The Factory Design Pattern in Software Development

Jeffrey Walley | 02/09/2025 | CTEC 295

Software design patterns are solutions used to commonly occurring software development problems. The “factory pattern” is a creational type of design pattern that provides an interface for creating objects. Its primary purpose is to encapsulate object creation, promoting loose coupling and enhancing code maintainability. By encapsulating the instantiation process, the “factory pattern” makes code more modular and easier to manage. Using a factory to create objects decouples code from the specific classes, which provides flexibility and scalability. The “factory pattern” also allows new types of objects to be added without modifying existing code, which is a good example of the “Open/Closed Principle” (OCP) of software design.

Four components of the factory pattern are the “creator”, “concrete creator”, “product”, and “concrete product”. The “creator” is an interface that declares the method of creating objects. A “concrete creator” are subclasses of the creator, and they specify the type of object created. The “product” is an abstract class that the “factory” will create, while the “concrete product” is a specific implementation of the “product” interface and relates to an object created by the “factory” method. The “factory pattern” is a form of object encapsulation which promotes flexibility and easier maintenance of code.

Java’s object-oriented principles make it easy to handle object creation in a way that keeps code manageable and reusable. The “factory pattern” separates object creation from an application’s main logic, making it easier to add new types without changing the existing code. This aligns with important design principles like the Single Responsibility Principle and the Open/Closed Principle. A common use case example for the “factory pattern” is creating a ‘vehicle’ common interface class, we can then create “concrete classes” that implement our ‘vehicle’ interface.

**Resources**

* <https://www.baeldung.com/java-factory-pattern> - Baeldung
* <https://www.geeksforgeeks.org/factory-method-for-designing-pattern/> - GeeksforGeeks
* <https://dev.to/mameen/understanding-the-factory-and-factory-method-design-patterns-113d> – Dev Community