**What is the Abstract Factory Pattern?**

The 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. This helps you keep everything organized and lets you switch between different types easily, following the same set of rules.

* Abstract Factory pattern is almost similar to [Factory Pattern](https://www.geeksforgeeks.org/design-patterns-set-2-factory-method/) and is considered as another layer of abstraction over factory pattern.
* Abstract Factory patterns work around a super-factory which creates other factories.
* Abstract factory pattern implementation provides us with a framework that allows us to create objects that follow a general pattern.
* So at runtime, the abstract factory is coupled with any desired concrete factory which can create objects of the desired type.

**Advantages of using Abstract Factory Pattern**

This pattern is particularly useful when the client doesn’t know exactly what type to create.

* **Isolation of concrete classes:**
  + The Abstract Factory pattern helps you control the classes of objects that an application creates.
  + Because a factory encapsulates the responsibility and the process of creating product objects, it isolates clients from implementation classes.
  + Clients manipulate instances through their abstract interfaces. Product class names are isolated in the implementation of the concrete factory; they do not appear in client code.
* **Exchanging Product Families easily:**
  + The class of a concrete factory appears only once in an application, that is where it’s instantiated.
  + This makes it easy to change the concrete factory an application uses.
  + It can use 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:**
  + 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. AbstractFactory makes this easy to enforce.

**Disadvantages of using Abstract Factory Pattern**

* **Complexity:**
  + Abstract Factory can introduce additional complexity to the codebase.
  + Having multiple factories and abstract product interfaces may be overkill for simpler projects.
* **Rigidity with New Product Types:**
  + Adding new product types (classes) to the system can be challenging.
  + You might need to modify not just the concrete factories but also the abstract factory interface, potentially impacting existing code.
* **Increased Number of Classes:**
  + As you introduce more abstract factories and product families, the number of classes in your system can grow rapidly.
  + This can make the code harder to manage and understand, particularly for smaller projects.
* **Dependency Inversion Principle Violation:**
  + In some cases, the Abstract Factory pattern may lead to a violation of the Dependency Inversion Principle, especially if client code directly depends on concrete factory implementations rather than the abstract interfaces.
* **Limited Extensibility:**
  + Extending the abstract factory hierarchy or introducing new product families might require modifications to multiple parts of the code, potentially leading to cascading changes and making the system less extensible.
* **Not Ideal for Simple Systems:**
  + The Abstract Factory pattern may be overkill for smaller, less complex systems where the overhead of defining abstract factories and products outweighs the benefits of the pattern.

**When to use Abstract Factory Pattern**

* **Multiple families of related products:** When your system needs to be configured with multiple families of related products, and you want to ensure that the products from one family are compatible with the products from another family.
* **Flexibility and extensibility:** If you need to allow for variations or extensions in the products or their families, the Abstract Factory pattern provides a way to introduce new product variants without modifying existing client code.
* **Encapsulation of creation logic:** The pattern encapsulates the creation of objects, making it easier to change or extend the creation process without affecting the client code.
* **Consistency across product families:** If you want to enforce consistency among the products created by different factories, the Abstract Factory pattern can help maintain a uniform interface.