Creational
Factory method - like abstract factory, but one method.
Abstract factory - many concrete factories providing common interface - like createButton for win, mac. Many methods coupled together for creating objects that are related.
Builder - object allowing sequential setting creation parameters and creating object at once
Lazy initialization - do not perform initialization until the object is used - flag + mutex in mt
Multiton - like singleton, but a map of named objects
Object pool - pool of objects, after usage returned to the pool, requests may be blocking or may fail
Prototype - create objects based on the pre-created prototype, avoid subclasses in favor of parameters set at runtime
Resource acquisition is initialization - tying resources to the lifetime of object
Singleton - no comments

Structural
Adapter/Wrapper - converts one interface into another via inheritance or member
Bridge - decouples interface from implementation. If one fixed implementation - pImpl.
Composite - one object or a group of objects have one common interface (see text editor)
Decorator - additional functionality keeping the same interface
Facade - provide one interface to the set of interfaces/components
Flyweight - minimize memory for large amount of objects by sharing the same info (font graphics in the word processor or QContainers implementation
Proxy - provides different functionality for the same interface, ATM to the bank, web proxy etc

Behavioral
Chain of responsibility - an object (request or command) is passed through a chain of objects which can process it and pass further or not
Command - putting the request into object, enables queueing, history, undo etc
Iterator - allows iteration over objects in an aggregate without exposing its internal structures
Mediator - decouples objects from each other easing maintaining applications (?)
Memento - allows saving/restoring internal object state. Originator provides memento objects to the outer world that contain originators state.
NullObject - object that implements an interface but does nothing. Used instead of returning NULL and checking against it
Observer - object with changing state notifies registered observers (observer interface)
Publisher/Subscriber - object publishes state changes in a form of messages to the subscibers. Sometimes a queue is involved, sometimes with message priorities.
State - object represents various states of some component. Each state may define separate implementation of set of operations (common interface) instead of set of switches.
Strategy - wrap a set of algorithms to perform some operations in wrapper and allow changing them at runtime.
Template method - defines algorithm/program skeleton with some steps and lets the subclasses implement these steps

Visitor - element->accept(visitor) { visitor->visit(this);} Double dynamic dispatch. Useful on iterations over larger number of objects (save doc elements, find, backup etc)

Concurrency

Active Object - async calling methods in object specific thread, callback for result, scheduler and a list of requests

Double checked locking - considered antipattern, require volatile keyword to work correctly

Read-Write Lock - multiple reads, single write

Scheduler - controls which thread or object can access resource with specified policy (avoiding starvation)

Thread pool - a few thread are pre-created and wait to perform tasks.

OS Messages - scheduler + active object + registration of handlers