软件体系结构

Zhenyan Ji

Beijing Jiaotong University

行为型模式--模板模式

模板模式

模板模式的思想是算法的一些部分定义良好并在基类中实现。算法其他部分有多种实现方式由派生类实现。

模板模式

目的

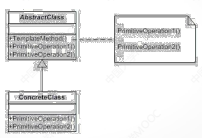
- 在一个操作中定义算法的骨架，而将一些步骤推迟到子类中实现。
- 模板方法使得子类在不改变算法结构的情况下重新定义该算法的某些特定步骤。

模板模式

动机

- 定义一个方法调用的操作的顺序，但允许子类提供它们特有的操作实现。

模板模式



模板模式

AbstractClass

- 定义了抽象的原语操作，子类实现算法的某些步骤。
- 实现定义算法骨架的模板方法，模板方法调用原语操作、AbstractClass里定义的操作、其他对象定义的操作。

模板模式

ConcreteClass

- 实现原语操作，执行算法的子类特定的步骤。

模板模式

Template类中的方法种类：

- Concrete methods
  定义所有子类的公共的基本操作。
- Abstract methods
  定义一些没有被具体实现的方法，这些方法会在派生类中实现。

模板模式

Hook methods

- 定义了操作的缺省实现，但在派生类中会被重写。

Template methods

- 调用模板类中其他abstract、hook、concrete方法。这些方法描述了算法执行的逻辑顺序，但没有实现算法的每个细节，也不允许子类对其进行重写。

模板模式：例子

```
public class Trip {
        public final void performTrip(){
                doComingTransport();
                doDayA();
                doDayB();
                doDayC();
                doReturningTransport();
        }
        public abstract void doComingTransport();
        public abstract void doDayA();
        public abstract void doDayB();
        public abstract void doDayC();
        public abstract void doReturningTransport();
}
```

模板模式：例子

```
public class PackageA extends Trip {
        public void doComingTransport() {
                System.out.println("The turists are comming by air ...");      }
        public void doDayA() {
                System.out.println("The turists are visiting the aquarium ..."); }
        public void doDayB() {
                System.out.println("The turists are going to the beach ...");    }
        public void doDayC() {
                System.out.println("The turists are going to mountains ...");   }
        public void doReturningTransport() {
                System.out.println("The turists are going home by air ...");     }
}
```

模板模式：例子

```
public class PackageB extends Trip {
        public void doComingTransport() {
                System.out.println("The turists are comming by train ...");     }
        public void doDayA() {
                System.out.println("The turists are visiting the mountain ..."); }
        public void doDayB() {
                System.out.println("The turists are going to the beach ...");     }
        public void doDayC() {
                System.out.println("The turists are going to zoo ...");     }
        public void doReturningTransport() {
                System.out.println("The turists are going home by train ...");   }
        }
```

模板模式

模板模式:

- 实现算法中不变的部分，让子类实现算法中可以变动的部分。
- 当重构系统代码时，识别出类之间的公共行为，将其封装在抽象基类中，从而避免了代码的重复。