CSE776-Design Patterns
Jim Fawcett

FACTORY METHOD PATTERN

Intent

- "Define an interface for creating an object, but let subclasses decide which class to instantiate"
  - It lets a class defer instantiation to subclasses at run time.
  - It refers to the newly created object through a common interface.

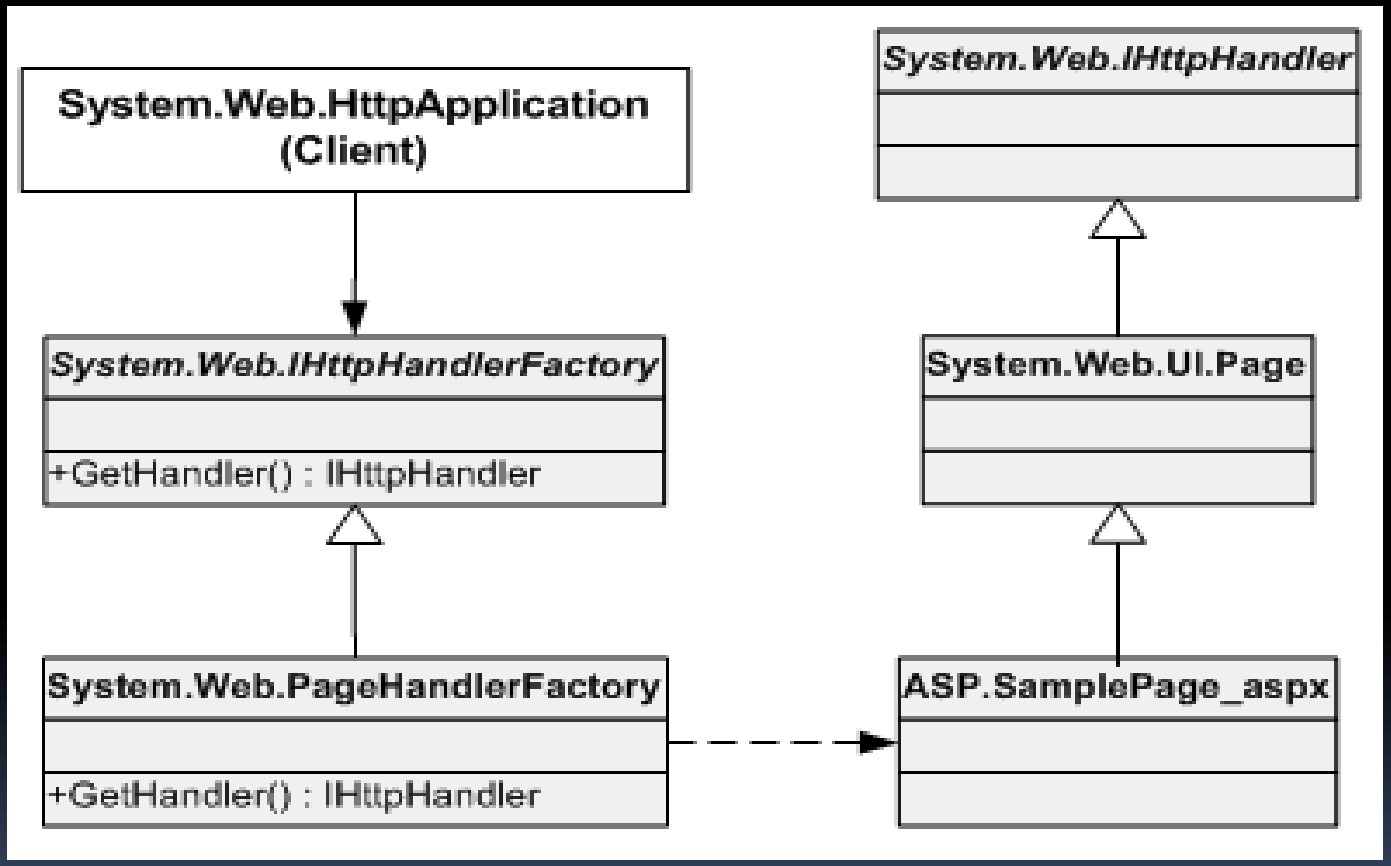
Also Known as

- Virtual Constructor
- The main intent of the virtual constructor idiom in C++ is to create a copy of an object or a new object without knowing its concrete type and this is exactly what the Factory Method does.

Motivation

- Frameworks:
  - Factory Method is used in frameworks where library code needs to create objects of types which may be sub classed by applications using the framework.
  - Since the library knows when an object needs to be created, but not what kind of object it should create, this being specific to the application, it can use the Factory Method.

Motivating Examples – Cont.



