

Bridge Pattern Tutorial

Written Date : October 8, 2009

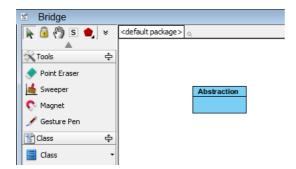
This tutorial is aimed to guide the definition and application of <u>Gang of Four (GoF)</u> bridge <u>design</u> <u>pattern</u>. By reading this tutorial, you will know how to develop a model for the bridge pattern, and how to apply it in practice.

Modeling Design Pattern with Class Diagram

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



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



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

Abstrac	tion	[_
		Add	•	
		Open Specification	Enter	
		Stereotypes	•	
		Model Element Properties	۱.	Visibility 🕨
		Sub Diagrams	•	Abstract
		Create Parent	•	ht.

5. Move the mouse cursor over the *Abstraction* class, and drag out **Aggregation** > **Class** to create an associated class *Implementor*.



- 6. Right-click on *Implementor*, and select **Model Element Properties** > **Abstract** to set it as abstract.
- 7. Right-click on the *Abstract* class, and select **Add** > **Operation** from the popup menu.

Abstrac	, ion		Implemento	r		
7		Add	۲.		Attribute	Alt+Shift+A
	B	Open Specification	Enter		Attribute with Getter and Setter	
		Stereotypes	•		Operation N	Alt+Shift+O
		Model Element Properties	; ▶		Constructor	
		Sub Diagrams	•		Template Parameter	
		Create Parent	+			

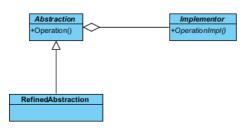
- 8. Name the operation Operation().
- 9. Right-click on the *Implementor* class, and select **Add** > **Operation** from the popup menu. Name the operation *OperationImpl()*.



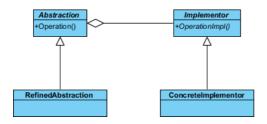
10. Right-click on *Implementor*, and select **Model Element Properties** > **Abstract** to set it as abstract.

n	Implementor				
<u> </u>	+OperationImpl/	Open Specification	Enter		
		Stereotypes	•		
		Model Element Properties	+	Visibility	+
		New Attribute	Alt+Shift+A	Scope	+
		New Operation	Alt+Shift+O	Type Modifier	+
		Referenced Diagrams	•	Abstract	
	Ū	Delete		Query K	
		Duplicate	Ctrl_F		

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



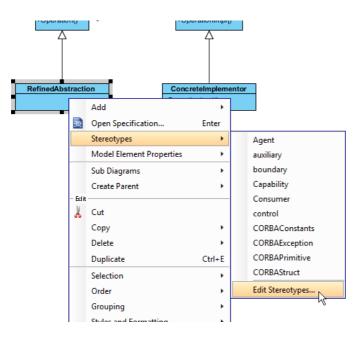
12. Repeat the previous step to create a subclass ConcreteImplementor from Implementor.



 ConcreteImplementor will inherits the operations from Implementor. Right-click on ConcreteImplementor and select Related Elements > Realize all Interfaces from the popup menu.

+Operation 2	2				
		Add	•]	
		Open Specification	Enter	-	Shaw Day and and Madel Flags and
		Stereotypes	•		Show Dependent Model Elements
		Model Element Properties		-	Navigate To Show Other Views
		Sub Diagrams Create Parent			Show Link
	- Edit				Visualize Related Model Element
	X	Cut		– Mo	del Transitor
		Сору	+	÷	Transit From
		Delete	•	3	Transit To
		Duplicate	Ctrl+E	7	Create Lifeline
		Selection	•	Ö	Create Model Element
		Order	+		Realize all Interfaces
		Grouping	•		Show Associations
		Styles and Formatting	•		Analysis
		Presentation Options	+	– Me	-
		Layer	•		Merge from Other Model Element(s)
		Define Design Pattern			Merge to Model Element
		Related Elements	•		Change Type
		Java Round-trip	•		
		C++ Round-trip	+		

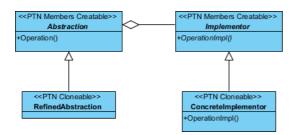
In practice, there may be multiple refined abstractions and/or concrete implementors. To represent this, stereotype the class *RefinedAbstraction* and *ConcreteImplementor* as PTN Cloneable. Right-click on *Abstraction* and select Stereotypes > Stereotypes... from the popup menu.



