# Abstract Factory Pattern Tutorial

Written Date : September 28, 2009

## What is the Abstract Factory Design Pattern?

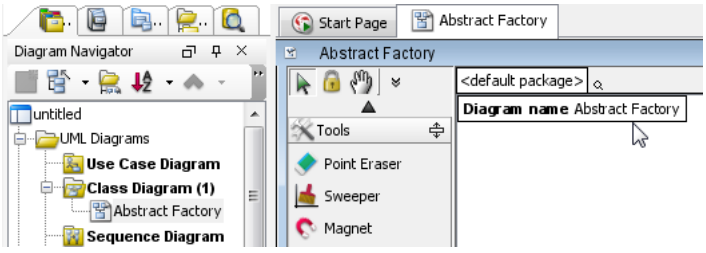
The Abstract Factory Design Pattern is a creational design pattern that enables the creation of families of related objects without explicitly specifying their concrete classes. It provides an interface for generating families of interconnected or dependent objects, abstracting away their specific implementations. This pattern effectively encapsulates a group of individual factories sharing a common theme.

The pattern operates by defining an abstract factory interface, which declares a set of methods for creating abstract product objects. Concrete factory classes then implement these methods to produce specific product objects. This approach facilitates the creation of cohesive families of objects designed to work together, thereby enhancing code flexibility and maintainability.

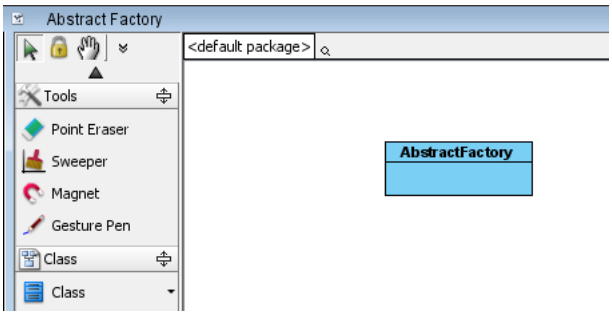
## Modeling the Design Pattern with a Class Diagram

1. Create a new project named *Design Patterns*.

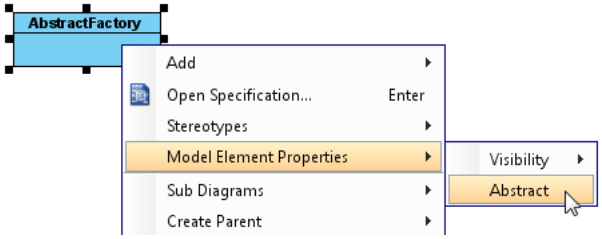
2. Create a class diagram named *Abstract Factory*.



3. Select **Class** from the diagram toolbar. Click on the diagram to create a class and name it *AbstractFactory*.

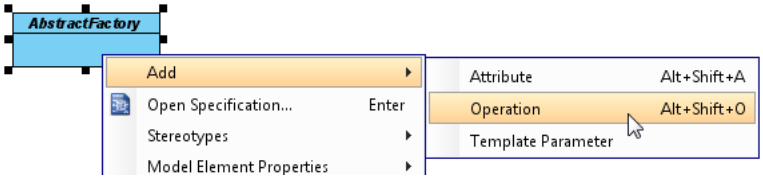



4. Set the *AbstractFactory* class as abstract by right-clicking on it and selecting **Model Element Properties** > **Abstract** from the popup menu.

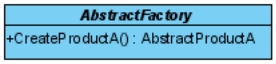



5. Create the abstract product classes *AbstractProductA* and *AbstractProductB*. Mark them as abstract. At this point, your diagram should resemble this:

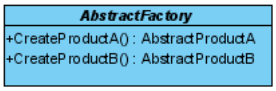

6. Right-click on *AbstractFactory* and select **Add** > **Operation** from the popup menu.



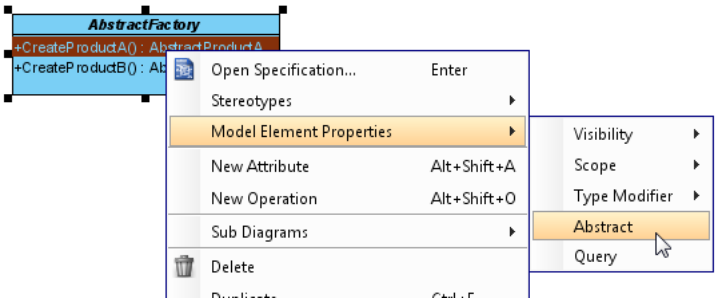
7. Name the operation *CreateProductA()* and set *AbstractProductA* as its return type.



