

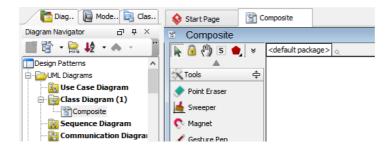
# **Composite Pattern Tutorial**

Written Date : October 7, 2009

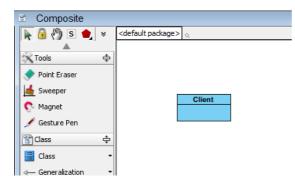
This tutorial is aimed to guide the definition and application of <u>Gang of Four (GoF)</u> composite <u>design</u> <u>pattern</u>. By reading this tutorial, you will know how to develop a model for the composite pattern, and how to apply it in practice.

## Modeling Design Pattern with Class Diagram

- 1. Create a new project *Design Patterns*.
- 2. Create a class diagram *Composite*.



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 *Component*.



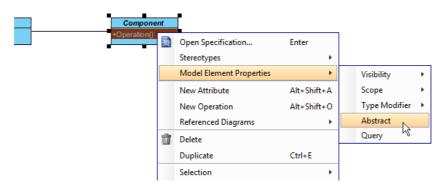
5. Right-click on *Component*, and select **Model Element Properties** > **Abstract** to set it as abstract.

Сотро	nent			
		Add	*	
		Open Specification	Enter	
		Stereotypes	×	
		Model Element Properties	•	Visibility 🕨
		Sub Diagrams	•	Abstract
		Create Parent	۲	K

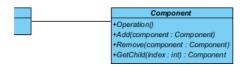
6. Right-click on the *Component* class, and select **Add** > **Operation** from the popup menu.

Сотр	onent				
		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	•		

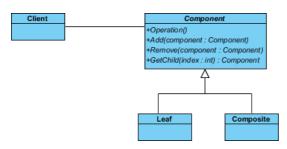
- 7. Name the operation *Operation()*.
- 8. Right-click on *Operation*, and select **Model Element Properties > Abstract** to set it as abstract.



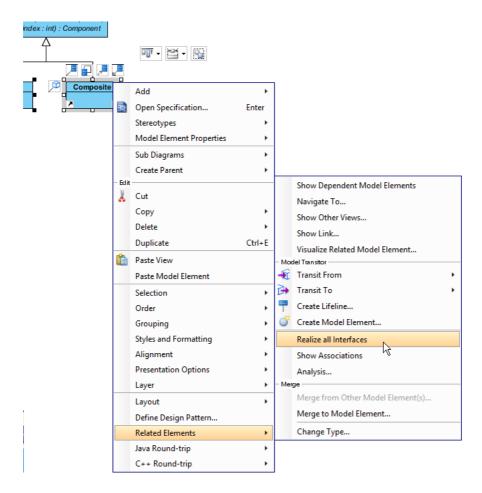
9. Repeat steps 6 to 8 to create operations Add(component : Component), Remove(component : Component), GetChild(index : int) : Component.



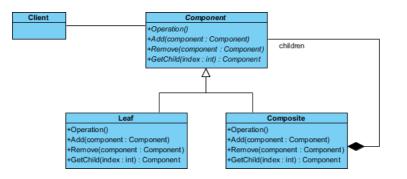
10. Move the mouse cursor over the *Component* class, and drag out **Generalization** > **Class** to create a subclass *Leaf*. Repeat this step to create another subclass *Composite*, from *Component*.



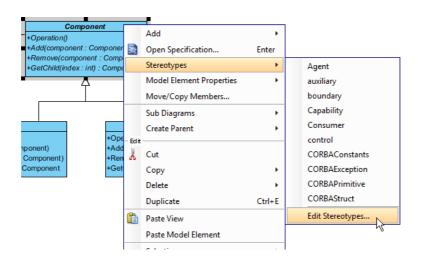
11. *Leaf* and *Composite* will inherit the operations from *Component*. Select *Leaf* and *Component*, right-click on them and select **Related Elements** > **Realize all Interfaces** from the popup menu.



12. Move the mouse cursor over the *Component* class, and drag out **Composition** > **Class** to *Component*. Name the Component's role as *children*.



