**Factory Method Design Pattern**

Factory Method design pattern uses subclass to create object of a derived class of another class which is not known.

**Intent**

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

So the intent is to create the object through subclass.

**Problem**

* A class has to instantiate subclass of another class but does not know which one.

We may have the scenario of Framework where a class has to create an object of derived class of another class, but it does not have information of that derived class. In other words we can say, user of the framework wants to have his own derived class and another class of framework has to create the object of derived class created by user. So it's obvious that derived class information will not be available as it is created by user of framework only.

**Solution**

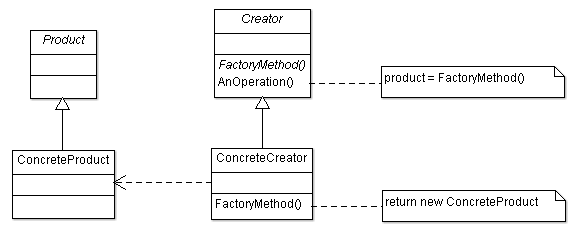
* Redefine a method in derived class which will decide which subclass to instantiate.

We can have an abstract method as factory method which is redefined in derived classes and in this method only required object is instantiated.

**Where it is applicable?**

* Framework and Libraries
* A class does not know the object it has to instantiate or it wants the objects to be created by its subclass.
* Parallel class hierarchies

**Structure**



**Participant classes**

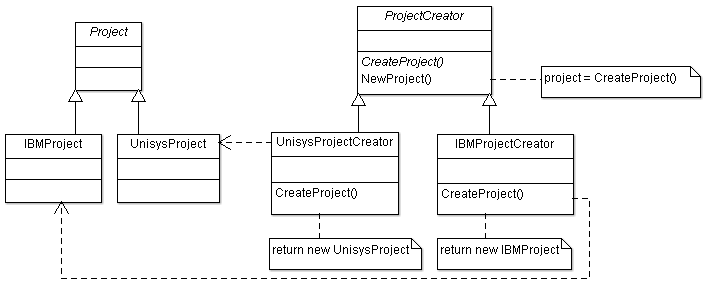
* **Product** class provides the interface for object to be created by method FactoryMethod().
* **ConcreteProduct** is derived from Product and implements the interface.
* **Creator** class has abstract method FactoryMethod(). It also has method AnOperation() which uses FactoryMethod().
* **ConcreteCreator** is derived from Creator and implements the FactoryMethod().

**How they work together?**

* Creator wants to create the object of ConcreteProduct but it does not have the information of this class. So method AnOperation() of class Creator calls the FactoryMethod() which is abstract method and implemented in subclass ConcreteCreator, so this instantiates the object of ConcreteProduct and returns the instantiated object to Creator.
* Sometimes it is required to pass some input and create objects based on that input.
* It provides lot of flexibility for plug-ins, parallel hierarchies and used heavily in different design.

**Example**

We have framework of IDE and we want to add the support for Unisys project and IBM project.



* UnisysProject and IBMProject classes are derived from abstract class Project.
* Suppose the ProjectCreator class wants to create new project which is object of UnisysProject. It has abstract method CreateProject() which is implemented in derived class UnisysProjectCreator. It calls CreateProject() method in NewProject(), so CreateProject() method of class UnisysProjectCreator instantiates the object of class UnisysProject and returns it to NewProject().
* So we can see the subclass has instantiated the object which was known later only.