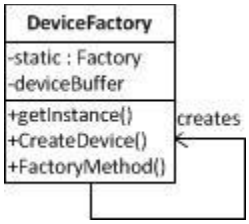
3 Abstract Factory Pattern [Gamma et al]

Abstract factory faciltates by providing a technique to encapsulate the of group of families of products termed as factories which produce a set of products with common interfaces. Neumerous examples of Abstract Factory Pattern are quoted in LogicalAJ, AspectJ, JUnit which implies the important role in dealing with group of objects. It can also be termed as factory of factories because it provides an abstract factory interface [1].

3.1 Factories as singleton [18]

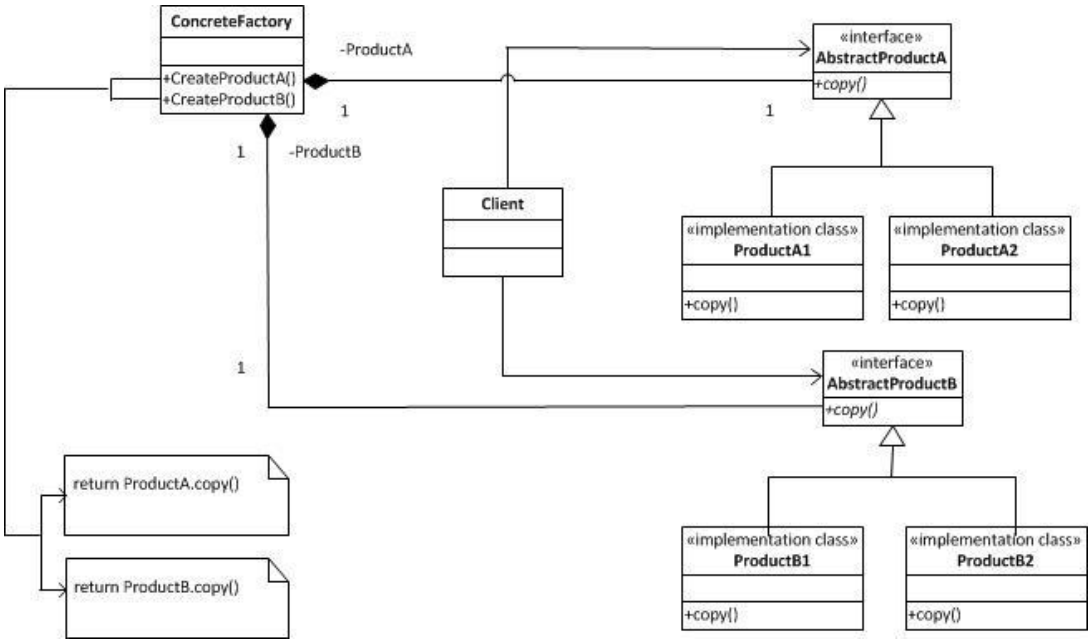
In this variant factories are considered as resource manager just like printer manager which recieves multiple requests but are able to entertain one at a time. [18] discuss an example where factory is used as a singleton.



