More GRASP Patterns (from Larman, Ch. 34)

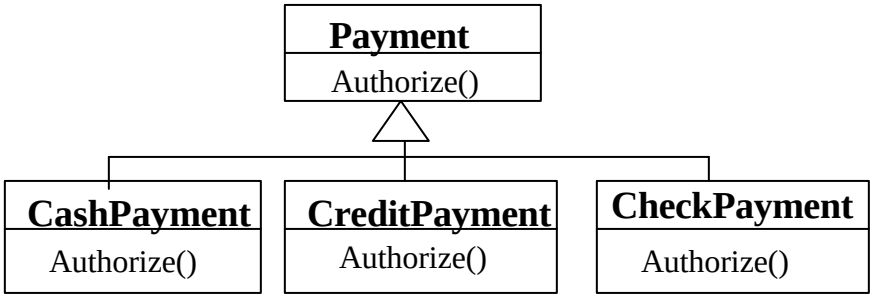
- Note that these "patterns" are really just very basic OO design principles
  - Polymorphism
  - Pure Fabrication
  - Indirection
  - Don't Talk to Strangers

GRASP Patterns--Polymorphism

- Problem: How to handle alternative behavior based upon type.
- E.g. Who is responsible for authorizing different types of payments.

Polymorphism--Continued

- Solution: use subclassing. Make subclass objects responsible for their own behavior and let proper behavior be invoked automartically by subclass type.



GRASP Pattern--Pure Fabrication

- Problem: Sometimes, during design, responsibilities need to be assigned that are not naturally attributable to any of the conceptual classes.
- Solution: Create an artificial class that does not represents anything in the problem domain.
- Note: Most of the GoF patterns involve fabricating new classes--e.g. observer, adapter, abstract factory, etc.

GRASP Pattern--Indirection

- Problem: To reduce direct couplings with objects which are subject to change.
- Solution: Use an intermediate object to mediate between other objects .
- The GoF Proxy, Bridge, and Mediator patterns utilize indirection.

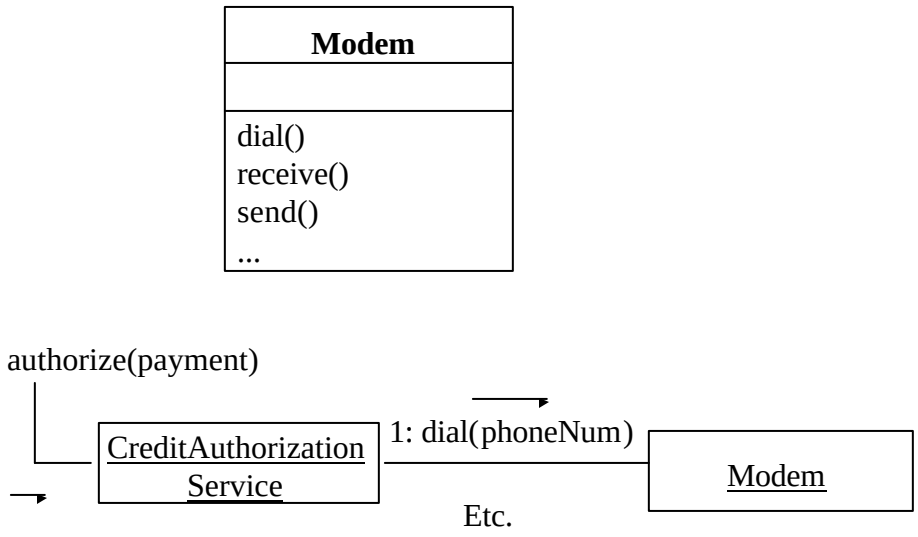
Indirection--A Simple Example

Consider a *CreditAuthorizationService* class that needs to use a Modem

Bad approach: Put low-level calls to the Modem API directly in the methods of the *CreditAuthorizationClass*

Better approach: Add an intermediate *Modem* class that insulates *CreditAuthorizationClass* from the Modem API.

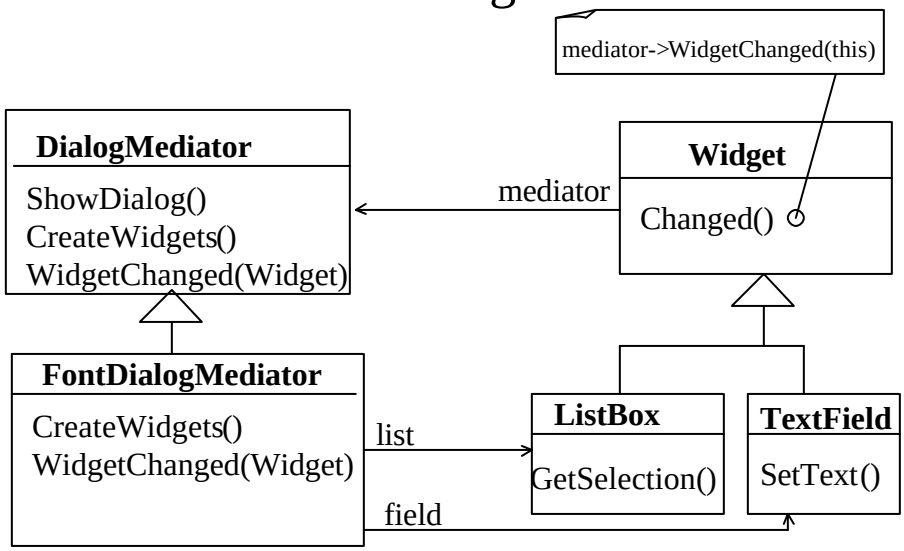
Indirection Example--Continued



Indirection--A More Complex Example

- Consider the implementation of a dialog box for a GUI.
- The dialog box uses a window to present a series of widgets, such as buttons, text fields, etc.
- There are many dependencies among widgets--e.g. a button may be disabled when an associated text field is empty.
- Different dialog boxes will have different dependencies.
- Hence, widget classes will have to be customized for each dialog--e.g. by subclassing

