

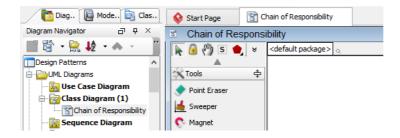
Chain of Responsibility Pattern Tutorial

Written Date : October 14, 2009

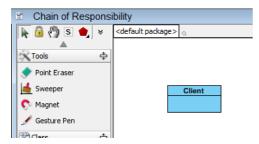
This tutorial is aimed to guide the definition and application of <u>Gang of Four (GoF)</u> chain of responsibility <u>design pattern</u>. By reading this tutorial, you will know how to develop a model for the chain of responsibility pattern, and how to apply it in practice.

Modeling Design Pattern with Class Diagram

- 1. Create a new project *Design Patterns*.
- 2. Create a class diagram Chain of Responsibility.



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



4. Move the mouse cursor over the *Client* class, and drag out **Association** > **Class** to create an associated class *Handler*.



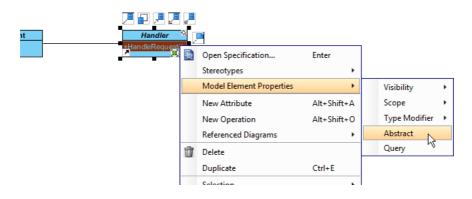
5. Right-click on *Handler*, and select **Model Element Properties > Abstract** to set it as abstract.

Handle	er	[
		Add	*	
		Open Specification	Enter	
		Stereotypes	×	
		Model Element Properties	×.	Visibility 🕨
		Sub Diagrams	•	Abstract
		Create Parent	+	45

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

Hand	ller	1			
		Add	•	Attribute	Alt+Shift+A
		Open Specification	Enter	Attribute with Getter and Setter	
		Stereotypes	•	Operation	Alt+Shift+O
		Model Element Properties	•	Constructor	
		Sub Diagrams	•	Template Parameter	
		Create Parent	•		

- 7. Name the operation *HandleRequest()*.
- 8. Right-click on *HandleRequest*, and select **Model Element Properties > Abstract** to set it as abstract.



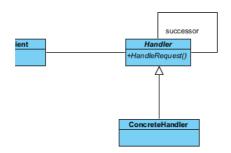
9. Move the mouse cursor over the *Handler* class, and click on the resource icon **Self Association** to create a self association.



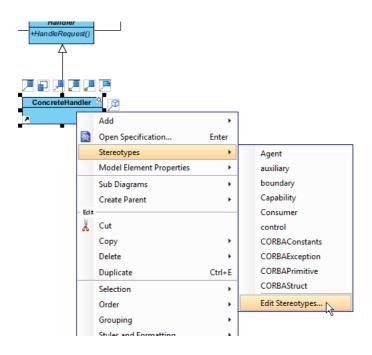
10. Name the association end successor.



11. Move the mouse cursor over the *Handler* class, and drag out **Generalization** > **Class** to create subclasses *ConcreteHandler*.



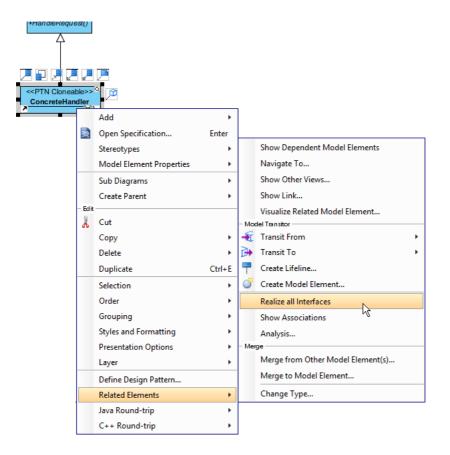
12. In practice, there may be multiple concrete handlers. To represent this, stereotype the class *ConcreteHandler* as **PTN Cloneable**. Right-click on *ConcreteHandler* and select **Stereotypes** > **Stereotype...** from the popup menu.



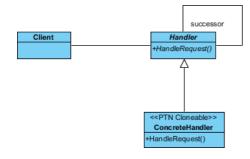
13. In the **Stereotypes** tab of the **Class Specification** dialog box, select **PTN Cloneable** and click > to assign it to *ConcreteHandler* class. Click **OK** to confirm.

\$		C	lass Sp	ecificatio	n			×
General Diagrams Class Cod		Operations ty Referen Java Annotatio		ns Chart F roiect Manao tereotypes Selected	Taggeo	Templat Quality Values	te Parameters Comments Constraints	2023
ORN O	I Abstract Pei I Component I Component I D Generatoi I Parameterizi I Persistable I User Type icipant itive rider Members Cre viceInterface sion Bean ct edef n y ypoint	r ed Type			N Cloneat		*	
Reset			ОК	Cano	el	Apply	He	p

14. We need make the concrete handlers inherit operations from the handle class. Right-click on *ConcreteHandler* and select **Related Elements** > **Realize all Interfaces** from the popup menu.

