From: Head First Design Patterns

**OO patterns:** Inheritance, interfaces, encapsulation

**Observers:** Monitoring objects.

**Decorators:** Instead of using inheritance, extend functionality without changing code using object compositions

**Factory:** Creating objects without using `new` (concrete).

**Singletons:** Only one instance to rule them all.

**Adapters:** Adapt one object for another to reuse functionality.

**Encapulating algorithms:** Subclasses pick and choose to customize algorithm.

**Iterators:** Iterate through complex data structures (e.g., list, stack, composite) in a customized fashion through a unified interface.

**Strategy and state:** State machines, change request..

**Object access:** Proxies control and manage access. Remote access example: RMI
Facade: Unified interface to simplify a complex subsystem.

Flyweight: Objects share data to reduce memory footprint. Shared parts held in external structures.

Bridge: Allow inserting an object to perform interface function.

Abstract factory: Interface to create unknown objects. Allows creating families of objects conveniently.

Builder: Share same constructor for multiple classes.

Prototype: Cloning objects from a prototype.

Mediator: Lets family of objects talk to each other without explicit connection.

Visitor: Flexible classes to transverse data structures.
Templates: Leave some functions undefined, to be defined in subclasses.

Strategy: Interchange algorithms to perform a task in different ways.

Observer/Listener: State changes sends notifications.

Memento/Serialization: Capture and save object state, to be restored later.