What is the Abstract Factory pattern, and what problem does it solve in software design?	Describe the structure of the Abstract Factory pattern. What are the main components involved?	What are the benefits of using the Abstract Factory pattern in software development? Provide some practical examples.
	Abstract Factory: This is an interface or an abstract class that declares a set of methods for creating abstract product objects. Each method in the abstract factory corresponds to the creation of a particular family of related objects. Concrete Factory: Concrete factories are implementations of the abstract factory interface. Each concrete factory is responsible for creating a specific family of related objects. It implements the methods declared in the abstract factory, thereby creating concrete product objects. Abstract Product: This is an interface or an abstract class that declares the interface for a type of product object created by the abstract factory. Each abstract product corresponds to a family of related objects. Concrete Product: Concrete products are the actual implementations of abstract product interfaces. These are the specific objects that are created by concrete factories. Each concrete product belongs to a particular family of related objects.	Encapsulation of object creation, Flexibility and extensibility Consistency and compatibility, Ease of testing, Promotes separation of concerns. Example: Game development: In a game development scenario, the Abstract Factory pattern can be used to create different types of game objects such as characters, weapons, and enemies. Each concrete factory could correspond to a different level or game scenario, providing objects tailored to that particular setting.
Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes. We can solve the problem of hard-code instantiation of products using interfaces.	To make an abstract factory, several interfaces are needed. It is recommended that each different product has its own distinct interface. Then, for each of these interfaces there exists a factory, outputs of our abstract factory, that implements an AbstractFactory interface.	The abstract factory is useful when we need to work with a plethora of different products, but dont want to depend on any specific class. The abstract factory will provide an interface to instantiate each respective instance.
An Abstract Factory is effectively a Factory, but instead of abstracting the subclass, it abstracts the family of classes altogether. In order to achieve this, it effectively becomes a factory of factories, each specialized in a given family of classes.	A estrutura do Abstract Factory inclui a interface Abstract Factory, que declara métodos para criar diferentes tipos de produtos abstratos. Seguem-se as fábricas concretas que implementam esta interface e produzem produtos concretos. Os produtos abstratos são as interfaces ou classes abstratas que definem os tipos de objetos a serem criados, e os produtos concretos são as implementações destas interfaces ou classes.	Os benefícios de usar o padrão Abstract Factory incluem a capacidade de trocar famílias de produtos facilmente, promover a consistência entre produtos, e reduzir a dependência do cliente em classes específicas de produtos. Por exemplo, em uma aplicação de interface de usuário, o Abstract Factory pode criar elementos de interface que sejam consistentes entre diferentes sistemas operativos. Uma fábrica pode produzir elementos para o Windows, enquanto outra produz elementos para o MacOS, permitindo que a mesma aplicação use diferentes temas de interface sem alterar o código do cliente.
O padrão Abstract Factory fornece uma maneira de encapsular um grupo de fábricas individuais que têm um tema comum sem especificar suas classes concretas. O problema que resolve é a necessidade de criar famílias de objetos relacionados sem especificar suas classes concretas.		

é um padrão creacional que permite produzir famílias de objectos relacionados sem especificar as suas classes concretas

produtos abstratos: declara interfaces para um set distinto mas relacionado de produtos que fazem uma família de produtos.
produtos concretos: várias implementações de produtos abstratos agrupados por variantes
abstract factory: interface que declara um set de métodos para criar cada produto abstrato
factories concretas: implementam métodos de criação da abstract factory. cada uma corresponde a uma variante específica de produtos e cria apenas essas variantes
cliente: comunica com os objetos e as interfaces abstratas

todos os produtos são compatíveis com os outros. evita tight coupling entre produtos concretos e o cliente; single responsability principle. open/closed principle.
usa-se quando o código precisa de trabalhar com várias famílias de produtos relacionados, por exemplo uma factory de mobilia com variantes de mobilia por exemplo moderna, vitoriana, vintage, etc e ter vários produtos como cadeiras, mesas, sofás, etc. em cada variante.

Os benefícios da abstract factory é fornecer a possibilidade da criação de uma família de objetos embora estes possam ser diferentes, por exemplo os produtos de um supermercado, existe a família de todos os produtos, mas entre esses pode haver fruta, bebidas, massas, etc..., embora existam semelhanças entre os produtos, por exemplo a maioria teria uma data de validade, existe também a diferença de tipo de produtos, sejam estes bebidas, comidas ou outros. Também podemos usar na criação de formas, embora todos tenham por exemplo o mesmo tipo de linha, a ligação entre vértices pode ser só no máximo com dois outros, existem as diferenças entre círculos, triângulos e retângulos ou quadrados, que têm numero de por exemplo arestas e vértices diferentes

A abstract factory é usada quando os objetos que queremos criar, embora estejam relacionados, não pertencem à mesma família, com isto é usada quando não queremos que estes objetos tenham necessariamente de pertencer a todas as classes existentes dos objetos

A abstract factory encapsula várias factories, estás factories depois, serão utilizadas conforme necessário, ou seja serão usadas as que são necessárias para um dado objeto

É mais complexo, mas é mais consistente e robutso

É uma fábrica que pode tratar de objetos de tipos relacionados, com diferentes interfaces.

Uma interface AbstractFactory, várias classes factory completas e várias interfaces para os produtos.

The Abstract Factory pattern is a design pattern in software engineering that provides an interface for creating families of related objects without specifying their concrete classes. It tackles the problem of creating objects that belong to distinct but related families, ensuring that they are compatible and adhere to a common theme or concept. By defining abstract interfaces for product families and concrete factories that produce objects within those families, the pattern allows for the creation of objects without directly referencing their concrete implementations. This promotes flexibility, decouples client code from specific object implementations, and facilitates the addition of new variants or configurations of objects without modifying existing code.

The Abstract Factory pattern involves organizing the creation of related objects into separate factories without specifying their concrete classes. Here's an example of the structure of the pattern:

Abstract Factory Interface/Class: This defines a set of methods for creating abstract product objects. It serves as an interface for creating families of related products without specifying their concrete classes.

Concrete Factory: Concrete implementations of the abstract factory interface. Each concrete factory is responsible for creating a specific family of related products. These factories produce concrete instances of products.

Abstract Product Interface/Class: This declares a set of methods that all concrete product classes must implement. It defines the interface for the products created by the abstract factory.

Concrete Product: Concrete implementations of the abstract product interface. Each concrete product represents a specific variation of a product created by the concrete factory.

The Abstract Factory pattern offers the following benefits in software development:

Encapsulation: Centralizing object creation logic promotes clean separation of concerns, making maintenance and modification easier.
Flexibility: Easily add new product variants or families without modifying existing client code, enhancing system flexibility and extensibility.
Consistency: Ensures that created objects belong to the same family and work together coherently, promoting consistency in design and implementation.
Dependency Injection: Supports dependency injection, facilitating loose coupling and improving testability.
Configurability: Allows for runtime configuration to produce different product variants based on parameters or preferences, enabling dynamic system adaptation.

