The Visitor Pattern

Use: The visitor pattern is used to implement plugin software architectures. Examples are plugins for browsers that allow you to add capabilities to the browser, and plugins for Photoshop that allow you to add various effects to images. Eclipse is based on a plugin architecture.

Advantages: The advantage is the ability to add features to an application without modifying and re-compiling the class library. Because the original library does not need to be changed to add new features, the plugins can be supplied by third-party developers, or even the end users of the application.

The problem: The problem that must be solved in the Visitor Pattern is how to allow a user to write an arbitrary method that has access to the underlying data structure of the application without knowing the signature of the method the user wants to write. One user might want a non-void method with no parameters that returns a value of some type, while another user might want to write a void method with several parameters.

The solution: The solution is for the application to provide a minimal complete set of operations that any other programmer can use to build the desired plugin. In addition, the application provides an abstract visitor class and a method named “accept,” which takes as a parameter an abstract class named “visitor” whose type is the abstract visitor. The programmer writes a plugin by declaring it to be a subclass of the abstract visitor class.

The signature mechanism: Because a plugin is a subclass of an abstract visitor, it is free to add any number of attributes of any type that are not present in its superclass. The key idea is that the attributes in a visitor correspond to the parameters in a non-plugin method.

Input parameters: If the original non-plugin method has an input parameter, then the plugin initializes the corresponding attribute in its constructor.

Output parameters: If the original non-plugin method has an output parameter, that is, a parameter called by reference, then the corresponding attribute is a reference variable. The plugin constructor initializes it to the address of the object to be changed.

Void methods: If the original non-plugin method is a void method, then the plugin only needs to implement the methods of its superclass, the abstract visitor class.

Non-void methods: If the original non-plugin method returns a non-void type, then the plugin implements an additional method named “result” that returns the same type.