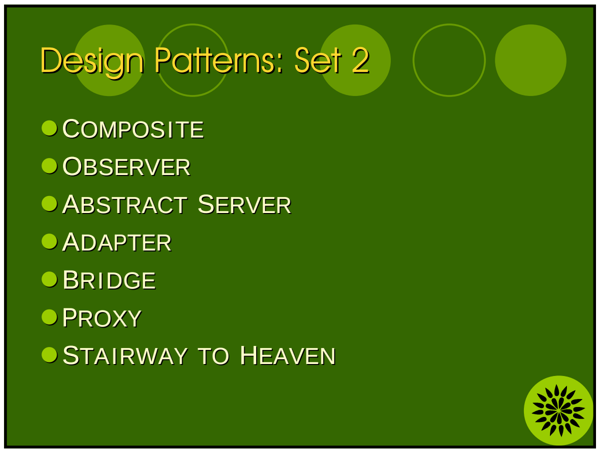
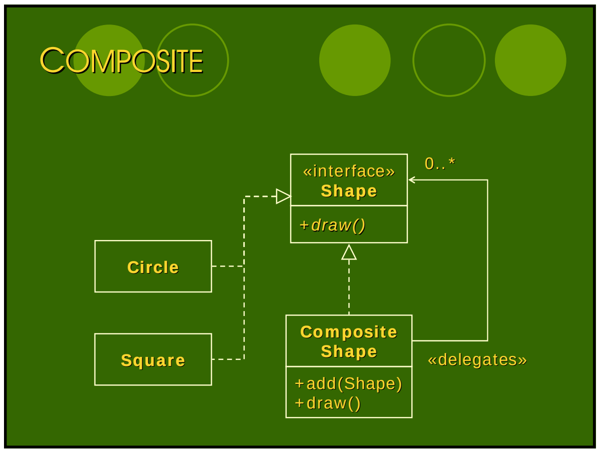
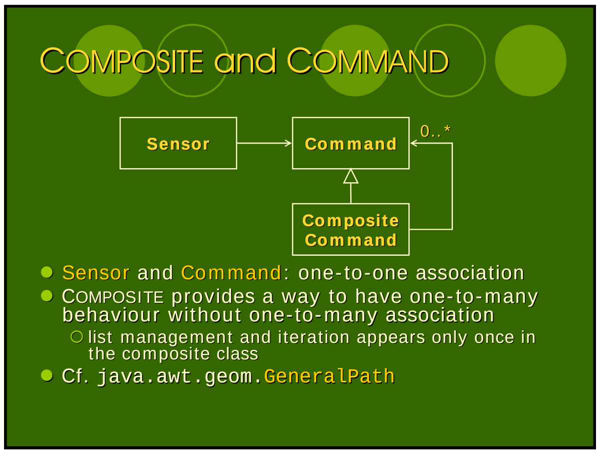


Design Patterns: Set 2

- COMPOSITE
- OBSERVER
- ABSTRACT SERVER
- ADAPTER
- BRIDGE
- PROXY
- STAIRWAY TO HEAVEN

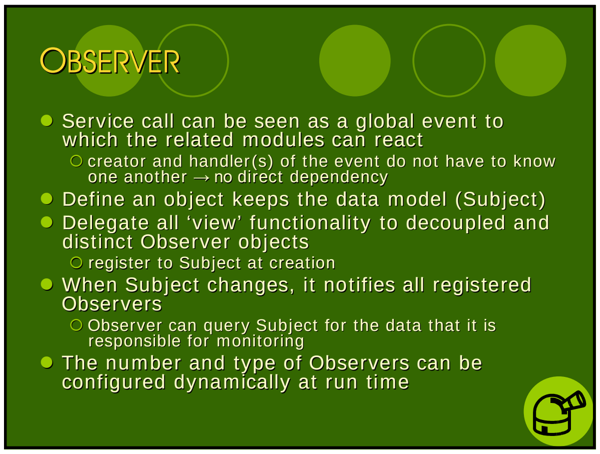


COMPOSITE



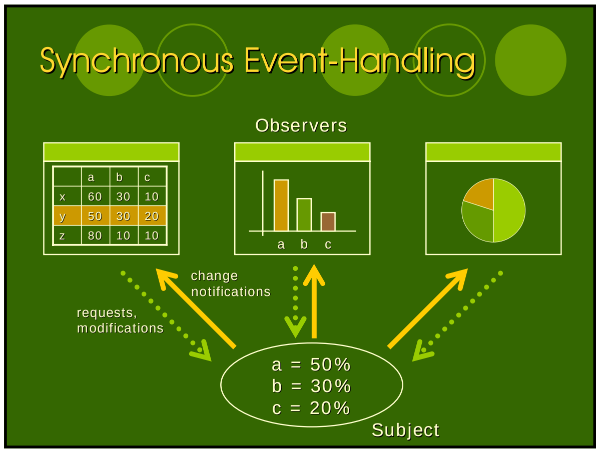
COMPOSITE and COMMAND

- Sensor and Command: one-to-one association
- COMPOSITE provides a way to have one-to-many behaviour without one-to-many association
  - list management and iteration appears only once in the composite class
- Cf. `java.awt.geom.GeneralPath`

