工厂模式

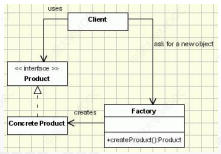
- 工厂模式可能是现代编程语言（Java、C#）中最常用的设计模式。
- 工厂模式有两种变种：
  - 工厂方法
  - 抽象工厂

工厂模式

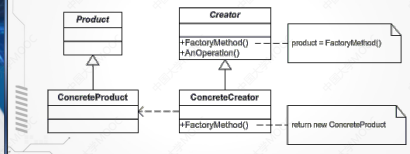
- **目的**
  - 创建对象的过程中没有暴露对象的初始化逻辑给用户。
  - 通过一个公共接口引用最新创建的对象。

工厂模式

- 工厂模式的类图



工厂方法模式



工厂方法模式

- 参与者

Product

- 定义工厂方法要创建的对象的接口

ConcreteProduct

- 实现Product接口

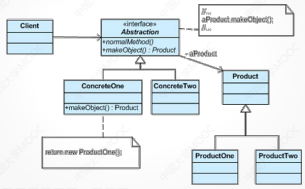
工厂方法模式

- Creator
  - 声明工厂方法,返回一个产品类型的对象.构造器也可能定义一种缺省的工厂方法实现,返回一个缺省的ConcreteProduct对象。
  - 可能会调用工厂方法创建一个产品对象

工厂方法模式

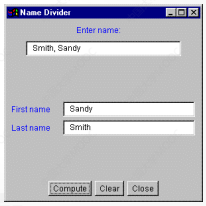
- ConcreteCreator
  - 重写工厂方法来获得一个ConcreteProduct的实例

工厂方法模式



例子

姓名的格式可以是姓在前，名在后。也可以是名在前，姓在后



例子: 基类

```
class Namer {
//a simple class to take a string apart into two names
        protected String last; //store last name here
        protected String first; //store first name here

        public String getFirst() {
                return first; //return first name   }
        public String getLast() {
                return last; //return last name   }
}
```

例子: 子类

```
class FirstFirst extends Namer { //split first last
    public FirstFirst(String s) {
                int i = s.lastIndexOf(" "); //find sep space
                if (i > 0) {
                        //left is first name
                        first = s.substring(0, i).trim();
                        //right is last name
                        last =s.substring(i+1).trim();
                        }
                else {
                        first = “”; // put all in last name
                        last = s; // if no space }
        } }
```

例子: 子类

```
class LastFirst extends Namer { //split last, first
        public LastFirst(String s) {
                int i = s.indexOf(","); //find comma
                if (i > 0) {
                        //left is last name
                        last = s.substring(0, i).trim();
                        //right is first name
                        first = s.substring(i + 1).trim();  }
                else {
                        last = s; // put all in last name
                        first = ""; // if no comma    }
    } }
```

例子：创建工厂

```
class NameFactory {
//returns an instance of LastFirst or FirstFirst
//depending on whether a comma is found
        public Namer getNamer(String entry) {
                int i = entry.indexOf(","); //comma determines name order
                if (i>0)
                        return new LastFirst(entry); //return one class
                else
                        return new FirstFirst(entry); //or the other
                }
}
```

例子：使用工厂

```
In our constructor for the program, we initialize an
instance of the factory class with
NameFactory nfactory = new NameFactory();
private void computeName() {
        //send the text to the factory and get a class
back
namer = nfactory.getNamer(entryField.getText());
        //compute the first and last names
        //using the returned class
        txFirstName.setText(namer.getFirst());
        txLastName.setText(namer.getLast());
}
```

工厂模式适用场景

- 什么时候使用工厂模式：
  - 当类不能预测要创建哪个类的对象时
  - 当类应用其子类去定义要创建的对象的类型时。
  - 当对象的创建要局部化时

工厂模式的几个变种

- 基类是抽象类，模式必须返回一个完全工作类。
- 基类包含缺省方法,在缺省方法无效时被子类继承。
- 通过传递不同参数给工厂从而返回不同类型的类实例。这种情况下，这些类共享同一个方法名，但方法实现不同。