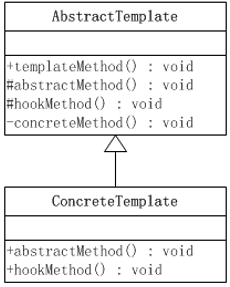
* 设计模式 – 模版方法模式（template method）：

准备一个抽象类，在一个模板方法中定义一个算法的骨架，而将一些步骤实现延迟到子类中。模板方法使得子类可以在不改变算法结构的情况下，重新定义算法中的某些步骤。将主要的方法定义为final，防止子类修改算法骨架，将子类必须实现的方法定义为abstract。而普通的方法（无final或abstract修饰）则称之为钩子方法。

**钩子方法作用：**1)作为可选内容，子类可以重写或者置之不理2)让子类有机会对模板方法中即将发生的或者已经发生的步骤做出反应3)作为控制条件，使得子类可以影响到抽象类中的算法流程

**模板方法模式是所有模式中最为常见的几个模式之一，是基于继承的代码复用的基本技术。**



**模板方法模式角色:**

**1）抽象模板(Abstract Template)角色：**

（1）定义了一个或多个抽象操作，以便让子类实现。这些抽象操作叫做基本操作，它 们是一个顶级逻辑的组成步骤。

（2）定义并实现了一个模板方法。这个模板方法一般是一个具体方法，它给出了一个

顶级逻辑的骨架，而逻辑的组成步骤在相应的抽象操作中，推迟到子类实现。顶级逻辑也有可能调用一些具体方法。

**2）具体模板(Concrete Template)角色又如下责任：**

**（1）**实现父类所定义的一个或多个抽象方法，它们是一个顶级逻辑的组成步骤。

（2）每一个抽象模板角色都可以有任意多个具体模板角色与之对应，而每一个具体模

板角色都可以给出这些抽象方法（也就是顶级逻辑的组成步骤）的不同实现，从而使得顶级逻辑的实现各不相同。

* 设计模式 – 模版方法模式优劣：

**优点**：1)模板方法模式在一个类中形式化地定义算法，而由它的子类实现细节的处理。2)模板方法是一种代码复用的基本技术。它提取了类库中的公共行为。3)模板方法模式导致一种反向的控制结构，这种结构有时被称为“好莱坞法则”，即“别找我们，,我们找你”通过一个父类调用其子类的操作(而不是相反的子类调用父类)，通过对子类的扩展增加新的行为，符合“开闭原则”。

**缺点**：每个不同的实现都需要定义一个子类，这会导致类的个数增加，系统更加庞大，设计也更加抽象，但是更加符合“单一职责原则”，使得类的内聚性得以提高。

* 设计模式 – 模版方法模式Demo：

解决某类事情的步骤有些是固定的，有些是会发生变化的，那么这时候我们可以为这类事情提供一个模板代码，从而提高效率 。

模板模式的步骤：

1. 先写出解决该类事情其中 的一件的解决方案。

2. 分析代码，把会发生变化的代码抽取出来独立成一个方法。把该方法描述成一个抽象的方法。

3. 使用final修饰模板方法，防止别人 重写你的模板方法。

示例：编写一个计算程序运行时间的模板。