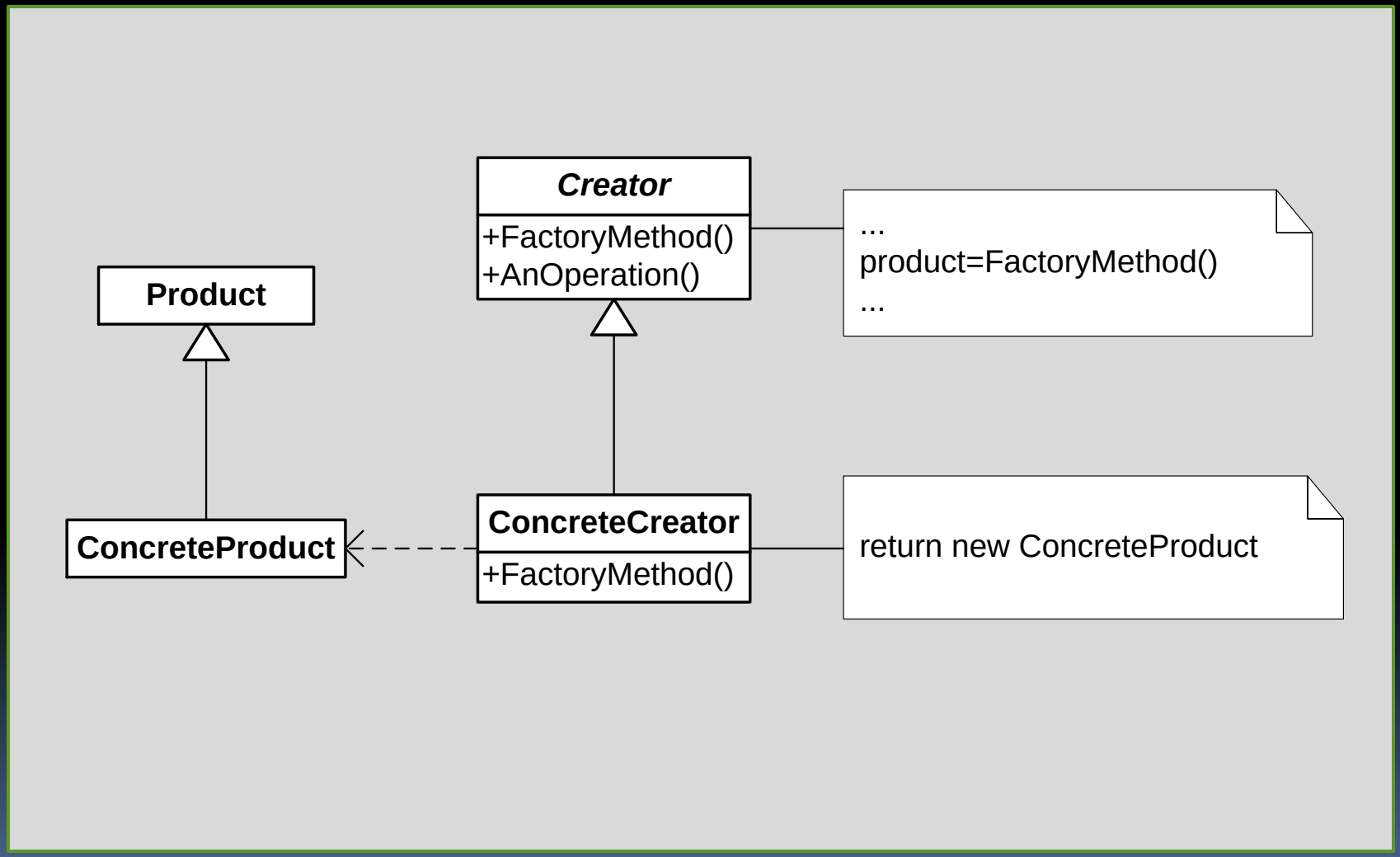
Forces

- We want to have a set of reusable classes which are flexible enough to be extended.
- The client does not know the type of object that needs to be created in advance and still wants to perform operations on them.

Applicability

- Factory Method is needed when:
  - A class can't anticipate the types of objects it must create.
  - A class wants its subclasses to specify the object to create.
  - The designer wants to localize knowledge of helper sub classes.

Basic Structure



Participants

- Product (IHttpHandler)
  - Defines the interface of objects the factory method creates.
- ConcreteProduct (ASP.SamplePage_aspx)
  - Implements the Product Interface
- Creator (IHttpHandlerFactory)
  - Declares the factory method and may provide a default implementation for it.
  - Defines the return type as Product.
- ConcreteCreator (PageHandlerFactory)
  - Overrides the factory method to return an instance of ConcreteProduct.