Practical Examples:
Database Access: In a database access library, an abstract factory can define methods for creating database connection objects, query builders, and data access objects. Concrete factories can implement these methods to produce connections and components tailored for different database management systems (e.g., MySQL, PostgreSQL, MongoDB).

Game Development: In a game development

"Abstract Factory pattern" é um padrão criativo que permite criar famílias de objetos relacionados sem especificar a classe concreta deles. Este padrão visa resolver problemas de coupling de classes semelhantes, pe. venda de mobília. Neste caso nós precisamos de uma relação de estilo (modern, victorian,...). Neste caso, o uso de Abstract Factory pattern é aconselhado.

Uma classe factory abstrata que terá metodos e componentes semelhantes a todos os tipos da factory em construção.
Classes factory concretas que implementam os casos especificos a essa implementação

Classe produto abstrata que declara a funcionalidade de produtos disntintos mas relacionados entre si
Classes produtos concretas que representam as varias implementaçoes dos produtos abstratos, agrupados por variantes

Permite que tenhamos a certeza que os produtos que são criados são compatíveis. Desde que o código crie objetos a partir desta interface, não nos temos de preocupar quanto à criação de variantes erradas de um produto que não se compare aos produtos já criados na app.

Existem vários benefícios em utilizar o padrão Abstract Factory em desenvolvimento de software:

- Desacoplamento: O código do cliente não precisa conhecer as classes concretas dos objetos que está utilizando, apenas a interface da fábrica abstrata e dos produtos abstratos. Isto facilita a substituição de famílias inteiras de objetos sem modificar o código do cliente.

-Flexibilidade: Como diferentes fábricas podem ser facilmente intercambiáveis, é possível alternar entre diferentes implementações de famílias de objetos durante a execução do programa. Isto é útil em situações em que se deseja mudar a configuração do sistema sem recompilar o código.

-Manutenibilidade: Ao organizar a criação de objetos relacionados em fábricas, o padrão Abstract Factory promove um código mais organizado e de fácil manutenção. As mudanças nas famílias de objetos podem ser feitas de forma isolada nas fábricas correspondentes, sem afetar o restante do sistema.

Exemplo prático:
Num jogo de estratégia onde temos diferentes tipos de unidades militares, como soldados, tanques e aviões. Cada tipo de unidade pode ter diferentes variantes, como soldados de

A estrutura do padrão Abstract Factory consiste nos seguintes componentes principais:

-Abstract Factory : Define uma interface para criar objetos relacionados sem especificar suas classes concretas.
-Concrete Factory : Implementa a interface da fábrica abstrata para criar conjuntos específicos de objetos.
-Abstract Product : Define uma interface para um tipo de objeto que a fábrica pode criar.
-Concrete Product: Implementa a interface do produto abstrato e define um objeto específico criado pela fábrica.

Esses componentes trabalham juntos para permitir a criação de famílias de objetos relacionados de maneira flexível e desacoplada.

O padrão Abstract Factory é um padrão de design de software que visa fornecer uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas. Este padrão resolve o problema de criar conjuntos de objetos relacionados sem acoplar o código cliente a classes específicas, promovendo assim a flexibilidade e a manutenibilidade do código.

A estrutura deste padrão é:
- Produtos Abstratos
- Produtos Concretos
- Fábrica Abstrata
- Fábrica Concreta
- Cliente

Os benefícios resultantes deste padrão são:
- Os produtos criados pela fábrica são compatíveis.
- Permite a criação de código low coupling
- Permite, também, a utilização dos dois primeiros princípios do SOLID.

O padrão Abstract Factory fornece uma interface para criar famílias de objetos relacionados ou dependentes sem precisar de especificar as suas classes concretas.

A estrutura do Abstract Factory inclui a interface Abstract Factory, que declara métodos para criar diferentes tipos de produtos abstratos. Seguem-se as fábricas concretas que implementam esta interface e produzem produtos concretos. Os produtos abstratos são as interfaces ou classes abstratas que definem os tipos de objetos a serem criados, e os produtos concretos são as implementações destas interfaces ou classes.

Os benefícios de usar o padrão Abstract Factory incluem a capacidade de trocar famílias de produtos facilmente, promover a consistência entre produtos, e reduzir a dependência do cliente em classes específicas de produtos. Por exemplo, em uma aplicação de interface de usuário, o Abstract Factory pode criar elementos de interface que sejam consistentes entre diferentes sistemas operativos. Uma fábrica pode produzir elementos para o Windows, enquanto outra produz elementos para o MacOS, permitindo que a mesma aplicação use diferentes temas de interface sem alterar o código do cliente.

O padrão Abstract Factory fornece uma maneira de encapsular um grupo de fábricas individuais que têm um tema comum sem especificar suas classes concretas. O problema que resolve é a necessidade de criar famílias de objetos relacionados sem especificar suas classes concretas.

Abstract Factory permite criar famílias de objetos relacionados sem especificar a sua classe, através de uma hierarquia que abranja várias plataformas e a construção de vários produtos.

Resolve o problema de modificar código existente quando queremos adicionar novos produtos ou família de produtos ao porgrama.

Interface Abstract Factory: Esta interface define os métodos para criar cada produto dentro de uma família.

Factories Concretas: Estas classes implementam a interface Abstract Factory e fornecem a lógica para criar variantes específicas de produtos.

Interfaces de Produto: Essas interfaces definem as operações comuns numa família de produtos. As classes de produtos concretos implementam essas interfaces.

Uma maior flexibilidade, consistência garantida e desacoplamento aprimorado.

Resolve o problema de uma família de produtos relacionados como cadeiras, sofá e mesa de café que precisam de corresponder ao mesmo estilo de decoração.

The Abstract Factory pattern offers several benefits in software development, enhancing flexibility, maintainability, and extensibility. Here are some of the key advantages:

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. This pattern is particularly useful when a system needs to be configured with multiple families of related products, ensuring that the products from one family are compatible with the products from another family.

The main problem the Abstract Factory pattern solves in software design is the need to create families of related objects without specifying their concrete classes

The structure of the Abstract Factory pattern involves several components:

Abstract Products: These are the interfaces or abstract classes that define the types of products that can be created. Each product type has its own interface

Concrete Products: These are the concrete implementations of the abstract product interfaces. Each concrete product represents a specific type of product that can be created by the factory

Abstract Factory: This is an interface that declares a set of creation methods for all products that are part of the product family. These methods return abstract product types represented by the interfaces extracted previously

Concrete Factories: These are the classes that implement the abstract factory interface. Each concrete factory corresponds to a specific product variant and is responsible for creating a complete family of products

Isolation of Concrete Classes: The pattern encapsulates the creation of objects, isolating clients from implementation classes. This means that clients manipulate instances through their abstract interfaces, and product class names are isolated in the implementation of the concrete factory, not appearing in client code. This isolation reduces dependencies and makes the system more flexible and easier to change.

Easy Exchange of Product Families: The Abstract Factory pattern allows for the easy exchange of product families. Since the class of a concrete factory appears only once in an application, it's easy to change the concrete factory an application uses. This can enable the use of various product configurations simply by changing the concrete factory. Because an abstract factory creates a complete family of products, the whole product family changes at once, promoting consistency among products.

Promoting Consistency Among Products: When product objects in a family are designed to work together, it's important that an application use objects from only one family at a time. The Abstract Factory pattern makes it easy to enforce

Abstract factory pattern é um padrão criacional que permite a criação de famílias de objetos relacionados entre si sem que seja necessária a especificação de classes concretas.

Abstract Products - declaram interfaces para um conjunto de produtos distintos mas relacionados que fazem parte de uma família de produtos.
Concrete Products - são várias implementações de produtos abstratos, agrupados por variantes. Cada produto abstrato deve ser implementado em todas as variantes fornecidas.
Abstract Factory - declara um conjunto de métodos para criação de cada um dos produtos abstratos.
Concrete Factories - implementam métodos de criação fábrica abstratos. Cada fábrica concreta corresponde a uma variante específica de produtos e cria apenas aquelas variantes do produto.
Client - O cliente utiliza as interfaces fornecidas pelo AbstractFactory e AbstractProduct para interagir com os produtos sem conhecer suas classes concretas.

-> é possível ter a certeza que os produtos que você obtém de uma fábrica são compatíveis entre si;
-> evita um vínculo forte entre produtos concretos e o código cliente.
-> é um princípio de responsabilidade única, permitindo a extração do código de criação do produto para um lugar, fazendo o código ser de fácil manutenção;
-> princípio aberto/fechado, em que se pode introduzir novas variantes de produtos sem danificar o código pré-existente.

Considerando um sistema de management de veículos, onde existem vários tipos de veículo e vários tipos de componentes (rodas, motor, pneus, etc.). Com o padrão Abstract Factory, é possível criar uma fábrica específica para os veículos e uma fábrica para as peças, sendo que cada uma produz os componentes relevantes para cada tipo de veículo.

The abstract factory pattern in software engineering is a design pattern that provides a way to create families of related objects without imposing their concrete classes, by encapsulating a group of individual factories that have a common theme without specifying their concrete classes.

O padrão Abstract Factory em engenharia de software é um padrão que fornece uma maneira de criar famílias de objetos relacionados sem impor suas classes concretamente, encapsulando um grupo de fábricas individuais que possuem um tema comum sem especificar suas classes concretas.

The abstract factory pattern consists on defining an interface for creating generic classes.

The abstract factory allows you to create new instances of child classes without specifying which classes they are. These objects are related to eachother but this way we don't have to specify that relation (class)

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 allows a client to create objects without knowing their exact types, promoting flexibility and interchangeability of components within a system.
The main components of the Abstract Factory pattern are:
-> Abstract Factory Interface;
-> Abstract Product Interfaces;
-> Concrete Products;
-> Client;

O Abstract Factory define um Factory Method por produto. Cada Factory Method encapsula o novo operador e as classes de produtos concretas e específicas da plataforma. Cada "plataforma" é então modelada com uma classe derivada de Factory.

Os elementos principais são:
Abstract Factory, Concrete Factories, Abstract Products, Concrete Products & Clients.

Client - Creates Products, calls Abstract Factory to instantiate them
Abstract Factory (interface)
Factory - implements Abstract Factory, instantiates Product
Product (interface)
Product

The abstract factory has an interface that defines all the factories in the family. The client uses the Abstract Factory to call for all the other factories without needing to specify what kind of object they want.

The Abstract Factory pattern encapsulates the creation of related objects, providing benefits such as encapsulation, flexibility, consistency, and separation of concerns. It is commonly used in scenarios such as GUI toolkits, database access libraries, and logging frameworks to abstract away object creation details, allowing for easy swapping of object families and promoting maintainability and scalability.
Let's consider a scenario where we're developing a fantasy strategy game. In this game, there are two factions: Orcs and Elves. Each faction has its own unique units, such as Orc warriors and Elf archers.

To implement this using the Abstract Factory pattern:

We define an interface called Unit that represents the common behavior of all units in the game, such as attacking.
We create concrete classes for each type of unit, such as Orc and Elf, which implement the Unit interface.
We define an abstract factory interface called FactionFactory that declares a method for creating units.
We implement concrete factory classes, such as OrcFactory and ElfFactory, each of which implements the FactionFactory interface and provides specific implementations for creating

O padrão Abstract Factory ajuda a controlar as classes de objetos que uma aplicação cria. Como uma fábrica encapsula a responsabilidade e o processo de criação de objetos de produto, ela isola os clientes das classes de implementação. Os clientes manipulam instâncias através de suas interfaces abstratas.

It is useful when we need to create an object but don't know the exact class yet.

Making an abstract factory ensures that we can create objects that are related to eachothers even if we don't know what kind of relation we want yet (aka in the main script)

O abstract factory é um padrão de projeto criacional, que resolve o problema de criar famílias inteiras de produtos sem especificar suas classes concretas. O abstract factory define uma interface para criar todos os produtos distintos, mas deixa a criação real do produto para as classes concretas de fábrica.

Este padrão envolve a interação entre uma Abstract Factory que fornece uma interface para criar várias Abstract Products, e Concrete Factories que implementam essas interfaces para produzir Concrete Products. O Client interage com essas abstrações sem precisar se preocupar com as implementações concretas
--Verificar se a independência da plataforma e a criação de serviços são um problema;
--Planear a relação plataforma/produto;
--Definir uma interface para o método de fábrica por produto;
--Definir uma classe derivada para cada plataforma que encapsula todas as referências ao operador new;
--O cliente deve deixar de fazer referência ao operador new, utilizando o método de fábrica para criar os produtos

Os benneficios de usar Abstract Factory patter no desenvolvimento de software são:
Isolamento das classes concreta se facilita a troca de famílias de produtos,promove a consistência entre produtos.

Um exemplo onde podemos aplicar esse padrão é um sistema de Jogos em que o tema deles varia, as quais podemos ter diferente como espacial, medieval entre outros.
Um Abstract Factory pode ser usado para criar os componentes do jogo como por exemplo carros, barcos, naves , etc.Com isso o utilizador ao mudar o tema não necessita de alterar o código existente,mas apenas a fábrica concreta que cria os temáticos temas.

