**Visitor Design Pattern**

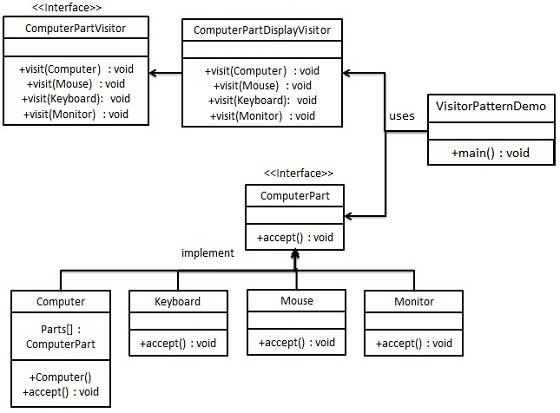
**If we want to perform a task on each and every object at that time we use this pattern.**

In Visitor pattern, we use a visitor class which changes the executing algorithm of an element class. By this way, execution algorithm of element can vary as and when visitor varies. This pattern comes under behavior pattern category. As per the pattern, element object has to accept the visitor object so that visitor object handles the operation on the element object.

**Implementation**

We are going to create a ComputerPart interface defining accept opearation.Keyboard, Mouse, Monitor and Computer are concrete classes implementing ComputerPart interface. We will define another interface ComputerPartVisitor which will define a visitor class operations. Computer uses concrete visitor to do corresponding action.

*VisitorPatternMain*, our demo class, will use *Computer* and *ComputerPartVisitor* classes to demonstrate use of visitor pattern.



**Hot Points**

* The visitor pattern is a great way to provide a flexible design for adding new visitors to extend existing functionality without changing existing code.
* The Visitor pattern comes with a drawback: If a new visitable object is added to the framework structure all the implemented visitors need to be modified. The separation of visitors and visitable is only in one sense: visitors depend of visitable objects while visitable are not dependent of visitors.
* Part of the dependency problems can be solved by using reflection with a performance cost.