【模板方法模式应用场景举例】

比如在玩“极品飞车”这款游戏，每辆车都有显示速度的仪表盘，但有可能 A 车显示的是蓝色的仪表盘，B 车显示的是红色的仪表盘，分析中可以发现，他们都有车速度的值，但显示的方式不太一样。其实模板方法就是最典型的“继承”的使用，大家平时百分百都可以用到，只是不知道叫模板方法模式:)!

【模板方法模式解释】

类型：行为模式

模板方法模式定义一个操作中算法的骨架，而将一些步骤延迟到子类中，使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。

【模板方法模式 UML 图】

【模板方法模式-JAVA 代码实现】

新建赛车的父类：

```
package car_package;
public class car_parent {
    private int speed;
    public int getSpeed() {
        return speed;
    }
    public void setSpeed(int speed) {
        this.speed = speed;
    }
    public void print_speed() {
        System.out.println("将速度" + this.getSpeed() + "取出来！");
    }
}
```

新建红色仪表盘的赛车实现类：

```
package car_imple;

import car_package.car_parent;

public class car_imple_red extends car_parent {

    @Override
    public void print_speed() {
        super.print_speed();
        System.out.println("将速度" + this.getSpeed() + "用红色的仪表盘显示车的速度");
    }

}
```

新建蓝色仪表盘的赛车实现类：

```
package car_imple;

import car_package.car_parent;
```

```
public class car_imple_blue extends car_parent {

    @Override
    public void print_speed() {
        super.print_speed();
        System.out.println("将速度" + this.getSpeed() + "用蓝色的仪表盘显示车的速度");
    }
}
```

新建客户端运行类：

```
package run_main;

import car_imple.car_imple_blue;
import car_imple.car_imple_red;
import car_package.car_parent;

public class run_main {

    public static void main(String[] args) {

        car_parent car_ref_red = new car_imple_red();
        car_ref_red.setSpeed(300);
        car_ref_red.print_speed();

        car_parent car_ref_blue = new car_imple_blue();
        car_ref_blue.setSpeed(400);
        car_ref_blue.print_speed();
    }
}
```

程序运行结果如下：

```
将速度300取出来!
将速度300用红色的仪表盘显示车的速度
将速度400取出来!
将速度400用蓝色的仪表盘显示车的速度
```

这就是模板方法模式，一个最常用，最容易理解的一个模式，将相同的功能抽象出来成一个父类，然后用子类做不同功能的实现。