O padrão Abstract Factory oferece uma estrutura flexível para a criação de famílias de objetos relacionados em sistemas de software. Ao encapsular a lógica de criação de objetos dentro de fábricas concretas, ele promove a abstração de classes concretas e permite que os clientes trabalhem com interfaces e classes abstratas.

Além disso, a abstração proporcionada pelo padrão Abstract Factory promove o desacoplamento entre clientes e classes concretas, o que facilita a extensibilidade do sistema. Novas famílias de objetos podem ser introduzidas adicionando-se novas fábricas concretas e classes de produtos correspondentes, sem exigir grandes alterações no código existente.

O padrão em questão é um padrão de design criacional usado em POO para fornecer uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas(abstração). "fábrica de fábricas".

A estrutura é caracterizada pela interação entre esses componentes. A fábrica abstrata fornece uma interface para criar famílias de produtos relacionados, enquanto as fábricas concretas implementam essa interface para criar objetos de produtos concretos. Os clientes interagem com a fábrica abstrata para criar objetos de produto, sem precisar conhecer as classes específicas dos produtos com os quais estão trabalhando. Isto promove um acoplamento frouxo e permite uma manutenção mais fácil e extensibilidade do sistema.

Outro benefício é o suporte à consistência. O padrão garante que objetos criados por uma fábrica pertençam à mesma família e sejam compatíveis entre si. Isso ajuda a manter a integridade do sistema e evita erros que podem surgir ao utilizar objetos incompatíveis juntos.

Além disso, o padrão Abstract Factory facilita o teste de software, permitindo que os clientes usem implementações fictícias (mock) de fábricas e produtos. Isso simplifica o processo de isolamento e teste de diferentes partes do sistema de forma independente, contribuindo para a qualidade geral do software.

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

Abstract Products - Declare interfaces for a set of distinct but related products
Concrete Products - Are various implementatios of abstract products
Abstract Factory - interface that declares a set of methods for creating each of the abstract products
Concrete Factories - implement creation methods of the abstract factory
Client - can work with any concrete factory/product variant, as long as it communicates with their objects via abstract interfaces.

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.

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 allows a client to create objects without having to specify their concrete types, making the code more flexible and maintainable.

The Abstract Factory pattern solves the problem of creating families of related or dependent objects in a way that promotes loose coupling between the client code and the concrete classes of the objects being created. It also allows for easier switching between different implementations of object families without changing the client code, making the system more easily extendable and maintainable.

The abstract factory design pattern makes it easier to create groups of objects that are related to each other. This in turn makes the code more segmented and well organized.

O padrão Abstract Factory é um padrão que permite criar famílias de objetos que estão relacionados sem especificar as suas classes concretas, tornando-os mais fléxiveis e extensíveis.
Facilita a criação de produtos que devem funcionar em conjunto, mantendo a consistência entre objetos de diferentes familias.

The main components involved in the Abstract Factory pattern are:

Abstract Factory: This is an interface or abstract class that declares a set of methods for creating related or dependent objects. Different concrete implementations of the abstract factory interface can create different sets of objects.

Concrete Factory: These are concrete implementations of the Abstract Factory interface. Each concrete factory is responsible for creating a specific family of related objects.

Abstract Product: This is an interface or abstract class that declares the common interface for the products created by the factory. Each product family typically consists of multiple related interfaces or classes.

Concrete Product: These are the actual product classes that implement the Abstract Product interface or inherit from the abstract product class. Each concrete product belongs to a specific family created by a concrete factory.

Client: The client is the code that uses the

The abstract factory pattern is similar to the factory pattern except it adds another layer to it. It creates an abstract factory that creates other factories.

Abstract Factory: Interface que declara um conjunto de métodos para a criação de cada um dos Abstract Products.
Concrete Factory: Implementação de métodos criacionais da Abstract Factory. Cada Concrete Factory corresponde a uma variante especifica de um produto e cria apenas estas variantes do produto.
Abstract Product: Declara interface para um conjunto distinto mas relacionado de produtos que vao criar uma família de produtos.
Concrete Product: Várias implementações das interfaces dos Abstract Products. Cada Abstract Product deve implementar todas as suas variantes.
Client: Usa as interfaces declaradas pelas Abstract Factory e Abstract Product, permitindo que seja independente das classes concretas.

The Abstract Factory Design Pattern is a creational design pattern that allows you to create families of related or dependent objects without specifying their concrete classes. Provides an interface for creating related or dependent objects, without having to specify their concrete classes12.

The main components are:

Abstract Factory:
It serves as a high-level model that defines a set of rules for creating families of related objects without specifying their concrete classes.
Declares a series of methods, each responsible for creating a specific type of object.
Ensures that concrete factories follow a common interface, providing a consistent way to produce sets of related objects.
Concrete Factories:
Implement the rules specified by the abstract factory.
Contain the logic to create specific instances of objects within a family.
Multiple concrete factories may exist, each adapted to produce a distinct family of related objects.
Abstract Products:
They represent a family of related objects, defining a set of common methods or properties.
They act as abstract types or interfaces that all concrete products within a family must follow.

This pattern is well suited for programs that utilize a lot of different products with a lot of different attributes/specifications. If you have a new product that differs a lot from previous products, instead of changing those products' factories you can instead create a new factory for the new product.

Um dos benefícios da utilização deste padrão é a independencia da implementação, permite trocar famílias de produtos facilmente sem alterar o código que produtos usa.
Por exemplo numa troca de temas de aplicações, alterar o tema de uma aplicação pode ser feito facilmente com diferentes factories que produzem conjuntos de elementos de UI que se encaixam em cada tema.

O padrão Abstract Factory expande o Factory pattern, oferecendo uma maneira de criar famílias de objetos relacionados sem especificar suas classes concretas.
Resolve o problema de classes que possuem muitas variações

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

Abstract Factory pattern should be used 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.

Server para simplificar várias Factories para um mesmo tipo de objeto.
Resolve o problema da criação desnecessária (bloat) de Factories.

O Abstract Factory é um padrão de projeto criacional que permite produzir famílias de objetos relacionados sem ter que especificar suas classes concretas.
Resolve o problema da criação de conjuntos de objetos que, naturalmente, devem ser usados juntos, garantindo que os objetos criados sejam compatíveis entre si.

