定义

应该有且仅有一个原因引起类的变更

说明

单一职责原则为我们提供了一个编写程序的准则，要求我们在编写类，抽象类，接口时，要使其功能职责单一纯碎，将导致其变更的因素缩

如果一个类承担的职责过多，就等于把这些职责耦合在一起。一个职责的变化可能会影响或损坏其他职责的功能。而且职责越多，这个类变低

举例说明

例如下面的工厂类，负责 将原料进行 预处理 和 加工成产品X和产品Y

```
public class Factory {

    private String preProcess(String material){
        return "*"+material+"*";
    }
    public String processX(String material) {
        return preProcess(material) +"加工成：产品X";
    }

     public String processY(String material) {
         return preProcess(material) +"加工成：产品Y";
     }
}
```

现因市场需求，优化 产品X 的生产方案，需要改变原料预处理的方式
将预处理方法

```
private String preProcess(String material){
     return "#"+material+"#";
}
```

在以下场景类中运行

```
public class Client {
    public static void main(String args[]) {
        Factory factory = new Factory();
        System.out.println(factory.processX("原料"));
        System.out.println(factory.processY("原料"));
    }
}
```

运行结果如下：

```
修改前：
*原料*加工成：产品X
*原料*加工成：产品Y
修改后：
#原料#加工成：产品X
#原料#加工成：产品Y
```

从运行结果中可以发现，在使产品X可以达到预期生产要求的同时，也导致了产品Y的变化，但是产品Y的变化并不在预期当中，这便导致程

那么按照单一职责划分

```
// 定义一个抽象类
public abstract class AFactory {

    protected abstract String preProcess(String material);

    public abstract String process(String material);
}


// x产品生成的独立实现
public class FactoryX extends AFactory{

    @Override
    protected String preProcess(String material) {
        return "*"+material+"*";
    }

    @Override
    public String process(String material) {
        return preProcess(material) +"加工成: 产品X";
    }
}

// y产品生成的独立实现
public class FactoryY extends AFactory{

    @Override
    protected String preProcess(String material) {
        return "*"+material+"*";
    }

    @Override
    public String process(String material) {
        return preProcess(material) +"加工成: 产品Y";
    }
}

public class Client2 {

    public static void main(String args[]) {
        produce(new FactoryX());
        produce(new FactoryY());
    }

    private static void produce(AFactory factory) {
        System.out.println(factory.process("原料"));
    }

}
```