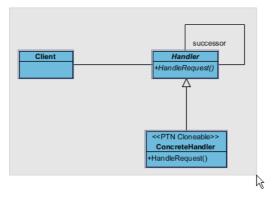


Up to now, the diagram should look like this:

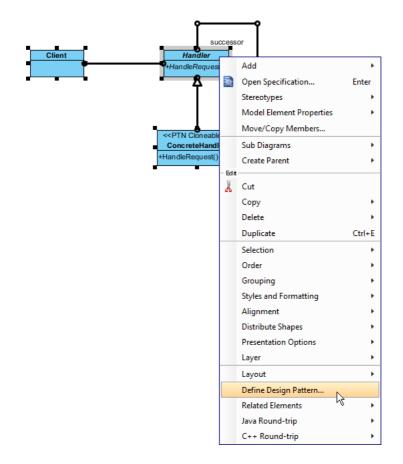


Defining Pattern

1. Select all classes on the class diagram.



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



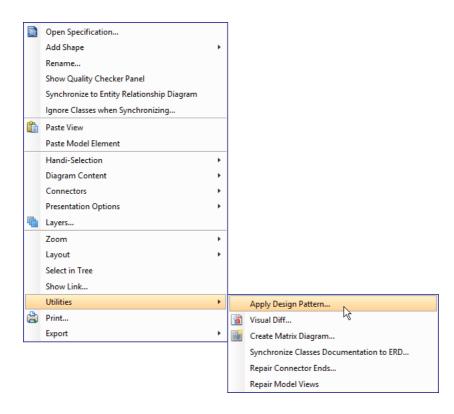
3. In the **Define Design Pattern** dialog box, specify the pattern name *Chain of Responsibility*. Keep the file name as is. Click **OK** to proceed.

\$	Define Design Pattern ×
Name:	Chain of Responsibility
File name:	Chain of Responsibility.pat
Location	
Save	to workspace:
○ Save	to directory:
Directory	/: C:\Users\John\Applications\Visual Paradigm 11.1\bin\vpworkspace\vp_design_pattern_repo v
Destination	n: C:\Users\John\Applications\Visual Paradigm 11.1\bin\vpworkspace\vp_design_pattern_repo\Chain of Responsibility.pat
	OK Cancel

Applying Design Pattern on Class Diagram

In this section, we are going to apply the chain of responsibility pattern in modeling a coin dispenser.

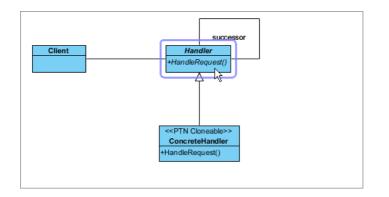
- 1. Create a new project Coin Dispenser.
- 2. Create a class diagram Domain Model.
- 3. Right-click on the class diagram and select **Utilities** > **Apply Design Pattern...** from the popup menu.



4.	In the Design	Pattern dialog box,	select Chain of	f Responsibility	from the list of	oatterns.
т.	in the Design i	allern dialog box,	Sciect Onain Of	responsionity		patterns.

\$	Design Pattern	×
Patterns: Chain of Responsibility	Client successor Handler +HandleRequest() +HandleRequest()	~
	Clent Clent ConcreteHandler ConcreteHandler HandleRequest HandleRequest HandleRequest HandleRequest HandleRequest HandleRequest Association successor Role1 successor	
Add Remove	OK Cancel	

5. Click on *Handler* in the overview.



6. Rename *Handler* to *CoinDispenser*, and operation *HandleRequest* to *Dispense* at the bottom pane.

📒 Handler	CoinDispenser	¥	
😂 HandleRequest	Dispense	¥	

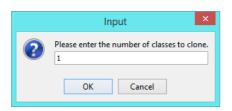
7. Click on *ConcreteHandler* in overview, and rename it to *DollarDispenser*, and operation *HandleRequest* to *Dispense*.

Client		Successor Handler +HandleRequest() < <ptn cloneable="">> ConcreteHandler +HandleRequest()</ptn>
Diagram Element 📃 (ConcreteHandler	×
Auto Rename		
ConcreteHandler	DollarDispenser	~ +
😝 HandleRequest	Dispense	×

8. We need one more concrete handler for dispensing cents. Keep *ConcreteHandler* selected, click on + and select **Clone...** from the popup menu.

eteHandler	DollarDispenser	~	+				
dleRequest	Dispense		(Clor	ne	Ν	
						W	

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

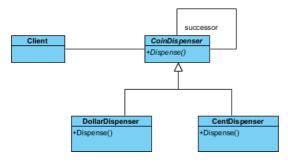


10. Rename ConcreteHandler2 to CentDispenser, and operation HandleRequest to Dispense.

E ConcreteHandler	DollarDispenser	~	+]
😂 HandleRequest	Dispense		~]
ConcreteHandler2	CentDispenser		Ý]
😝 HandleRequest	Dispense		Ý]

11. Click **OK** to apply the pattern to diagram.

12. Tidy up the diagram. Here is the result:



Resources

- 1. Chain of Responsibility.pat
- 2. Design Patterns.vpp

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/)