Factory Method Pattern Tutorial

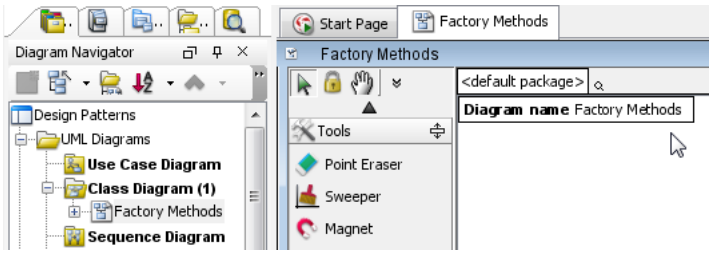
Written Date : September 28, 2009

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

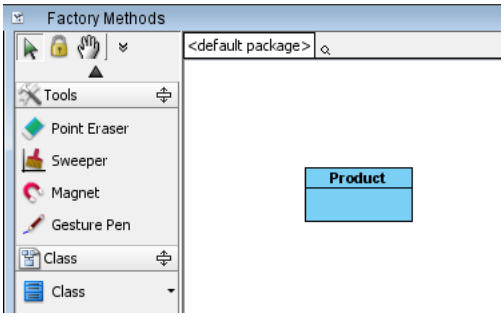
Modeling Design Pattern with Class Diagram

1. Create a new project *Design Patterns*.

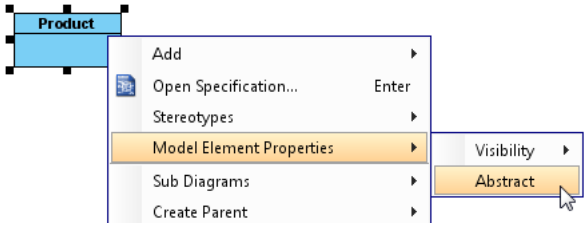
2. Create a class diagram *Factory Method*.



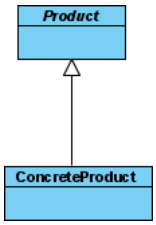
3. Select **Class** from diagram toolbar. Click on diagram to create a class. Name it as *Product*.



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

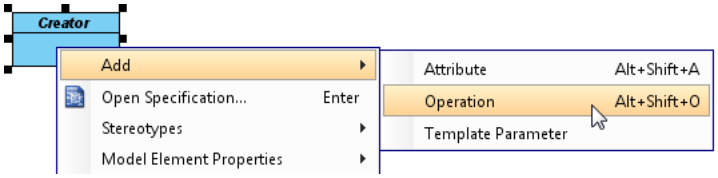


5. Move the mouse cursor over the *Product* class, and drag out **Generalization** > **Class** to create a subclass *ConcreteProduct*.

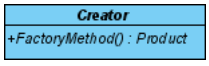


6. Create a class *Creator*, and set it as abstract.

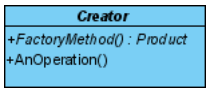
7. Right click on the *Creator* class, and select **Add** > **Operation** from the popup menu.



8. Name the operation *FactoryMethod()*, and make it return *Product*.
9. Right click on *FactoryMethod()*, and select **Model Element Properties** > **Abstract** to set it as abstract.