3.1 UML Diagram and source code of factories as singleton [18]

3.2 Pluggable Factories [21] [22]

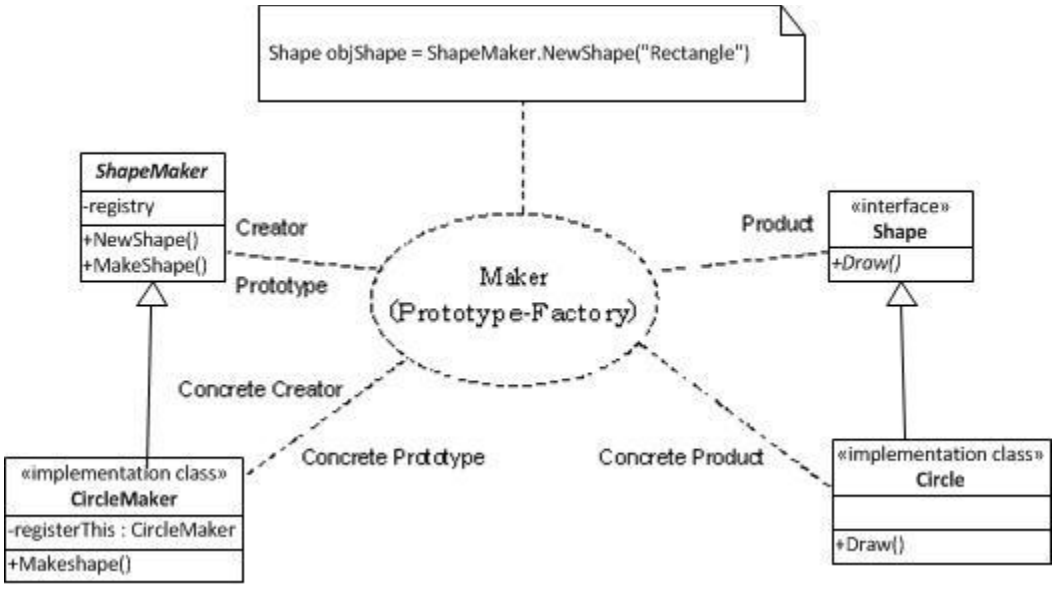
In this variant new functionality to an existing code is termed as "plugins". Without changing the common code and dynamically loading of factory classes at run time is the basic idea of this implementation. This method advocates its use by removing the dependency on subclasses (concrete factories). Therefore a dynamic loading mechanism is introducted which auto detects any subclasses that inherit from it. The variant is a based on compound implementation of both Abstract factory and Prototype pattern [21]. Abstract factory is used to defer the instantiation of subclasses and the prototype is applied to create the prototypical instance forwarding the responsibility to the subclasses that they voulenterily create themseleves.



3.2 UML Diagram of pluggable factories quoted in [21]

3.3 Industrial Strength Pluggable Factories [23]

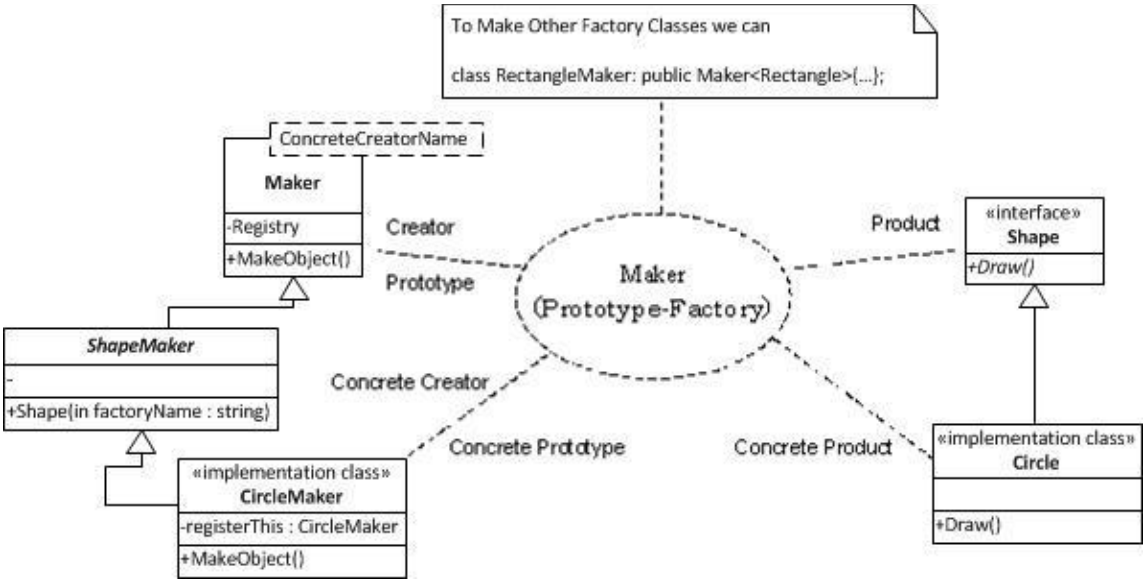
This variant is an extension of the solution provided by Vlissides [22]. Which adopts the pluggability mechanism i.e dynamically adding subclasses to the abstract factory heirarcies. It introduces a container which the author [23] refer as "Maker" object that hold the responsbility of registering its subclasses. This is also a compound variant of abstract factory and prototype pattern.



3.3 UML Diagram of industrial strength pluggable factories [23]

3.4 Generic Factories [23]

This variant facilities Maker Classes to be independent of making decision of creating iteself i.e. any class in an application can be used as plugin with the use of language “generics” feature. This variant uses template class paramterization to create objects. This benefits greatly and reduce the number of classes which was a higlighted problem in the discussed variants of abstract factory.



3.4 UML Diagram of generic factories [23]