A Quick Survey of Some Other Patterns

From Gamma et al.

Other patterns

- Gamma et al. define 23 O.O. design patterns and other practitioners have identified dozens more.
- The following selection outlines some of the ones, not covered earlier, I've found most useful. All are in Gamma et al.

Abstract Factory (Creational)

- Use factory methods to create a family of related objects.
  - E.g. In java.awt a factory object creates a set of widgets that work together (scrollbars, windows, menus)

Singleton (Creational)

- Ensure that only one instance of a class is created

```
class Quangle {
    private static Quangle singleton = null ;
    // Constructor is private
    private Quangle() { … } ;
    public static Quangle getInstance() {
        if( singleton==null ) singleton = new Quangle() ;
        return singleton ; }
    … }
```

  - Use with caution. Forcing client code to call static methods prevents reuse with subclass.

Adapter (Structural)

(aka Wrapper)

- Use a class to adapt another class to conform to an expected interface.
  - Whereas the Proxy pat. adds functionality while keeping the same interface, the Adapter pat. changes the interface while leaving the same functionality.

Façade (Structural)

- Provide a single interface to a set of objects.
  - E.g. In TurtleTalk a single class provides interfaces to the compiler, the interpreter, and the maze (i.e. all aspects of the model).
  - This simplifies the clients by reducing the number of classes they are directly coupled to and the number of objects they need to interact with.

Iterator (Behavioural)

- Give sequential access to the items of a collection without exposing its representation.
  - Allows generic algorithms to operate on a variety of collections (sets, lists, maps, etc)
  - java.util.Iterator

    public boolean hasNext()

    public Object next() ; // Mutator
  - Issues abound when you consider insertions and deletions from the underlying data structure.

Mediator (Behavioural)

- Use an object to mediate all interaction between two or more other objects.
  - The mediator class calls on the other classes so that neither has to call (and hence depend on) the other
  - E.g. In TM a class called TMBigApplet mediates communication between model side and view side (each of which is represented by a façade object)

Template Method (Behavioural)

- Vary details of algorithm by filling in abstract methods.

```
abstract class AbstractParent {
    …
    public void templateMethod() {
        part of algorithm
        hookMethod() // "down call"
        another part of algorithm }
    protected abstract void hookMethod() ; }
    … }
class Child1 extends AbstactParent {
    protected void hookMethod() { implementation 1 } }
```