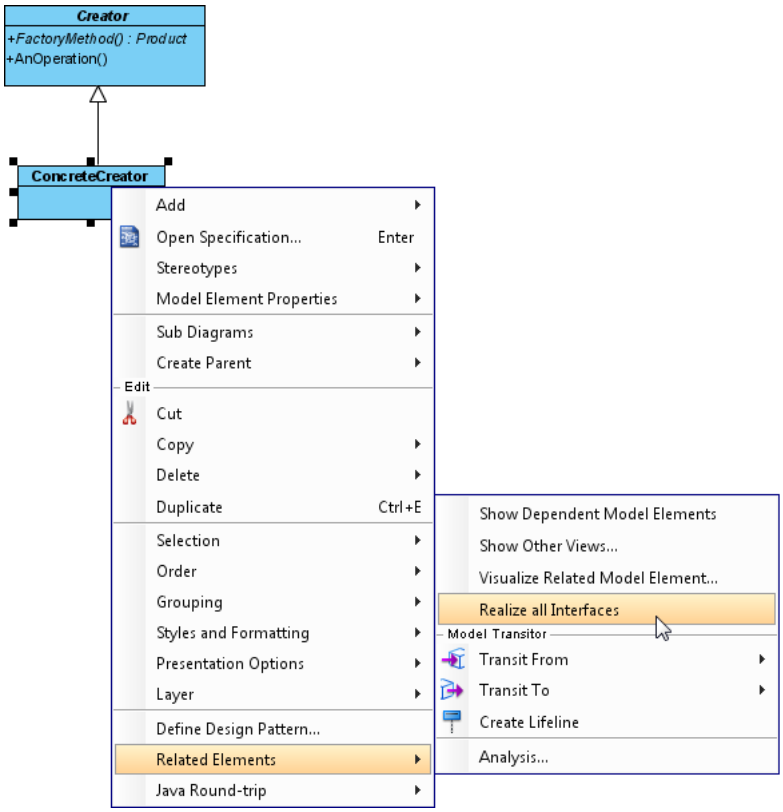


10. Add a non abstract operation *AnOperation()* to *Creator*.

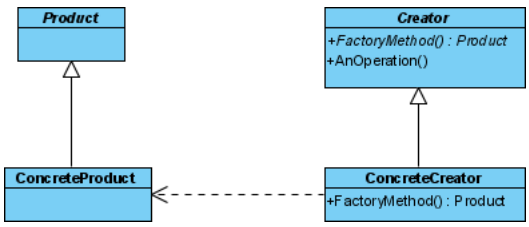


11. Move the mouse cursor over the *Creator* class, and drag out **Generalization** > **Class** to create a subclass *ConcreteCreator*.

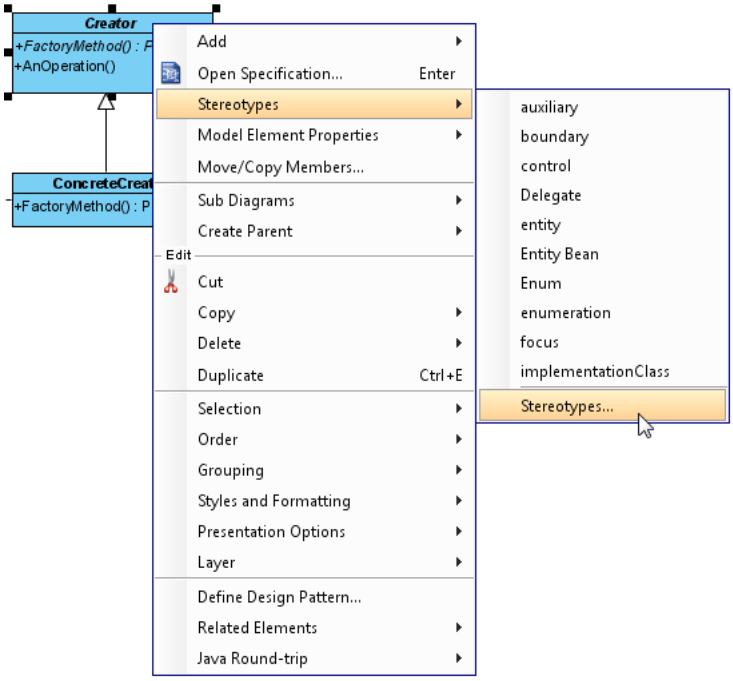
12. Make *ConcreteCreator* inherit the abstract operations provided from *Creator* by right clicking on *ConcreteCreator*, and selecting **Related Elements** > **Realize all Interfaces** from the popup menu.



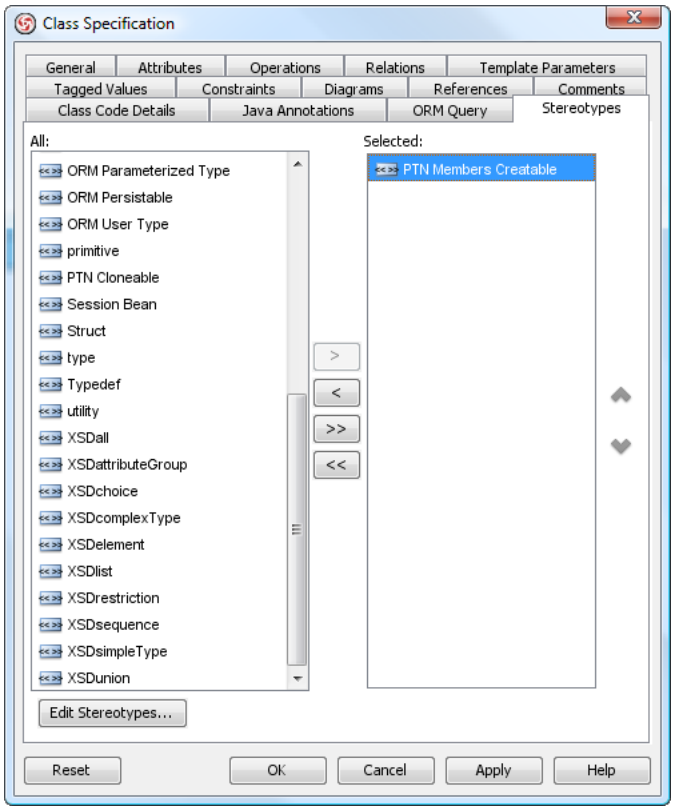
13. In practice, the *FactoryMethod* in *ConcreteCreator* is expected to return an instance of *ConcreteProduct*. Therefore, add a dependency between *ConcreteCreator* and *ConcreteProduct*. Move the mouse cursor over the *ConcreteCreator* class, and drag out **Dependency** > **Class** to *ConcreteProduct*. Up to now, the diagram should look like this:



14. There may be more than one operations in the *Creator* class. To represent this, stereotype the *Creator* class as **PTN Members Creatable**. Right click on **Creator** class and select **Stereotypes** > **Stereotype** from the popup menu.

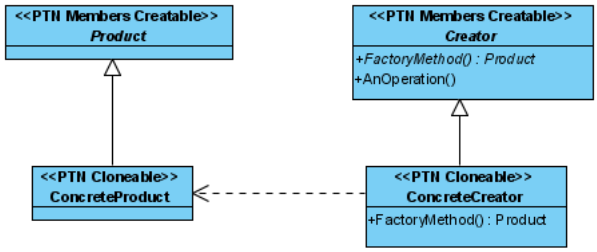


15. In the **Stereotypes** tab of class specification, select **PTN Members Creatable** and click > to assign it to the class. Click **OK** to confirm.



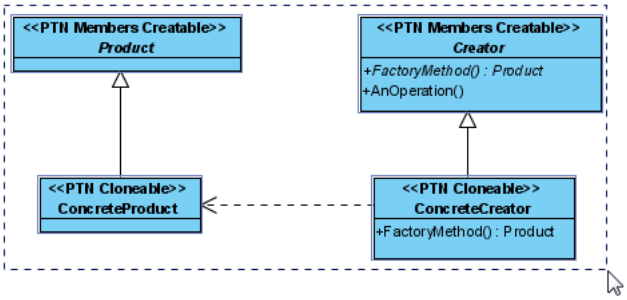
16. The *Product* class should also have its own operations. Repeat step 14 and 15 to stereotype it as **PTN Members Creatable**.

17. There may be multiple concrete products and creator. Let's repeat step 14 and 15 to stereotype *ConcreteProduct* and *ConcreteCreator* as **PTN Cloneable**. The diagram should look like this:

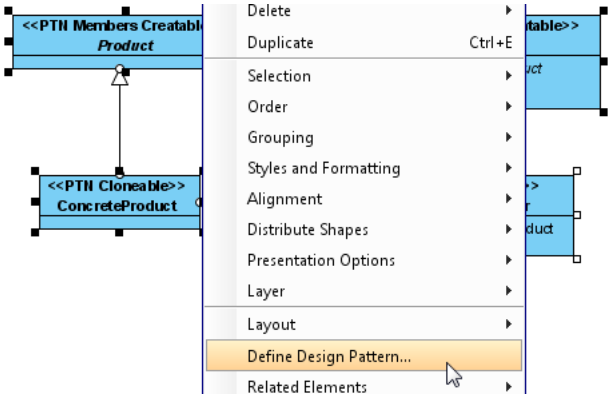


Defining Pattern

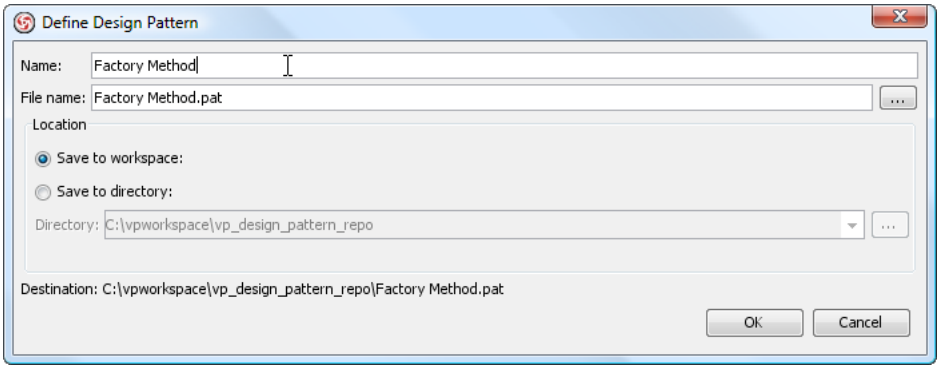
1. Select all classes on the class diagram.



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 *Factory Method*. Keep the file name as is. Click **OK** to proceed.

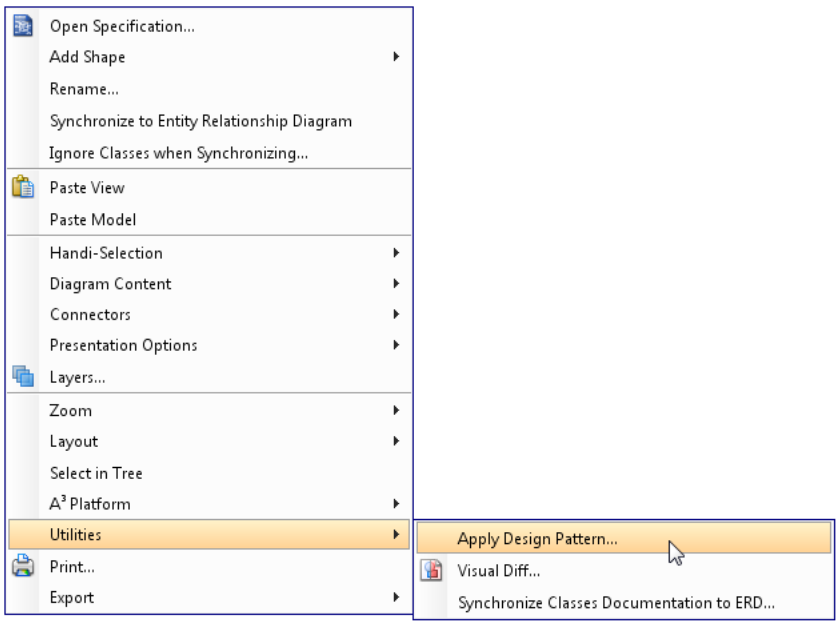


Applying Design Pattern on Class Diagram

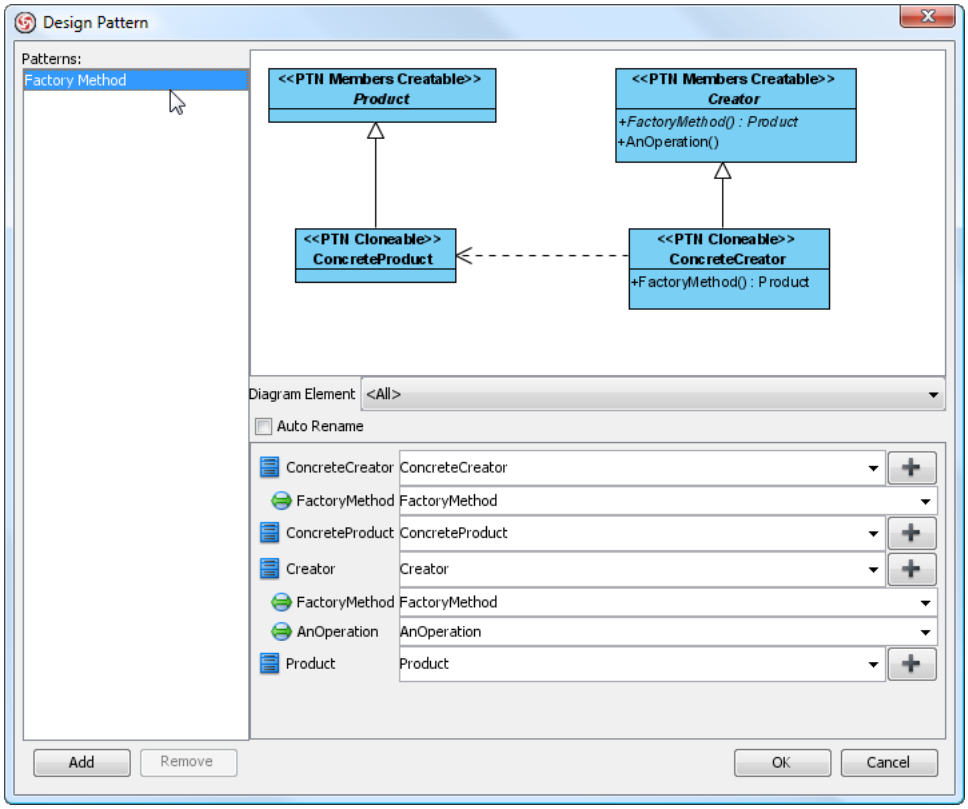
In this section, we will try to make use of the factory method pattern to model a part of a text editor.

1. Create a new project *Text Editor*
2. Create a class diagram *Domain Model*.

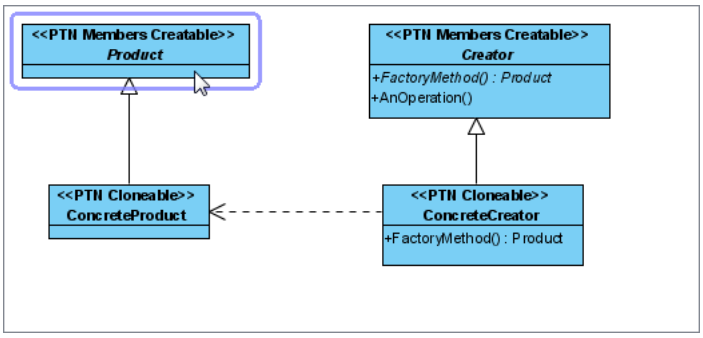
3. Right click on the class diagram and select **Utilities** > **Apply Design Pattern...** from the popup menu.



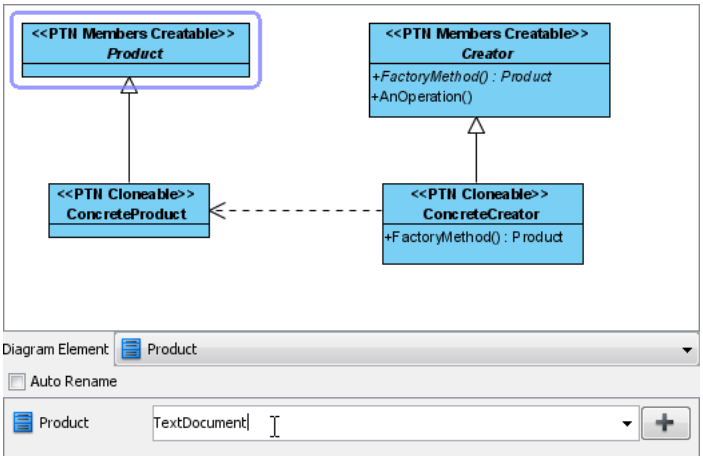
4. In the **Design Pattern** dialog box, select *Factory Method* from the list of patterns.



