Creational patterns

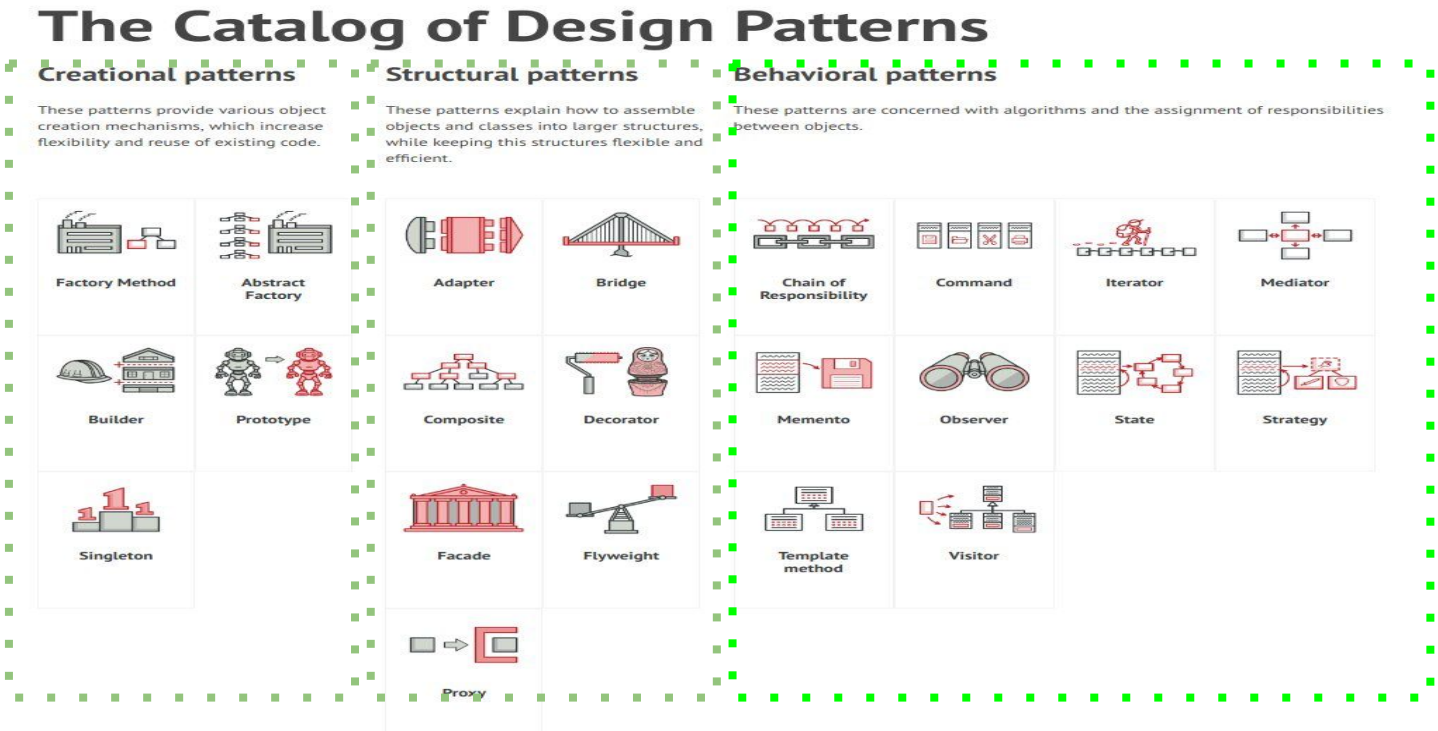
ABSTRACT FACTORY

Upcode Software
Engineer Team

-2023-

CONTENT

The Catalog of Design Patterns

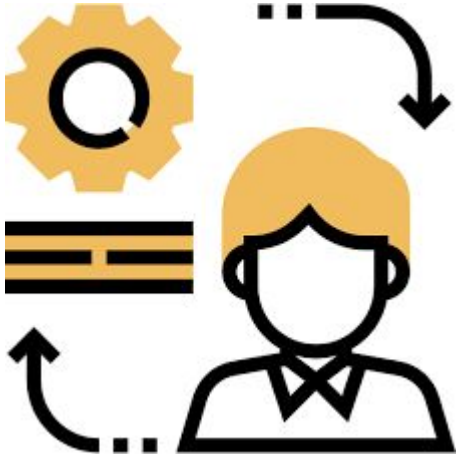


What's a design pattern?

- **Design patterns** are *typical solutions to commonly occurring problems in software design*.
- They are like pre-made **blueprints that you can customize to solve a recurring design problem in your code.**
- Patterns are often confused with algorithms, because both concepts describe typical solutions to some known problems. **While an algorithm always defines a clear set of actions that can achieve some goal, a pattern is a more high-level description of a solution.** The code of the same pattern applied to two different programs may be different.

Why should I learn patterns? (1/n)

- The truth is that you might manage to work as a programmer for many years without knowing about a single pattern.
- A lot of people do just that. Even in that case, though, you might be implementing some patterns without even knowing it.
- So why would you spend time learning them?

