Template method pattern

In software engineering, the **template method pattern** is a behavioral design pattern that defines the program skeleton of an algorithm in a method, called template method, which defers some steps to subclasses.[1] It lets one redefine certain steps of an algorithm without changing the algorithm's structure.[2] This use of "template" is unrelated to C++ templates.

1 Introduction

In the *template method* of this design pattern, one or more algorithm steps can be overridden by subclasses to allow differing behaviors while ensuring that the overarching algorithm is still followed.

In object-oriented programming, first a class is created that provides the basic steps of an algorithm design. These steps are implemented using abstract methods. Later on, subclasses change the abstract methods to implement real actions. Thus the general algorithm is saved in one place but the concrete steps may be changed by the subclasses.

The Template Method pattern thus manages the larger picture of task semantics, and more refined implementation details of selection and sequence of methods. This larger picture calls abstract and non-abstract methods for the task at hand. The non-abstract methods are completely controlled by the template method, but the abstract methods, implemented in subclasses, provide the pattern's expressive power and degree of freedom. Template Method's abstract class may also define hook methods that may be overridden by subclasses.[3] Some or all of the abstract methods can be specialized in a subclass, allowing the writer of the subclass to provide particular behavior with minimal modifications to the larger semantics. The template method (that is non-abstract) remains unchanged in this pattern, ensuring that the subordinate non-abstract methods and abstract methods are called in the originally intended sequence.

The Template Method pattern occurs frequently, at least in its simplest case, where a method calls only one abstract method, with object oriented languages. If a software writer uses a polymorphic method at all, this design pattern may be a rather natural consequence. This is because a method calling an abstract or polymorphic function is simply the reason for being of the abstract or polymorphic method. The Template Method pattern may be used to add immediate present value to the software or with a vision to enhancements in the future. It is strongly related to the Non-Virtual Interface (NVI) pattern.

The Template Method pattern implements the Protected Variations GRASP principle, like the Adapter pattern does. The difference is that Adapter gives the same interface for several operations, while Template Method does so only for one.

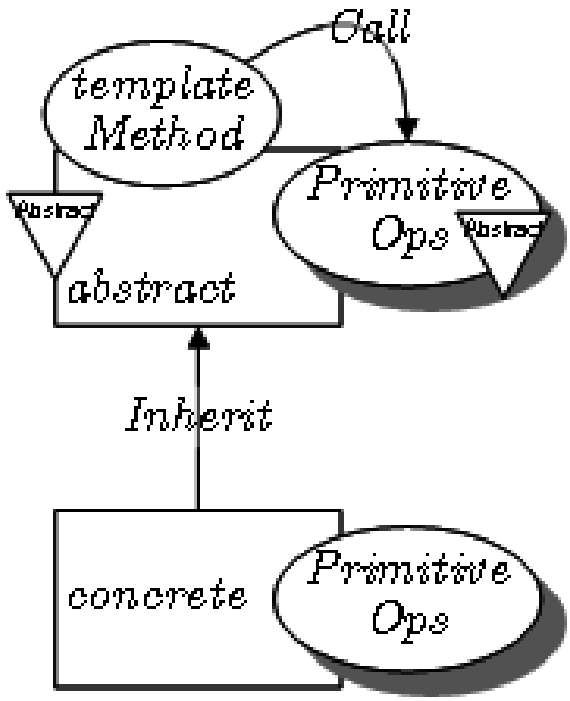
2 Structure

Template method: UML class diagram.

3 Usage

The template method is used in frameworks, where each implements the invariant parts of a domain's architecture, leaving "placeholders" for customisation options. This is an example of inversion of control. Reasons to use the template method are to:[4]

- Let subclasses implement (through method overriding) behavior that can vary.
- