Patterns

GRASP:

- Creator
- Information Expert
- Low Coupling
- High Cohesion
- Controller
- Polymorphism
- Indirection
- Pure Fabrication
- Protected Variations

GoF:

- Adapter
- Factory
- Composite
- Facade
- Observer
- Decorator
- Strategy
- Singleton

Creator

Assign B the responsibility of creating A if:

- B contains, composes, or aggregates A
- B records A
- B has the data to init. A.

Contraindications:

- Complex creation logic
- Factory more appropriate.

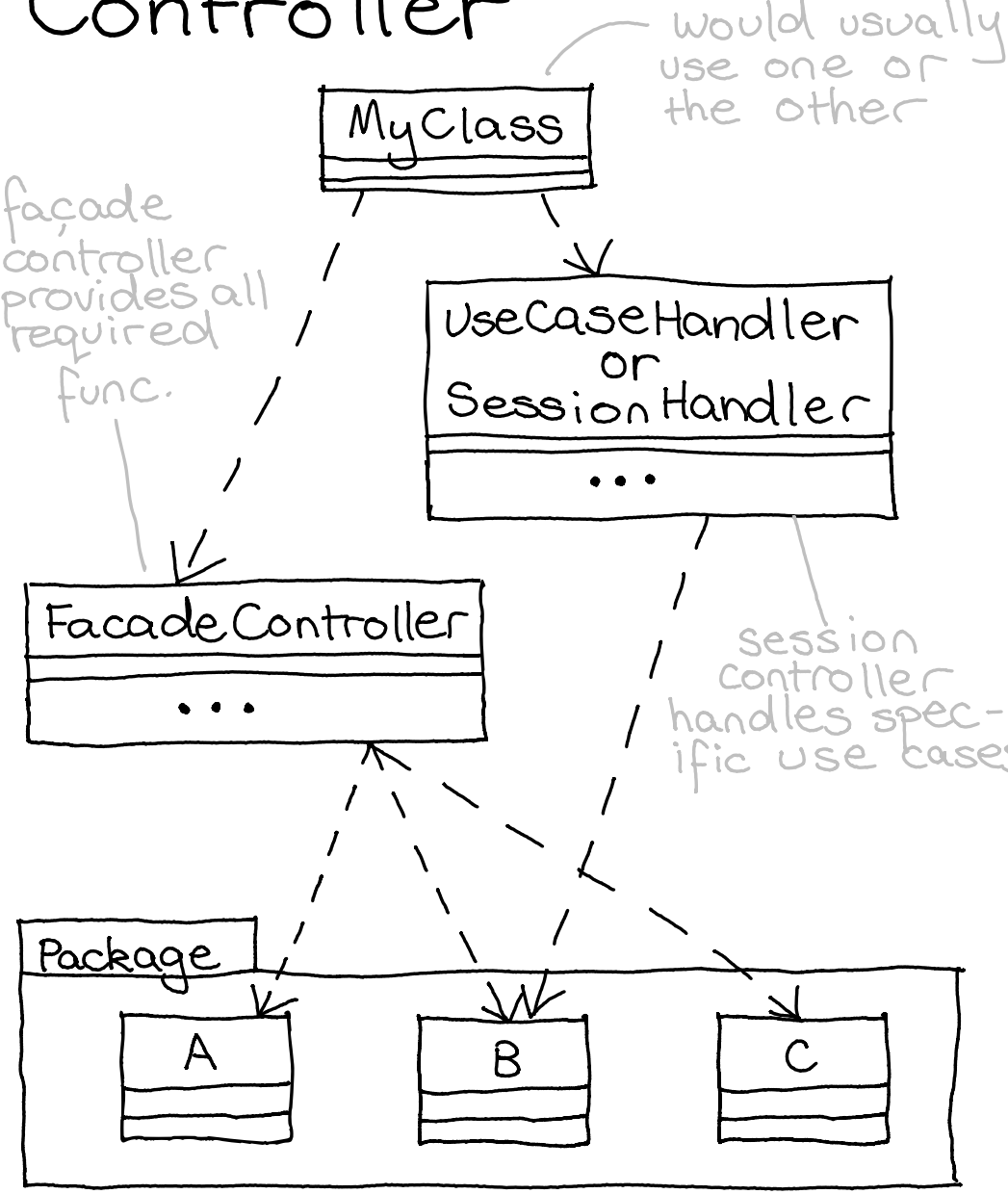
Information Expert

- Assign a responsibility to obj. X if X has the information to fulfil it.

