Understanding Template method pattern

2

1

From what I understand, Template method is nothing but ordinary method that calls virtual or abstract methods defined in child class. Am I right, or is there something else important about this pattern that I miss?

```
abstract class Foo {
  public void IamTemplateMethod() { // which will be called in child class object
    method1(); // because this...
    method2(); // ...or this method was called in me
  }
  public virtual void method1() { ... } // to be overriden in child class
  public abstract void method2() { ... } // to be defined in child class
}
```

If I am right, are there any other common ways to implement Template method?

design-patterns template-method-pattern

asked Feb 25 '14 at 13:41

Jan Turoň

19.2k 16 86 131

2 Answers

6

Yes. Most patterns are nothing special, but just smart approaches that seems to suit certain situations well, but still using normal OO principles (inheritance, polymorphism, abstraction etc.).

What the template method is saying is that sometimes, you need to do some common logic, with some sub-class specific logic interleaved with it. So the specific logic that you want to leave for each sub-class is defined as an abstract / virtual method that is left for the concrete class to implement, while the common business logic goes around it.

class to override the bits you want it to.

Think of it like a document template. The headings, footer and common elements will be there fixed and always the same, and then the specific details of what the specific document is being used for fill the blanks in between.

edited Feb 25 '14 at 13:55

answered Feb 25 '14 at 13:48

jbx
12.6k 10 62 115

ah - finally I understand why to use `final` keyword, thanks! :-) – Jan Turoň Feb 25 '14 at 13:56

Cool :). You can also use it with classes (not to allow anyone to override them) and with variables (in which case their value will not be allowed to change once they are initialised, they will be immutable) – jbx Feb 25 '14 at 13:57

2

Template pattern provide a common sequence following for all the children of that method. So Template Pattern defines a final method which tells the sequence of execution.

```
abstract class Foo {
    public void final initilialize(){
        method1();
        method2();
        method3();
    }
    public void method1(){...}
    public void method2(){...}
    public void method3(){...}
}
```

Now child classes can extend Foo class. And Reference can be created as :

```
Foo obj1=new child();
```

For more information look into http://www.tutorialspoint.com/design_pattern/template_pattern.htm

answered Feb 25 '14 at 13:52

Is this still a Template method pattern when method1() - method3() are not virtual or abstract, so calling initialize() will not call those method from child class? – Jan Turoň Feb 25 '14 at 14:05

Ah, I see: Java syntax where all methods can be overriden. – Jan Turoň Feb 25 '14 at 14:11

yups.. and so you can enforce methods to be executed in a specified sequence in java. – Prateek Feb 25 '14 at 15:49