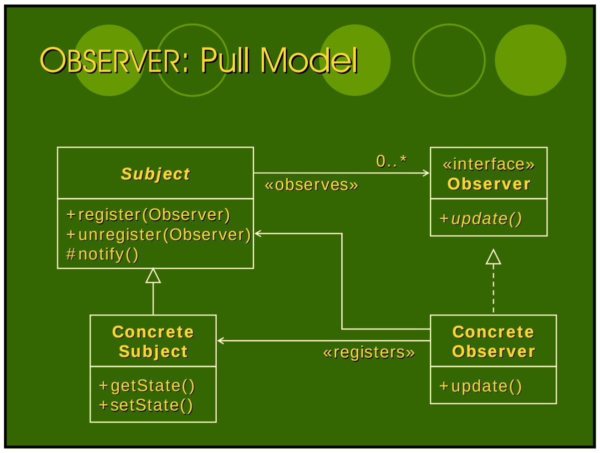


OBSERVER

- Service call can be seen as a global event to which the related modules can react
  - creator and handler(s) of the event do not have to know one another → no direct dependency
- Define an object keeps the data model (Subject)
- Delegate all 'view' functionality to decoupled and distinct Observer objects
  - register to Subject at creation
- When Subject changes, it notifies all registered Observers
  - Observer can query Subject for the data that it is responsible for monitoring
- The number and type of Observers can be configured dynamically at run time

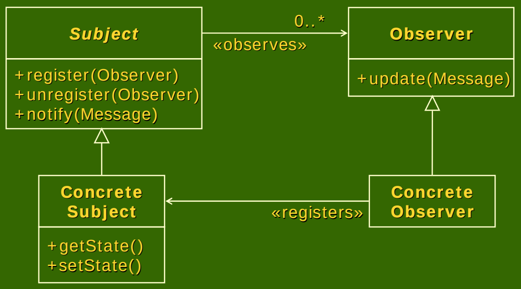


Synchronous Event-Handling



OBSERVER: Pull Model

OBSERVER: Push Model

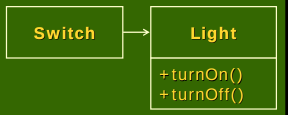


Features

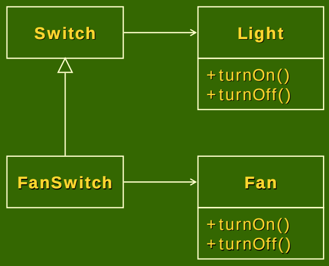
- Use Observer when
  - an abstraction has two aspects, one dependent on the other
  - a change to one object requires changing others, and you do not know how many objects need to be changed
  - an object should be able to notify other objects without assumptions about these objects
- Observer is a widely used pattern: once you understand it, you see uses for it everywhere
  - you can register observers with all kinds of objects rather than writing those objects to explicitly call you
- Cf.
  - `java.awt.event.ActionListener`
  - `java.util.Observer` and `java.util.Observable`

ABSTRACT SERVER – A Motivating Example

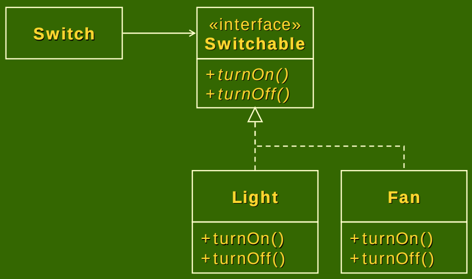
- Design software for a simple table lamp
  - switch: on/off
  - light: on/off
- The simple design violates
  - DIP
  - OCP



Example: A Bad Way to Extend Switch



Example: Extending Switch with ABSTRACT SERVER



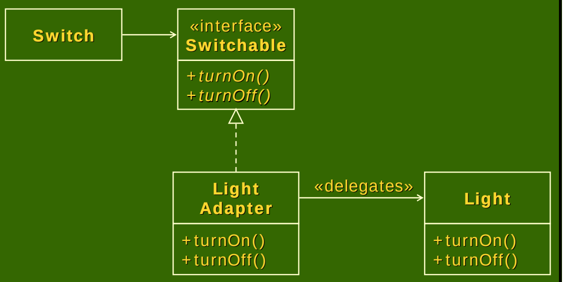
Who Owns the Interface?

- Interfaces belong to the client, not to the derivative
  - Switch cannot be deployed without Switchable
  - Switchable can deployed without Light
- Inheritance hierarchies usually should not be packaged together
  - package clients with the interfaces they control
- Cf.
  - `java.io.Closeable`, `java.io.Flushable`
  - `javax.swing.table.TableModel`

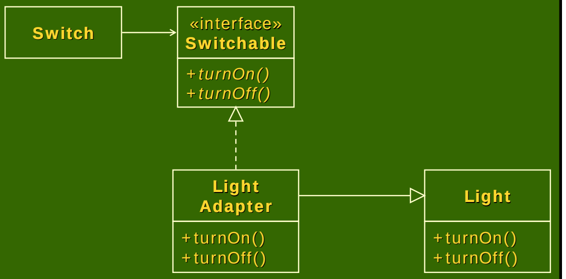
ADAPTER

- Potential SRP violation in ABSTRACT SERVER:
  - Light and Switchable may not change for the same reasons
  - what if Light cannot be inherited?
- Solution: add a class that can be adapted to the interface
  - drawback: extra classes, instantations