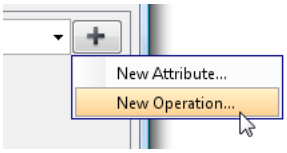
5. Click on **Product** in the overview.



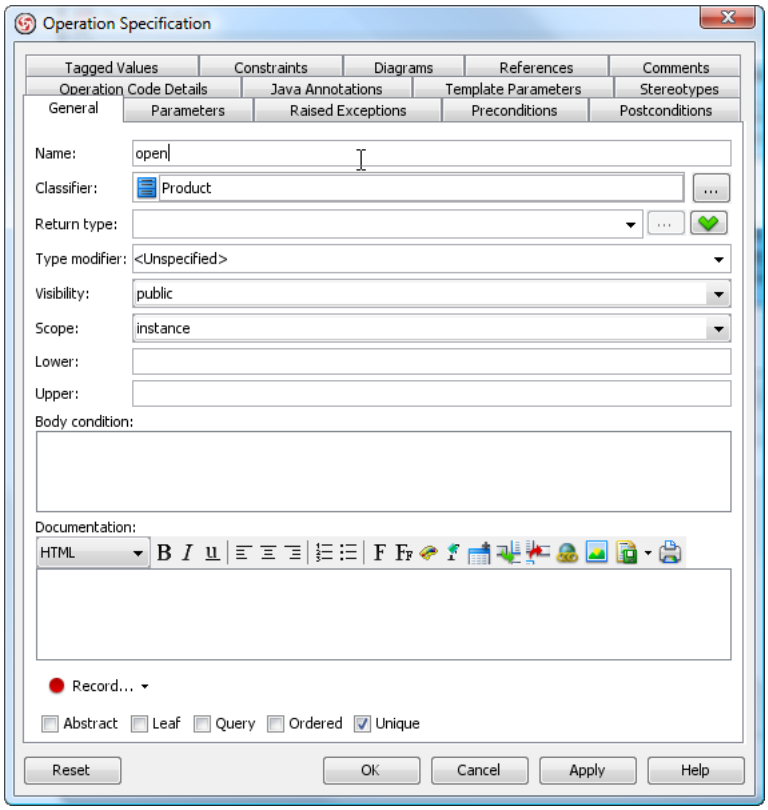
6. Rename it to *TextDocument* at the bottom pane.



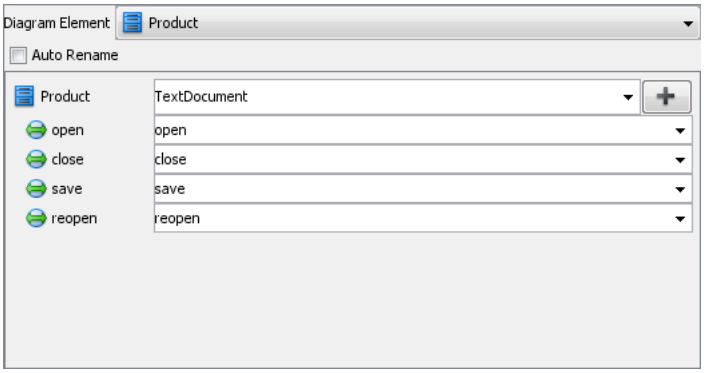
7. Click on +, and select **New Operation...** from the popup menu. We shall create the operations available in the *TextDocument* class.