Produto: Esta é a interface ou classe abstrata que define as operações comuns a todos os objetos que podem ser criados pela fábrica.
Concrete Products:São as classes reais que herdam do Produto e implementam seus métodos.
Creator: Esta é a classe abstrata ou interface que declara o método da fábrica.
Factories:São subclasses da classe Criador que sobrepõem o método da fábrica para especificar o produto concreto que elas criam

Abstract Factory, Concrete Factories, Abstract Products, Concrete Products, Client

É uma interface com os métodos comum a todas as Factories seguido de uma ou mais componente(s) que implementam os métodos abstratos

A estrutura deste padrão é dividida em AbstractFactory (Interface que declara um conjunto de métodos para criar cada um dos objetos abstratos), o ConcreteFactory(Implementações específicas da interface AbstractFactory, cada uma destinada a criar objetos de uma certa família), o AbstractProduct (Interfaces para uma família de objetos de produtos que são relacionados), o ConcreteProduct (Implementações específicas das interfaces de produtos) e o Client(Usa as interfaces declaradas pela AbstractFactory e AbstractProduct).

Garante que todos os objetos dentro de uma família sejam criados usando a mesma fábrica, promovendo consistência em seu estilo e comportamento.

The first thing the Abstract Factory pattern suggests is to explicitly declare interfaces for each distinct product of the product family (e.g., chair, sofa or coffee table). Then you can make all variants of products follow those interfaces. For example, all chair variants can implement the Chair interface; all coffee table variants can implement the CoffeeTable interface, and so on.

Simplifica o código reduzindo o número de Factories para um mesmo objeto

Temos 3 figuras, e cada figura tem dois tipos de que pode ser
Em vez de criarmos duas factories para cada figura, podemos abstrair as factories, e simplesmente implementa-las para cada tipo de figura.

Ex: Rectangle pode ser Square e Non-Square
Factory Square e Factory Non-Square pode ser convertido numa interface:

```
Interface  Rect Factory{
   create shape;
}
```

Factory Square implements Rect Factory { ... }
Factory Non-Square implements Rect Factory { ... }

Benefícios:

Garante que os produtos obtidos são compatíveis entre si.
Esconde os detalhes de implementação dos produtos, expondo apenas suas interfaces.
Facilita a adição de novas famílias de produtos sem alterar o código cliente.

Exemplos práticos:

Imaginemos uma loja de mobília e pretendemos criar varias variantes de diferentes peças de mobília. Usamos a Abstract Factory para criar objetos que combinam entre si

Abstract Factory is a creation design the allows the production of families of related objects without specifying their concrete classes. It solves problems such as:
- Making an application independent of how its objects are created
- Making a class independent of how the required objects are created
- Create relations between families of objects or dependent objects

It contains abstract Factories, concrete Factories and Clients

-You can be sure that the products you're getting from a factory are compatible with each other, like a couch that matches the design of the chairs, or a charger that can charge your phone
-Follows the Single Responsibility principle and the Open/Closed principle, helping the code by being easier to support and introduce new variants without breaking the code
-You can avoid tight coupling between the concrete products and the client

The Abstract Factory pattern offers several benefits, including enhanced flexibility by enabling easy substitution of entire families of related objects like database types or UI components without altering client code; decoupling client code from specific implementations, thereby simplifying maintenance and extension of the codebase; ensuring consistency among objects produced by a factory, fostering visual consistency and compatibility within the system; facilitating scalability by simplifying the addition of new product families, such as supporting new document types in a document processing application; and aiding in testing by allowing for seamless substitution of mock or stub implementations, improving overall testability. Practical examples of Abstract Factory pattern usage encompass various domains, including GUI toolkits like Swing or Qt, where it can create diverse UI components based on different themes; in database access layers, where it facilitates the creation of database-related objects tailored to various vendors such as MySQL or Oracle; within document processing applications for generating different document formats like PDF or HTML; and in software interacting with the underlying operating system, offering OS-specific object creation, such as file system objects for Windows, macOS, and Linux. Through abstract factories, these examples demonstrate the pattern's utility in promoting

The abstract factory pattern is a design pattern that provides an interface for creating families of related or dependent objects without specifying their concrete classes. By encapsulating the creation process within factory interfaces, it reduces coupling between client code and specific implementations. This fosters flexibility, allowing for runtime switching between different families of products, facilitating adaptation to varied requirements or scaling by adding new product families. Additionally, it ensures consistency by guaranteeing that objects created by a factory are compatible with each other, which is particularly useful when creating multiple objects that need to work together or have dependencies.

The main components in the abstract factory pattern are : Abstract products, concrete products, the abstract factory and concrete factories. Abstract Products declare interfaces for a set of distinct but related products which make up a product family. Concrete Products are various implementations of abstract products, grouped by variants. Each abstract product must be implemented in all given variants. The Abstract Factory interface declares a set of methods for creating each of the abstract products. Concrete Factories implement creation methods of the abstract factory. Each concrete factory corresponds to a specific variant of products and creates only those product variants. Although concrete factories instantiate concrete products, signatures of their creation methods must return corresponding abstract products. This way the client code that uses a factory doesn't get coupled to the specific variant of the product it gets from a factory. The Client can work with any concrete factory/product variant, as long as it communicates with their objects via abstract interfaces.

É um padrão que nos permite criar familias de obejetos relacionados sem especificar cada objetos concretamente.
Criar uma abstração de objetos por uma familia de objetos ajuda na escalabilidade.

Temos uma abstratct factory a criar uma abstração de factorys concretas.

Garantir compatibilidade e os mesmo da factory pattern.

There is an abstract class interface that represents the concrete classes. The concrete classes are extented from that class and implement these interfaces. Then there is a creator method in the Factory Interface that creates a set of new objects related to the abstract class according to the provided input.

Isolates concrete classes
Makes exchanging product families easy
Promotes consistency among products

An example would be UI Framework Integration, an application that can switch between different UI frameworks. An Abstract Factory can create UI elements specific to the chosen framework, ensuring compatibility and consistent usage within the application.

The Factory pattern deals with creating objects of a single type, while the Abstract Factory pattern deals with creating objects of related types.

To sum up it provides an interface for creating families of related objects, without specifying concrete classes.

Uma vantagem é que existe um encapsulamento na criação de objetos e desacoplamento entre cliente e interfaces concretas.

Uma Fábrica Abstrata fornece uma interface que premite cirar objetos sem especificar a sua classe. Isto permite que o código do cliente trabalhe com famílias de objetos sem precisar conhecer as classes específicas sendo instanciadas.

Abstract and Concrete Factory. Abstract and Concrete Product. A Client.

A Abstract Factory premite a criação de diferentes Factorys para diferentes produtos.

Num jogo, podemos ter uma fábrica abstrata de personagens que produz diferentes tipos de personagens, como heróis, vilões e NPCs. Cada tipo de personagem é representado por uma interface abstrata, e as fábricas concretas criam instâncias concretas desses personagens.

O padrão de desenho Abstract Factory, é um padrão de design que fornece uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas. Ele é útil quando um sistema precisa ser independente de como seus objetos são criados, compostos e representados.

