Visitor:

**Other name (if any)**

Sometimes, it is also known as a **behavioral pattern**.

What it does

**It decouples the operations from an object structure. It takes all the methods of a class to another class. It has two parts visitor and visitable/element. Visitor holds functions. Visitable/Element is the main object without functions. It supports double dispatch. In single dispatch, dispatch of function depends on one object (the caller) but in double dispatch, dispatch of function depends on two objects (the caller and the argument). There are the two most important methods that are used in visitor patterns. The accept() method accepts a visitor. It is provided by the Visitable class. The visit() method is called every time when we visit an element. It is implemented by the Visitor class.**

**Where to use**

1. It is used when we have to perform an operation on a group of similar kind of Objects. With the help of visitor pattern, we can move the operational logic from the objects to another class.

**Steps**

1. Visitor Interface: It may be an interface or an abstract class. The interface contains the visit operation for all types of visitable classes.  
2. **Concrete Visitor Class:** All the visit methods that are declared in abstract visitor must be implemented in the Concrete Visitor class. Each visitor is responsible for different operations.  
3. Visitable/Element Interface: It has only one method which is accept. It will take any visitor class as parameter.  
4. **Concrete Visitable/Element Class:** It is a class that implements the Visitable interface or class in which accept operation is defined. The object of the visitor class is passed through the object of using the accept operation.  
5. Client Code: It creates visitable/element object using visitable/element interface and concrete visitable/element class. It also creates visitor object using visitor interface and concrete visitor class. Then we will pass the visitor object to the accept function of the concrete visitable/element object.

**Special cases (if any)**

placeholder

**Advantages**

1. If the logic of operation changes, then we need to make change only in the visitor implementation rather than doing it in all the item classes.  
2. Adding a new item to the system is easy, it will require change only in visitor interface and implementation and existing item classes will not be affected.

**Disadvantages**

1. It is difficult to extend if there are several implementations of the visitor interface.

**Code**

Coding Concept

**Difference with similar pattern**

Strategy Design Pattern:

**Diagram**

Coding Concept + Tutorialspoint