Visitor Pattern Tutorial
Written Date: October 28, 2009

This tutorial is aimed to guide the definition and application of Gang of Four (GoF) visitor design pattern. By reading this tutorial, you will know how to develop a model for the visitor 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 Visitor.
3. Select Class from diagram toolbar. Click on 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 _Visitor_.

5. Right-click on _Visitor_, and select _Model Element Properties > Abstract_ to set it as abstract.

6. Right-click on the _Visitor_ class, and select _Add > Operation_ from the popup menu.

7. Name the operation _VisitConcreteElement(ConcreteElement)_.

8. Right-click on _VisitConcreteElement_, and select _Model Element Properties > Abstract_ to set it as abstract.
9. Move the mouse cursor over the **Visitor** class, and drag out **Generalization > Class** to create subclasses **ConcreteVisitor**.

10. We need to make the concrete visitors inherit operations from the visitor class. Right-click on **ConcreteVisitor** and select **Related Elements > Realize all Interfaces** from the popup menu.

11. Move the mouse cursor over the **Client** class, and drag out **Association > Class** to create an associated class **ObjectStructure**.
12. Move the mouse cursor over the ObjectStructure class, and drag out Association > Class to create an associated class Element.

![Diagram](attachment:image.png)

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

14. Right-click on the Element class, and select Add > Operation from the popup menu. Name the operation Accept(Visitor).

15. Right-click on Accept(Visitor), and select Model Element Properties > Abstract to set it as abstract. Up to now, the diagram becomes:

![Diagram](attachment:image.png)
16. Move the mouse cursor over the **Element** class, and drag out **Generalization > Class** to create subclasses **ConcreteElement**.

17. We need to make the concrete elements inherit operations from the element class. Right-click on **ConcreteElement** and select **Related Elements > Realize all Interfaces** from the popup menu.
18. In practice, there may be multiple `ConcreteVisitor` class. To represent this, stereotype the `ConcreteVisitor` class as **PTN Cloneable**. Right-click on `ConcreteVisitor` class and select **Stereotypes > Stereotypes...** from the popup menu.
19. In the **Stereotypes** tab of class specification, select **PTN Cloneable** and click > to assign it to the class. Click OK to confirm.

20. Repeat step 18 and 19 on **ConcreteElement**.
21. There may be multiple operations in Visitor, Element and ConcreteElement. To represent this, stereotype them as PTN Members Creatable. Repeat steps 18 and 19 to stereotype Visitor, Element and ConcreteElement as PTN Members Creatable. Up to now, the pattern 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.

![Define Design Pattern dialog box](image)

3. In the **Define Design Pattern** dialog box, specify the pattern name **Visitor**. 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 visitor pattern to model the visiting of elements in a room.

1. Create a new project **My Room**.

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 **Visitor** from the list of patterns.

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

6. Rename it to **RoomElements** at the bottom pane.
7. Select **Element** in overview. At the bottom pane, rename it to **RoomElement**. Rename parameter **Visitor** in **Accept()** to **RoomElementVisitor**.

8. Select **ConcreteElement** in overview. At the bottom pane, rename it to **Chair**. Rename parameter **Visitor** in **Accept()** to **RoomElementVisitor**.

9. We need 2 more concrete elements for bed and table. Keep **ConcreteElement** selected, click on the + button and select **Clone**... from the popup menu.
10. Enter 2 to be the number of classes to clone. Click **OK** to confirm.

11. At the bottom pane, rename `ConcreteElement2` and `ConcreteElement3` to `Bed` and `Table`. Rename parameter `Visitor` to `RoomElementVisitor`.

12. Select `Visitor` in overview. At the bottom pane, rename it to `RoomElementVisitor`. Rename operation `VisitConcreteElement` to `VisitChair`, and parameter `ConcreteElement` to `Chair`.
13. We need 2 more operations for visiting bed and table. Keep Visitor selected, click on the + button and select New Operation... from the popup menu.


15. Open the Parameters tab.

16. Click Add... at the bottom, and create parameter Chair in Parameter Specification dialog box. Click OK to go back to the Operation Specification dialog box.
17. In the **General** page, check **Abstract**.

18. Click **OK** to confirm editing.

19. Repeat steps 13 to 18 to create one more abstract operation **VisitTable** which have **Table** as parameter.

20. Select **ConcreteVisitor** in overview. At the bottom pane, rename it to **RoomElementPaintVisitor**. Rename operation **VisitConcreteElement** to **VisitChair**, and parameter **ConcreteElement** to **Chair**.
21. We need one more visitor for printing elements’ information. Keep ConcreteVisitor selected, click on the + button and select Clone... from the popup menu.

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

23. At the bottom pane, rename ConcreteVisitor2 to RoomElementInfoVisitor. Rename operation VisitConcreteElement to VisitChair, and parameter ConcreteElement to Chair.

24. Click OK to confirm editing and apply the pattern to diagram.
25. Tidy up the diagram. It should become:

![Diagram of Visitor Pattern]

**Resources**
1. Design Patterns.vpp
2. Visitor.pat

**Related Links**
- Full set of UML tools and UML diagrams

[Visit Visual Paradigm home page](https://www.visual-paradigm.com/)
[Visit Visual Paradigm tutorials](https://www.visual-paradigm.com/tutorials/)