

More GRASP Patterns

Polymorphism
Pure Fabrication
Indirection
Don't Talk to Strangers



Polymorphism

- **Problem**
 - How to handle alternatives based on type.
 - How to create pluggable software components.
- **Solution**
 - When related behaviours vary by type, assign responsibility using polymorphic operations to the types for which the behaviour varies.





Definition of Polymorphism

- Giving the same name to services in different objects when the services are similar or related.
- *Corollary*
 - Do not test for the type of an object and use conditional logic to perform varying alternatives based on type.



Polymorphism

- Most important basic strategic pattern in object-oriented design.
- Easily extended to handle new variations.
- *AKA*
 - Do it myself
 - Don't ask, "what kind?"

Polymorphism Example



- **Problem**

- Who should be responsible for authorizing different kinds of payment in the point-of-sale example in the text?
- The behaviour of authorizing varies by the kind of payment:
 - Cash
 - Credit
 - Check

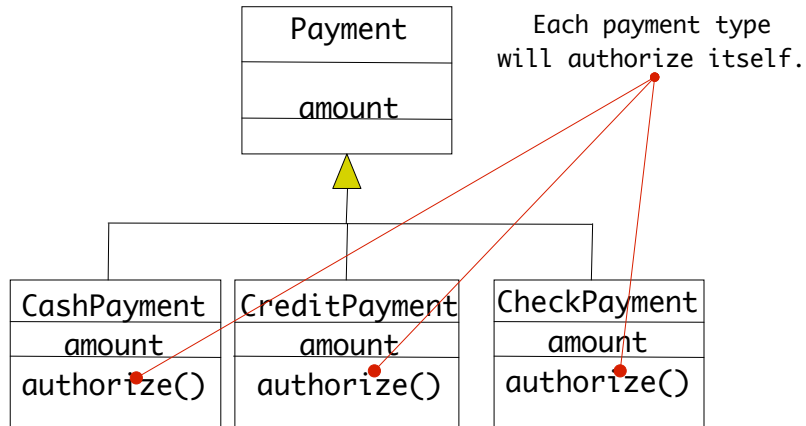
Polymorphism Example



- **Solution**

- Assign the responsibility for authorizing to each payment type, implemented with a polymorphic authorize operation.
- The implementation of each authorize operation will be different.

Polymorphism Example



Polymorphism Example



- Adding a new type of payment, e.g. debit card.
 - The debit payment class will have its own polymorphic authorize operation.
 - Minor impact on existing design.

Pure Fabrication



Pure Fabrication



- **Problem**

- There are situations in which assigning responsibilities only to domain classes lead to poor cohesion or coupling or low reuse potential.

- **Solution**

- Assign a highly cohesive set of responsibilities to an artificial class - a fabrication.

Pure Fabrication



- A class should be designed with high potential of reuse - responsibilities are small and cohesive (fine-grained).
- This will create a function-centric object.
- Part of the high-level object-oriented service layer in an architecture.

Pure Fabrication Example



- ***Sale*** instances must be saved in a relational database.
 - *Expert* might assign this responsibility to the ***Sale*** class itself.
- But....Let's look at the characteristics of this addition to our design.



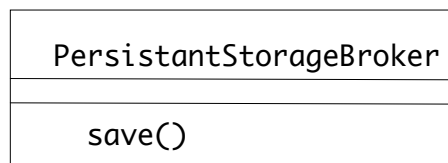
Pure Fabrication Example

- Saving to a database requires
 - A large number of supporting database operations.
 - Not related to other *Sale* operations = *low cohesion*.
 - Coupling to the database interface.
 - Coupling to another system's interface, not even another domain object!
- Saving objects in a RDB is a general task – think about the issue of reuse!



Pure Fabrication Example

- **Solution**
 - Create a new class that is solely responsible for saving objects in a persistent store, such as a RDB.
 - This class is a *Pure Fabrication*.



Pure Fabrication Example



- This new class contributes to good design.
 - *Sale* maintains its high cohesion and low coupling.
 - *PersistentStorageBroker* class is cohesive.
 - *PersistentStorageBroker* class is generic and reusable.

Pure Fabrication



- **Benefits**
 - High cohesion
 - High reuse potential
- **Potential Problems**
 - Not object-centric: may lead to a process-oriented design implemented in an object-oriented language.

Indirection



Indirection



- **Problem**
 - How can you de-couple objects?
- **Solution**
 - Assign responsibility to an intermediate object to mediate between other components so that they are not directly coupled.



