C15: Design Patterns

Gamma,Helm,Johnson,Vlissides (GOF)

Adapter

- How to use an appropriate class given the wrong interface?
- Define adapter as an intermediary, does little work itself, mostly translates and passes on (cf. electrical travel adapter)
- E.g.: use Vector as Set (either by inheritance or composition), or class InputStreamReader

Composite

- How to create complex objects using only simple parts?
- Provide a collection of simple objects and allow them to be nested arbitrarily.
- E.g.: AWT containers (btw, figure 15.2 is actually an example of a widget-relationship diagram)

Strategy

- How can the algorithm solving a particular problem be changed easily and dynamically?
- Define a bunch of algorithms with a common interface, let the client select
- E.g.: classes implementing LayoutManager

Observer

- How to have loosely coupled objects act in synchronicity?
- Maintain a list of objects “listening” to some other object, notify all about changes inside the latter
- E.g. all kinds of Listeners in Java, even class Observable and interface Observer (notifyObservers(), update(Observable,Object)

Flyweight

- How to reduce storage cost for objects with similar state?
- Share all common state
- E.g. information about a class (name, method signatures, …) is stored in an instance of class Class, every instance of a particular class points to the same Class instance thus sharing all this information.

Abstract factory

- How to create instances of related classes without specifying/revealing actual representations?
- Use a method returning a new value characterised by some interface or parent class only.
- E.g. elements() returning a enumerator for whatever collection, FontMetric, …

Factory method

- How to have a method return different types of objects for various subclasses?
- Subclass overrides, returns whatever, but must still be a subclass of the return-type; client has to cast appropriately
- E.g. clone() method

Iterator

- How to access all elements of some aggregate linearly without exposing the implementation of the aggregate?
- Provide a mediator object just for access
- E.g. Enumeration for collection classes

Decorator (filter/wrapper)

- How to attach additional behaviour to an object dynamically?
- Combine is-a and has-a relationship: create a class wrapping around the original value, adding new methods, not changing old ones
- E.g.: InputStreamReader

Proxy

- How to hide details, such as transmission protocols to remote objects?
- Have a proxy acting as a surrogate/placeholder for another class; the proxy can perform substantial computation, or be smart (e.g. cache), …
- E.g. that's how RMI (remote method invocation) works

Bridge

- How to decouple an abstraction from its implementation (such that the latter can vary independently)?
- Remove implementation detail from the abstraction, put it into a datafield (composition)
- E.g. WindowPeer class