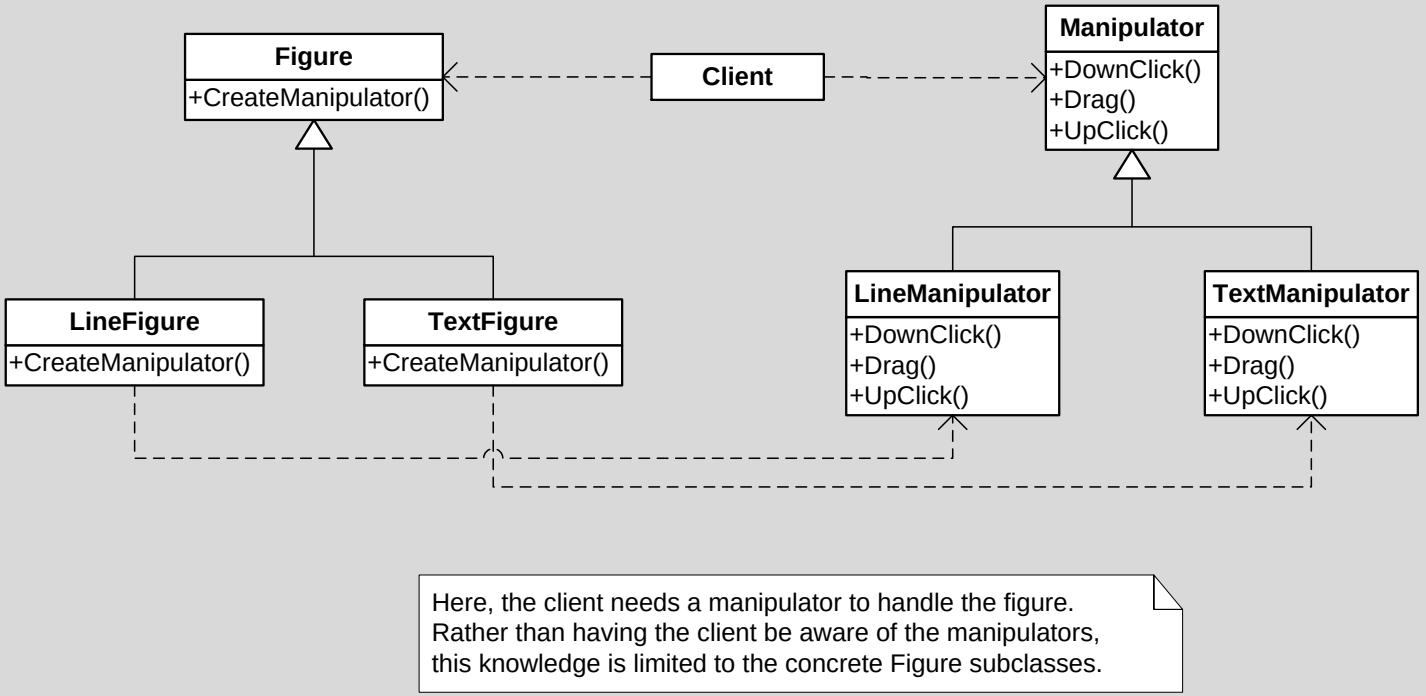
Collaborators

- The Creator relies on the subclass's factory method to return an instance of appropriate ConcreteProduct object.
- The Creator executes some sequence of operations on the object or simply returns a reference to Product (bound to the ConcreteProduct object) to the client.

Consequences

- The client code deals only with the product interface, therefore it can work with any user defined Concrete Product classes (decoupling subclass details from client classes).
- New concrete classes can be added without recompiling the existing client code.
- It may lead to many subclasses if the product objects requires one or more additional objects. (Parallel class hierarchy)

Consequences – Cont.



Implementation

- Two major varieties
  - Abstract Creator class with no default implementation
  - Concrete Creator with default implementation.
- Other variations:
  - Parameterized Methods
  - Templates

Parameterized Factory Methods

```
class Creator {
public:
  virtual Product* Create(ProductID id) {
      if (id == P1) return new MyProduct;
      if (id == P2) return new YourProduct;
      // other products ...

      return 0;
  }
};

// You can subclass the Creator to handle more IDs
Product* MyCreator::Create(ProductID id) {
  if (id == P3) return new TheirProduct;

  // Handle other IDs
  return this->Creator::Create(id);
};
```

Templatized Factory Methods

```
class Creator {
public:
   Creator() {
        // You won't call factory method here (why?)
        // Use lazy initialization instead
   }
   virtual Product* CreateProduct() = 0;
};

template <class T>
class StandardCreator: public Creator {
public:
   virtual Product* CreateProduct() {
        return new T;
   }
}

// In the Client
StandardCreator<MyProduct> myCreator;
```

Known Uses

- It is a pervasive pattern.
- It is used in several places in the Java API. For example, URLConnection has a method getContent that returns the content as an appropriate object (html, gif etc.)
- .Net Framework Class Library
  Factory method is used in:
  - Systems.Collections.IEnumerable,
  - System.Net.WebRequest
  - System.Security.Cryptography

Related Patterns

- Abstract Factory
- Template Methods
- Prototypes

References

- Design Patterns, Elements of Reusable Object-Oriented Software, Erich Gamma, et. al., Addison-Wesley, 1994, ISBN 0-201-63361-2
- http://www.ondotnet.com/pub/a/dotnet/2003/08/11/factorypattern.html
- http://en.wikibooks.org/wiki/More_C%2B%2B_Idioms/Virtual_Constructor