Example: Object-Form Adapter



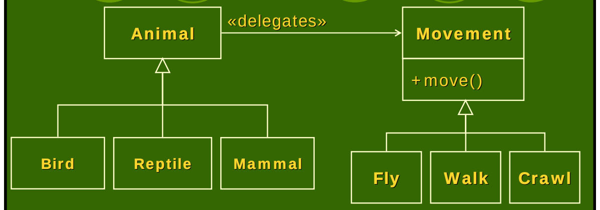
Example: Class-Form Adapter



BRIDGE – A Motivating Example

- Modelling animal characteristics
  - each type of animal can have different number of legs (integer)
  - each type of animal can have different type of movement: fly, walk or crawl
  - an animal must be able to return the number of legs when asked
  - an animal must be able to calculate how long it would take to move a distance given the type of terrain
- Variation in number of legs: a member variable with get/set methods
- Variation in movement type:
  - a member variable to indicate the type and to select different code for movement
    - tight coupling, messy code
  - animal types are derived from a base class
    - need to manage subtypes of animals
    - no animals with more than one type of movement
    - subtyping based on one property; what about classifying them as mammals, reptiles and birds?

Example: Bridging Two Hierarchies



- Encapsulate the behaviour (i.e. movement) into a class
- The animal class contains an object that has the appropriate behaviour

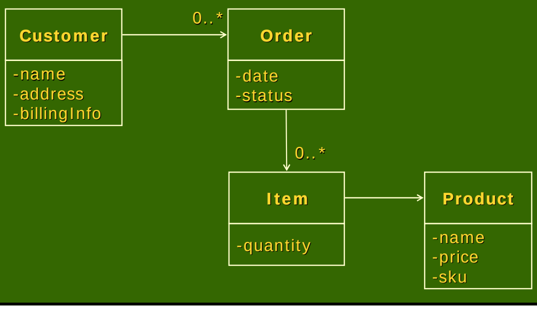
Commonality and Variability

- Commonality analysis
  - what are the common elements among the elements
  - define a family to which the elements belong and a context where things vary
  - find the structure that is unlikely to change over time
- Variability analysis
  - how things vary within the context of commonality (variability only makes sense within a given commonality)
  - find the structure that is likely to change
- Shortly: When the type hierarchy has more than one degree of freedom
  - separate the hierarchies
  - tie them together with a bridge

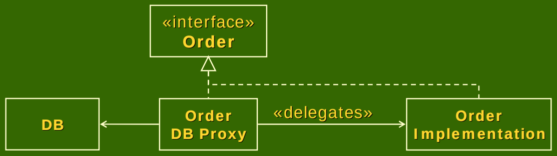
PROXY

- Allows to cross a barrier without either of the participants knowing about it
  - database
  - network
- Theory: PROXY can be inserted in between two collaborating objects without them knowing about it
- Reality: not so trivial…

Example: Web Shopping Cart



Example: Order Proxy



- Interface that declares all the methods that clients need to invoke
- Class that implements those methods without knowledge of the database
- Proxy that knows about the database

Example: Order Proxy (cont'd)

```
public interface Order {
    public String getCustomerId();
    public void addItem(Product p, int quantity);
    public int total();
    }

public class OrderImp implements Order {
    private List<Item> itsItems;

    public int total() {
        int total = 0;
        for (Item item : itsItems)
            total += item.getProduct().getPrice() * item.getQuantity();
        return total;
    }
    /* rest of the implementation omitted */
}
```

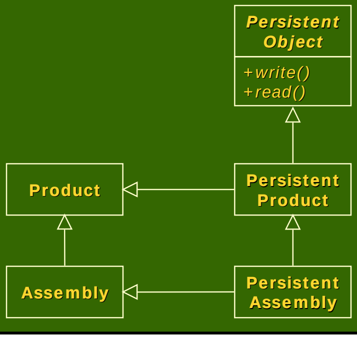
Example: Order Proxy (cont'd)

```
public class OrderProxy implements Order {

    public int total() {
        OrderImp imp = new OrderImp(getCustomerId());
        ItemData[] itemDataArray =
                                DB.getItemsForOrder(orderId);
        for (ItemData item : itemDataArray)
            imp.addItem(new ProductProxy(item.sku),
                        item.qty);
        return imp.total();
    }

    /* rest of the implementation omitted */
}
```

STAIRWAY TO HEAVEN

- Achieves dependency inversion (like PROXY)
- Employs a variation on the class form of ADAPTER
- Only useful in languages supporting multiple inheritence
- Completely seperates knowledge of the database away from the business rules of the application

STAIRWAY TO HEAVEN (cont'd)



Reading for the Next Week

- Section 6: The ETS Case Study
  - Chapter 28: VISITOR
  - Chapter 29: STATE
  - Chapter 30: The ETS Framework