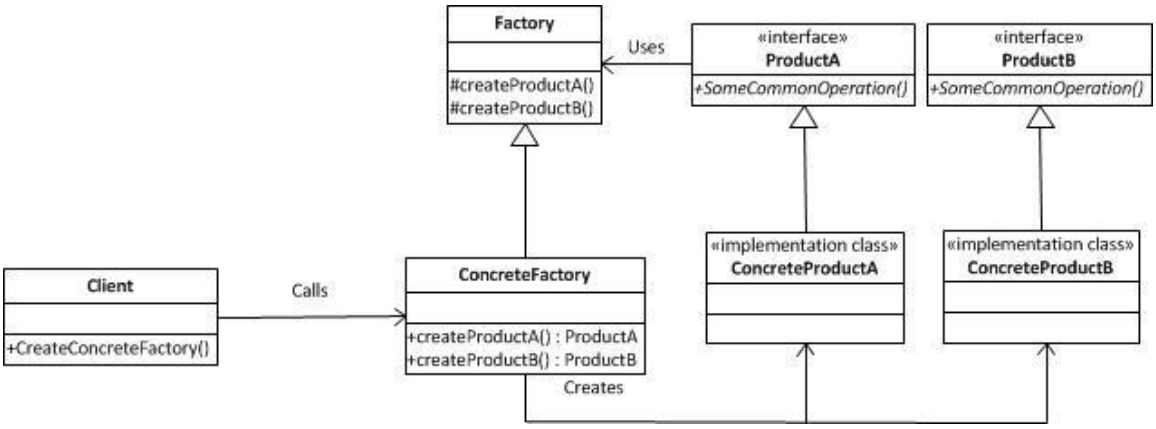
2. Factory Method Pattern [Gamma et al]

This pattern is typically implemented as virtual methods, sometimes it is also known as virtual constructor [1] . Objects are represented as abstract which is type cased when required just like how libraries work. Factory method hides the instantiation logic with a method that is overided across all the subclasses. The intent is to create similar kind of objects through factory methods. This pattern has been reported high frequency of usage in applications like Microsoft COM, MFC, J2EE etc. [16] discusses a section about the occurance of patterns in academic literature, forums, web in general and gives a ranking based on disscussion made by the research community. [18] discusses a practical example of factory method pattern. Following are the implmentation variants:

2.1 Different Product types inside factory class [20]

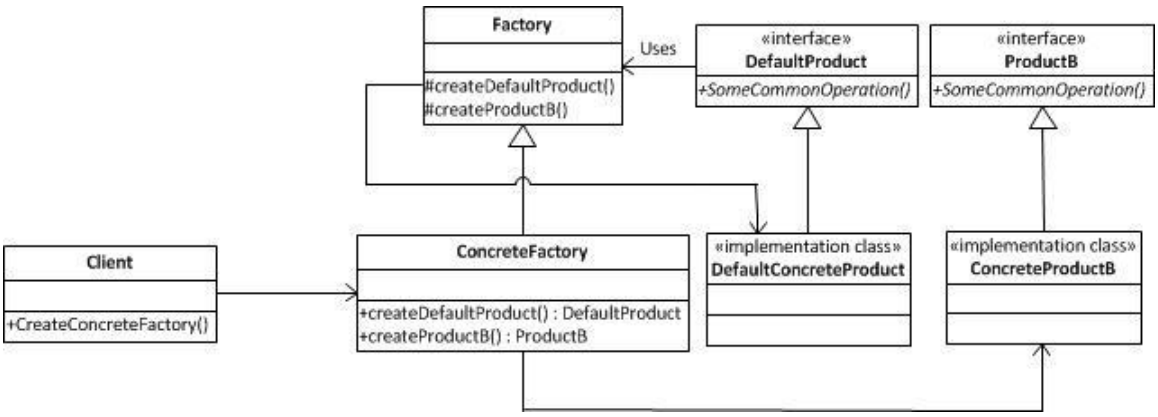
This variant modifies the idea provided by the basic implementation in which a single factory method is used to create same kind of products. This idea implements multiple kind of products to be created or accessed using different factory methods. The structure of the pattern is the same but the implementation is modified to support more than one type of product [20]. This variant could be an alternative of abstract factory.



