软件体系结构作业 13

姓　名：洪伟鑫

专　业：软件工程

年　级：2022 级

学　号：37220222203612

2025 年 4 月 22 日

1、阅读 Abstract Factory 的例子的代码，举例说明使用 Abstract Factory 模式的其他应用。

抽象工厂的例子还有如下：

多主题文档生成器

抽象工厂：DocumentFactory

具体工厂：PDFFactory, HTMLFactory, WordFactory

抽象产品：Header, Paragraph, Table

具体产品：PDFHeader, HTMLHeader, WordHeader 等

应用场景：将同一份内容以不同格式输出，保持各种格式的一致性

支付系统集成

抽象工厂：PaymentFactory

具体工厂：AlipayFactory, WeChatPayFactory, PayPalFactory

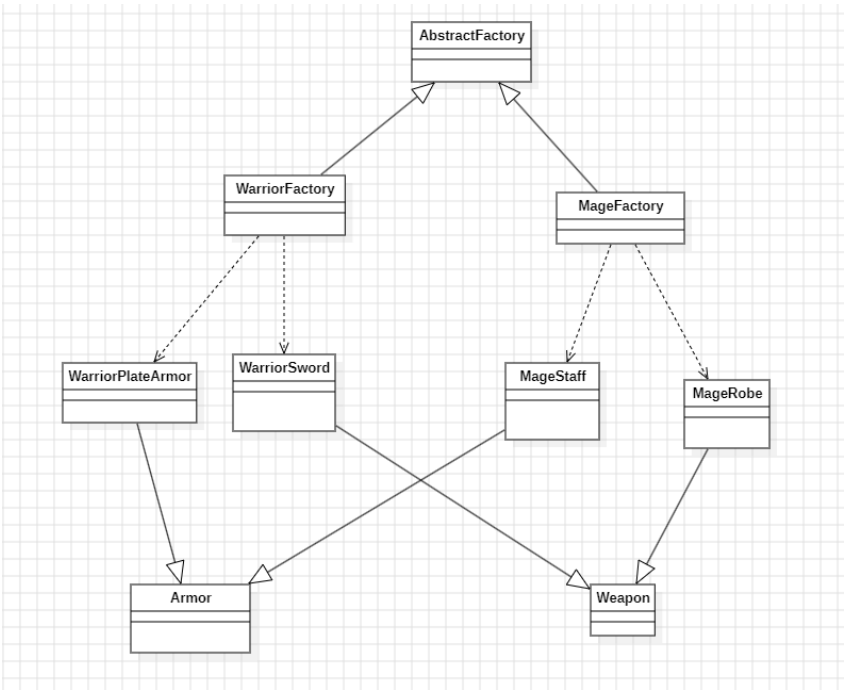
抽象产品：PaymentForm, PaymentProcessor, PaymentValidator

具体产品：各支付平台对应的具体实现

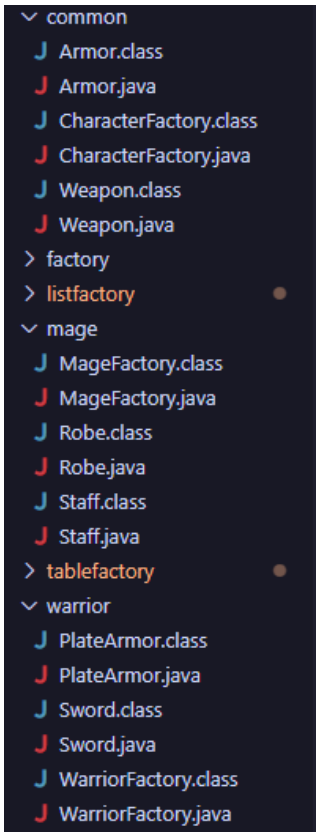
应用场景：集成多种支付方式，统一支付处理流程

这里以实现一个 RPG 角色创建的例子来说明：

1. **抽象产品**：
   Weapon：武器抽象类
   Armor：防具抽象类
2. **具体产品**：
   武器：Sword（剑）、Staff（法杖）
   防具：PlateArmor（板甲）、Robe（法袍）
3. **抽象工厂**：
   CharacterFactory：角色工厂抽象类
4. **具体工厂**：
   WarriorFactory：战士工厂
   MageFactory：法师工厂



目录如下



代码：

```java
package warrior;

import common.CharacterFactory;
import common.Weapon;
import common.Armor;

public class WarriorFactory extends CharacterFactory {
    @Override
    public Weapon createWeapon() {
        return new Sword();
    }

    @Override
    public Armor createArmor() {
        return new PlateArmor();
    }
}
```

以战士产品族为例，类中有两个方法，分别创建两个具体的产品等级结构（Weapon、Armor）

```java
import common.CharacterFactory;
import common.Weapon;
import common.Armor;
import warrior.WarriorFactory;
import mage.MageFactory;

public class GameTest {
    Run | Debug
    public static void main(String[] args) {
        // 创建战士装备
        CharacterFactory warriorFactory = new WarriorFactory();
        Weapon warriorWeapon = warriorFactory.createWeapon();
        Armor warriorArmor = warriorFactory.createArmor();

        System.out.println(x:"=== 战士装备 ===");
        warriorWeapon.attack();
        warriorArmor.defend();

        // 创建法师装备
        CharacterFactory mageFactory = new MageFactory();
        Weapon mageWeapon = mageFactory.createWeapon();
        Armor mageArmor = mageFactory.createArmor();

        System.out.println(x:"\n=== 法师装备 ===");
        mageWeapon.attack();
        mageArmor.defend();
    }
}
```

在测试类中创建了具体的战士工厂，然后调用对应的方法产生具体产品（PlateArmor 和 Sword）

```java
package warrior;

import common.Weapon;

public class Sword extends Weapon {
    public Sword() {
        super(name:"精钢剑", damage:15);
    }

    @Override
    public void attack() {
        System.out.println("使用" + name + "进行近战攻击，造成" + damage + "点伤害！");
    }
}
```

```java
package warrior;

import common.Armor;

public class PlateArmor extends Armor {
    public PlateArmor() {
        super(name:"精钢板甲", defense:25);
    }

    @Override
    public void defend() {
        System.out.println("穿戴" + name + "，获得" + defense + "点物理防御！");
    }
}
```

法师工厂同理:

```java
package mage;

import common.CharacterFactory;
import common.Weapon;
import common.Armor;

public class MageFactory extends CharacterFactory {
    @Override
    public Weapon createWeapon() {
        return new Staff();
    }

    @Override
    public Armor createArmor() {
        return new Robe();
    }
}
```

```java
package mage;

import common.Armor;

public class Robe extends Armor {
    public Robe() {
        super(name:"魔法长袍", defense:15);
    }

    @Override
    public void defend() {
        System.out.println("穿戴" + name + ", 获得" + defense + "点魔法防御！");
    }
}
```

```java
package mage;

import common.Weapon;

public class Staff extends Weapon {
    public Staff() {
        super(name:"魔法杖", damage:20);
    }

    @Override
    public void attack() {
        System.out.println("使用" + name + "释放魔法，造成" + damage + "点魔法伤害！");
    }
}
```

运行结果如下:

```
PS E:\大三资料\大三下课程资料\体系结构\PPT\Code\Abstract Factory> java GameTest
=== 战士装备 ===
使用精钢剑进行近战攻击，造成15点伤害！
穿戴精钢板甲，获得25点物理防御！

=== 法师装备 ===
使用魔法杖释放魔法，造成20点魔法伤害！
穿戴魔法长袍，获得15点魔法防御！
```