# Visitor – Pattern B

### Authors: Georgi Chishirkov – 2530090

### Ilia Nikushev – 2551721

Site of referrence : <http://www.blackwasp.co.uk/visitor.aspx>

Description of the pattern:

* Allows you to add methods to classes of different types without much altering to those classes
* You can make completely different methods depending on the class used
* Allows you to define external classes that can extend other classes without majorly editing them

Description of the implementation:

Our implementation of the pattern handles a restaurant ordering app where the cost of certain items with or without their extras are added up together. We have 2 basic interfaces:

Lets start with the first one, <<IRestaurantItem>> defines basic methods for each of the concrete items you can order. In this case they are fries, steak and tea. Each of the concrete items has its own added behavior. For fries we have ketchup, for steak we have a sauce and for the tea you can choose to serve in a cup or jug.

The second interface is <<IRestaurantVisitor>>, it defines the Visit method for each of the concrete items of the previous interface. It also has its own 2 concrete classes. RestaurantVisitor is a normal customer of the restaurant with the added behavior of a first time coupon which gives him a discount if this is his first visit. SpecialRestaurantVisitor could be a any person that has a special discount. It also implements the interface and uses a discount but for this one you can specify a different amount (between 1-5 for each item he gets).

PRO’s, CON’s and similarity to other patterns : The Visitor pattern is like a more powerful Command pattern because the visitor may initiate whatever is appropriate for the kind of object it encounters.

+ Adding new operations easily

+ Single visitor object is used to visit all elements of the data structure

- If a new visitable object is added to the framework structure all the implemented visitors need to be modified