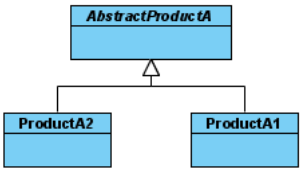

8. Also, add an operation *CreateProductB()*, with *AbstractProductB* as its return type.



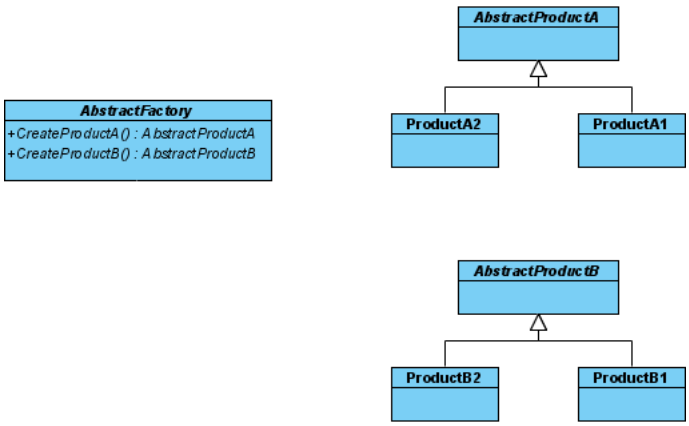

9. Set both operations as abstract by right-clicking on each operation and selecting **Model Element Properties** > **Abstract** from the popup menu.



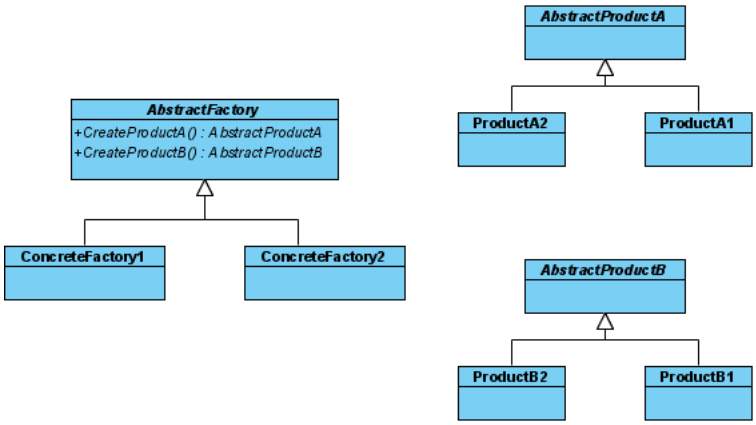

10. Move the cursor over *AbstractProductA*, then use the resource icon **Generalization** > **Class** to create two subclasses, *ProductA1* and *ProductA2*.



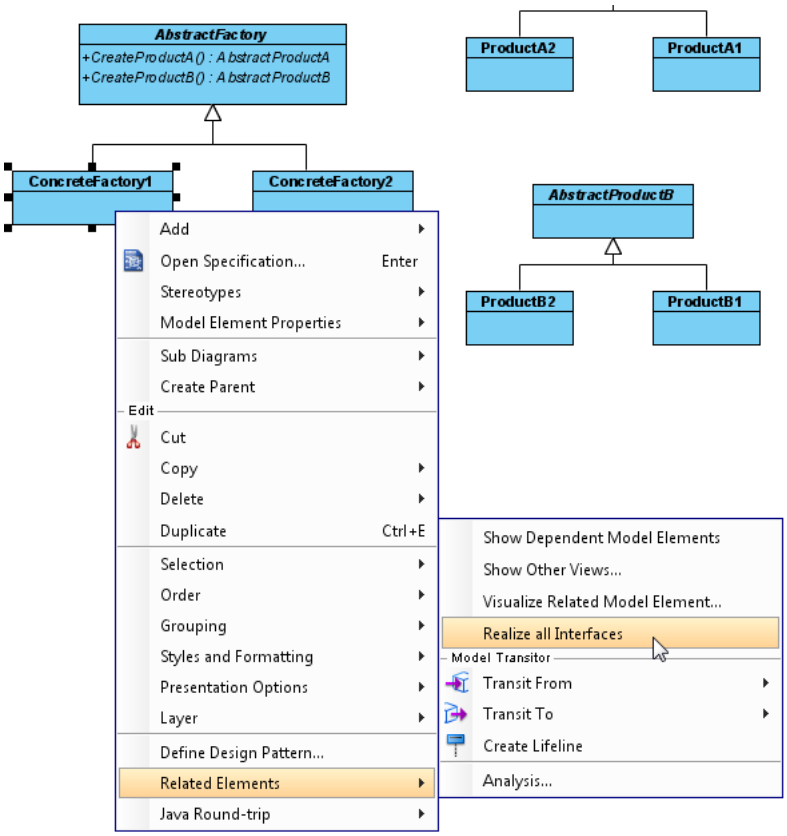
11. Similarly, create subclasses *ProductB1* and *ProductB2* from *AbstractProductB*. At this point, your diagram should look like this:



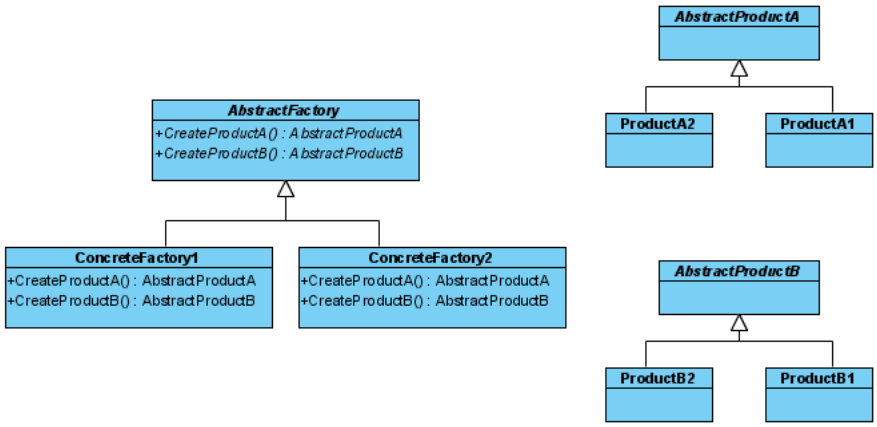

12. Create subclasses *ConcreteFactory1* and *ConcreteFactory2* from *AbstractFactory*.



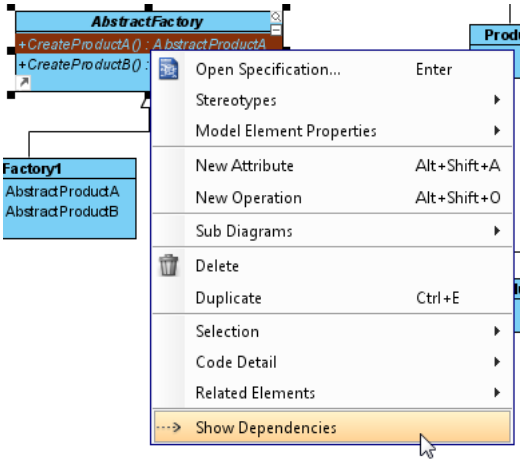
13. Inherit operations from *AbstractFactory* by right-clicking on *ConcreteFactory1* and selecting **Related Elements** > **Realize all Interfaces** from the popup menu.



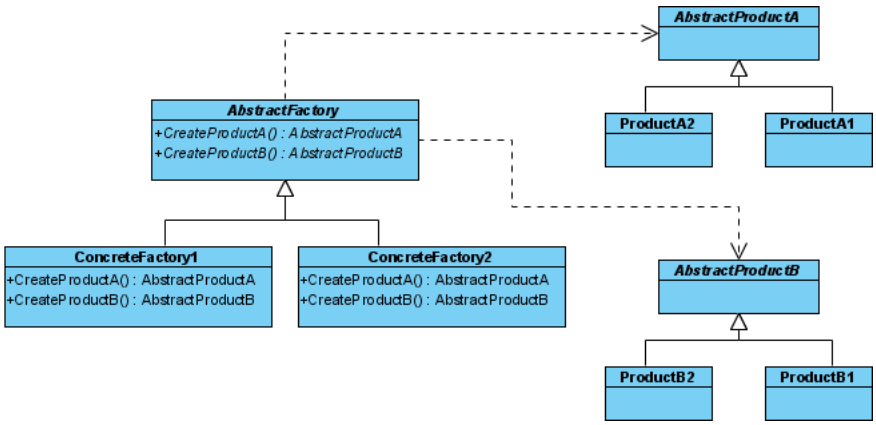

14. Repeat step 13 for *ConcreteFactory2*. Your diagram should now appear as follows:



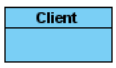
15. Link the *Factory* and *Product* hierarchies by visualizing their dependencies. Right-click on *AbstractFactory*'s operation *CreateProductA* and select **Show Dependencies** from the popup menu.




