Behavioral Patterns

- ~~Chain of Responsibility~~ (requests through a chain of candidates)
- ✓ Command (encapsulates a request)
- ~~Interpreter~~ (grammar as a class hierarchy)
- ✓ Iterator (abstracts traversal and access)
- Mediator ← (indirection for loose coupling)
- ~~Memento~~ (externalize and re-instantiate object state)
- Observer later (defines and maintains dependencies)
- ~~State~~ (change behaviour according to changed state)
- ✓ Strategy (encapsulates an algorithm in an object)
- ~~Template Method~~ (step-by-step algorithm w/ inheritance)
- ✓ Visitor (encapsulated distributed behaviour)

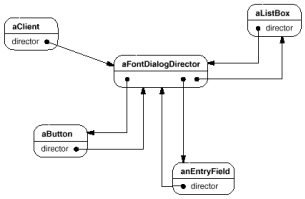
Mediator

- Defines an object that encapsulates how a set of objects interact.
  - promotes loose coupling by keeping objects from referring to each other explicitly
  - lets you vary their interaction independently

Motivation

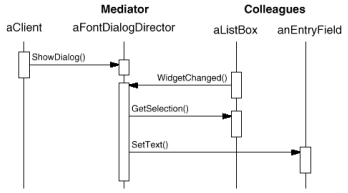
- A collection of widgets that interact with one another.
  - e.g., certain familes may not have certain weights
    - disable 'demibold' choice

Motivation



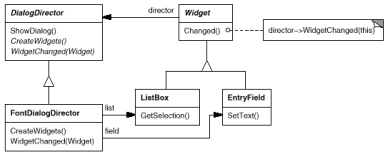
- Create a mediator to control and coordinate the interactions of a group of objects.

Motivation



- *e.g.,*
  - list box selection moving to entry field
  - entryField now calls WidgetChanged() and enables/disables
  - entry field does not need to know about list box and *vice-versa*

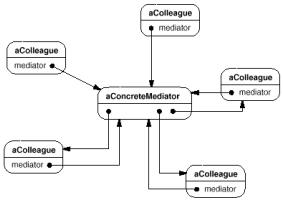
Motivation



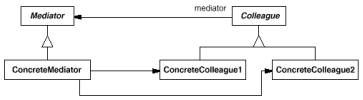
Applicability

- A set of objects communicate in a well-defined but complex manner
- reusing an object is difficult because it refers to and communicates with many other objects
- a behavior that's distributed between several classes should be customizable without a lot of subclassing

Structure



Structure



- Mediator
  - defines an interface for communicating with Colleague objects
- ConcreteMediator
  - knows and maintains its colleagues
  - implements cooperative behavior by coordinating Colleagues
- Colleague classes
  - each Colleague class knows its Mediator object
  - each colleague communicates with its mediator whenever it would have otherwise communicated with another colleague

Consequences

- limits subclassing
  - localizes behaviour that otherwise would need to be modified by subclassing the colleagues
- decouples colleagues
  - can vary and reuse colleague and mediator classes independently
- simplifies object protocols
  - replaces many-to-many interactions with one-to-many
  - one-to-many are easier to deal with
- abstracts how objects cooperate
  - can focus on object interaction apart from an object's individual behaviour
- centralizes control
  - mediator can become a monster