8. In the **Operation Specification** dialog box, enter *Open* as operation name.

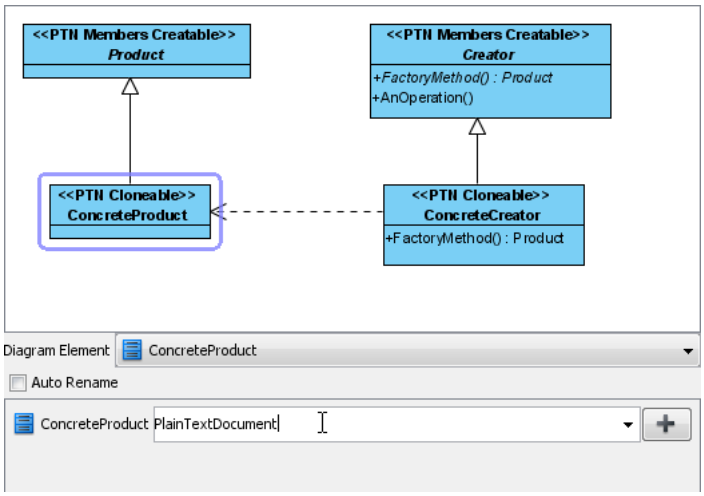


9. Repeat step 7 and 8 to create operations *close*, *save*, *reopen*.

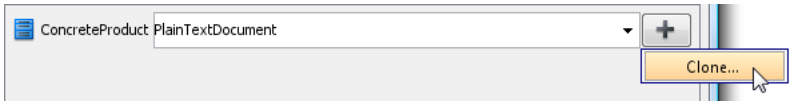


10. Click on *ConcreteProduct* in the overview.

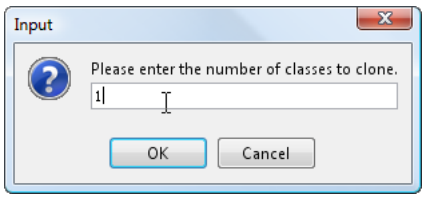
11. Rename *ConcreteProduct* to *PlainTextDocument*.



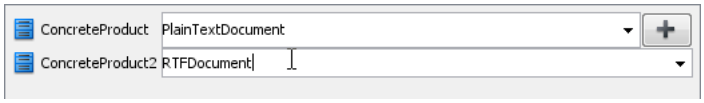
12. We need to process one more document type for RTF document. Keep *ConcreteProduct* selected and click the + button, then select **Clone...** in the popup menu.



13. Enter *1* to be the number of classes of clone. Click **OK** to confirm.

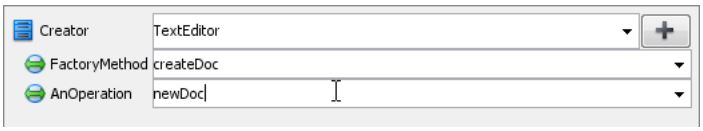


14. Enter *RTFDocument* as class name.

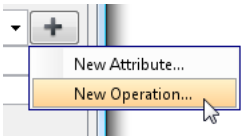


15. Select the *Creator* class in overview.

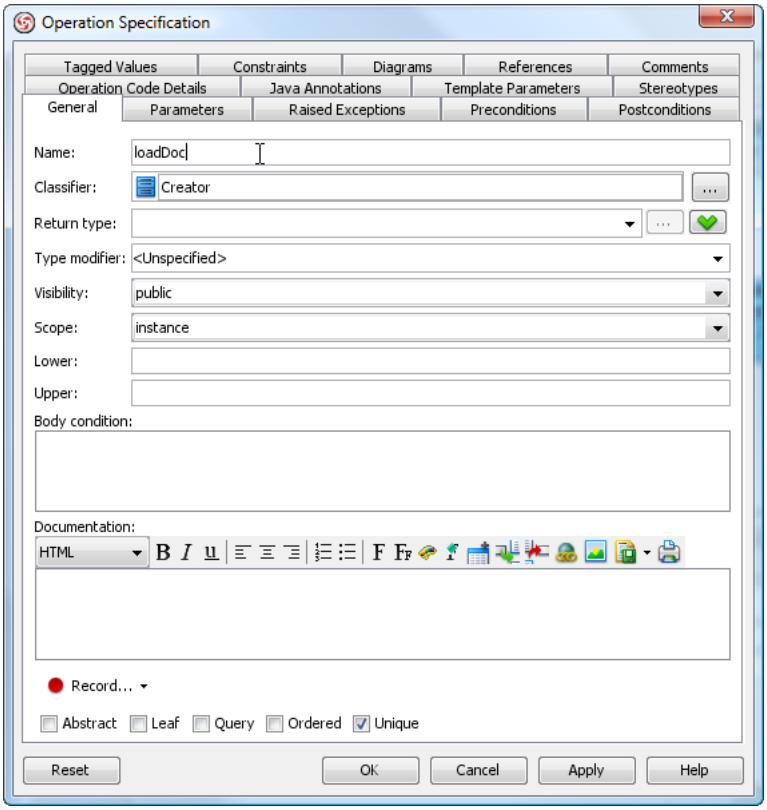
16. Rename *Creator* to *TextEditor*, operations *FactoryMethod* to *createDoc*, *AnOperation* to *newDoc*.



