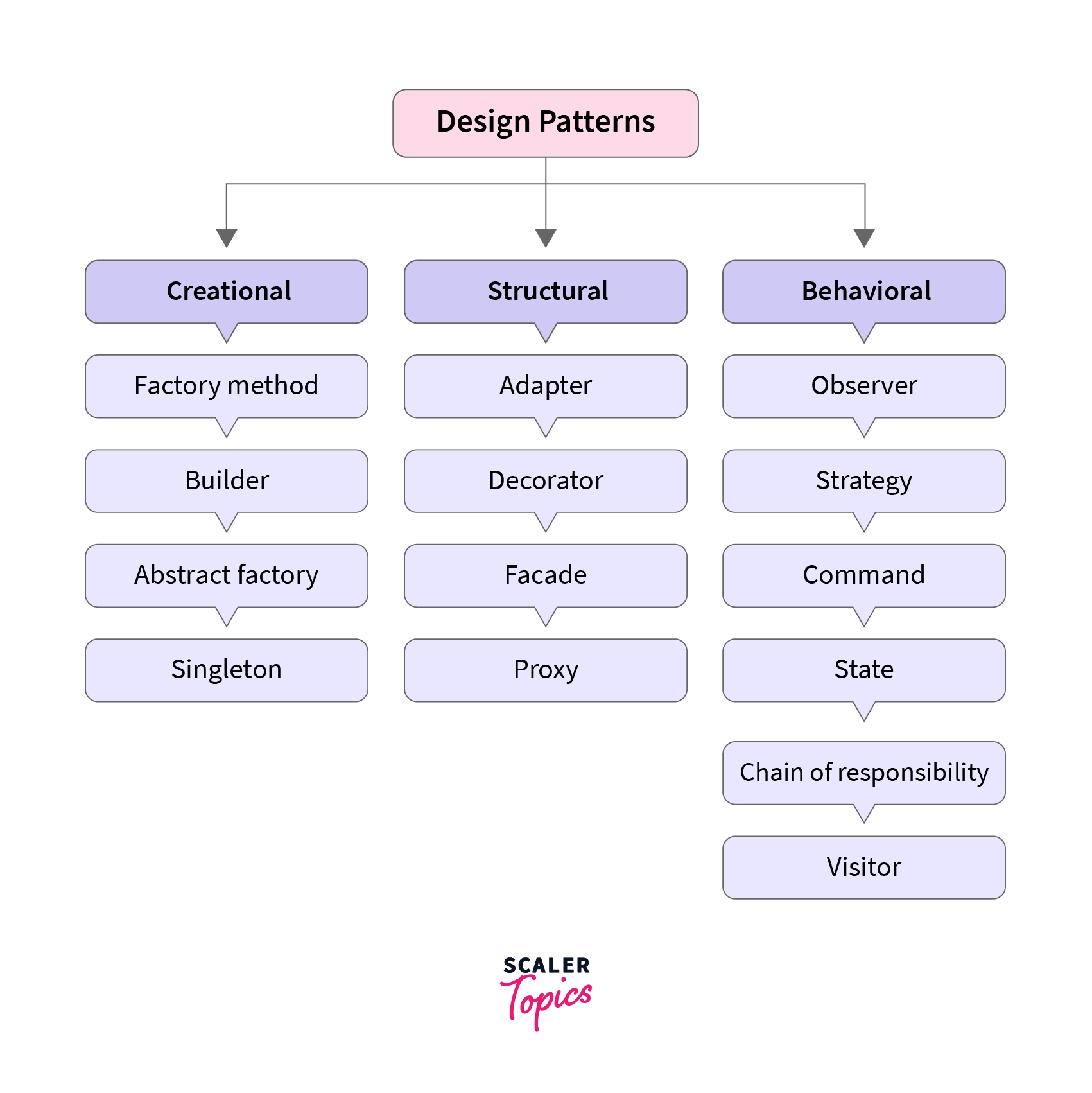
Creational patterns are focused towards how to instantiate an object or group of related objects.

Creational design patterns are design patterns that deal with object creation mechanisms, trying to create objects in a manner suitable to the situation.



Creation Pattern Types:

There are three basic classifications of patterns:

Creational, Structural, and Behavioral patterns.

Creational Patterns

ABFPS (Abraham Became First President of States).

* Abstract Factory: Creates an instance of several families of classes [ Create different Instance of F1,F1,F3 ]
* Builder: Separates object construction from its representation : [ Build Objects in Multiple Steps ]
* Factory Method: Creates an instance of several derived classes . [ Create an different instance from C1,C2,C3 ]
* Prototype: A fully initialized instance to be copied or cloned. [Clone & reuse object ]
* Singleton: A class in which only a single instance can exist.[ one instance per class]

**Note**: The best way to remember Creational Pattern is by remembering ABFPS (Abraham Became First President of States).

Structural Patterns

* Adapter: Match interfaces of different classes [CA 🡪 CIF 🡪 CB]
* Bridge: Separates an object’s abstraction from its implementation [High Levels -🡪 Low Levels Objects abstractions]
* Composite: A tree structure of simple and composite objects. [Uniform Object tree/Hierarchy creations from different type of related Objects structure]
* Decorator: Add responsibilities to objects dynamically. [Objects A 🡪 Add Multiple Flavours and resposibility]
* Façade: A single class that represents an entire subsystem . [ Abstract complex processing steps in group of activity and expose façade interface to client.] [Façade -🡪 S1,S2, S3…Sn]
* Flyweight: A fine-grained instance used for efficient sharing. [Share Object instance for efficient sharing of costly resources]
* Proxy: An object representing another object .[Deferred full fledge object creation until it is needed.]

**Note**: To remember Structural Pattern best is (ABCDFFP) : ABCD can Fly with Façade Proxy.

[ Anybody ABCD can dance & fly - weight with package in Façade with light weight proxy object. ]

Behavioral Patterns

* Mediator: Defines simplified communication between classes
* Memento: Capture and restore an object's internal state
* Interpreter: A way to include language elements in a program
* Iterator: Sequentially access the elements of a collection
* Chain of Resp: A way of passing a request between a chain of objects
* Command: Encapsulate a command request as an object
* State: Alter an object's behavior when its state changes
* Strategy: Encapsulates an algorithm inside a class
* Observer: A way of notifying change to a number of classes
* Template Method: Defer the exact steps of an algorithm to a subclass
* Visitor: Defines a new operation to a class without change

**Note**: Just remember Behavioral Pattern Music....... 2 MICS On TV (MMIICCSSOTV).