16. Repeat step 15 for the operation *CreateProductB*. Your diagram should now look like this:

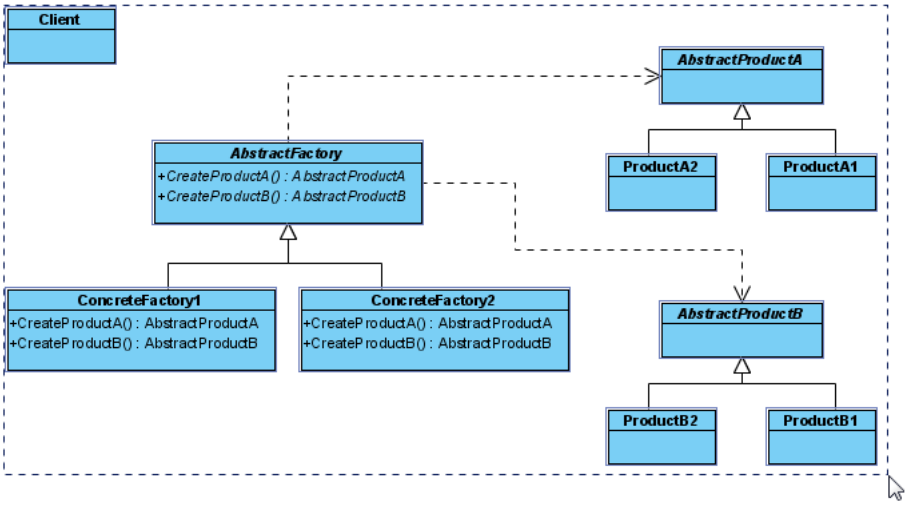



17. Finally, create the *Client* class.

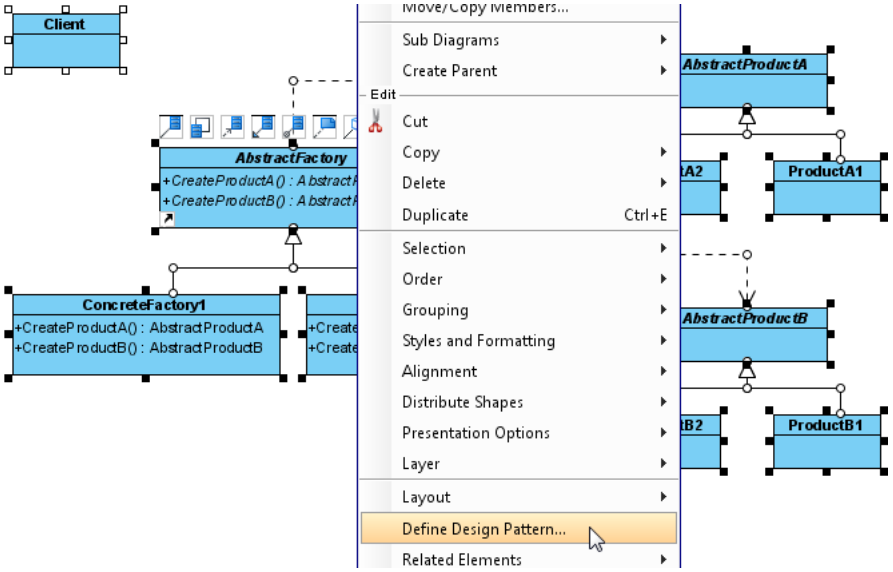



## Defining the Pattern

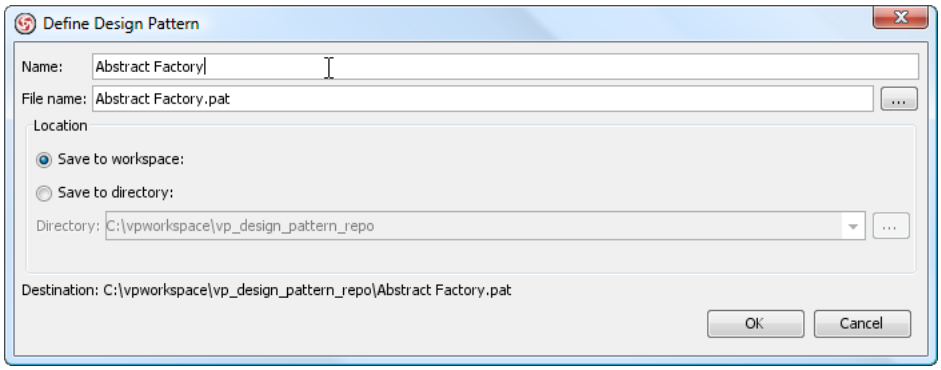
