Abstract factory design pattern:  
**Other name (if any)**

An Abstract Factory Pattern is also known as **Kit.**

**What it does**

This is a creational design pattern. It is a factory of factory. It creates family of classes.

**Where to use**

1. It is used when we can group together different type of classes in different groups.  
2. It's helpful in scenarios where different parts of your program need different sets of related objects.  
3. **When you want to ensure consistency among related objects.**  
4. **When you need to support multiple platforms or variations.**

**Steps**

1. Interface Base Class: create an interface for base class  
2. Implemented Child Class: It will implements the base class. Here the main code will be written for that specific class.  
3. Interface Factory Class: There will be a function.  
4. Implement Child Factory Class: Here the function in the interface factory class will return the specific implemented child class which will be part of that specific Group of implemented child factory class based on the condition.  
5. Abstract factory class: There will be a function which will return the specific implemented child factory class based on the condition.  
6. Client code: It will create object using abstract factory class. Pass the value to the function of the abstract factory class. Then call the function of the specific implemented child class.

**Advantages**

1. It organizes object creation.  
2. It ensures consistency.  
3. It's easy to switch between different versions or setups.  
4. You can add new types of objects without changing existing code.  
5. It makes code easier to manage.  
6. You can easily switch between different groups of objects as needed.

**Disadvantages**

1. Implementing the pattern can add complexity to the codebase, especially in simpler projects.  
2. It may introduce additional classes and interfaces, increasing the size of the code.  
3. It might be challenging for developers new to the pattern to understand and use effectively.  
4. The pattern may lead to a proliferation of factory classes, making maintenance harder.  
5. Excessive use of abstraction can obscure the logic of object creation, making debugging difficult.  
6. Introducing factories may add unnecessary dependencies between components.  
7. Using the pattern in simple scenarios can violate the principle of keeping things simple.

**Code**

Tutorialspoint

**Difference with similar pattern**

**Factory Pattern**: It's a simpler version of Abstract Factory, focusing on creating individual objects rather than families of related objects.  
Maybe prototype but [[later]]

**Diagram**

Tutorialspoint