Indirection Example

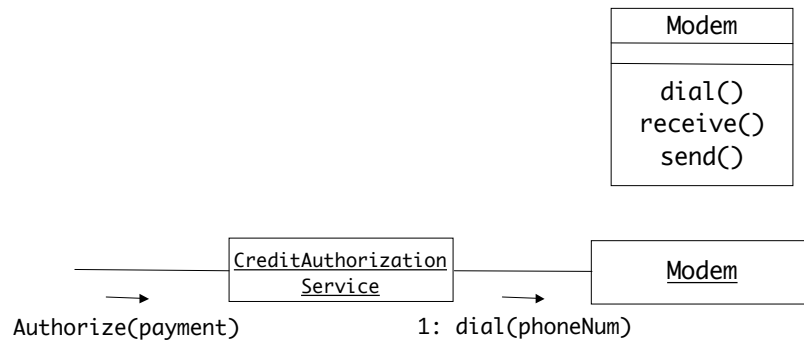
- The *POST* app needs to use a modem to transmit a credit payment request.
- The OS provides a low-level call for doing this.
- **Problem**
 - What class in the *POST* app talks to the modem??



Indirection Example

- **Solution**
 - Add a class that is responsible for talking to the modem.
 - Because of the nature of the task, this class will be highly coupled to the OS; if the class needs to be ported to another OS then it will require modification.
 - An intermediate Modem class between this new class and the OS API will function as a device proxy.

Indirection Example



Indirection



- **Examples**
 - Device proxy and publish-subscribe or observer
- **Benefits**
 - Low coupling

Don't Talk to Strangers



Don't Talk to Strangers



- **Problem**

- If a client object has to use a service or obtain information from an indirect object, how can it do so without being coupled to knowledge of the internal structure of its direct server or indirect objects?

Don't Talk to Strangers



- **Solution**

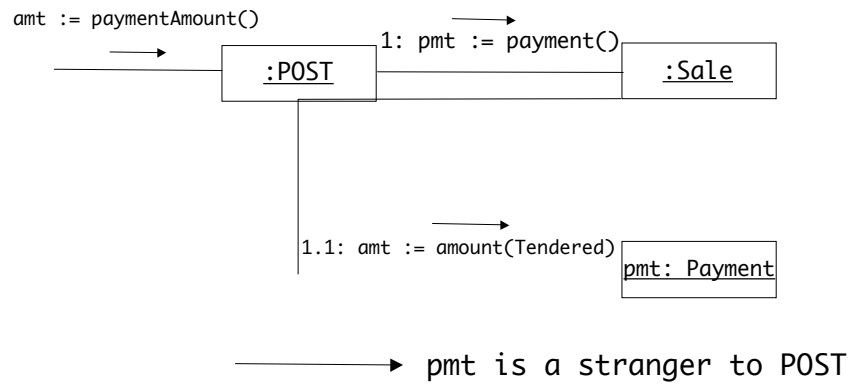
- Assign the responsibility to a client's direct object to collaborate with an indirect object so that the client does not need to know about the indirect object.

Don't Talk to Strangers

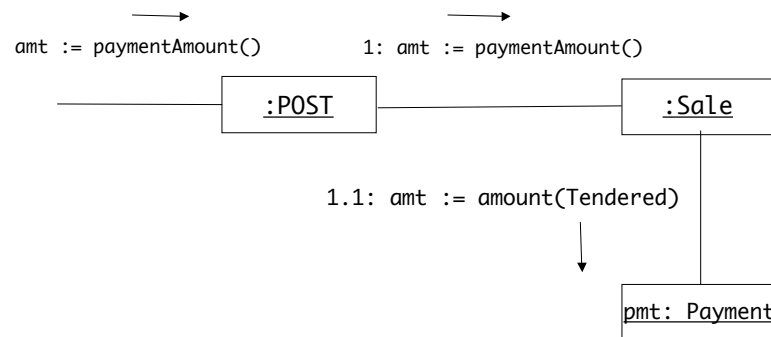


- Places constraints on what objects you should send messages to within a method.
- General solution to support this principle is known as promoting the interface.

Don't Talk to Strangers Example



Don't Talk to Strangers Example





Don't Talk to Strangers

- Also known as the Law of Demeter.
- Breaking the law!
 - *Pure fabrication*: considered acceptable to get visibility to an object X via a broker and then send messages to X directly.
- **Benefits**
 - Low coupling