Abstract Factory Design Pattern

Abstract Factory is a creational design pattern. It allows to create groups of objects that are related to each other without specifying their concrete class types.

For the example of using that pattern I created a small GUI library that is capable of producing GUI elements for the specified platform. For the user, interface of the library is pretty simple: You just need to specify the platform that you are currently using (which also could be easily extracted from the platform you are running the code on), and use factory methods to build the GUI:

```
GUIFactory factory = GUI.createFactory(OS.Windows);
Button button = factory.createButton("Hello");
```

Abstract Factory suggests creating interfaces for each distinct GUI element. But in my case I used an abstract class for that that implements all the required interfaces instead of implementing them in each distinct platform class. Each distinct platform class (LinuxButton, WindowsButton) extends abstract class (Button) and overrides the render method to be able to correctly render itself in the given platform.

Also, for each platform (Linux, Windows etc.) there is a factory class that implements the GUIFactory interface which provides function for creating GUI elements.

As the code snippet above shows, there is a static GUI class that provides an interface for creating a factory for the given platform.

The main drawback is that the code structure looks complicated since there will be plenty of classes and interfaces when the library will grow.

On the other hand, the pattern gives you a lot of benefits: factory products are compatible with each other, there is no tough coupling between products and clients code and the code is pretty scalable for introducing new variants of products without need of changing the existing clients code.