant system which delivers both Chinese and Western meal sets.

1. Create a new project *Restaurant*.
2. Create a class diagram *Meal Preparation*.
3. Right click on the class diagram and select **Utilities > Apply Design Pattern...** from the popup menu.
4. In the Design Pattern dialog box, select Abstract Factory from the list of patterns.

5. Click on AbstractProductA in the preview.
6. Rename \textit{AbstractProductA} to \textit{MainCourse} at the bottom pane.

7. Similarly rename \textit{ProductA1} to \textit{WesternMainCourse}, and \textit{ProductA2} to \textit{ChineseMainCourse}. 


10. Also rename the operations from `CreateProductA` and `CreateProductB` to `PrepareMainCourse` and `PrepareDessert`.
11. Similarly, rename `ConcreteFactory1` to `WesternMealFactory`, and rename operations `CreateProductA` and `CreateProductB` to `PrepareMainCourse` and `PrepareDessert` accordingly.

12. Similarly, rename `ConcreteFactory2` to `ChineseMealFactory`, and rename operations `CreateProductA` and `CreateProductB` to `PrepareMainCourse` and `PrepareDessert` accordingly.

13. Finally, rename `Client` to `Restaurant`.
14. Click **OK** to confirm the changes, and apply the pattern. The following diagram is obtained.

Resources
1. Design Patterns.vpp
2. Abstract Factory.pat

Related Links
• Full set of UML tools and UML diagrams

Visual Paradigm home page
(https://www.visual-paradigm.com/)

Visual Paradigm tutorials
(https://www.visual-paradigm.com/tutorials/)