1. Select all classes on the class diagram.




2. Right-click on the selection and choose **Define Design Pattern...** from the popup menu.



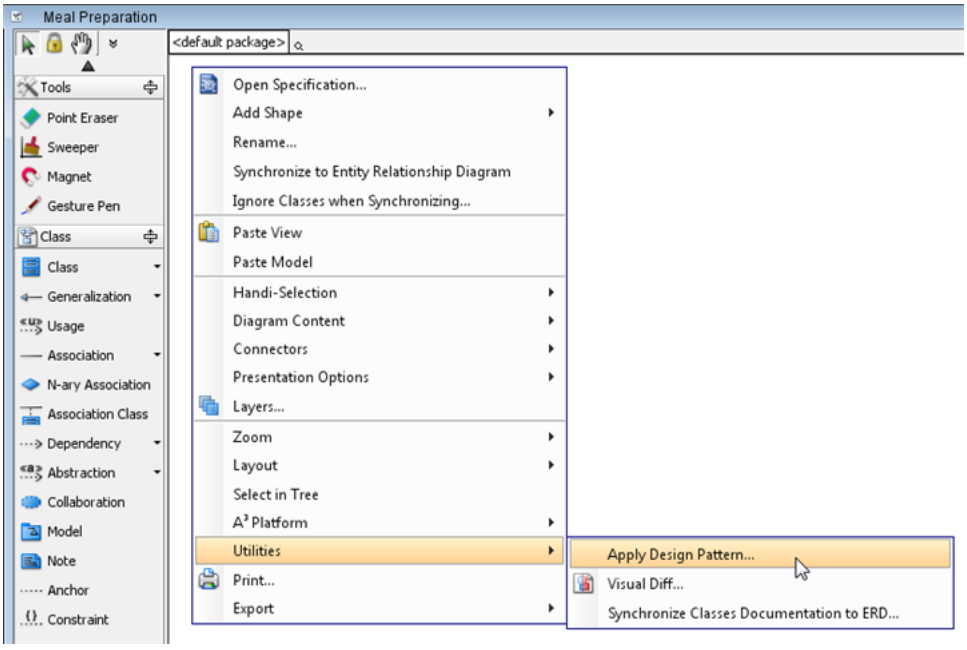
3. In the **Define Design Pattern** dialog box, specify the pattern name as *Abstract Factory*. Keep the file name as is. Click **OK** to proceed.



