

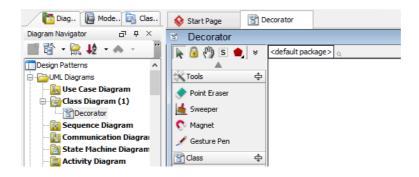
# **Decorator Pattern Tutorial**

Written Date : October 8, 2009

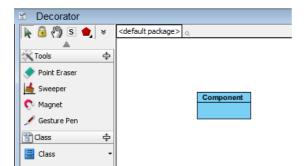
This tutorial is aimed to guide the definition and application of <u>Gang of Four (GoF)</u> decorator <u>design</u> <u>pattern</u>. By reading this tutorial, you will know how to develop a model for the decorator pattern, and how to apply it in practice.

## Modeling Design Pattern with Class Diagram

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



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



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

Comp	onent			
		Add	•	
		Open Specification	Enter	
		Stereotypes	•	
		Model Element Properties	۱.	Visibility 🕨
		Sub Diagrams	•	Abstract
		Create Parent	•	NG.

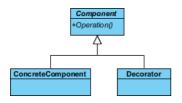
5. Right-click on the *Component* class, and select **Add** > **Operation** from the popup menu.

Compon	ent			_
		Add	×	Attribute Alt+Shift+A
		Open Specification	Enter	Attribute with Getter and Setter
		Stereotypes	•	Operation Alt+Shift+C
		Model Element Properties	•	Constructor
		Sub Diagrams	+	Template Parameter
		Create Parent	•	

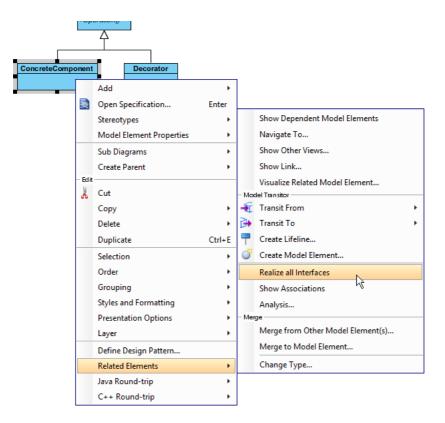
- 6. Name the operation *Operation()*.
- 7. Right-click on *Operation*, and select **Model Element Properties > Abstract** to set it as abstract.

Compone +Operation	_				
	Ŕ	Open Specification	Enter		
		Stereotypes	•		
		Model Element Properties	•	Visibility	+
		New Attribute	Alt+Shift+A	Scope	•
		New Operation	Alt+Shift+O	Type Modifier	•
		Referenced Diagrams	•	Abstract	
	Ŵ	Delete		Query 너	
		Duplicate	Ctrl+E		
		0.1			

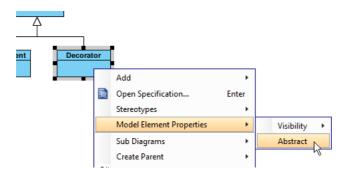
8. Move the mouse cursor over the *Component* class, and drag out **Generalization** > **Class** to create a subclass *ConcreteComponent*. Repeat this step to create another subclass *Decorator*.



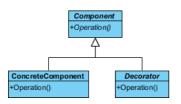
 ConcreteComponent should inherit the operations from Component. Select ConcreteComponent, right-click on and select Related Elements > Realize all Interfaces from the popup menu.



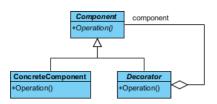
10. Decorator is an abstract class. Right-click on the *Decorator* class, and select **Model Element Properties** > **Abstract** to set it as abstract.



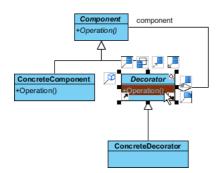
11. Decorator should also inherit the operations from *Component*. Select *Decorator*, right-click on and select **Related Elements** > **Realize all Interfaces** from the popup menu.



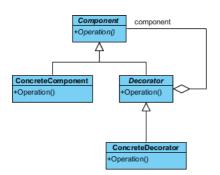
12. Move the mouse cursor over the *Decorator* class, and drag out **Aggregation** > **Class** to *Component*. Name the *Component*'s role as *component*.



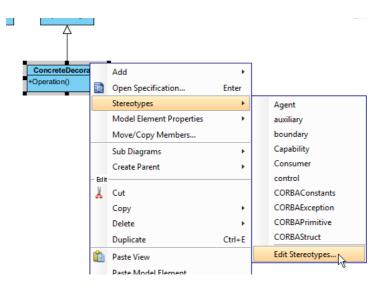
- 13. Move the mouse cursor over the *Decorator* class, and drag out **Generalization** > **Class** to create a subclass *ConcreteDecorator*.
- 14. We shall make *ConcreteDecorator* implements the decorator operation. Select *Operation* in *Decorator*.



15. Press the Ctrl key, and drag to ConcreteDecorator.



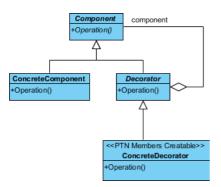
16. In practice, there may be added behaviors in concrete decorators. To represent this, stereotype the class *ConcreteDecorator* as **PTN Members Creatable**. Right-click on *ConcreteDecorator* and select **Stereotypes > Stereotypes...** from the popup menu.



17. 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.

