**Abstract Factory**

Motivation

Abstract Factory design pattern defines a framework, which produces objects that follow a general pattern and at runtime this factory is paired with any concrete factory to produce objects that follow the pattern. In other words, the Abstract Factory is a super-factory which creates other factories (Factory of factories).

Intent

* Abstract Factory offers the interface for creating a family of related objects, without explicitly specifying their classes.

Implementation

The classes that participate to the Abstract Factory pattern are:

**-AbstractFactory** - declares a interface for operations that create abstract products.

**-ConcreteFactory** - implements operations to create concrete products.

**-AbstractProduct** - declares an interface for a type of product objects.

**-Product** - defines a product to be created by the corresponding ConcreteFactory; it implements the AbstractProduct interface.

**-Client** - uses the interfaces declared by the AbstractFactory and AbstractProduct classes.

The AbstractFactory class is the one that determines the actual type of the concrete object and creates it, but it returns an abstract pointer to the concrete object just created. This determines the behavior of the client that asks the factory to create an object of a certain abstract type and to return the abstract pointer to it, keeping the client from knowing anything about the actual creation of the object.

The fact that the factory returns an abstract pointer to the created object means that the client doesn't have knowledge of the object's type. This implies that there is no need for including any class declarations relating to the concrete type, the client dealing at all times with the abstract type. The objects of the concrete type, created by the factory, are accessed by the client only through the abstract interface.

The second implication of this way of creating objects is that when the adding new concrete types is needed, all we have to do is modify the client code and make it use a different factory, which is far easier than instantiating a new type, which requires changing the code wherever a new object is created.