2.1. UML Diagram of multiple product types inside factory class [20]

2.2 Default Product Implementation [20]

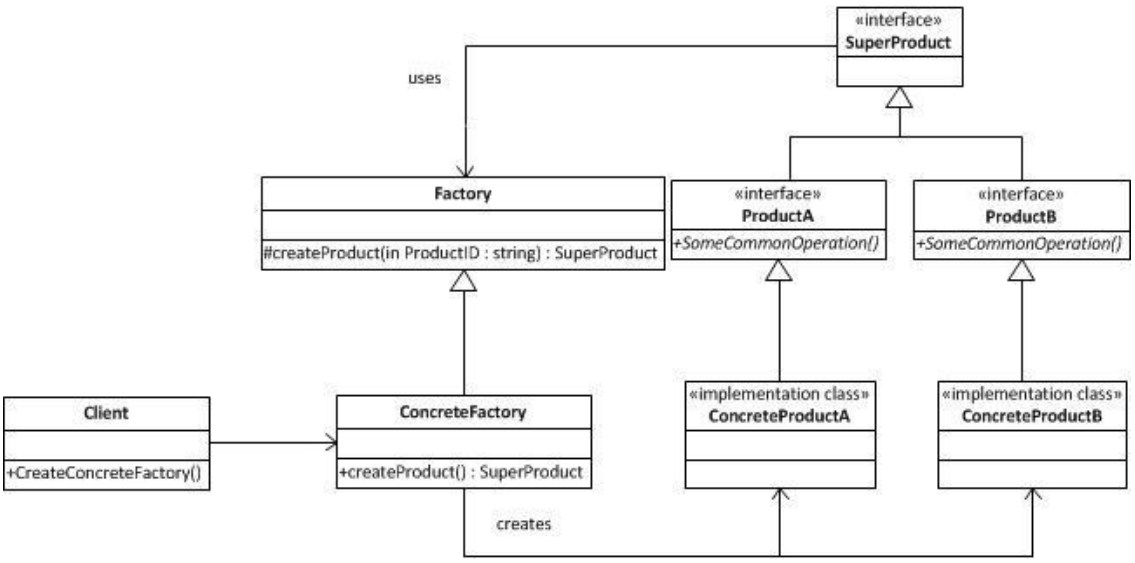
This variant extendes the idea of the the variant discussed above by providing default implementation of a product. A direct association occurs between the factory and default product.



2.2 UML Diagram of default product implementation [20]

2.3 Parameterized Factory Method [20]

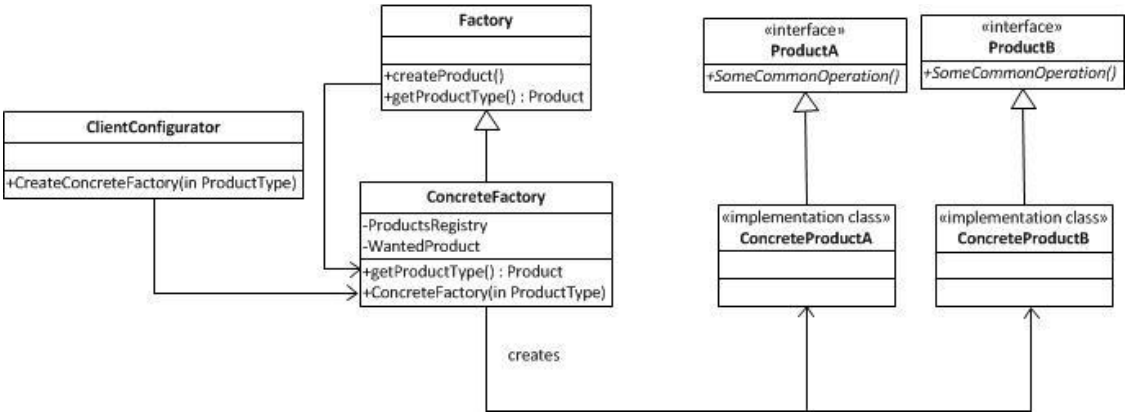
This variant highlights the concept of parameterized factory method interface. It benefits by providing a single factory method that is used to return multiple type of products. So instead of multiple factory methods this gives the oppurtunity to create the instances using a single method in which single abstract product is implemented by each concrete product.



2.3 UML Diagram of parameterized factory [20]

2.4 Delegated construction of products from Factory method [20]

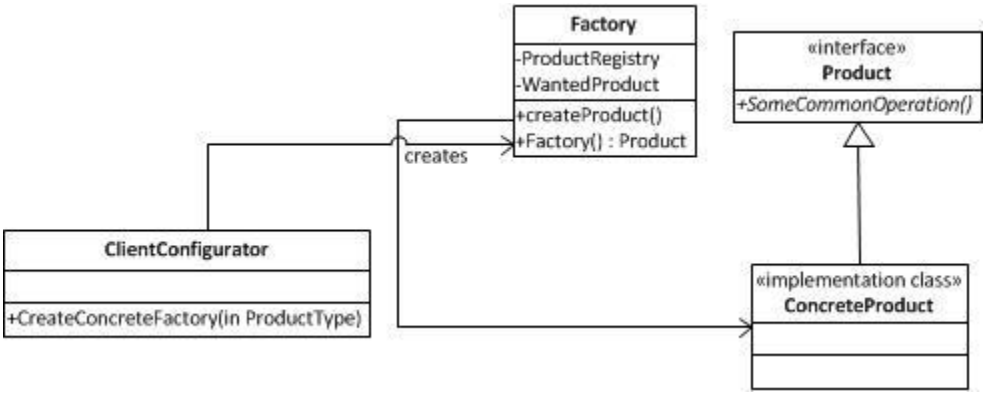
This method uses language feature mechanism to dynamically create types at runtime. Factory method calls another method which is overrided by the implementation class of the factory which return the product type which is intended to be created. Client configurator is used to select a particular time from the registry of concrete types.



