

Study questions for Lecture 14 - Answers

1. Questions on pages 179, 183, 254, 268 in the Patterns book.

See book for answers

Deliverables

1. Consider ways in which these patterns might appear in your project. If it sounds good, use them.

Consider: Adaptor, Singleton, Friends, Proxy, Iterator.

I will describe them wrt to DS.

a) Adaptor.

Suppose that the original DL subsystem interface contained a method `logon(name)`, But the GUI was designed to use a name field "(first name, second name)" which it expected to send in a `logOn` message to the DL. We could create an adaptor ADL for the DL. Instances of the constructor for the DL would be changed to instances of the constructor for ADL. The calls from the GUI would not have to be changed, and the code in the original DL could remain the same also. The methods in ADL would simply call the corresponding methods in the DL, except for the `logOn` method which would concatenate the first and second names to create a single name which it passes to the corresponding `logOn` method in the wrapped DL.

b) Singleton

Suppose that statistics are kept, during any session, on how many dates are asked for and returned. We might want to store this information in our file when we log out. We could use a singleton global for this (with its static methods and variables), to make sure that not more than one gets created, so that all information is properly compiled and merged.

c) Friends

Suppose that the DL has returned a member record for a suggested date. Suppose that some of the properties of the member record are contained in an associated object called `phys-props`, and the GUI wishes to display one of these, say, the politics. If the code is written so that it calls a method of `memberRecord` to return a `PhysProps` object, from which it then asks for the politics information, then the GUI has knowledge of the implementation of `memberRecord`. Instead, `memberRecord` should be written so that it has a method to return the politics information directly. In this way, the GUI is only asking "friends" for information, not "friends of friends".

d) Proxy

When we build our version 1, we could use a proxy which "fakes" a DB, for the purposes of implementing a limited version 1. Later we could simply change its

inner definition when the real DB is defined. For example, it might be changed to a remote proxy if the DB is stored in an another address space

e) Iterator

When we implement the vector for the in-memory copy of the DB, we need to iterate through its members. We could use an iterator for this. In fact, we could set things up so that different storage mechanisms could be used, provided they all support the same iterator helper class.