Contraindications:

- Assigning to X causes high coupling.
- Should be done w/ helper / controller.

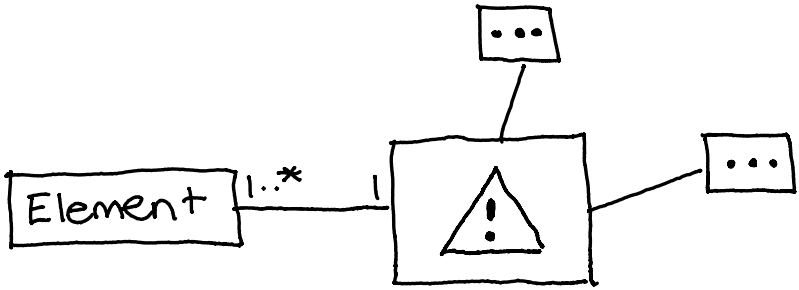
Controller



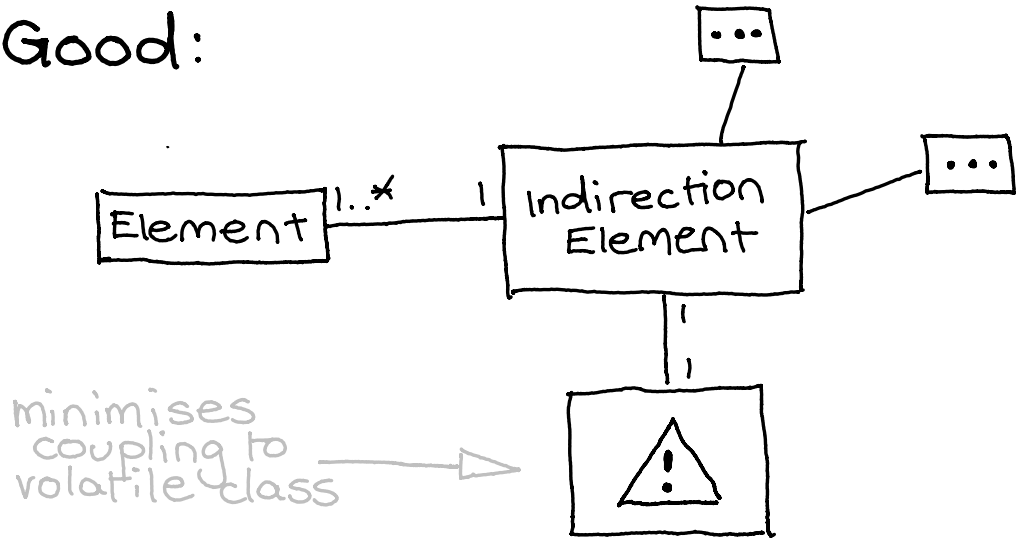
- Façade controller might be used to take inputs from UI layer.

Indirection

Bad:



Good:



- ⚠ is a volatile class.

Protected Variation

The use of objects with a stable interface when working with variable or evolving systems to avoid cascading undesirable effects.

