Visitor

"'T is some visitor," I muttered, "tapping at my chamber door;
Only this and nothing more."

-- Edgar Allen Poe, The Raven

Problem: *You need to add a new method to a hierarchy of classes, but the act of adding it will be painful or damaging to the design.*

This is a common problem. For example, suppose you have a hierarchy of `Modem` objects. The base class has the generic methods common to all modems. The derivatives represent the drivers for many different modem manufacturers and types. Suppose also that you have a requirement to add a new method, named `configureForUnix`, to the hierarchy. This method will configure the modem to work with the UNIX operating system. It will do something different in each modem derivative, because each different modem has its own particular idiosyncrasies for setting its configuration, and dealing with UNIX.

Unfortunately adding `configureForUnix` begs a terrible set of questions. What about Windows, what about MacOs, what about Linux? Must we really add a new method to the `Modem` hierarchy for every new operating system that we use? Clearly this is ugly. We'll never be able to close the `Modem` interface. Every time a new operating system comes along we'll have to change that interface and redeploy all the modem software.

The VISITOR family of design patterns.

The Visitor family allows new methods to be added to existing hierarchies without modifying the hierarchies.

The patterns in this family are:

- VISITOR

 Copyright (C) 2002 by Robert C. Martin. All Rights Reserved.