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

General Attributes Operations Relations Chart Relations Template Parameters Diagrams Traceability References Project Management Quality Comments Class Code Details Java Annotations Stereotypes Tagged Values Constraints All: Selected: Selected:	Second Class	Specification	x
Image: Provider Image: Provider	General Attributes Operations Relation Diagrams Traceability References Class Code Details Java Annotations All: Image: State of the state o	Ations Chart Relations Template Parameters Project Management Quality Comments Stereotypes Tagged Values Constraints Selected:	a^ ™
Reset OK Cancel Apply Help	Image: Service and Serv	>> <<	

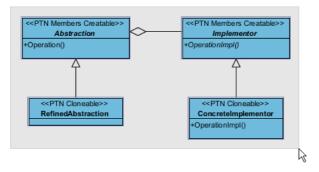
16. Repeat steps 14 and 15 on *concreteImplementor*.



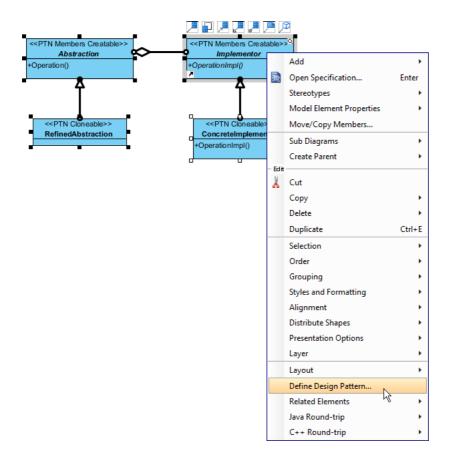
17. In practice, there may be multiple operations and/or operationImpls. To represent this, stereotype the class *Abstraction* and *Implementor* as **PTN Members Creatable**. Repeat steps 14 and 15 to stereotype *Abstraction* and *Implementor* as **PTN Members Creatable**.

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

\$	Define Design Pattern
Name:	Bridge
File name:	Bridge.pat
Location	
Ŭ	to workspace:
	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\Bridge.pat
	OK Cancel

Applying Design Pattern on Class Diagram

In this section, we are going to apply the bridge pattern to model a report generator for various report types.

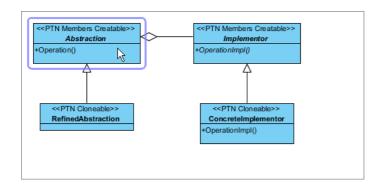
- 1. Create a new project *Diagram Editor*.
- 2. Create a class diagram *Domain Model*.
- 3. Right-click on the class diagram and select **Utilities** > **Apply Design Pattern...** from the popup menu.

	Open Specification]	
	Add Shape	•		
	Rename			
	Show Quality Checker Panel			
	Synchronize to Entity Relationship Diagram			
	Ignore Classes when Synchronizing			
i	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

>	[Design Pattern		
Patterns: Bridge	< <ptn cre<br="" members="">Abstraction +Operation() <<ptn cloneabl<br="">RefinedAbstract</ptn></ptn>	+Oper +Oper	N Members Creatable>> Implementor ationImpl() < <ptn cloneable="">> oncreteImplementor erationImpl()</ptn>	
	Diagram Element <all></all>			
	Auto Rename	Abstraction	~	
	Auto Rename	Operation	~	
	Auto Rename Auto Rename Abstraction Operation ConcreteImplemento	Operation ConcreteImplementor	~	
	Auto Rename Abstraction Geration Geration Geration GerationImpl	Operation ConcreteImplementor OperationImpl	~	
	Auto Rename	Operation ConcreteImplementor OperationImpl Implementor	~	
	Auto Rename	Operation ConcreteImplementor OperationImpl Implementor OperationImpl	~	+ + +
	Auto Rename	Operation ConcreteImplementor OperationImpl Implementor	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	+ + + + +
	Auto Rename	Operation ConcreteImplementor OperationImpl Implementor OperationImpl	~	+ + + + +

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

5. Click on *Abstraction* in the overview.



6. Rename *Abstraction* to *ReportGenerator* and operation *Operation* to *generate* at the bottom pane.

Abstraction	ReportGenerator	· +
😝 Operation	generate	~

7. Select *RefinedAbstraction* in the overview pane.

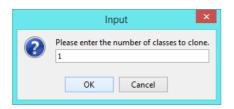
8. Rename *RefinedAbstraction* to *HTMLReportGenerator*.