Este padrão resolve o problema de criar famílias de objetos relacionados sem especificar suas classes concretas. Isso permite que o sistema seja flexível e independente de como os objetos são criados, facilitando a substituição de famílias de objetos sem alterar o código cliente. Além disso, o Abstract Factory pode ser usado para criar diferentes versões de um sistema, como versões para diferentes plataformas ou versões com diferentes conjuntos de recursos.

A estrutura do padrão de projeto Abstract Factory é composta por quatro componentes principais:

1- AbstractFactory: Uma interface ou classe abstrata que define os métodos para criar famílias de objetos relacionados ou dependentes.
2- ConcreteFactory: Implementações concretas da interface ou classe abstrata da AbstractFactory. Cada ConcreteFactory cria uma família de objetos relacionados.
3- AbstractProduct: Uma interface ou classe abstrata que define o tipo de objetos que a fábrica pode criar.
4- ConcreteProduct: Implementações concretas da interface ou classe abstrata do AbstractProduct. Cada ConcreteProduct é um objeto dentro de uma família de objetos relacionados.

Os principais componentes envolvidos são a AbstractFactory, que é a interface para criar famílias de objetos, e as ConcreteFactory, que implementam essa interface para criar famílias específicas de objetos. Os AbstractProduct e ConcreteProduct representam os objetos que são criados pelas fábricas.

Abstract Factory: Declares an interface for creating a family of related objects without specifying their concrete classes.

Concrete Factory: Implements the Abstract Factory interface to create concrete product objects.

Abstract Product: Declares an interface for a type of product object.

Concrete Product: Represents the actual product objects that the factory creates. These classes implement the Abstract Product interface.

O padrão de projeto Abstract Factory oferece vários benefícios no desenvolvimento de software, especialmente em sistemas complexos onde a criação de objetos é intrincada ou onde a criação de objetos de diferentes famílias é necessária. Aqui estão alguns dos principais benefícios:

1- Abstração da Criação de Objetos: O padrão Abstract Factory abstrai o processo de criação de objetos, permitindo que o código cliente não precise saber os detalhes de como os objetos são criados. Isso torna o código mais limpo e mais fácil de manter.
2- Flexibilidade: Permite que o sistema seja flexível em relação às classes de objetos que podem ser criadas. Isso é especialmente útil em sistemas que podem precisar criar diferentes tipos de objetos em diferentes circunstâncias.
3- Desacoplamento: Reduz o acoplamento entre o código cliente e as classes de objetos concretos. Isso significa que o código cliente não precisa depender diretamente das classes concretas, o que facilita a modificação ou substituição de implementações sem afetar o código cliente.
4- Reutilização de Código: Como o padrão Abstract Factory encapsula a lógica de criação de objetos, essa lógica pode ser reutilizada em diferentes partes do sistema sem duplicação de código.

Abstract Factory lets you produce families of related objects without specifying their concrete classes. Having to change your code in a big way after new types of the same object are required.

Encapsulates object creation.
Promotes loose coupling.
Supports product family variations.

	Abstract Products, concrete products, abstract factory, concrete factories, client Abstract Products will declare an interface for a set of distinct but related products which make up a product family. Concrete Products are various implementations of abstract products, that are grouped by variants. Each abstract product must be implemented in all given variants. The Abstract Factory interface declares a set of methods that will be used to create each of the abstract products. Concrete Factories implement creation methods of the abstract factory. Each concrete factory corresponds to a specific variant of products and creates only those product variants.	
Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes. It falls under the creational design pattern and provides a way to encapsulate a group of factories that have a common link without highlighting the concrete classes.	Although concrete factories instantiate concrete products, signatures of their creation methods must return corresponding abstract products. This way the client code that uses a factory doesn't get coupled to the specific variant of the product it gets from a factory. The Client can work with any concrete factory/product variant, as long as it	With this pattern we can be sure that the products we're getting from a factory are compatible with each other, we avoid tight coupling between products and client code and we follow the Single Responsability Principle, because we can extract the product creation code into one place, making the code easier to support and the Open/Closed Principle since we can introduce new variants of products without breaking existing client code.
É responsável para resolver problema de criar famílias de objetos relacionados sem depender de suas classes concretas.	Essa e uma estrutura que permite criar famílias de objetos relacionados de forma flexível, sem depender das classes concretas dos produtos.	flexibilidade na criação de objetos, facilidade na adição de objetos
Abstract Factory pattern é um padrão de desenho que propoem uma abstração à interface de criação de objetos. Ou seja, não precisamos de especificar diretamente a implementação do "Factory".	Abstract Factory - declara os métodos abstratos de criação. Classe utilizada pelos "clientes" Concrete Factories - implementação desses métodos dependendo do tipo de "Factory" em especifico	Uma maior flexibilidade, fornece mais um layer de abstração do lado da construção. Permitindo ao cliente ter essa abstração e não saber qual o "Factory" mas sim o produto retornado ser o que ele pediu.
O Abstract Factory pattern é um padrão de design creacional que permite a criação de objetos bastante similares sem a especificação das suas classes concretas. Desta forma ajuda-nos a encapsular um grupo de Factories que tenham características em comum, sem indicar as classes específicas.	Os componentes principais deste padrão são vários produtos abstratos, que implementam interfaces para produtos diferentes mas relacionados, as quais são implementadas por classes concretas de produtos, cada uma representando uma variante. A estrutura precisa também de uma interface abstrata que indique os métodos de construção dos produtos abstratos. Esta é implementada por várias Factories.	O benefício deste padrão é a criação de objetos bastante similares sem a especificação das suas classes. Em softwares de produtos com várias variâncias, porém similares, como em softwares de lojas de roupa, é bastante util.
Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes.	Abstract Products -> declare interfaces for a set of distinct but related products which make up a product family. Concrete Products -> are various implementations of abstract products, grouped by variants. Abstract Factory -> interface declares a set of methods for creating each of the abstract products. Concrete Factories -> implement creation methods of the abstract factory. Each concrete factory corresponds to a specific variant of products and creates only those product variants	Give us another layer of abstraction in construction phase.

É um padrão que permite instanciar familias de objetos relacionados sem especificar a sua classe concreta

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.

A abstract Factory pattern visa criar familias de objetos que estão relacionados sem expecificar a sua classe.
Quando usamos objetos que possuem diversas formas possiveis, na sua criação podemos criar a instancia errada do objeto, por isso usamos uma interface para cada variação de familia de produtos.

- interface de produtos abstrata: são declaradas interfaces para familias de objetos relacionados entre si
- Produtos concretos: implementações dos produtos abstratos
- abstract factory: declara metodos para criar cada uma das familias de objetos
- concrete factories: Factories que implementam os metodos da abstract factory. Cada concrete factory corresponde a cada uma das familias de objetos e apenas cria estes

The main components involved in Abstract Factory design are

Abstract Factory
The Abstract Factory is a foundational interface responsible for declaring a suite of methods to create various abstract products. It serves as a template for factories, guiding the production of product families without dictating their concrete implementations. This abstraction ensures that the creation process is uniform across different environments, promoting consistency and interchangeability of product families.

