template method模式【抽取通用处理逻辑 --- 抽象类实现了通用逻辑，其他类继承抽象类，实现业务不同逻辑】

场景：

（1）我们有多种优惠策略

（2）不同的优惠策略在计算的价格的时候，有一些通用的基础逻辑

（3）每种优惠策略还有一些是自己特殊的价格计算的逻辑

public abstract class AbstractClass {

   public void templateMethod() {

       System.out.println("处理优惠打折相关的基础通用逻辑");

       // 但是对于优惠打折具体的处理逻辑，交给不同的折扣类型子类自己去实现

       method1();

       method2();

       method3();

  }

   protected abstract void method1();

   protected abstract void method2();

   protected abstract void method3();

}

public ConcreteClass extends AbstractClass {

   public void method1() {

       System.our.println("实现优惠打折的部分逻辑");

  }

   public void method2() {

       System.our.println("实现优惠打折的部分逻辑");

  }

   public void method3() {

       System.our.println("实现优惠打折的部分逻辑");

  }

}

**这个模式一定要多用用，尤其是对于有多种不同实现的一些场景**，比如说，不同的折扣类型，不同的优惠券类型，不同的商品类型，那肯定涉及到不同的处理逻辑。但是可以将他们共同的基础逻辑抽取到父类中，然后将要子类实现的逻辑留空，交给子类去实现。

这种场景应该是很常见的， 我们从这个阶段开始，后面每个阶段，只要遇到类似场景，就会去使用这种设计模式

应该是**最高频**使用的设计模式，高频到了，我们都不用再这里举个什么例子了，因为在任何一个系统中，一定会出现说，多个类中都有相同的代码，此时就可以使用模板方法设计模式，将多个类中通用的逻辑抽取到一个父类中去

853769620