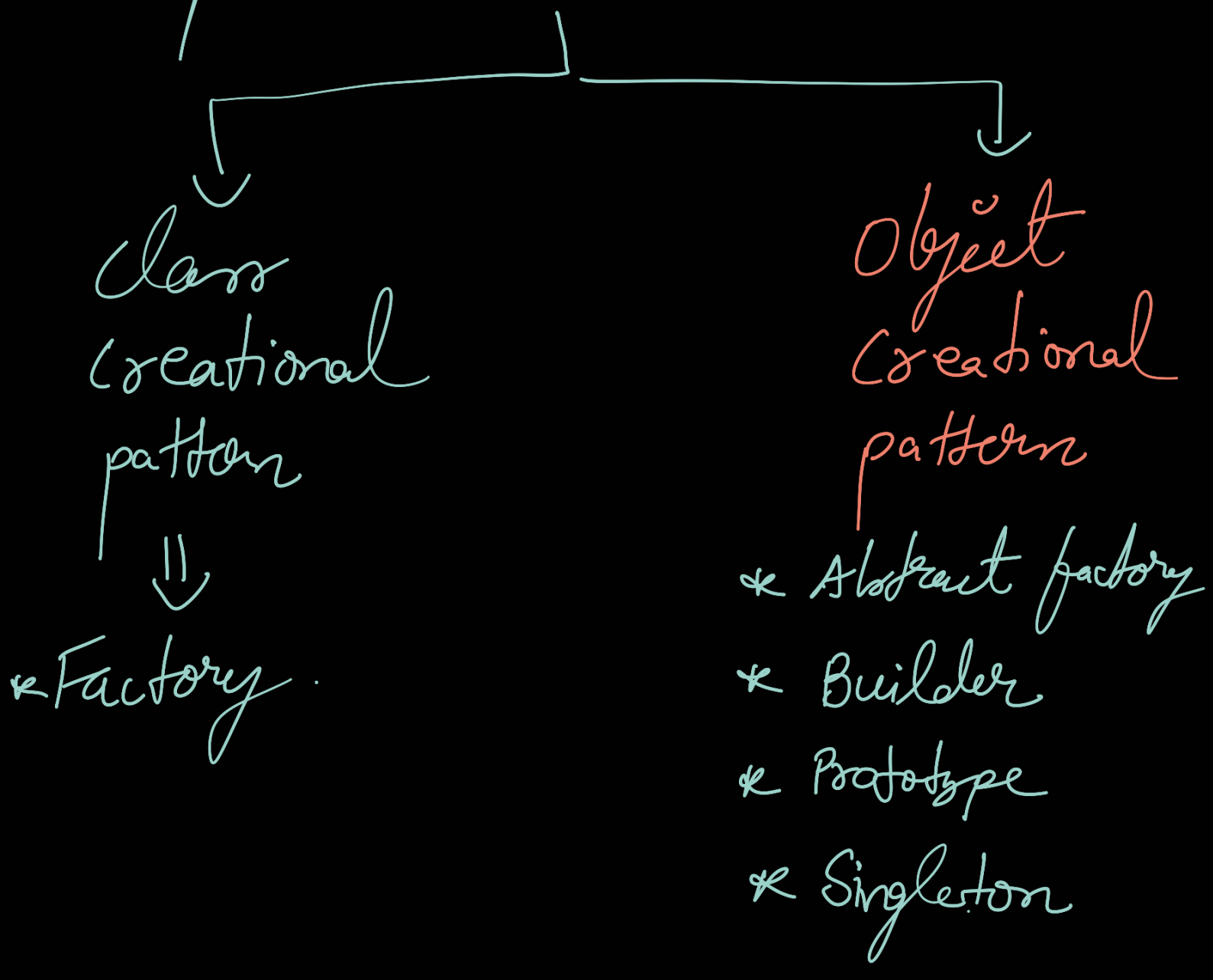
Creational Design Pattern

* Abstract the instantiation process
* Make system independent of how objects are created, composed.



① Factory Design Pattern

* Define interface for creating object, but let subclasses decide which class to instantiate
* Factory Method lets a class defer instantiation to subclasses

↳ Other Name → Virtual Constructor

Library Creator → Library Consumer

Creation of object handled by this

which object to create passed by this

This pattern is helpful when:

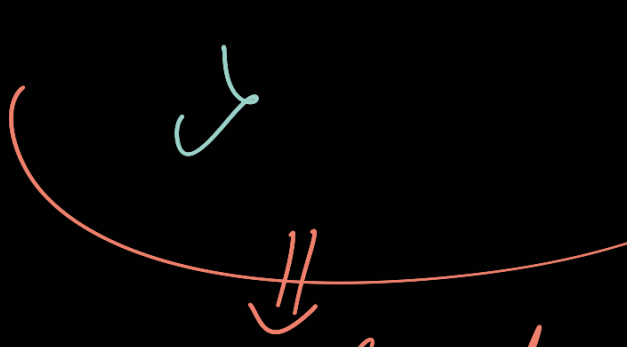
- The exact type of the object to be created isn't known until runtime.
- A system needs to be independent of how its products are created.

* Used to create reusable framework and libraries like GUI Library.

class A || class B || class C

obj A | obj B | obj C

```
Switch (object type)
{
    if ( )
        return obj A
    else if ( )
        return obj B
    else if ( )
        return obj C
```



This code has to be repeated everywhere we want object of class A/B/C

We can move this switch logic of creating object inside factory class.

```
class CreationFactory {
public :
    createObject ()
        { switch case
```