Factory and Abstract Factory Pattern - Team Segmentation Fault Enjoyers

Members

- Baring, Jethro
- Muñoz, Jhudiel Van
- Buhat, Marion Emmanuel

History

The Factory and Abstract Factory Pattern was officially introduced as a design pattern in the book "Design Patterns: Elements of Reusable Object-Oriented Software,"

Factory Pattern

Definition

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

Usage

- When a class can't anticipate the class of objects it must create
- When a class wants its subclasses to specify the objects it creates
- Classes delegate responsibility to one of several helper subclasses, and you want to localize the knowledge of which helper subclass is the delegate

Pros

- Loose coupling
- Encapsulation
- Code reuse
- Extensibility

Cons

- Complexity
- Overhead

The Factory Method Pattern is a creational design pattern that provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created. Instead of directly calling a constructor to create an object, a factory method is used, which is declared in an interface or an abstract class and implemented by concrete subclasses. This pattern promotes loose coupling and flexibility in object creation, allowing the client code to depend on interfaces rather than concrete classes. It is particularly useful when a class cannot anticipate the class of objects it needs to create or when the client code needs to be unaware of the actual classes being instantiated.

Abstract Factory Pattern

Definition

Abstract Factory patterns work around a super-factory which creates other factories. It is also called the factory of factories.

Usage

- When the system needs to be independent of how its object are created, composed, and represented.
- When you want to provide a library of objects that does not show implementations and only reveals interfaces.
- When the family of related objects has to be used together, then this constraint needs to be enforced.
- When the system needs to be configured with one of a multiple family of objects.

Pros

- You can be sure that the products you're getting from a factory are compatible with each other.
- You avoid tight coupling between concrete products and client code.
- Single Responsibility Principle. You can extract the product creation code into one place, making the code easier to support.
- Open/Closed Principle. You can introduce new variants of products without breaking existing client code.

Cons

The code may become more complicated than it should be, since a lot of new interfaces and classes are introduced along with the pattern.

The Abstract Factory Pattern is a creational design pattern that provides an interface for creating families of related or dependent objects without specifying their concrete classes. It involves creating abstract classes for object creation, with each abstract class representing a family of related objects. Concrete subclasses implement these abstract classes to produce specific instances of the related objects. This pattern promotes the creation of consistent object families, ensures compatibility between objects within a family, and allows for easy interchangeability of entire families of objects. It is particularly useful when the system needs to be independent of how its objects are created, composed, and represented.