**Visitor Design Pattern**

Visitor pattern 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.

For example, think of a Shopping cart where we can add different type of items (Elements). When we click on checkout button, it calculates the total amount to be paid. Now we can have the calculation logic in item classes or we can move out this logic to another class using visitor pattern. Let’s implement this in our example of visitor pattern.

### Visitor Design Pattern Java Example

To implement visitor pattern, first of all we will create different type of items (Elements) to be used in shopping cart.

ItemElement.java

Notice that accept method takes Visitor argument. We can have some other methods also specific for items but for simplicity I am not going into that much detail and focusing on visitor pattern only.

Let’s create some concrete classes for different types of items.

Book.java

Fruit.java

Notice the implementation of accept() method in concrete classes, its calling visit() method of Visitor and passing itself as argument.

We have visit() method for different type of items in Visitor interface that will be implemented by concrete visitor class.

ShoppingCartVisitor.java

Now we will implement visitor interface and every item will have it’s own logic to calculate the cost.

ShoppingCartVisitorImpl.java

Lets see how we can use visitor pattern example in client applications.

ShoppingCartClient.java

When we run above visitor pattern client program, we get following output.

Book ISBN::1234 cost =20

Book ISBN::5678 cost =95

Banana cost = 20

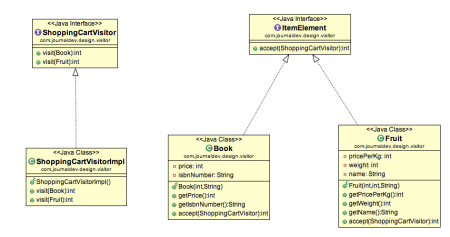
Apple cost = 25

Total Cost = 160

Notice that implementation if accept() method in all the items are same but it can be different, for example there can be logic to check if item is free then don’t call the visit() method at all.

### Visitor Design Pattern Class Diagram

Class diagram for our visitor design pattern implementation is:

[](https://cdn.journaldev.com/wp-content/uploads/2013/07/visitor-pattern.png)

### Visitor Pattern Benefits

The benefit of this pattern is that 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.

Another benefit is that 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.

### Visitor Pattern Limitations

The drawback of visitor pattern is that we should know the return type of visit() methods at the time of designing otherwise we will have to change the interface and all of its implementations. Another drawback is that if there are too many implementations of visitor interface, it makes it hard to extend.