Use the ***Visitor Pattern*** when you want to add capabilities to a composite of objects and encapsulation is not important.

***Scenario – Objectville diner needs more nutritional information***

// new methods

getHealthRating()

getCalories()

getProtein()

getCarbs()

// new methods

getHealthRating()

getCalories()

getProtein()

getCarbs()

***Visitor*** works hand in hand with a **Traverser**. The Traverser knows how to navigate to all of the objects in a Composite. The Traverser guides the Visitor through the Composite so that the Visitor can collect state as it goes. Once state has been gathered, the Client can have the Visitor perform various operations on the state. When new functionality is required, only the Visitor must be enhanced.

getHealthRating() getState()

getCalories() getState()

getProtein() getState()

getCarbs() getState()

***Benefits:***

* Allows you to add operations to a Composite structure without changing the structure itself.
* Adding new operations is relatively easy.
* The code for operations performed by the Visitor is centralized.

***Uses and Drawbacks:***

* The Composite classes’ encapsulation is broken when the Visitor is used.
* Because the traversal function is involved, changes to the Composite structure are more difficult.