# 简单工厂模式

## 1. 模式介绍

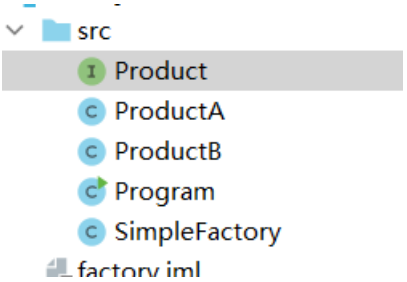
简单工厂模式（Simple Factory Pattern）是设计模式中的一种，它属于创建型模式，但不属于 GoF（Gang of Four，即四位设计模式的作者）提出的 23 种经典设计模式之一。然而，由于其简单性和实用性，它在实际编程中经常被使用。

## 2. 模式用法

简单工厂模式的主要目的是根据传入的参数（通常是字符串或其他标识符）来动态地创建并返回某一类（或该类的子类）的实例。在简单工厂模式中，通常会有一个工厂类，它包含了必要的逻辑来判断应该创建并返回哪一个类的实例。

## 3. 代码实例：

代码结构：




接口：

```
public interface Product {
    2 usages  2 implementations
    void use();
}
```

产品 A:

```
1 usage
public class ProductA implements Product{

    2 usages
    @Override
    public void use() {
        System.out.println("Using Product A");
    }
}
```

产品 B:

```
1 usage
public class ProductB implements Product{
    2 usages
    @Override
    public void use() {
        System.out.println("Using Product B");
    }
}
```

工厂类：

```
2 usages
public class SimpleFactory {
    2 usages
    public static Product createProduct(String type) {
        if ("A".equals(type)) {
            return new ProductA();
        } else if ("B".equals(type)) {
            return new ProductB();
        } else {
            return null; // 或者抛出一个异常
        }
    }
}
```

运行：

```
*/
public class Program {
    public static void main(String[] args) {
        // 使用简单工厂创建并使用产品A
        Product productA = SimpleFactory.createProduct( type: "A");
        if (productA != null) {
            productA.use();
        }

        // 使用简单工厂创建并使用产品B
        Product productB = SimpleFactory.createProduct( type: "B");
        if (productB != null) {
            productB.use();
        }
    }
}
```

结果：

```
"C:\Program Files\Java\jdk-1
Using Product A
Using Product B
```

## 4.优缺点

简单工厂模式的优点主要包括以下几点：

易于理解和实现：简单工厂模式的设计和实现都非常直观，它通过一个集中的工厂类来负责创建对象，使得代码更加简洁和易于理解。

解耦：将对象的创建与使用分离，降低了代码的耦合度。客户端代码不再需要关心对象的创建细节，只需要通过工厂类来获取所需的对象即可。

集中管理：所有的对象创建逻辑都集中在工厂类中，这使得对对象的创建进行统一管理和维护变得容易。当需要修改对象的创建逻辑时，只需要修改工厂类即可，而不需要修改客户端代码。

代码复用：如果多个地方都需要创建相同的对象，可以通过工厂类进行复用，避免代码的重复。

可扩展性（相对简单的扩展）：虽然简单工厂模式本身在添加新产品时需要修改工厂类，但相比客户端直接实例化对象的方式，工厂类提供了更集中的修改点。在简单的应用场景中，这种扩展性是可以接受的。

隐藏实现细节：客户端代码不需要知道对象是如何被创建的，只需要知道如何调用工厂类来获取对象即可。这有助于隐藏实现细节，提高代码的可维护性。

符合单一职责原则（在一定程度上）：工厂类负责创建对象，而客户端代码负责使用对象。这在一定程度上符合单一职责原则，即每个类都应该只有一个引起变化的原因。

需要注意的是，虽然简单工厂模式具有以上优点，但它也存在一些缺点，如违反了开闭原则（在添加新产品时需要修改工厂类）、不易于扩展复杂的对象层次结构等。因此，在选择使用简单工厂模式时需要根据具体的应用场景和需求进行权衡。