Indirection--Using a Mediator



Note: This is a GoF pattern: Mediator Pattern

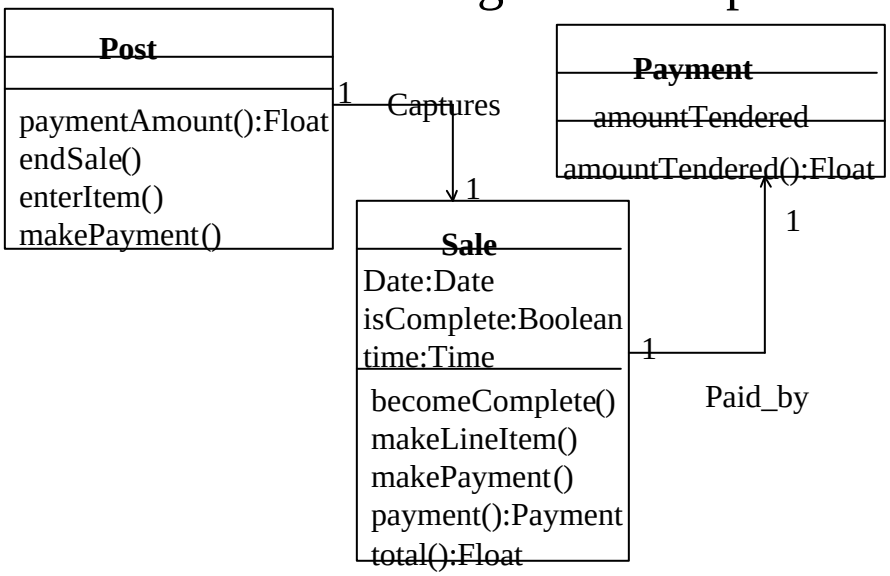
GRASP Patterns--Don't Talk to Strangers

- Problem: To avoid high coupling. Specifically, to limit the degree of knowledge that client objects need to possess regarding the internal structure and connections of server objects.

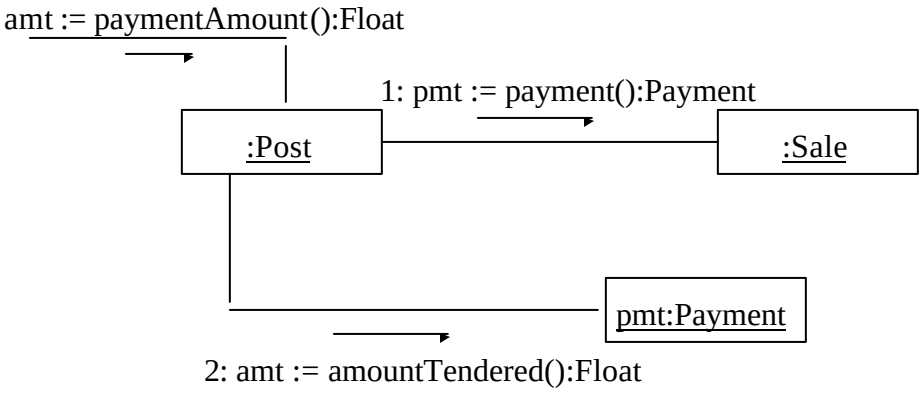
Don’t Talk to Strangers (Law of Demeter)

- Within a method, other methods should be invoked only upon the following objects:
  - The *this* object (*self*)
  - A parameter of the method
  - An attribute of *self*
  - An element of a collection that is an attribute of *self*
  - A local object of the method.
- Intent is avoid coupling a client to indirect objects or knowledge of the internal structure of direct objects

Don’t talk to Strangers--Example

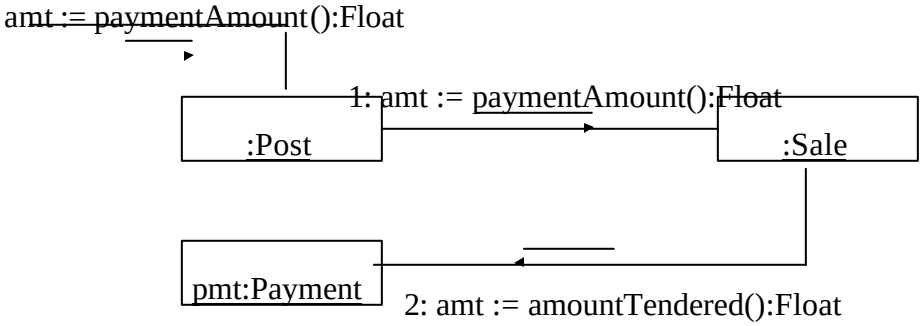


Don’t Talk to Strangers Example--Continued



Note: This is similar to the notion of “Stamp Coupling” in our earlier discussion of coupling and cohesion.

Don’t Talk to Strangers Example--Better Approach



Note: In this case :Post delegates the paymentAmount() method to :Sale. A paymentAmount() method must be added to the Sale class.