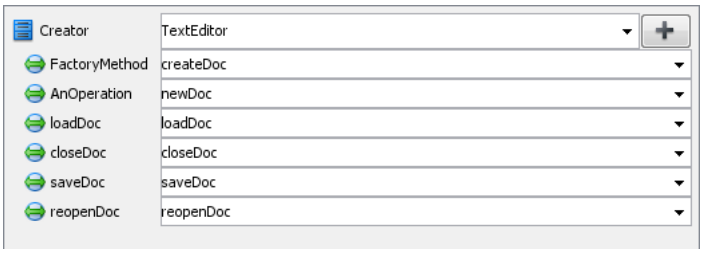
17. We need more operations. Click on +, then select **New Operation...** from the popup menu.



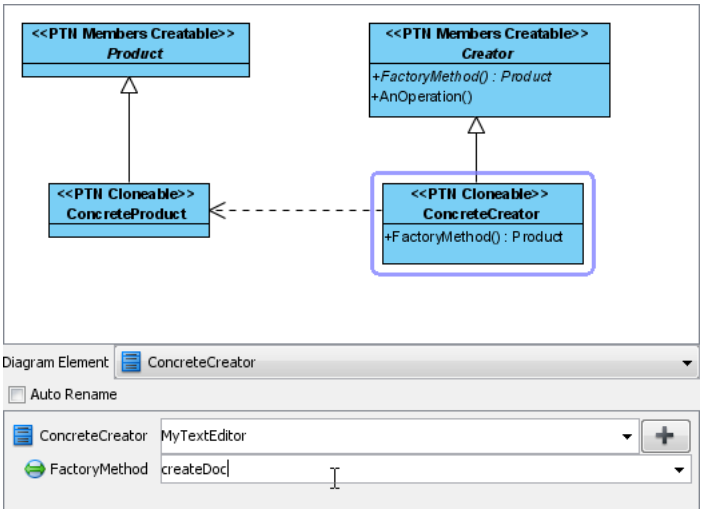
18. In the **Operation Specification** dialog box, enter *loadDoc* as name.



19. Repeat the previous steps to create operations *closeDoc*, *saveDoc*, *reopenDoc*.

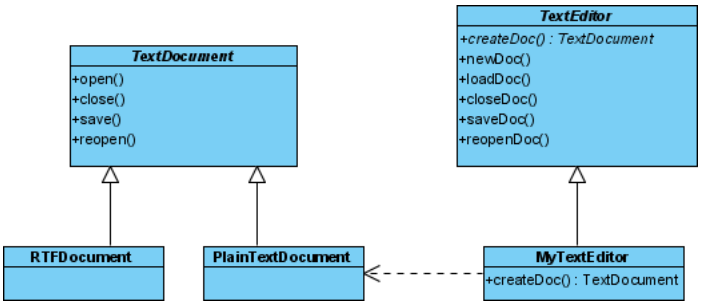


20. Select *ConcreteCreator*. Rename *ConcreteCreator* to *MyTextEditor* and operation *FactoryMethod* to *createDoc*.



21. Click **OK** to confirm editing and apply the pattern to diagram.

22. Tidy up the diagram. It should become:



Resources

1. Design Patterns.vpp
2. Factory Method.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/)