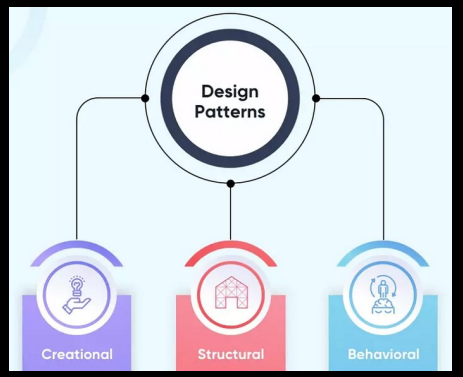


Creational Design Patterns

Ajith de Silva @epita.fr | 2025

EPITA
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Creational Design Patterns :: About

Concerned with Object-Oriented concept

Offer how to create instances form objects in:

… Flexible

… Efficient

… Maintainable

Provide INTERFACE for creating Instances in SUPPER CLASS

Allows SUBCLASSES to alter TYPE of instance to create

Great Decoupling with other code

Centralize creational logic

Creational Design Patterns

1. **Singleton**
   - *… One INSTANCE.*
   - *… Global access*
2. **Factory**
   - *… Interface to create instances in supper class.*
   - *… Subclass decides type of Instance`*
3. **Abstract Factory**
   - *… Group similar Factory based on Instances types*
4. **Builder**
   - *… Create complex instance step by step*
5. **Prototype**
   - *… Clone Existing instance*

Structural Design Patterns

1. **Adaptor/Wrapper**

   … Makes incompatible objects to interact. (Change one interface to another)

   Bridge

2. *… Split large set of closely related objects and make those independent*

   Composite

   … Group similar Factory based on Instances types

3. **Decorator**

   … Group similar Factory based on Instances types

4. **Facade**

   … Clone Existing instance

5.

Solution Founders : GoF

1. **Eric Gamma + Richard Helm**

   … met in 1990

   … initiated how to solve this problem

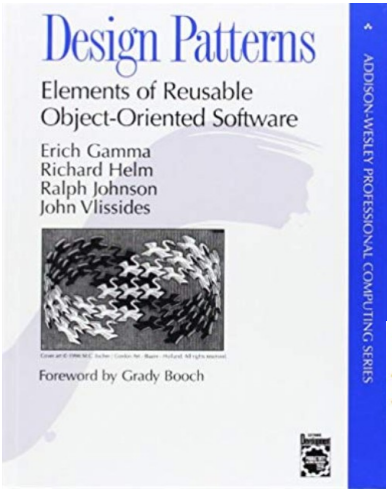
2. **They met Ralph Johnson & John Vlissides**

   … continue defining generic solutions for recurring same problems

   … had multiple meetings to finalize

   … known as Gang of Four (GoF)

3. **They invented/defined “DESIGN PATTERNS” in late 1994**



4. **Sold over 500,000 copies in > 13 languages**

What is Design Pattern ???

1. **Blue Print. Can NOT use as IT IS.**

   … kind of how to do specification.

   … NOT implementation. Language independent

2. **Generic way of solving recurring problems**

   … Eg: how to create file

   … how to share & organize …

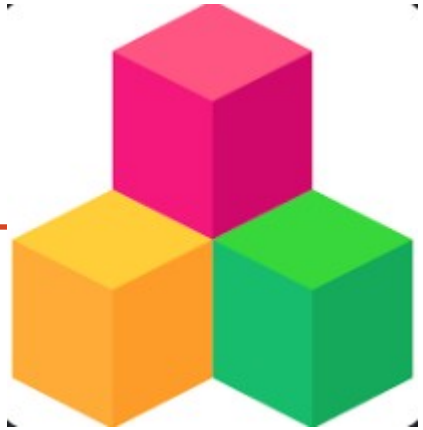
3. **Helps to Design SW that understandable easily.**

   … if someone says Folder, all understand it

   … same way, if someone says “Singleton”, all understand it

4. **Proved & Tested with many programming languages**

Main Categories



1 **Creational**

How to create?

Flexibility

Reuse

2 **Strcutural**

How to assemble?

Efficient & flexible structures

3 **Behavioral**

Responsibility?

Between objects

When to Use Design Patterns ???

1. **Not advisable to use as we want ...**

2. **FIRST Identify the PROBLEM that needs to SOLVE**

   ... needs more time for this

3. **Evaluate Input - > Process - > Output**

   ... will give better view on what to solve & how

4. **Perform multiple Design Approaches**

   ... use appropriate design pattern for each

5. ***Select the BEST based on Trade-offs***

   ... Feasibility, Cost (Time + Money), Development & Operations cost

Caution !!

1. **Can NOT use to solve every solution**

2. **Don’t OVER user | Don’t Misuse**

   … solution will be more complex

   … wrong implementation

3. **Some cases, needs to modify pattern to achieve the goal**

   … when pattern does not match (new variation of existing problem)

4. **Extra Overhead**

   … affect performance since additional layers

In Brief ...

1. Essential Tool for SW Engineers
2. Blue Print of How to solve recurring problems
3. Helps to stay UP with Best practices
4. Helps to create reusable, maintainable & scalable software
5. Communication among team members made easy