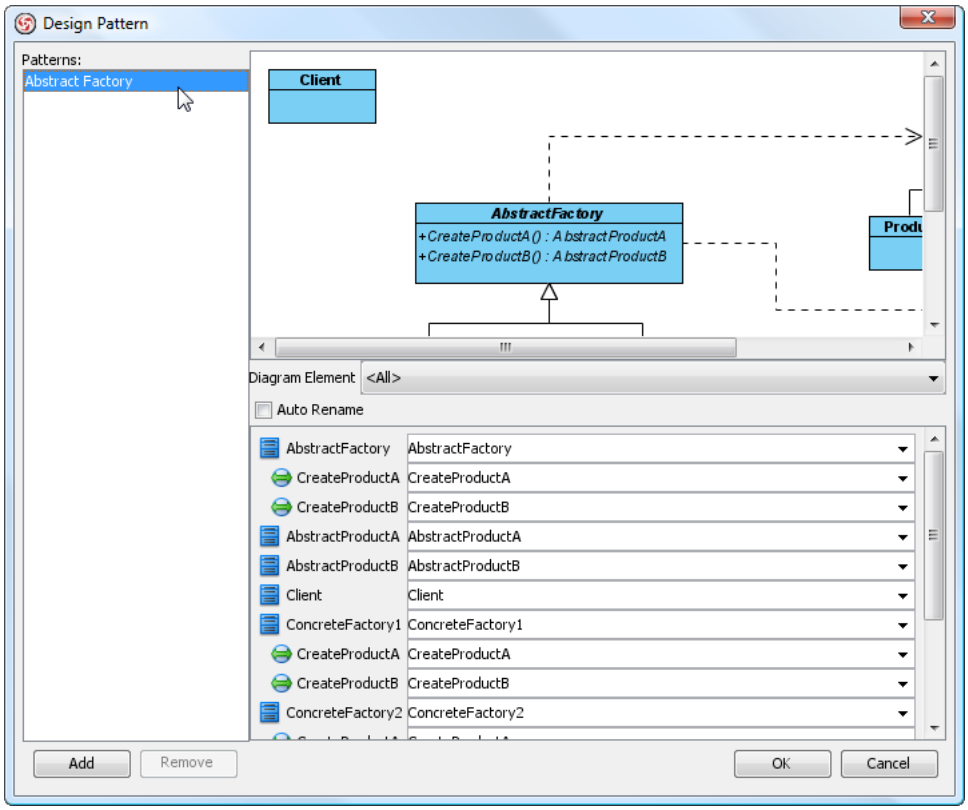

## Applying the Design Pattern to a Class Diagram

In this section, we will apply the Abstract Factory pattern to model a restaurant system that delivers both Chinese and Western meal sets.

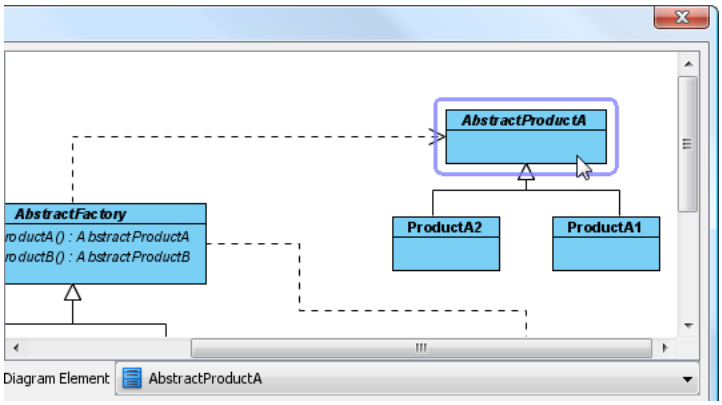
1. Create a new project named *Restaurant*.
2. Create a class diagram named *Meal Preparation*.
3. Right-click on the class diagram and select **Utilities** > **Apply Design Pattern...** from the popup menu.



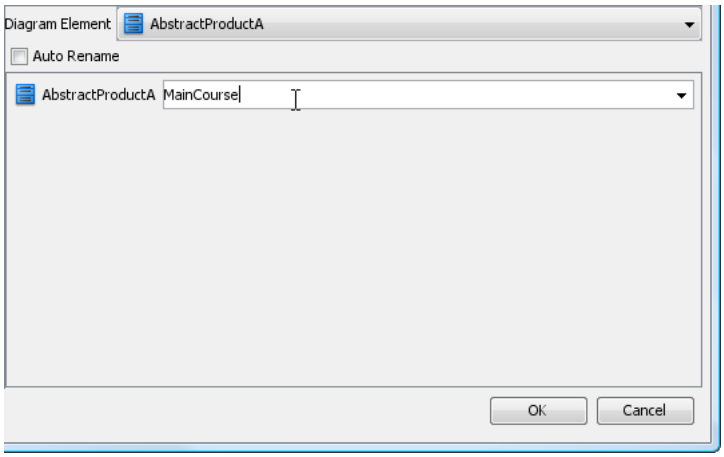
4. In the **Design Pattern** dialog box, select *Abstract Factory* from the list of patterns.