2.4 UML Diagram of delegated construction using factory method [20]

2.5 No concrete factory class [20]

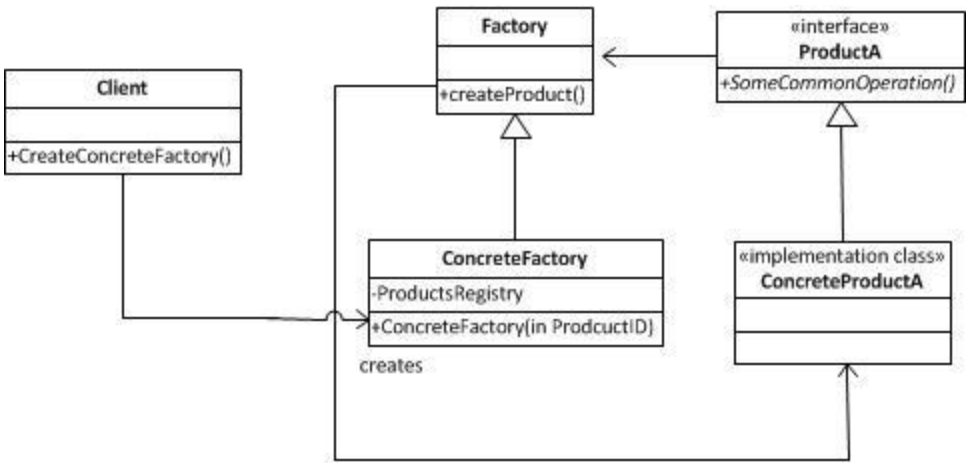
This variant implements the factory method pattern in a unique way by eleminating the subclass concrete creator. The creator is configured by a product key which is already in its registry when the creator class is loaded and the factory method is used to return that selected product.



2.5 UML Diagram of the no concrete factory class in factory method variant [20]

2.6 Product creation from repository types using delegation mechanism [20]

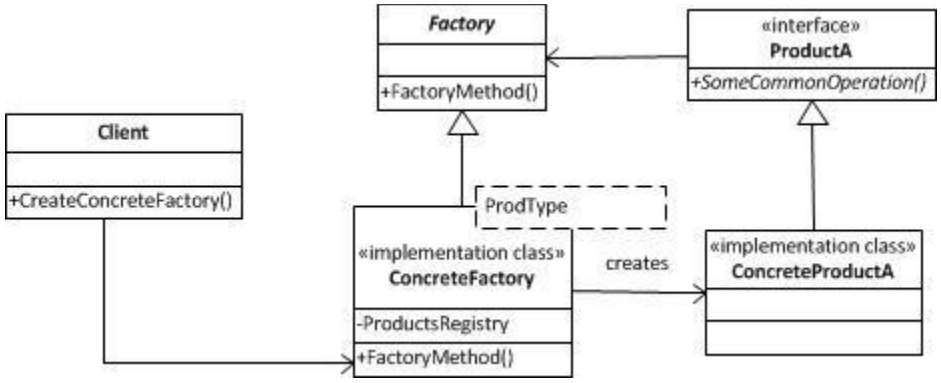
In this variant the product selection is made from the repository maintained at the concrete creator class and eliminating the need to override factory method. The selection takes place at the constructor of the concrete class [20].



2.6 UML diagram of product creation from repository using delegation [20]

2.7 One concrete creator for all [20]

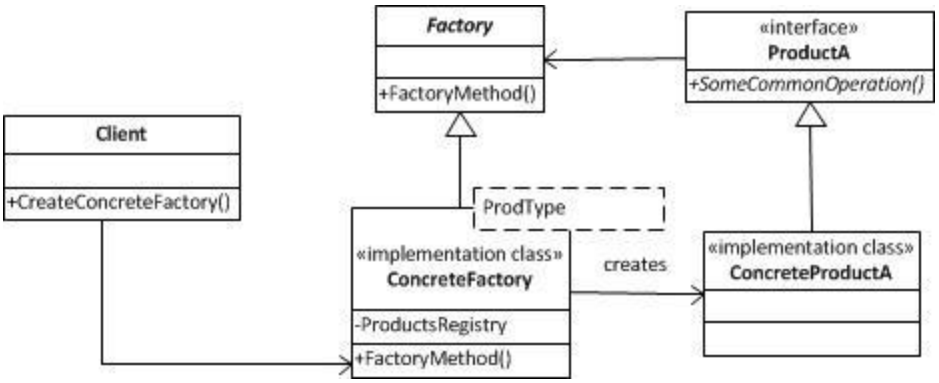
This variant is an extention of the parameterized factory method implementation with an addition of language feature (generics) which reduces the number of concrete creators to only one [20]. The number of product types served by the factory method can be in a number.



2.7 UML diagram of the one creator for all [20]

2.8 Single concrete class to select product from repository withouth any configuration client [20]

This variant is a similar implementation of the one discussed above, which basically introduce the product selection to be handled internally by the concrete factory class from the registry.



2.8 UML Diagram of single concrete creator with repository [20]