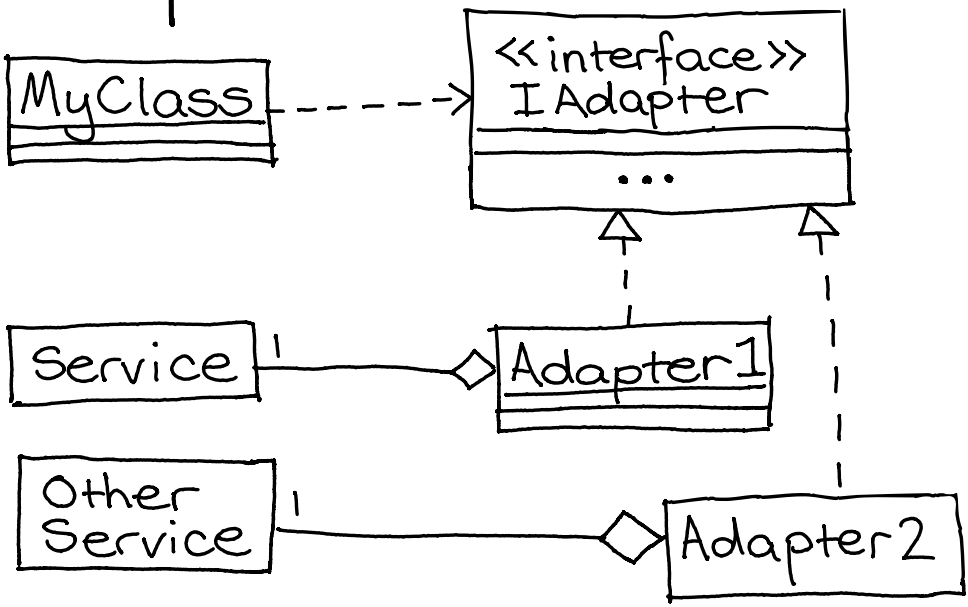
- Indirection protects from var. by offering the interface of the indirection obj.
- Polymorphism protects from variations in obj. behaviour.

Pure Fabrication

Creation of an artificial or convenience class which doesn't exist in the problem domain.

- Should be assigned a highly cohesive set of responsibilities.

Adapter

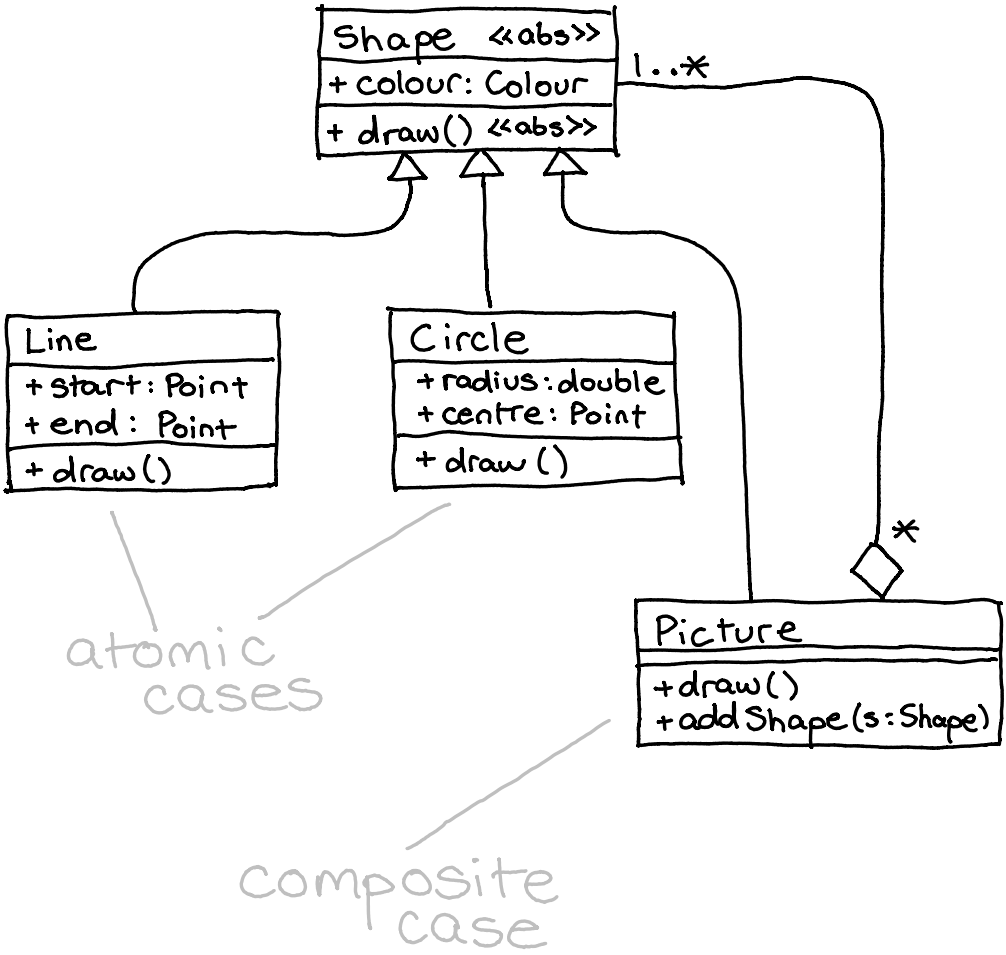


- Adapter can be used instead of the external class.