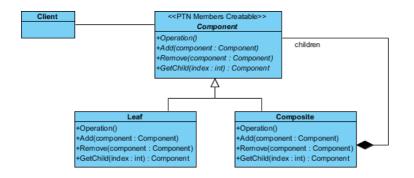
13. In practice, there may be multiple operations in *Component*. To represent this, stereotype the class *Component* as **PTN Members Creatable**. Right-click on *Component* and select **Stereotypes > Stereotypes...** from the popup menu.



14. In the **Stereotypes** tab of the **Class Specification** dialog box, select **PTN Members Creatable** and click > to assign it to *Component* class. Click **OK** to confirm.

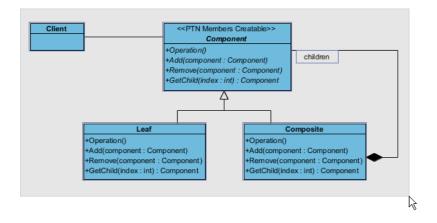
<b>\$</b>		C	lass Spec	ificatio	ı		×
General At Diagrams Class Code D All: Class Code D All: Class Code D ORM At ORM At ORM At ORM D ORM Particip ORM Us ORM U	Traceability etails Ja ss bostract Persis component Generator arameterized ersistable ser Type ant e r ineable Interface n Bean	Operations Reference va Annotatio stable	Relations ces Proie	Chart R ect Manage eotypes Selected:	elations ement Taggeo	Templat Quality d Values	ters 📝 ents
Struct							
eset			▼ OK	Canc	el	Apply	Help

Up to now, the diagram should look like this:

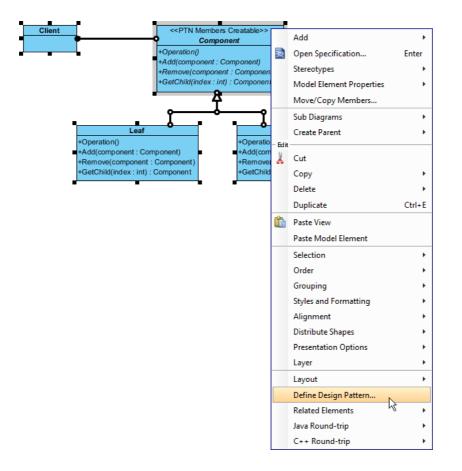


# **Defining Pattern**

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

<b>\$</b>	Define Design Pattern ×					
Name:	Composite					
File name:	Composite.pat					
Location						
Ŭ	to workspace: to directory:					
Directory	Directory: C:\Users\John\Applications\Visual Paradigm 11.1\bin\vpworkspace\vp_design_pattern_repo					
Destination	Destination: C: \Users\John\Applications\Visual Paradigm 11.1\bin\vpworkspace\vp_design_pattern_repo\Composite.pat					
	OK Curci					

## **Applying Design Pattern on Class Diagram**

In this section, we are going to apply the composite pattern to model a furniture shop's furniture catalog.

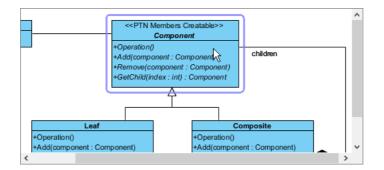
- 1. Create a new project Furniture Shop.
- 2. Create a class diagram Domain Model.
- 3. Right-click on the class diagram and select **Utilities** > **Apply Design Pattern...** from the popup menu.

	Open Specification		
	Add Shape	۲	
	Rename		
	Show Quality Checker Panel		
	Synchronize to Entity Relationship Diagram		
	Ignore Classes when Synchronizing		
(în)	Paste View		
	Paste Model Element		
	Handi-Selection	۲	
	Diagram Content	۲	
	Connectors	۲	
	Presentation Options	۲	
6	Layers		
	Zoom	۲	
	Layout	۲	
	Select in Tree		
	Show Link		
	Utilities	•	Apply Design Pattern
٢	Print		Visual Diff
	Export	•	Create Matrix Diagram
			Synchronize Classes Documentation to ERD
			Repair Connector Ends
			Repair Model Views

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

<b>\$</b>		Design Pattern		×
Patterns: Composite	Client	+Operation() +Add(component	nent : Component)	children
		Leaf +Operation() +Add(component : Component)	+Operation()	mposite
	<	<		>
	Auto Rename			*
	Client Component			· +
	Add	Add		
		component Remove		
	<u> </u>	component		
		GetChild		<u> </u>
		index Composite		
	Operation			· · ·
Add Remove			ОК	Cancel

5. Click on *Component* in the overview.



6. Rename *Component* to *Furniture*, and the parameters *component* in various operations to *furniture* at the bottom pane.