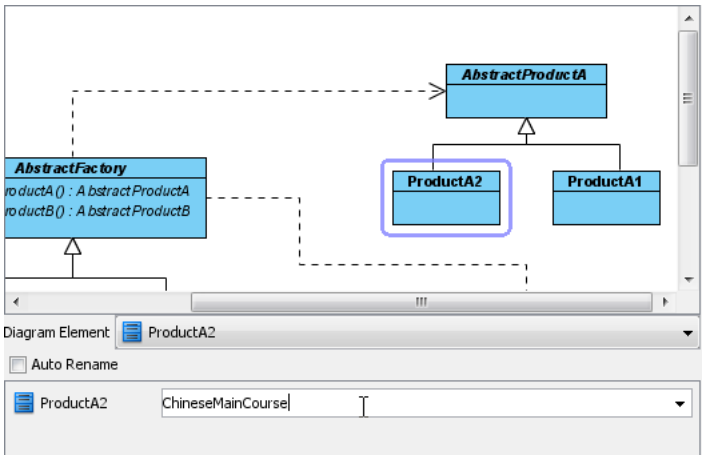

5. Click on *AbstractProductA* in the preview pane.



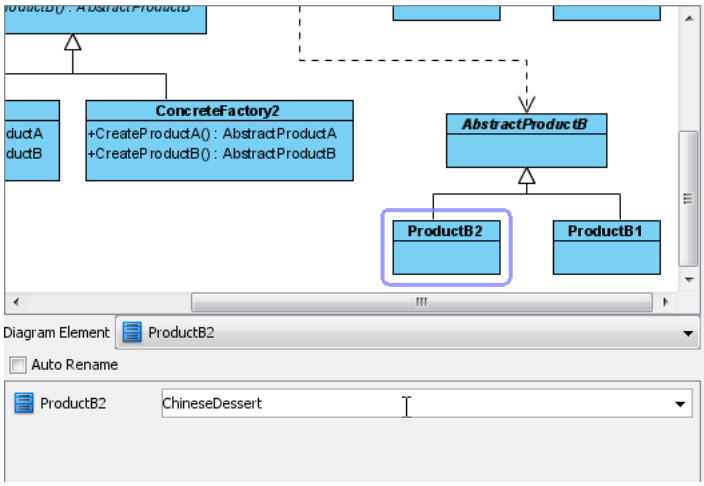
6. Rename *AbstractProductA* to *MainCourse* in the bottom pane.



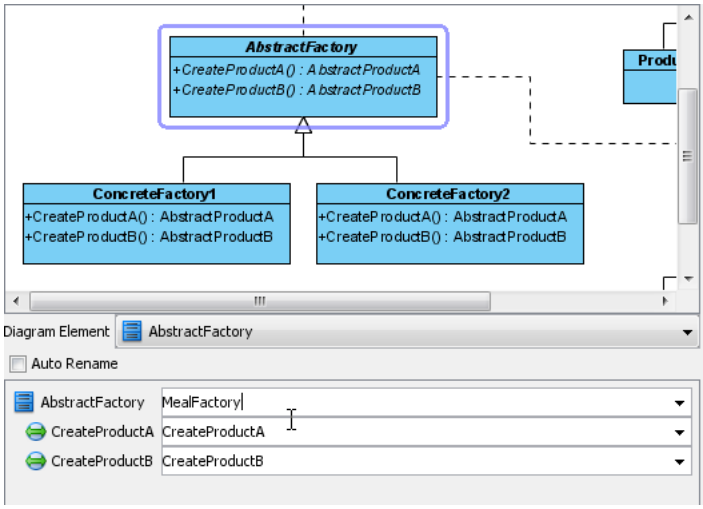

7. Similarly, rename *ProductA1* to *WesternMainCourse* and *ProductA2* to *ChineseMainCourse*.



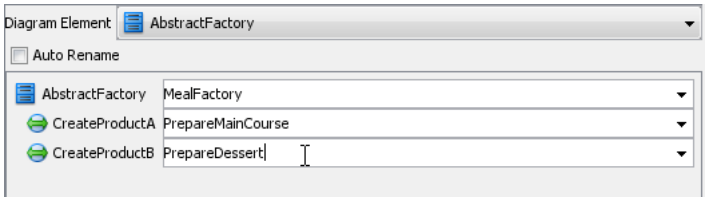
8. Rename *AbstractProductB*, *ProductB1*, and *ProductB2* to *Dessert*, *WesternDessert*, and *ChineseDessert*, respectively.



