Design Patterns

|  |  |  |
| --- | --- | --- |
| Date | Description | Author |
| 04-12-2012 | Initial version. | Kewin |
|  |  |  |
|  |  |  |
|  |  |  |

# Composite Pattern

We have identified in our class diagram that we want the underlying system of how we organize documents into folders and other folders.

We will do this by having both the document and the folder class inherit from the interface “IFileSystemComponent”.

A folder will then contain a list of IFileSystemComponent’s which then obviously can be both documents and other folders.

# Information Expert

Our Storage class is an information expert, in that it holds all logic on how to read from and write to files in the file system, as well as holding all information on files currently in the file system.

# Controller & Indirection

Our controller does in fact serve 2 purposes, it both acts as what its name as, a controller, and also an indirection between the external user and vital components of our system.

The indirection responsibility of the controller is that it makes sure that a user does not have any direct contact with neither our storage nor our server. These are both vital components that we do not want the user to meddle with in other ways than we have planned.

As a controller our controller takes the users input and then calls relevant functions in other modules like our information expert, the Storage.