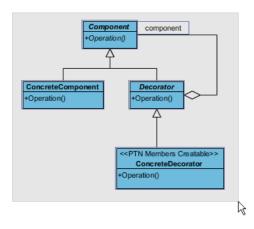
<b>\$</b>		Class	Speci	ificatio	n			×
		ferences		Chart R ct Manao otypes		Templat Quality Values	te Parame Comm Constra	ents aints
CRM Com	Driven Bean Type s stract Persistable nponent enerator ameterized Type sistable r Type nt eable terface Bean	~	> <	Selected:		rs Creatab		*
Reset		OK	(	Cano	el	Apply		Help

Up to now, the diagram should look like this:

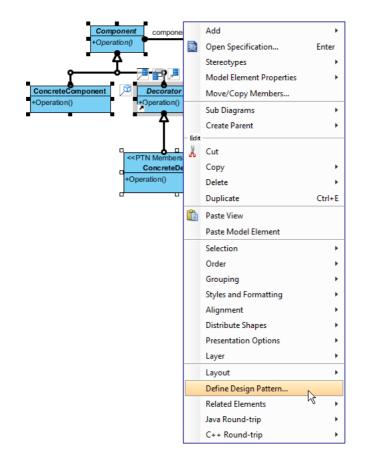


# **Defining Pattern**

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 *Decorator*. Keep the file name as it. Click **OK** to proceed.

<b>\$</b>	Define Design Pattern ×
Name:	Decorator
File name:	Decorator.pat
Location	h under an
Ŭ	to workspace: to directory:
Director	y: 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\Pecorator.pat

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

In this section, we are going to apply the decorator pattern to model a domain model of diagram editor.

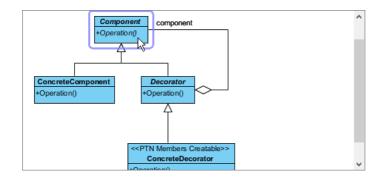
- 1. Create a new project *Diagram Editor*.
- 2. Create a class diagram Domain Model.
- 3. Right-click on the class diagram and select **Utilities** > **Apply Design Pattern...** from the popup menu.

	Open Specification		]	
	Add Shape	•		
	Rename			
	Show Quality Checker Panel			
	Synchronize to Entity Relationship Diagram			
	Ignore Classes when Synchronizing			
<b>i</b>	Paste View			
	Paste Model Element			
	Handi-Selection	•		
	Diagram Content	۲		
	Connectors	۲		
	Presentation Options	۲		
•	Layers			
	Zoom	•		
	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

<b>&gt;</b>	Design Pattern	
Patterns: Decorator	Component +Operation() ConcreteComponent +Operation() ConcreteDecorator +Operation() ConcreteDecorator +Operation()	,
	Diagram Element   <all></all>	
	Component Component	~
	Operation Operation Operation ConcreteComponent	~
	Operation Operation	<b>v</b>
	ConcreteDecorator ConcreteDecorator Operation Operation	<b>+</b>
	Decorator Decorator	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
	Operation Operation	~
	Association Role 1	
	Role2 component	
	оче	Cancel

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

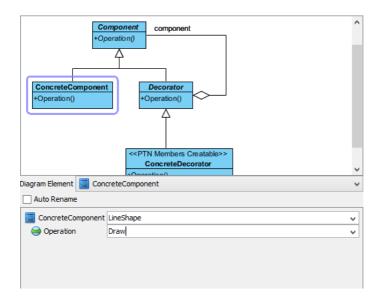
5. Click on *Component* in the overview.



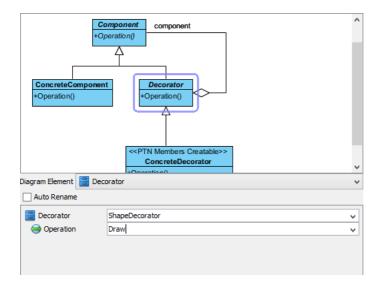
6. Rename *Component* to *Shape* at the bottom pane, and operation *Operation* to *Draw*.

Component	Shape	v
😂 Operation	Draw	¥

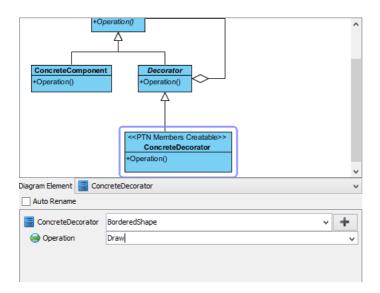
7. Select *ConcreteComponent* in overview, and rename it to *LineShape*, and its operation *Operation* to *Draw* at the bottom pane.



8. Select *Decorator* in overview, and rename it to *ShapeDecorator*, and its operation *Operation* to *Draw* at the bottom pane.



9. Select *ConcreteDecorator* in overview, and rename it to *BorderedShape*, and its operation *Operation* to *Draw* at the bottom pane.



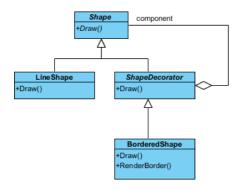
10. We need several operations for additional behaviors in *BorderedDecorator*. Click on the + button at the bottom pane and select **New Operation...** from the popup menu.

ConcreteDecorator	BorderedShape v	+				
😝 Operation	Draw	New Attribute				
			Ne	ew (	Operation	N
						N

11. In the **Operation Specification** dialog box, enter *RenderBorder* as operation name. Click **OK** to confirm.

Ś	•		Ope	eration Specifi	cation			×		
	Stereotypes									
	General	n Code Detail Parameters	-	Java Annotations d Exceptions	Preconditions	ate Paramete Postcone				
	Name:	RenderBorde	er							
	Classifier:	Concre	teDecorator							
	Return type:					¥	*			
	Type modifier:	<unspecified< td=""><td>4&gt;</td><td></td><td></td><td></td><td>~</td><td></td></unspecified<>	4>				~			
	Visibility:	public					~			
	Scope:	instance					*			

### 12. Click **OK** to apply the pattern to diagram. This is the final diagram:



Resources

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

#### **Related Links**

• Full set of UML tools and UML diagrams



Visual Paradigm tutorials

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