工厂方法

波波老师~研发总监/资深架构师

开放封闭原则（The Open Close Principle）

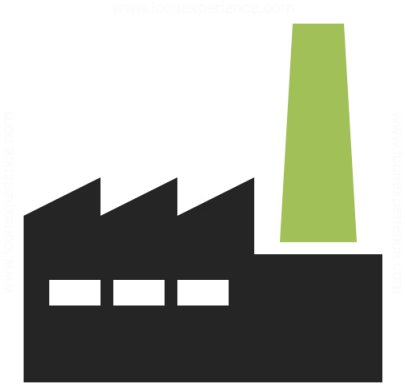
- SOLID软件设计原理之一
- 软件实体应该对扩展开放，对修改封闭。

简单工厂的问题

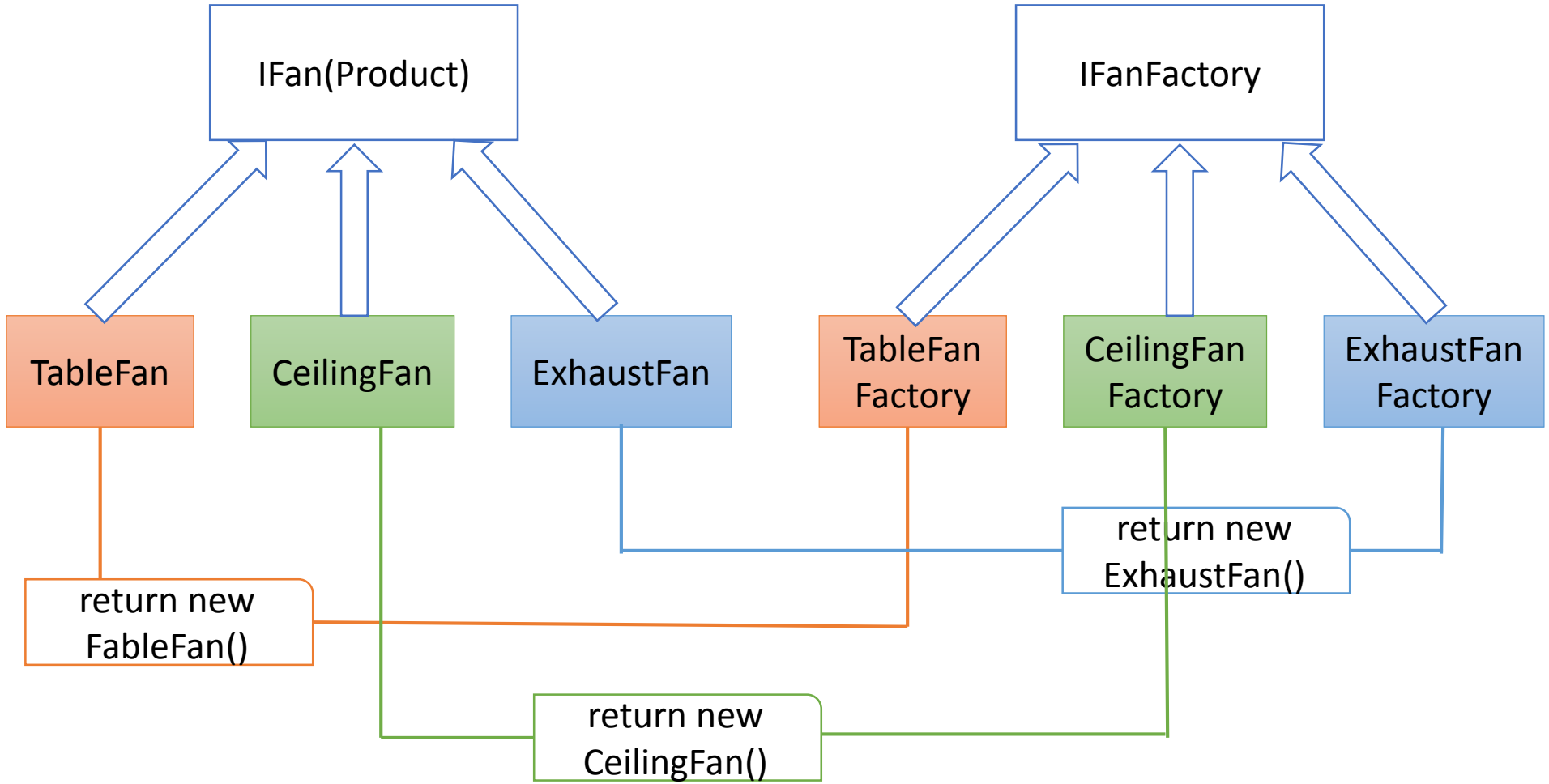
```java
package io.spring2go.corespring.simplefactory;

public class FanFactory implements IFanFactory {

    @Override
    public IFan createFan(FanType type) {
        switch (type) {
        case TableFan:
            return new TableFan();
        case CeilingFan:
            return new CeilingFan();
        case ExhaustFan:
            return new ExhaustFan();
        default:
            return new TableFan();
        }
    }

}
```

