① <u>Factory Method pattern</u>

    It create object based on I/P at runtime.

<u>Adv</u> - ① Loose coupling
     ② Encapsulation → Easy to change

<u>Applⁿ</u> - Java Collection API.

    By using subclass you want to create object, flexibility in code that time used.

<u>Dis</u> → complexity override Factory method

what happen if it doesn't Exist :-

    Leading tight coupling
modify many part for change.

② <u>Abstract Factory method</u>,

    you need to create families of related objects without specifying their concrete classes.

Adv → Security
Consistency → object-created within Form
Scalable → Adding new Product

Appl^n → button, text boxes,
database connection Library.

Dis → Complex to implement
Hard to Extend

IF doesn't Exist → individualy instantiale
Familes of object

Inconistence.
UI compont mixed accidenty
breaking UI

Singleton Pattern

Adv - ① Lazy initolization
obj is created only when it is
needed

Appl^n

Database connection
configuration setting

Dis - ① Global State → tight coupling
    ② multiple thread at a time

If doesn't Exist.

    multiple Instance created
    errors. conflict Log File

Builder pattern

Adv = Flexibility → Allow to construct complex Obj
Redable code   by specify only neede
                            part

    Sequence No matter

App^n → construct pizza obj with many
        Ingredient.

        ② User cond^n APPS

Dis - ① complex

If don't Exist
Constructor with many parameter
my Code hard or error.

Optional parameter would
require overloading constructor.

Ⓑ Prototype pattern.

    Performance
    Decoupling

App^n → Game Development → Character
                                             clone

Dis → Complexly
       Hard code

what If doun't Exist →

    monualy crate new obj Each tine
    performance Low.
    Time consyne