**ABSTRACT FACTORY**

O padrão abstrato de fábrica fornece uma maneira de encapsular um grupo de fábricas individuais que têm um tema comum sem especificar suas classes concretas. [1] No uso normal, o software cliente cria uma implementação concreta da fábrica abstrata e, em seguida, usa a interface genérica da fábrica para criar os objetos concretos que fazem parte do tema. O cliente não sabe (nem se importa) com quais objetos concretos obtém de cada uma dessas fábricas internas, uma vez que utiliza apenas as interfaces genéricas de seus produtos. [1] Esse padrão separa os detalhes da implementação de um conjunto de objetos de seu uso geral e depende da composição do objeto, pois a criação de objetos é implementada em métodos expostos na interface de fábrica. [2]

Um exemplo disso seria uma classe de fábrica abstrata que fornece interfaces para criar vários produtos (por exemplo, e ). O sistema teria qualquer número de versões concretas derivadas da classe como ou , cada uma com uma implementação diferente de e que criaria um objeto correspondente como ou . Cada um desses produtos é derivado de uma classe abstrata simples, como ou da qual o cliente está ciente. O código do cliente obteria uma instância apropriada do e chamaria seus métodos de fábrica. Cada um dos objetos resultantes seria criado a partir da mesma implementação e compartilharia um tema comum (todos eles seriam objetos extravagantes ou modernos). O cliente só precisaria saber como lidar com o abstrato ou a classe, não a versão específica que obteve da fábrica de concreto. DocumentCreatorcreateLetter()createResume()DocumentCreatorFancyDocumentCreatorModernDocumentCreatorcreateLetter()createResume()FancyLetterModernResumeLetterResumeDocumentCreatorDocumentCreatorLetterResume

Uma fábrica é o local de uma classe concreta no código no qual os objetos são construídos. A intenção ao empregar o padrão é isolar a criação de objetos de seu uso e criar famílias de objetos relacionados sem ter que depender de suas classes concretas. [2] Isso permite que novos tipos derivados sejam introduzidos sem alteração no código que usa a classe base.

O uso desse padrão possibilita o intercâmbio de implementações concretas sem alterar o código que as utiliza, mesmo em tempo de execução. No entanto, o emprego desse padrão, como acontece com padrões de design semelhantes, pode resultar em complexidade desnecessária e trabalho extra na escrita inicial do código. Além disso, níveis mais altos de separação e abstração podem resultar em sistemas mais difíceis de depurar e manter.