Concrete Factory
Concrete Factories are specific implementations of the Abstract Factory interface. Each Concrete Factory is tasked with creating a set of products that belong to a single variant, thereby specializing in the production of a particular product family. These factories instantiate and return products in accordance with the abstract product interfaces, thus maintaining a separation between product creation and product use.

Abstract Product
The Abstract Product outlines a standard

Os principais componentes são os objetos abstratos que possuem um familia. Os grupos de objetos abstratos que são agrupados dependento do seu tipo de instancia.A factory que é o interface que trata de criar os objetos de uma determinada instancia.

- podes garantir que todos os objetos criados por uma fabrica são compativeis entre si
- todas as vantagens da factory

Benefits
Promotes Consistency Among Products: Ensures that related products designed to work together are compatible and enforces consistency among them.

Decouples Client Code from Concrete Implementations: Clients interact with products through abstract interfaces, reducing the dependency on concrete implementations and making it easier to introduce new variants of products without changing the client code.

Supports Product Families and Variants: Ideal for systems that need to manage or extend multiple families of products or where products come in several variants but must follow a common theme or interface.

Enhances Scalability and Flexibility: New product families can be introduced with ease by defining new specific factory classes, without altering existing client code, making the system more scalable and flexible.

Facilitates Open/Closed Principle: The system can introduce new variants of products or new product families by extending existing factories, thus the system is open for extension but closed for modification.

Examples:

Nos ao usarmos a "Abstract Factory" temos a certeza que os que resultam da criação sao compativeis entre si. Evitamos assim "tight coupling" entre os produtos expecificos e o "main". E mantem em comum as vantagens do Factory Method seguindo o "Single Responsibility Principle" e o "Open/Closed Principle".

		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.
Abstract Factory is a creational design pattern that lets you produce families of related objects without specifying their concrete classes. It solves the problem that happens when someone tries to update their classes and must change part of the core code.	1 - Abstract Products declare interfaces for a set of distinct but related products which make up a product family. 2 - Concrete Products are various implementations of abstract products, grouped by variants. Each abstract product (chair/sofa) must be implemented in all given variants (Victorian/Modern). 3 - The Abstract Factory interface declares a set of methods for creating each of the abstract products. 4 - Concrete Factories implement creation methods of the abstract factory. Each concrete factory corresponds to a specific variant of products and creates only those product variants. 5 - Although concrete factories instantiate concrete products, signatures of their creation methods must return corresponding abstract products. This way the client code that uses a factory doesn't get coupled to the specific variant of the product it gets from a factory. The Client can work with any concrete factory/product variant, as long as it communicates with their objects via abstract interfaces.	Abstract Factory: The GUIFactory interface declares a set of methods that create different abstract products, such as createButton() and createCheckbox(). This interface represents the Abstract Factory, which defines the contract for creating families of related products. Concrete Factories: The WinFactory and MacFactory classes are Concrete Factories. They implement the GUIFactory interface and provide the specific implementations for creating Windows and macOS products, respectively. Abstract Products: The Button and Checkbox interfaces define the base interfaces for different product families. Each product family has its own interface, and variants of the product families implement these interfaces.
É necessário que encapsule as suas dependências. Este encapsulamento deve ser previsto no desenho do software. Com o intuito de criar uma interface para criar famílias de objetos relacionados sem especificar a sua classe, através de uma hierarquia que abranja várias plataformas e a construção de vários produtos, evitando a utilização do operador new. Assim cria-se uma interface por produto, definindo para cada uma uma subclasse para cada plataforma e um método abstract factory com a interface e uma subclasse para cada plataforma, definindo um método para cada objeto, sendo o devolvido o correspondente a essa plataforma.	O padrão Abstract Factory é um padrão de design que fornece uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas. A estrutura desse padrão envolve a definição de interfaces para fábricas abstratas, que por sua vez têm métodos para criar objetos distintos dentro de uma família. Os principais componentes incluem as interfaces abstratas para produtos e fábricas, classes concretas que implementam essas interfaces, e um cliente que utiliza essas fábricas para criar objetos sem se preocupar com as classes específicas. Esse padrão promove a modularidade, flexibilidade e facilita a substituição de famílias de objetos sem alterar o código do cliente.	O padrão Abstract Factory traz muitos benefícios, como a criação de famílias de objetos coerentes e compatíveis que seguem um tema ou configuração específica. Também isola classes concretas do código do cliente, tornando o código mais abstrato e desacoplado. Um exemplo seria fazer uma fábrica de mobiliuário, um de estilo moderno e outro de estilo victoriano.
It is a creational design pattern that lets you produce families of related objects without specifying their concrete classes. It solves the problem of having diverse types and regarding each type having possible variants of those types which can provide problems in code structuration.	The main components involved are: 1 -> Abstract Products - declare interfaces for a set of distinct but related products 2 -> Concrete Products - various implementations of abstract products, grouped by variants 3 -> Abstract Factory - declares a set of methods for creating each of the abstract products. 4 -> Concrete Factories - implement creation methods of the abstract factory	A restaurant can have three different types of dishes (ex: meat, fish and dessert) and from those types there must be variants of those types (ex: meat can be pork, chicken, etc.). It is necessary to create interfaces for every type of dish and each variant must implement the direct interface. It is then necessary to create an abstract factory.

Abstract Factory allows you to create an intrinsic relation between objects that share the same characteristics of an abstract concept. It solves the problem of categorizing objects that are different share abstract similarities. And you also don't need to change existing code when adding new products or families.

Abstract factory: This is an interface or abstract class defining methods for creating abstract product objects.
Concrete factory: These are concrete implementations of the abstract factory interface.
Abstract product: this is an interface or abstract class defining the interface for a family of related products. It doesn't necessarily define the implementation details but declares the methods that concrete products must implement.
Concrete product: These are concrete implementations of the Abstract product interface. Each concrete product corresponds to a specific variant or type of product in the family of products.
Client: This is the code that uses the Abstract Factory and the products it creates. It remains independent of the concrete classes of products and works with the products only through their abstract interfaces.

The Abstract Factory pattern is like a chair-making factory that can create different types of chairs. You ask the factory for a chair, but you don't know exactly what kind it will be. It could be a wooden chair, a plastic chair, or a metal chair. The factory decides based on what you need. This is helpful because if you want to change the type of chair later, you can do it without changing how you ask the factory. It keeps things organized and makes it easy to add new types of chairs later on.

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

What problems does it solve?
It's particularly useful in scenarios where different families of related objects need to be created interchangeably or where object creation is complex and needs to be abstracted away from client code.