Factory

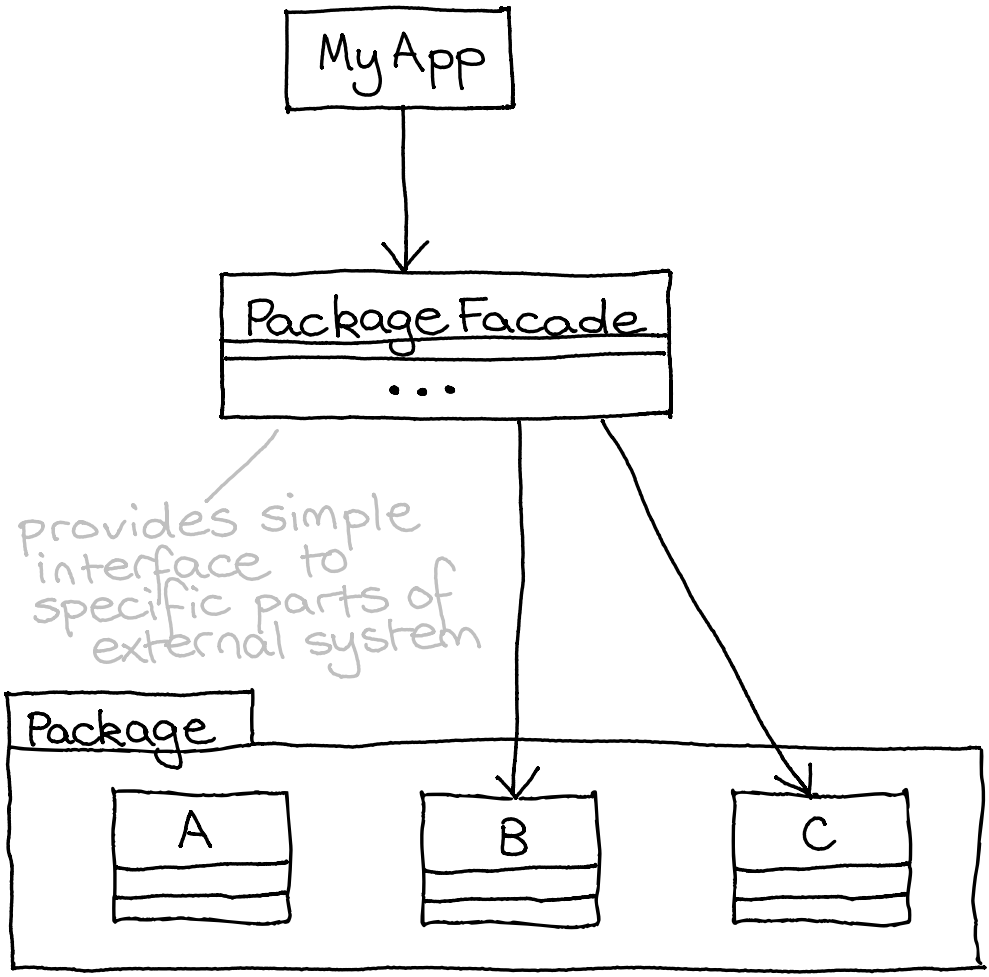
- Implements complex object creation logic.
- Can be singleton
- Can have methods to create different obj. types
- Returned objects should be cast to Interface type shared between all factory products.