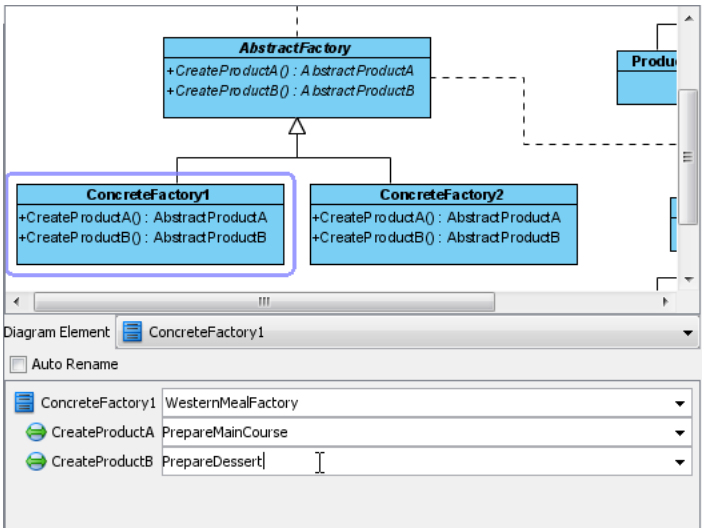

9. Now, address the factory branch. First, rename *AbstractFactory* to *MealFactory*.



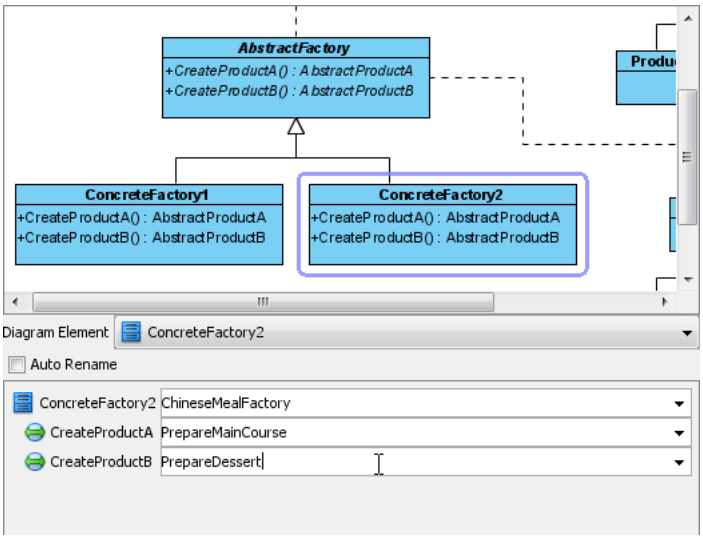

10. Also, rename the operations from *CreateProductA* and *CreateProductB* to *PrepareMainCourse* and *PrepareDessert*.



11. Similarly, rename *ConcreteFactory1* to *WesternMealFactory*, and update its operations *CreateProductA* and *CreateProductB* to *PrepareMainCourse* and *PrepareDessert*, respectively.

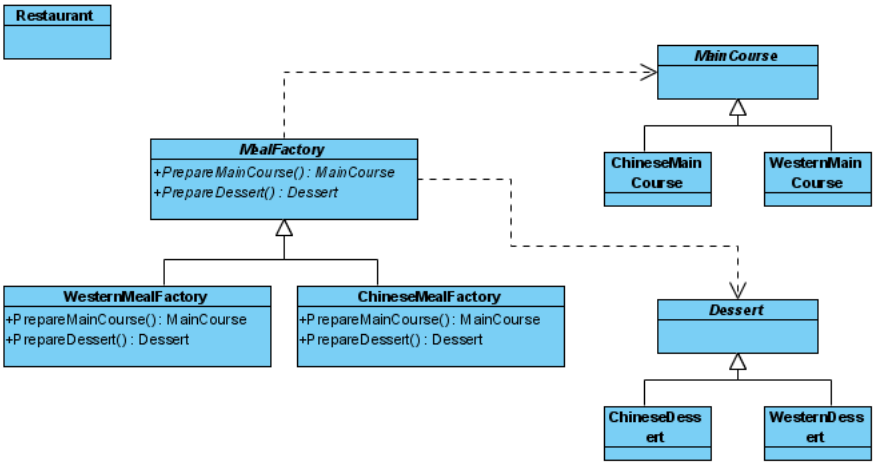



12. Likewise, rename *ConcreteFactory2* to *ChineseMealFactory*, and update its operations *CreateProductA* and *CreateProductB* to *PrepareMainCourse* and *PrepareDessert*, respectively.




13. Finally, rename *Client* to *Restaurant*.

14. Click **OK** to confirm the changes and apply the pattern. The following diagram will be generated:




Resources

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

Related Links

- Full set of UML tools and UML diagrams

Visual Paradigm

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

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