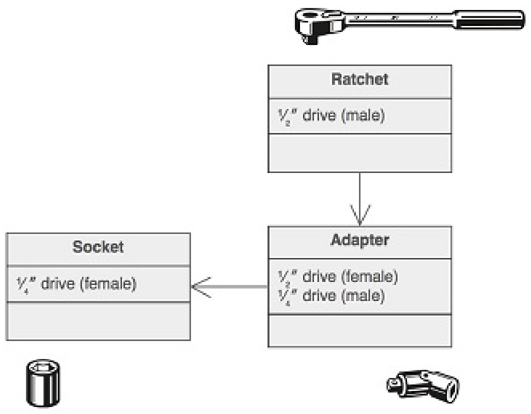
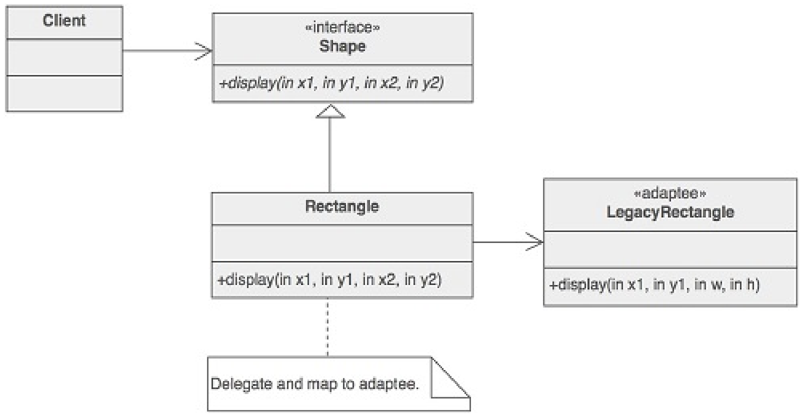
Structural

General:

- ease the design by identifying a simple way to realize relationships between entities
- about class and object composition
- use inheritance to compose interfaces
- define ways to compose objects to obtain new functionality

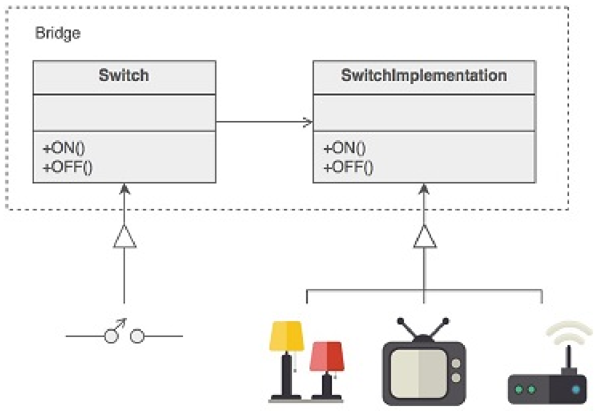
Adapter:

- Definition/Use
  - 'adapts' one interface for a class into one that a client expects
  - used when a client class has to call an incompatible provider class
  - an "off the shelf" component offers compelling functionality but its "view of the world" is not compatible
  - wrap an existing class with a new interface
- Structure
  - 

  - 
Client
«interface»
Shape
+display(in x1, in y1, in x2, in y2)
Rectangle
+display(in x1, in y1, in x2, in y2)
«adaptee»
LegacyRectangle
+display(in x1, in y1, in w, in h)
Delegate and map to adaptee.

- Notes
  - put the adapter term in the name of the adapter class to indicate the use of the pattern to the other developers

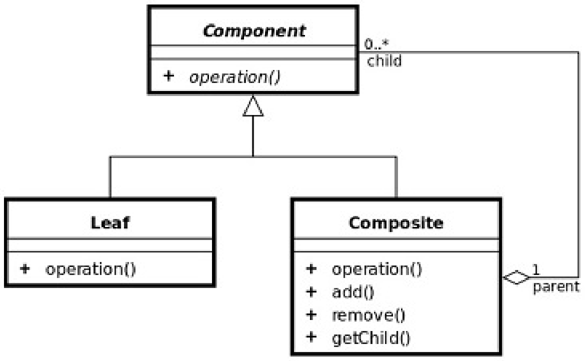
Bridge:

- Definition/Use
  - decouple an abstraction from its implementation so that the two can vary independently
  - useful when a code often changes for an implementation as well as for a use of code
  - decouple an abstraction from its implementation so that the two can vary independently
- Structure
  - 

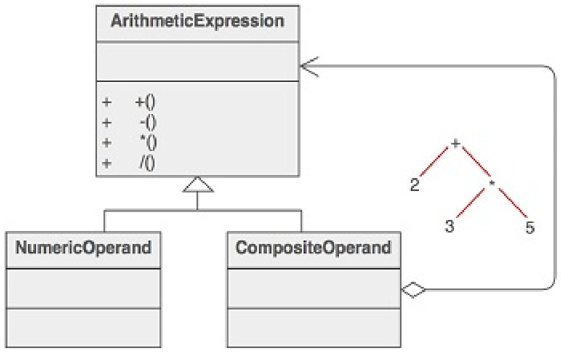
- Notes
  - design the separation of concerns: what does the client want, and what do the platforms provide

Composite:

- Definition/Use
  - a tree structure of objects where every object has the same interface
  - application needs to manipulate a hierarchical collection of "primitive"(leaf) and "composite" objects
- Structure



–

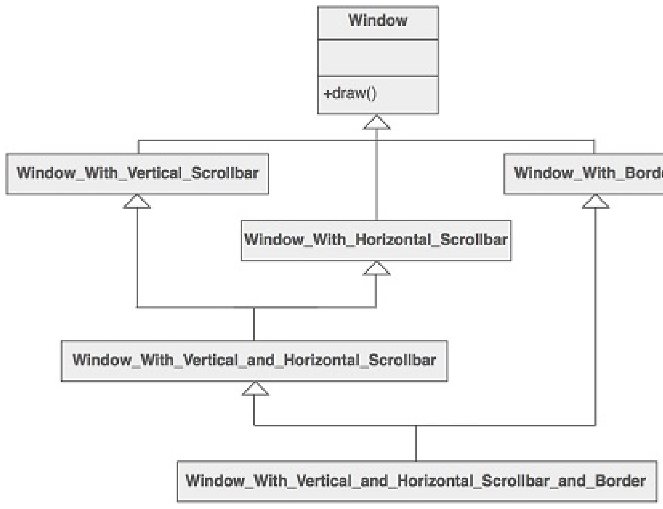


–

- Examples
  - GUI, widgets organized in a tree and operations (resize, repainting) on all widgets processed using pattern
- Notes
  - consider the heuristic, ”containers that contain containees, each of which could be a container”

Decorator (Wrapper):

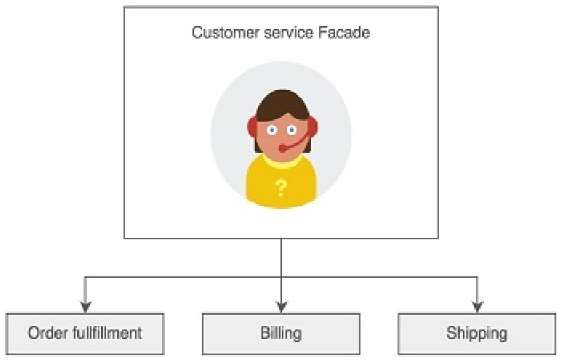
- Definition/Use
  - add additional functionality to a class at runtime where subclassing would result in an exponential rise of new classes
  - client-specified embellishment of a core object by recursively wrapping it
- Structure

– 
Window
+draw()
Window_With_Vertical_Scrollbar
Window_With_Border
Window_With_Horizontal_Scrollbar
Window_With_Vertical_and_Horizontal_Scrollbar
Window_With_Vertical_and_Horizontal_Scrollbar_and_Border

  *
```
Widget* aWidget = new BorderDecorator(
  new HorizontalScrollBarDecorator(
    new VerticalScrollBarDecorator(
      new Window( 80, 24 ))));
aWidget->draw();
```

- Notes
  - ensure the context is: a single core (or non-optional) component, several optional embellishments or wrappers, and an interface that is common to all

Farcade:

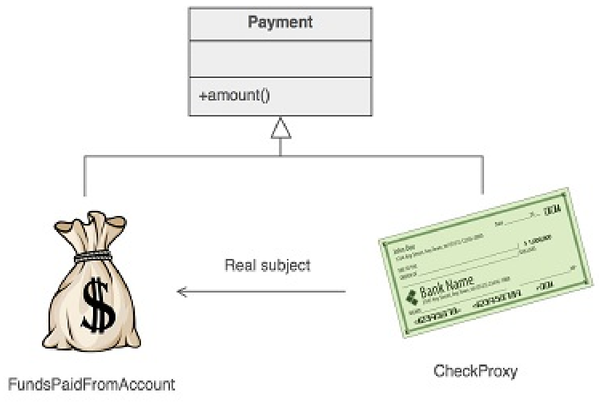
- Definition/Use
  - create a simplified interface of an existing interface to ease usage for common tasks
  - hides the complexities of the system and provides an interface to the client from where the client can access the system
- Structure
  - 

- Notes
  - often singletons because only one facade object is required
  - client uses (is coupled to) the facade only

Flyweight:

- Definition/Use
  - a large quantity of objects share a common properties object to save space
  - each ”flyweight” object is divided into two pieces
    - the state-dependent (extrinsic) part: stored or computed by client objects, and passed to the Flyweight when its operations are invoked
    - the state-independent (intrinsic) part: stored (shared) in the Flyweight object
- Example
  - in video games, it is usual that you have to display the same sprite (i.e. an image of an item of the game) several times
    - it would highly use the CPU and the memory if each sprite was a different object
    - so the sprite is created once and then is rendered at different locations in the screen
    - this problem can be solved using the flyweight pattern
    - the object that renders the sprite is a flyweight

Proxy:

- Definition/Use
  - a class functioning as an interface to another thing
  - provide a surrogate or placeholder for another object to control access to it
- Structure
  - 

Structural Notes

- Example
  - ProxyImage and RealImage

Comparison:

- adapter makes things work after they're designed, bridge makes them work before they are
- composite and decorator have similar structure diagrams, reflecting the fact that both rely on recursive composition to organize an open-ended number of objects
- decorator and proxy have different purposes but similar structures