Composite



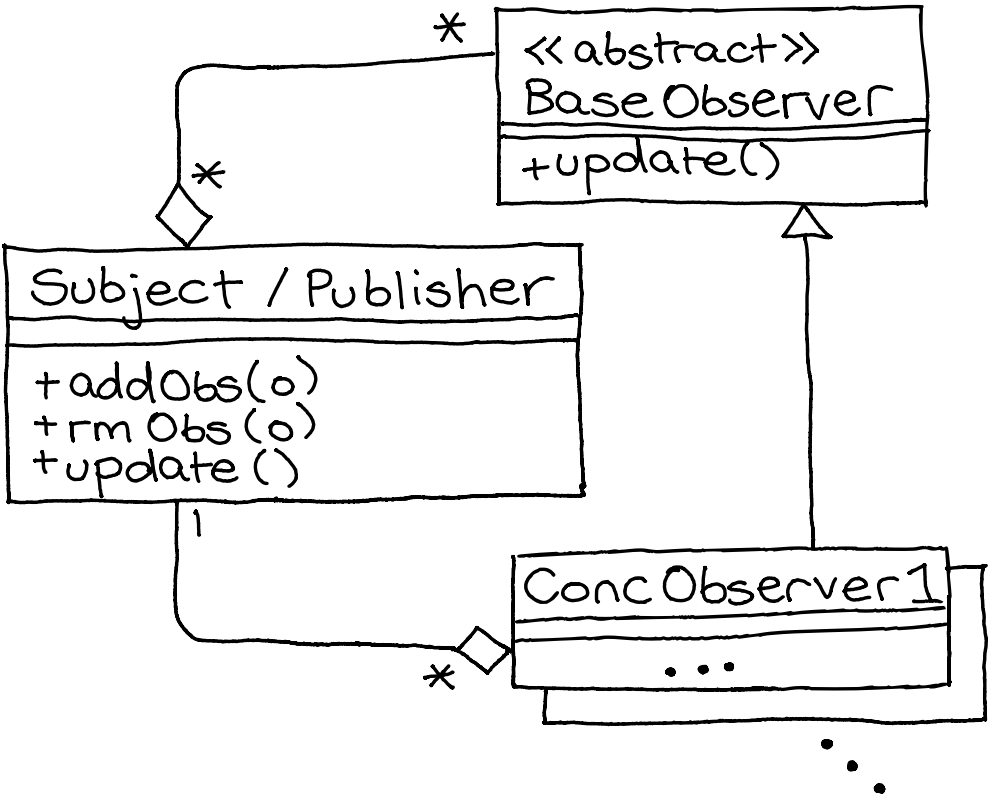
- Composite case aggregates atomic cases
  > its methods call the methods of the contained objects.

Façade



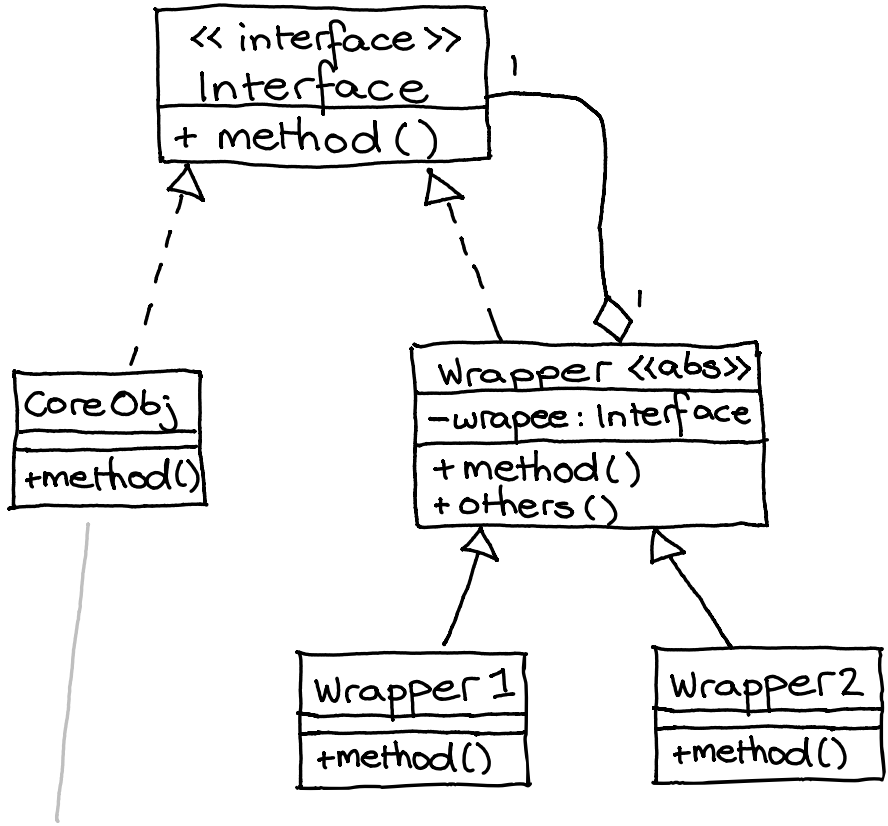
- Different to adapter which would instead provide an interface to components w/ similar purposes
- Different to controller which is designed to provide a unified interface to handle user input.

