Command Pattern Tutorial
Written Date: October 14, 2009

This tutorial is aimed to guide the definition and application of Gang of Four (GoF) command design pattern. By reading this tutorial, you will know how to develop a model for the command 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 Command.
3. Select Class from diagram toolbar. Click on the diagram to create a class. Name it as Invoker.
4. Move the mouse cursor over the Invoker class, and drag out Aggregation > Class to create an associated class Command.
5. Right-click on Command, and select Model Element Properties > Abstract to set it as abstract.

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

7. Name the operation Execute().

8. Right-click on Execute, and select Model Element Properties > Abstract to set it as abstract.
9. Move the mouse cursor over the Command class, and drag out Generalization > Class to create subclasses ConcreteCommand.

10. We need make the concrete commands inherit operations from the command class. Right-click on ConcreteCommand and select Related Elements > Realize all Interfaces from the popup menu.

11. Right-click on the ConcreteCommand class, and select Add > Attribute from the popup menu. Enter state as attribute name.
12. Move the mouse cursor over the `ConcreteCommand` class, and drag out `Association > Class` to create an associated class `Receiver`.

![Diagram showing association between Receiver and ConcreteCommand]

13. Right-click on the `Receiver` class, and select `Add > Operation` from the popup menu. Enter `Action` as operation name.

![Diagram showing operation on Receiver]

14. Create a `Client` class near the `Receiver` class.

![Diagram showing association between Client and Receiver]

15. Move the mouse cursor over the `Client` class, and drag out `Association > Class` to create an associated class `Receiver`.

![Diagram showing association between Client and Receiver]

16. Move the mouse cursor over the `Client` class, and drag out `Dependency > Class` to create an associated class `ConcreteCommand`. Up to now, the diagram becomes:

![Complete diagram showing all relationships]
17. In practice, there may be multiple concrete handlers. To represent this, stereotypes the class *ConcreteCommand* as **PTN Cloneable**. Right-click on *ConcreteCommand* and select **Stereotypes** > **Stereotypes...** from the popup menu.

18. In the **Stereotypes** tab of the **Class Specification** dialog box, select **PTN Cloneable** and click > to assign it to *ConcreteCommand* class. Click **OK** to confirm.
19. There may be multiple actions that the receiver can perform. To represent this, stereotype the class Receiver as PTN Members Creatable. Up to now, the diagram becomes:

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.

![Diagram of Command Pattern](image)

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

![Define Design Pattern Dialog](image)

**Applying Design Pattern on Class Diagram**

In this section, we are going to apply the command pattern in modeling a document editor.

1. Create a new project Document 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.
4. In the **Design Pattern** dialog box, select **Command** from the list of patterns.

![Diagram of Design Pattern](image)

5. Select **Invoker** in overview.

![Diagram of Invoker](image)

6. At the bottom pane, rename **Invoker** to **ToolbarButton**.

![Rename Invoker to ToolbarButton](image)
7. **Select Command** in overview. At the bottom pane, rename Command to DocumentCommand.

![Diagram showing the Command pattern with Command renamed to DocumentCommand](image)

8. **Select ConcreteCommand** in overview. At the bottom pane, rename ConcreteCommand to OpenCommand.

![Diagram showing the ConcreteCommand with renamed OpenCommand](image)

9. **We need 2 more concrete commands for closing and saving a document.** Press on the + button and select **Clone...** from the popup menu.

![Diagram showing the Clone button](image)
10. Enter 2 to be the number of classes to clone.

11. Rename `ConcreteCommand2` to `CloseCommand`, `ConcreteCommand3` to `SaveCommand`.
12. Select Receiver in overview. At the bottom pane, rename Receiver to Document, and operation Action to Load.

![Diagram of Command Pattern]

13. Create more operations for closing and saving documents. Click on the + button and select New Operation... from the popup menu.

![Operation Specification]

14. In the Operation Specification, enter Close as name. Click OK to confirm.
15. Repeat steps 13 and 14 to create operation Save.

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

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

Resources
1. Command.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/)