			_
< <ptn creatable="" members="">></ptn>	~	< <ptn creatable="" members="">></ptn>	
Abstraction	\sim	Implementor	
+Operation()		+OperationImpl()	
<ptn cloneable="">> RefinedAbstraction</ptn>		<ptn cloneable="">> ConcreteImplementor +OperationImpl()</ptn>	
Diagram Element 📋 RefinedAbstra	ction		~
Auto Rename			
RefinedAbstraction HTMLRe	portGenerator		+

9. Besides HTML report generator, we need also a Plain Text report generator. Click on the + button at the bottom pane, beside *Abstraction*, and select **Clone...**

E RefinedAbstraction	HTMLReportGenerator	~	+	
			CI	one

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



11. Rename RefinedAbstraction2, the cloned class, to PlainTextReportGenerator.

RefinedAbstraction	HTMLReportGenerator	× +
RefinedAbstraction2	PlainTextReportGenerator	*

12. Select Implementor in the overview pane.

13. Rename Implementor to ReportGeneratorImpl, and OperationImpl to generateTOC.

< <ptn cre.<br="" members="">Abstraction +Operation()</ptn>	3>>	<pre><ptn creatable="" members="">> Implementor +OperationImpl() </ptn></pre>	
Diagram Element 📄 Impl	ementor		Ŷ
Auto Rename			
Implementor	ReportGeneratorImpl		~ +
😝 OperationImpl	generateTOC		~

14. We need 2 more operations for generating content and appendix. Click on the + button and select **New Operation...** from the popup meun.

Implementor	ReportGeneratorImpl ~	+		
😝 OperationImpl	generateTOC	1	Vew	Attribute
		New Operation		Operation

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

Operation Specification					×	
Reference Stereotype Operati General Name: Classifier: Return type:		Raised	Relations Relations Java Annotations Exceptions	Quality Constraints Templ Preconditions	Commen Traceab ate Parameter: Postcondit	sility
Type modifier Visibility: Scope: Lower: Upper: Body condition	 <unspecified></unspecified> public instance 					
Documentatio B I <u>u</u> = Record. Abstract	····		Trdered ♥ Uniq	ue	Apply	##

16. Repeat the previous steps to create another abstract operation generateAppendix.

Implementor	ReportGeneratorImpl	~ +
😂 OperationImpl	generateTOC	*
😝 generateContent	generateContent	*
😝 generateAppendix	generateAppendix	*

17. Select ConcreteImplementor in overview. Rename ConcreteImplementor to SimpleReportGeneratorImpl, and operation OperationImpl to generateTOC.

ConcreteImplementor	SimpleReportGeneratorImpl	~ +
😝 OperationImpl	generateTOC	~
😝 generateContent	generateContent	~
😝 generateAppendix	generateAppendix	~

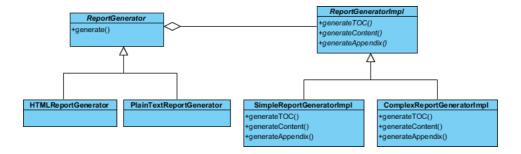
18. Similar to *RefinedAbstraction*, here we need to have another concrete implementor for generating complex report. Click on the + button and select **Clone...** from the popup menu.

ConcreteImplementor	SimpleReportGeneratorImpl 🗸 🗸	+		
😝 OperationImpl	generateTOC	(Clor	ne
😝 generateContent	generateContent	×	Π	
😝 generateAppendix	generateAppendix	~		

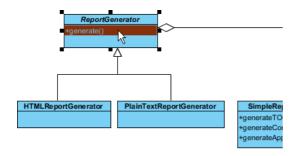
- 19. Enter 1 to be the number of classes to clone. Click **OK** to confirm.
- 20. Rename ConcreteImplmentor2, the cloned class, to ComplexReportGeneratorImpl, and operation OperationImpl to generateTOC.

ConcreteImplementor	SimpleReportGeneratorImpl	× +
😝 OperationImpl	generateTOC	~
😝 generateContent	generateContent	~
😂 generateAppendix	generateAppendix	~
ConcreteImplementor2	ComplexReportGeneratorImpl	~
😝 OperationImpl	generateTOC	~
😂 generateContent	generateContent	~
😂 generateAppendix	generateAppendix	~

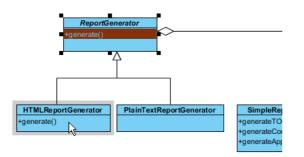
21. Click **OK** to confirm. Here is the diagram formed:



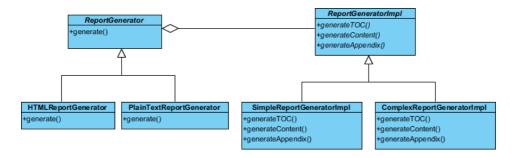
22. We want *HTMLReportGenerator* and *PlainTextReportGenerator* to implement their own way of generating repot. Select the generate operation is *ReportGenerator*.



23. Press the Ctrl key, and drag to HTMLReportGenerator. Replease the mouse button afterwards.



24. Repeat the previous steps to create the generate method in *PlainTextReportGenerator*. Here is the completed diagram:



Resources

- 1. Bridge.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/)

https://www.visual-paradigm.com/tutorials/bridgedesignpattern.jsp