Observer



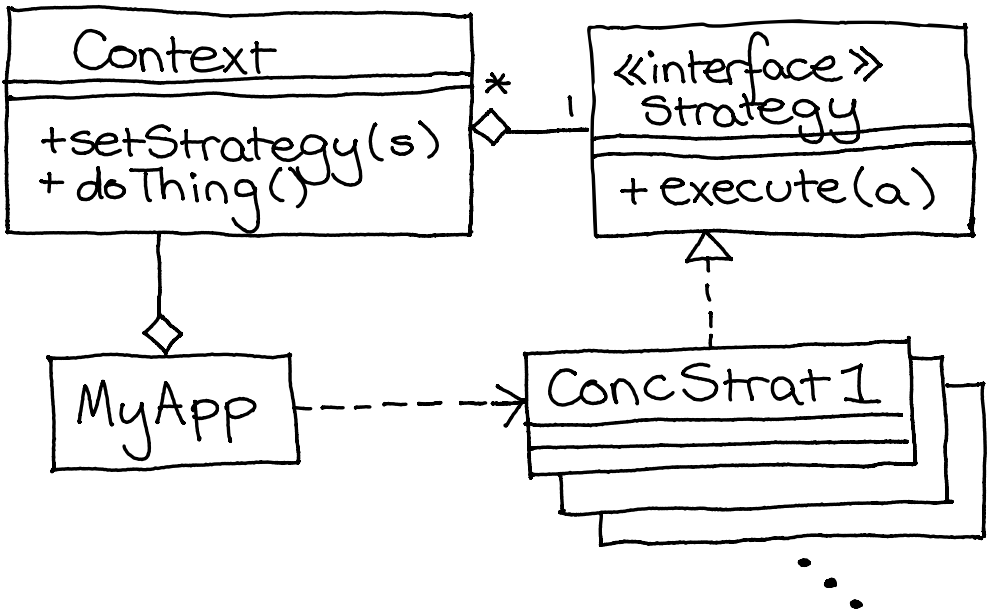
- There can be an abstract publisher if multiple are required.
- Publisher = Subject
- Observer = Listener = Subscriber

Decorator



add attributes to the core object, not the interface / super class.

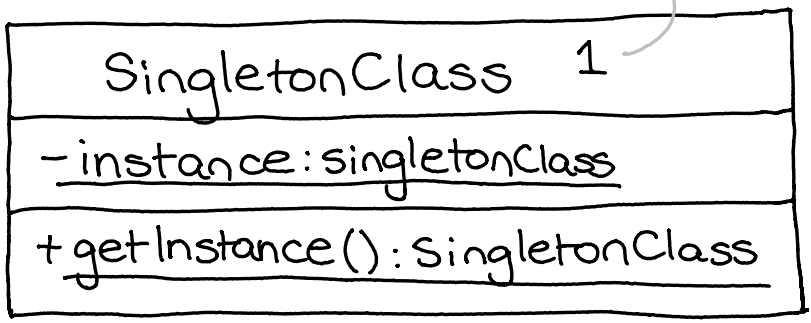
Strategy



- Context stores one strategy which it is using.
- MyApp creates the relevant strategy and uses +setStrategy(s) to set it.

Singleton

labelled w/ 1 in UML



- getInstance() either instantiates or returns existing instance.
- instance & getInstance() both static.