**Factory**

# Intent

Define an interface for creating an object, but let subclasses decide which class to instantiate.

# Also Known As

Virtual Constructor

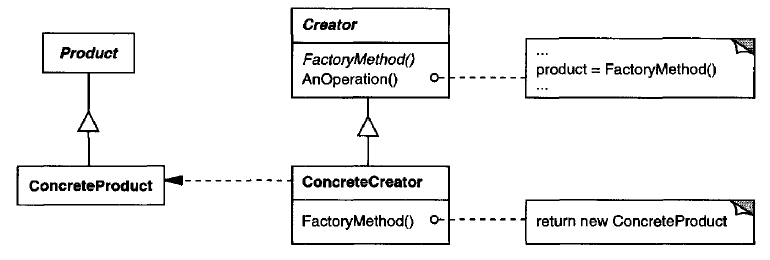
# Motivation

# Applicability

Use the Factory Method pattern when

* a class can't anticipate the class of objects it must create.
* a class wants its subclasses to specify the obje cts it creates.
* classes delegate responsibility to one of several helper subclasses, and you want to localize the knowledge of which helper subclass is the delegate.

# Structure



# Participants

**Product (Document)**

defines the interface of objects the factory method creates

**ConcreteProduct (MyDocument)**

implements the Product interface

**Creator (Application)**

declares the factory method, which returns an object of type Product

Creator may also define a default implementation of the factory method that

returns a default ConcreteProduct object

may call the factory method to create a Product object

**ConcreteCreator (MyApplication)**

overrides the factory method to return an instance of a Concrete Product

# Collaborations

Creator relies on its subclasses to define the factory method so that it returns an instance of the appropriate ConcreteProduct.

# Consequences

The code only deals with the Product interface; therefore, it can work with any user-defined ConcreteProduct classes.

**Disadvantage of factory methods**

Clients might have to subclass the Creator class just to create a particular ConcreteProduct object

A utility class that creates an instance of a class from a family of derived classes.

Define an interface for creating a single object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.

The factory method pattern is a creational pattern that uses factory methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created.

This is done by creating objects by calling a factory method—either specified in an interface and implemented by child classes, or implemented in a base class and optionally overridden by derived classes—rather than by calling a constructor.

**Also known as Virtual Constructor**

Solves problems like:

* How can an object be created so that subclasses can redefine which class to instantiate?
* How can a class defer instantiation to subclasses?

Describes how to solve such problems:

* Define a separate operation (factory method) for creating an object
* Create an object by calling a factory method

Factory Method makes a design more customizable and only a little more complicated. **Other design patterns require new classes, whereas Factory Method only requires a new operation.**

The creation is deferred to run-time.

1. Client knows about abstract base class but not concrete subclass.
2. Run-time creation of objects.
3. Subclasses create objects.

Actually, the factory method is a way of circumventing following limitations of C++ constructor:

**No return type**: A constructor cannot return a result, which means that we cannot signal an error during object initialization. The only way of doing it is to throw an exception from a constructor.

**Naming**: A constructor should have the same name as the class, which means we cannot have two constructors that both take a single argument.

**Compile time bound**: At the time when we create an object, we must specify the name of a concrete class which is known at compile time. There is no way of dynamic binding constructors at run time.

**There is no virtual constructor**: We cannot declare a virtual constructor. If we are constructing derived object, the compiler calls the base class constructor first, and the derived class hasn't been initialized yet. This is the reason why we cannot call virtual methods from the constructor.

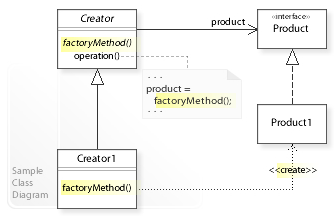
# What's the difference between Abstract Factory Pattern and Factory Method?

* Abstract Factory design pattern creates Factory
* Factory design pattern creates Products

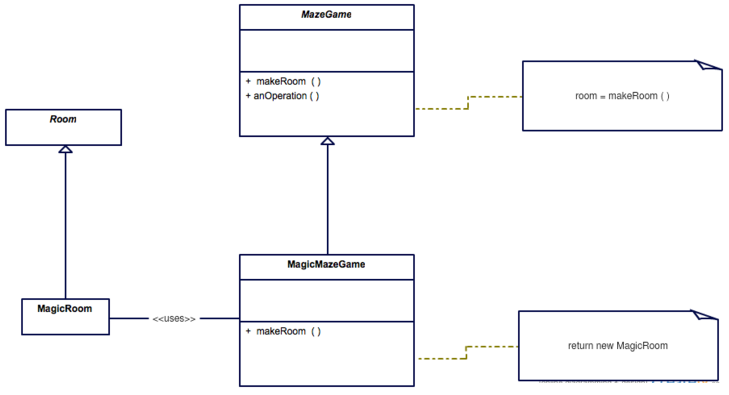
# Factory Method - Abstract Creator Class

# Factory Method - Concrete Creator Class

# Structure



UML class diagram



Class Diagram

# Application

1. In ADO.NET, IDbCommand.CreateParameter is an example of the use of factory method to connect parallel class hierarchies.
2. In Qt, QMainWindow::createPopupMenu is a factory method declared in a framework that can be overridden in application code.
3. In Java, several factories are used in the javax.xml.parsers package. e.g. javax.xml.parsers.DocumentBuilderFactory

or

javax.xml.parsers.SAXParserFactory.

# References

<https://sourcemaking.com/design_patterns/factory_method>

<https://en.wikipedia.org/wiki/Factory_method_pattern>

# END