工厂方法定义

- 定义一个用于创建对象的接口，让子类决定具体实例化哪个类。
- Java中用得最多的模式之一。



关系图


IFan(Product)
IFanFactory
TableFan
CeilingFan
ExhaustFan
TableFan Factory
CeilingFan Factory
ExhaustFan Factory
return new ExhaustFan()
return new FableFan()
return new CeilingFan()

实现~接口

```
package io.spring2go.corespring.factorymethod;

public interface IFan {

    public void swithOn();

    public void switchOff();

}
```

```
package io.spring2go.corespring.factorymethod;

public interface IFanFactory {
    IFan createFan();
}
```

实现~产品

```java
package io.spring2go.corespring.factorymethod;

// 台扇
public class TableFan implements IFan {

    @Override
    public void swithOn() {
        System.out.println("The TableFan is swithed on ...");
    }

    @Override
    public void switchOff() {
        System.out.println("The TableFan is swithed off ...");
    }

}
```

```java
package io.spring2go.corespring.factorymethod;

// 吊扇
public class CeilingFan implements IFan {

    @Override
    public void swithOn() {
        System.out.println("The CeilingFan is swithed on ...");
    }

    @Override
    public void switchOff() {
        System.out.println("The CeilingFan is swithed off ...");
    }
}
```

```java
package io.spring2go.corespring.factorymethod;

public class ExhaustFan implements IFan {

    @Override
    public void swithOn() {
        System.out.println("The ExhaustFan is swithed on ...");
    }

    @Override
    public void switchOff() {
        System.out.println("The ExhaustFan is swithed off ...");
    }
}
```

实现~工厂

```java
package io.spring2go.corespring.factorymethod;

public class TableFanFactory implements IFanFactory {

    @Override
    public IFan createFan() {
        return new TableFan();
    }

}
```

```java
package io.spring2go.corespring.factorymethod;

public class CeilingFanFactory implements IFanFactory {

    @Override
    public IFan createFan() {
        return new CeilingFan();
    }

}
```

```java
package io.spring2go.corespring.factorymethod;

public class ExhaustFanFactory implements IFanFactory {

    @Override
    public IFan createFan() {
        return new ExhaustFan();
    }

}
```

添加新产品和工厂

```java
package io.spring2go.corespring.factorymethod;

// 螺旋桨式通风扇
public class PropellerFan implements IFan {

    @Override
    public void swithOn() {
        System.out.println("The PropellerFan is swithed on ...");
    }

    @Override
    public void switchOff() {
        System.out.println("The PropellerFan is swithed off ...");
    }

}
```

```java
package io.spring2go.corespring.factorymethod;

public class PropellerFanFactory implements IFanFactory {

    @Override
    public IFan createFan() {
        return new PropellerFan();
    }

}
```

客户端

```java
package io.spring2go.corespring.factorymethod;

//客户端代码
public class FactoryMethodMain {

    public static void main(String[] args) {
        IFanFactory fanFactory = new PropellerFanFactory();

        // 使用工厂方法创建一个电扇
        IFan fan = fanFactory.createFan();

        // 使用创建的对象
        fan.swithOn();

        fan.switchOff();
    }


}
```

好处

- 客户和产品制造逻辑解耦
- 遵循OCP，新需求无需改代码
- 易于单元测试
  - 没有switch(or if/else)
- 公共制造逻辑可以抽取到抽象类
  - 例如BaseFanFactory

Spring框架应用

- Spring容器基于工厂模式
  - 创建Spring beans
  - 管理Spring beans生命周期
- 工厂接口
  - BeanFactory
  - ApplicationContext
- 工厂方法
  - getBean()

课后思考

- 简单工厂和工厂方法都只能制造一类产品，如果需要创建一族相关产品，有什么好的设计模式？

参考

- Factory Patterns – Factory Method Pattern(by Snesh Prajapati)
  - https://www.codeproject.com/Articles/1135918/Factory-Patterns-Factory-Method-Pattern

代码

- https://github.com/spring2go/core-spring-patterns

波波微课

- 关于波波微课
  - 十多年研发经验老司机波波老师主导
  - 致力于使用新媒体技术提升学习成效
  - 主题面向Java，Spring，面向对象开发和微服务等
  - 关注工程师的成长
- 理念
  - 交互式的课程体验
  - 贴近一线企业实践
- 方法
  - 短视频，平均10分钟，最长不超过15分钟
  - 一个视频专注讲清楚一个主题
  - 50%原理+50%案例代码
  - 所有代码和ppt在github上可免费获得