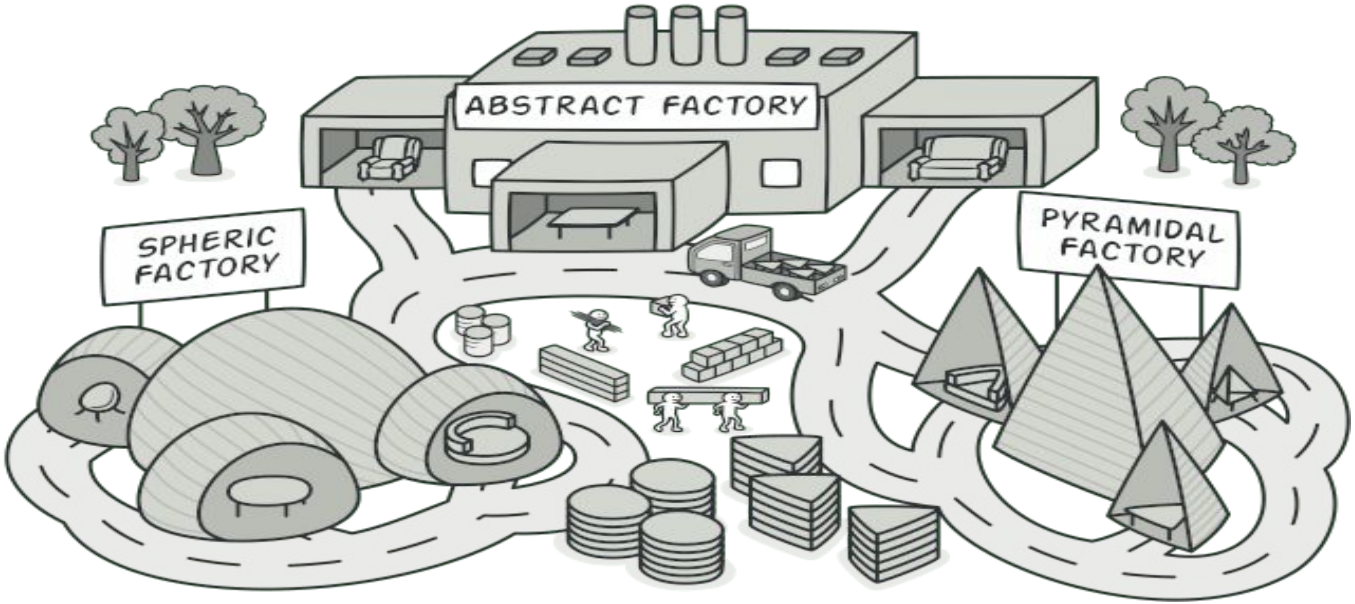


Why should I learn patterns? (2/n)

- **Design patterns** are a toolkit of tried and tested solutions to common problems in software design. Even if you never encounter these problems, knowing patterns is still useful because it teaches you **how to solve all sorts of problems using principles of object-oriented design.**
- **Design patterns** define a common language that you and your teammates can use to communicate more efficiently. You can say, “Oh, just use a Singleton for that,” and everyone will understand the idea behind your suggestion. No need to explain what a singleton is if you know the pattern and its name

What is Abstract Factory ?

Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.



Where to use Abstract Factory ? (1/n)

- Use the Abstract Factory when your code needs to work with various families of related products, but you don’t want it to depend on the concrete classes of those products—they might be unknown beforehand or you simply want to allow for future extensibility.
- The Abstract Factory provides you with an interface for creating objects from each class of the product family. As long as your code creates objects via this interface, you don’t have to worry about creating the wrong variant of a product which doesn’t match the products already created by your app.

Where to use Abstract Factory ? (2/n)

- Consider implementing the Abstract Factory when you have a class with a set of Factory Methods that blur its primary responsibility.

- In a well-designed program each class is responsible only for one thing. When a class deals with multiple product types, it may be worth extracting its factory methods into a standalone factory.

How to use Abstract Factory ? (1/n)

1. Map out a matrix of distinct product types versus variants of these products. Creational Design Patterns / Abstract Factory
2. Declare abstract product interfaces for all product types. Then make all concrete product classes implement these interfaces.
3. Declare the abstract factory interface with a set of creation methods for all abstract products.
4. Implement a set of concrete factory classes, one for each product variant.

How to use Abstract Factory ? (2/n)

5. Create factory initialization code somewhere in the app. It should instantiate one of the concrete factory classes, depending on the application configuration or the current environment. Pass this factory object to all classes that construct products.

6. Scan through the code and find all direct calls to product constructors. Replace them with calls to the appropriate creation method on the factory object.

SOURCE code.

1. Abstract Factory (GitHub.com)
2. Abstract Factory Computer Types (GitHub.com)

Summary.

Abstract Factory patterns work around a super-factory which creates other factories. This factory is also called as factory of factories. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern.

Reference

1. Dive into Design Pattern (page 87)
2. Elements of Reusable Object-Oriented Software (page 87)
3. Abstract Factory Pattern in Java (www.baeldung.com)
4. How to use abstract factory (YouTube (sodoCode))

Thank you!

Presented by **Abbos Fayziboev**

(abbosfayziboev@gmail.com)