Design Patterns – C#

Creational : Instances Objects – LifeCycle Runtime Or CompileTime

Structural : Comunicating Between Independent Classes - Wrapping function

Behavioral : Communicate Behavior Between Classes Entities – Responsibility – Proccess - Loosly Couply

---

Creational : ***Singelton***

Use For : Session user - Access to a file config-connection

Creational : ***Abstract***

Use For : Posting product-package

Creational : ***Factory***

Use For : All kind of datasources (Sql-Xml) – switch_case - Example : car models

Creational : ***Builder***

Use For : Complex objects-step by step-director - builder - fastfood

Creational : ***Prototype***

Use For : Clone *shallow,deep

Creational : ***Multiton***

Use For :Group of singelton by key - camera controller

Structural : ***Adapter***

Use For : Repo data access

Structural : ***Composite***

Use For : Hierachical object models – Example : manager employeers

Structural : ***Decorator***

Use For : Provide vehicles for hire during track days , Extendition functionality = additional operations

Structural : ***Facade***

Use For : Hidden complex business logic

Structural : ***Flyweight***

Use For : Minimize resource –Intrinsic (stateless-save in flyweight objects-shared), Extrinsic (stateful-save out of flyweight objects-uUnShared) – Hashtable – Example : war strategy

Structural : ***Proxy Design***

Use For : Provide a placeholder object – public interface

Use For : Cache , virtual, remote, protection , Smart proxies

---

Behavioral : ***Chain Of Responsibility***

Use For : Linked list of handler– successor = next handler – Example : try_cache and vending_machine

Behavioral : ***Command***

Use For : Queue(seq) of command, stack(rollbacking commands) receiver(robot-functionality), invoker(controller=> enqueue,undo command) – Example : sqlCommand

Behavioral : ***Interpreter Like Composite***

Use For : Hierarchy of expression (terminal node = leaf = just Operation, non-termiranl = nodes = operator or operation) – domain_spec or notation – performance not important – reverse func

Behavioral : ***Visitor***

Use For : Separate data structure and functionality – traverse hierarchy for generate salary or example increase salary – callBack (this) – accept=>visitor=>element

Behavioral : ***Mediator***

Use For : Transmit message – increase maintability – Example : online presenter with attendee

Behavioral : ***Memento***

Use For : BackupRestore/snapshot objects – undo functionality

Behavioral : ***Observer***

Use For : Event model - dependent objects update auto/notify state – observer object that subscribe to a subject – Example : Logger System

Behavioral : ***State***

Use For : Changed state at runtime so do not need switch-case and If to If – Example : audio player = context

Behavioral : ***Strategy***

Use For : Group Of Algorithms For Diff Strategy Game

Behavioral : ***Template***

Use For : Multi-Step algorithms - similar strategy the key diff is the ability to vary parts of the algorithm rather than replacing the algo in its entierly

Behavioral : ***Iterator***

Use For : Accessed seq without knowledge of its structure –mehodes = iterator = IEnumerator , array Data= aggregate = IEnumerable

---

Misc : ***Non-Virtual Interface***

Use For : How to override – sub class do not need to recompile – write protected method in public method as “public interface(it is not related to Interface OOP)”

Misc: ***Null Object***

Use For : Often created with singelton and strategy and factory method – Example : reporting check status

Misc: ***Object Pool Design***

Use For : Improve performance - Example : Connection pool sql server- two List (available-in use)

Misc: ***Service Locator***

Use For : Decouple a class from its dependency by centralize service locator object - Example : PeymentTerm with PeymentMonth