The Abstract Factory pattern consists of several key components that work together to facilitate the creation of families of related or dependent objects while keeping the client code decoupled from the actual implementations. The main components involved:
- Abstract Products
- Concrete products
- Abstract Factory
- Concrete Factories
- The client

- 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.
- Abstraction

Example: In a web application, you may need to generate different types of reports (for example PDF, Excel) based on user preferences. By using a factory pattern, you can abstract away the details of report generation and provide a common interface for creating reports. This allows clients to interact with the factory interface without needing to know the specifics of how each type of report is generated, promoting abstraction and simplifying client code.
- Flexibility

Example: Consider a system where different payment gateways (for example PayPal) need to be integrated to process payments. By using a factory pattern, you can create a PaymentGatewayFactory that dynamically selects the appropriate payment gateway based on user preferences or system configuration. This allows the system to adapt to different payment gateway requirements without requiring

Abstract Factory Pattern is a way of organizing how you create groups of things that are related to each other. It provides a set of rules or instructions that let you create different types of things without knowing exactly what those things are.

Abstract Factory - high level blueprint
Concrete Factories - implements rules specified by the abstract factory
Abstract product - family related products

Possibility to create another factory to simplify the creation of products

Abstract Factory: é um padrão criacional que permite produzir famílias de objetos relacionados sem especificar as suas classes concretas. Problema: Ao criar novos objetos ou famílias de objetos, não queremos alterar o código existente.	Existe uma interface geral que contém métodos para a criação de produtos de certo tipo. Existem várias implementações dessa interface que sobrescrevem os seus métodos e lá instanciam a sua família de objetos. São declaradas as interfaces do grupo de objetos que estão relacionados, posteriormente, são criados novos objetos que podem ser agrupados nos grupos	Encapsula objetos com lógica criacional. Promove baixo acoplamento. Garante compatibilidade entre as famílias de objetos e o objeto instanciado. Por exemplo se nós tivermos um conjunto de objetos que se relaciona : camisola T-shirt camisa; Existem várias variantes, como por exemplo a cor de cada uma delas: amarelo, azul e verde. Não é necessário que o cliente saiba qual o tipo de cor para definir o objeto.
Encapsulate the way to create objects, diferent ways to create the same object (ex: various types of sofa).	An interface (let's call it Factory) contains some object creation methods, that will be shared by many factory classes that implement it. Each factory overrides the methods and returns the objects that are superseded by the interface.	When there are multiples factories that have common properties, abstracting all them in a single interface allows to encapsulate the object creation logic. There is no need to worry about each factory's creational methods, because all of them are factories that create similar products between them.
É um design pattern que permite a criação de famílias de objetos relacionados ou dependentes sem especificar as suas classes concretas. Resolve o problema de criar famílias de produtos e garantir que apenas produtos da mesma família sejam usados juntos. É útil quando o software precisa ser independente de como os objetos são criados e compostos.	FábricaAbstrata: Classe abstrata ou interface que decalra métodos de criação do objeto do tipo ProdutoAbstrato. ProdutoAbstrato: Declara os métodos que são implementados por classes do tipo ProdutoConcreto FábricaConcreta: Estende ou implementa a FabricaAbstrata. Cria internamente um objeto do tipo ProdutoConcreto, mas esse objeto é retornado como um ProdutoAbstrato ProdutoConcreto: Estende ou implementa a classe ProdutoAbstrato. Nessa classe são implementados os métodos declarados em ProdutoAbstrato. Essa é a classe que faz uma instância concreta ser criada.	O padrão Abstract Factory no desenvolvimento de software oferece vários benefícios: Isolamento de classes concretas: ajuda a controlar as classes de objetos criados ao encapsular a responsabilidade e o processo de criação de objetos de produto, isolando os clientes das classes de implementação Promoção da Consistência entre Produtos: garante que uma aplicação utilize objetos de apenas uma família de cada vez, garantindo consistência entre objetos de produto relacionados projetados para trabalhar em conjunto. Troca Fácil de Famílias de Produtos: permite alterações fáceis na fábrica concreta utilizada, possibilitando várias configurações de produtos ao mudar a fábrica concreta, o que por sua vez altera toda a família de produtos de uma só vez. Cenários como Implementação Multi-Cloud, Fornecedores de Segurança e Módulos de Conformidade em aplicações de computação em nuvem e segurança são exemplos práticos válidos
O Abstract Factory tem objetivo de cria famílias de objetos relacionados ou dependentes sem especificar suas classes concretas; Resolve o problema de querer-se trabalhar com diversas famílias de produtos relacionados, mas não quer depender de classes concretas desses produtos.	Os componentes principais sao: Abstract Product; Abstract Factory; Concrete Factory; Concrete Product; Client Code;	É útil ao nosso código quando precisamos criar várias famílias desses objetos - troca fácil de famílias de produtos. Exemplo, as camadas de abstração de banco de dados.
É um padrão onde há um conjunto de vários tipos de fábricas diferentes, todas implementações concretas de uma interface ou classe abstrata, mas que fabricam os mesmos tipos de objetos. Resolve o problema de termos maneiras diferentes de fabricar um mesmo objeto.	Para além das fábricas em si, há a interface comum que todas as fábricas implementam, para além das classes dos objetos a fabricar.	Se tiver famílias de objetos que podem ser classificados por duas propriedades distintas. Por exemplo, mobília: pode ser classificada pelo tipo: "cadeira", "mesa", "cama"; mas também por estilo: vitoriano, clássico, arte nova. Como a mobília de um mesmo tipo tem muitos elementos em comum, posso ter uma fábrica para cada estilo, que cria cada tipo de mobília nesse estilo. "CreateVictorianFurnitire(String type)".

É um padrão de projeto criacional que fornece uma interface para criar famílias de objetos relacionados ou dependentes sem especificar as suas classes concretas. Ele resolve o problema de como criar objetos que pertencem a uma mesma família, mas sem acoplar o código cliente às classes concretas desses objetos. O principal objetivo é encapsular a criação de objetos relacionados em uma hierarquia de fábricas. Ele define uma interface abstrata para criar todos os produtos, mas deixa a criação real dos produtos para as subclasses concretas da fábrica. Cada fábrica concreta corresponde a uma variante específica dos produtos e cria apenas esses produtos.

- Interface Abstract Factory
- Fábricas Concretas
- Produtos Abstratos
- Produtos Concretos

Isola as classes concretas: o código cliente não conhece as classes concretas, apenas trabalha com suas interfaces abstratas.
Facilita a troca de famílias de produtos: Você pode trocar toda uma família de produtos simplesmente trocando a fábrica concreta.
Promove a consistência entre produtos: Quando os objetos de uma família são projetados para trabalharem juntos, é importante que uma aplicação use objetos de apenas uma família de cada vez.