Component	Furniture	~ <b>+</b>
😂 Operation	Operation	*
😝 Add	Add	¥
component	furniture	
😂 Remove	Remove	~
component	furniture	
😂 GetChild	GetChild	*
index	index	

7. Rename Operation to ShowPrice.

Component	Furniture	~ <b>+</b>	
😑 Operation	ShowPrice		~
😝 Add	Add		~
component	furniture		
😂 Remove	Remove		~
component	furniture		
😝 GetChild	GetChild		~
index	index		

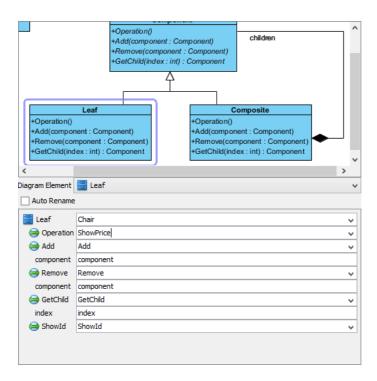
8. Besides the operation *ShowPrice*, we also need one more operation for *ShowId*. Keep *Component* selected, click on the + button at the bottom pane, and select **New Operation...** from the popup menu.

Component	Furniture v	÷		
😝 Operation	ShowPrice		New	Attribute
😝 Add	Add		New	Operation N
component	furniture			·
😝 Remove	Remove	~		
	1 - 14			

9. In the **Operation Specification** dialog box, name the operation *Showld*. Check **Abstract** at the bottom of dialog box.

\$	•		Opera	ation Specific	ation		×	
	Reference	-				Comments	*	
	Stereotypes							
			Code Details Java Annotations Template Parameters					
	General	Parameters	Raised E	Exceptions	Preconditions	Postconditions	+	
	Name:	ShowId						
	Classifier:	Compo	nent					
	Return type:					¥		
	Type modifier:	<unspecifie< td=""><td>d&gt;</td><td></td><td></td><td>~</td><td></td></unspecifie<>	d>			~		
	Visibility:	public				¥		
	Scope:	instance				~		
	Lower:							
	Upper:							
	Body condition	:						
	Documentation	:						
	BIu≣	ΞĘΞ	🗄 F Ff 🇞 📑	<u> - 📽 🦿 + -</u>	<b>•</b>	**		
	Record							
	✓ Abstract	Leaf	Query C	Ordered 🔽 Uniq	ue			
[	Reset			ОК	Cancel	Apply Hel	p	

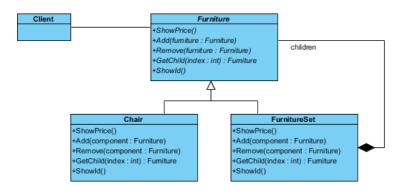
10. Select *Leaf* in overview, and rename it as *Chair* at the bottom pane. Rename also the operation *Operation* to *ShowPrice*. Note that if the option **Auto Rename** is on, rename of operation is not needed as this will be done automatically.



11. Select *Composite* in overview, and rename it as *FurnitureSet* at the bottom pane. Rename also *Operation* to *ShowPrice*. Click **OK** to apply the pattern to diagram.

	+Operation() +Add(component : Cd +Remove(component +GetChild(index : int) 	: Component)
+GetChild(ind	ex : int) : Component	+GetChild(index : int) : Component
<		>
Diagram Element	📑 Composite	
Auto Rename		
Composite	FurnitureSet	~
😝 Operation	ShowPrice	~
😂 Add	Add	~
component	component	
😝 Remove	Remove	~
component	component	
😝 GetChild	GetChild	~
index	index	
😝 ShowId	ShowId	~

#### This is the result:



## Resources

- 1. <u>Composite.pat</u>
- 2. <u>Design Patterns.vpp</u>

### **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/)