Command & Mediator

10/21/2009

Opening Discussion

- Do you have any questions about the reading for today?

Command

- This pattern encapsulates a request for some action to be performed in an object.
- This allows you to support a log of recent operations or implement things like an undo function.
- It also allows you to write code that sends requests to objects without knowing what the request is, or the details of the object it is being sent to.
- In the most general sense, the abstract Command has one method: execute().

Example

- GoF uses menus as an example of command. The writer of the MenuItem class doesn't know what needs to happen for a given item. Instead, it allows a command to be attached and when the item is selected, that command is executed.
- The Java event model (since 1.1) is based on the Command pattern. In this case the listeners are the Command objects. They specify what should happen for each type of event that Listener waits for.

Benefits and Drawbacks

- These are basically an OO way of doing callback functions (functors). As they are objects they are typically more flexible.
- Can be used for undo/redo, logging, and transactions.
- You can assemble simple commands into larger ones.
- Easy to vary the commands as you don't have to change the classes that send the commands, just set them up with a different type of command.

Mediator

- This pattern introduces an object that encapsulates the interaction between other objects.
- This decouples the other objects and reduces their dependencies on one another. This allows you to change the way that class A alters class B without changing either A or B. As a result, A and B will both be more reusable.
- An AbstractMediator defines the types of messages that can be passed around.

Example

- The GoF book uses the example of a dialog box where altering one selection changes another selection. For example, picking one option in a drop box might make some other options unusable, might change the contents of some other drop box.
- When a change is made in one element of the dialog box, a message is sent to the mediator which can then decide what to do with other elements of the dialog box.

Benefits and Drawbacks

- It limits subclassing so you don't create many different subclasses that interact differently, instead you create new mediators.
- It reduces the number of connections between classes so you don't have many-to-many relations and instead have one-to-many relations.
- The mediator itself can get very complex and hard to maintain, but it makes the colleague classes simpler.

Progress Presentations

- Who wants to give their progress reports today?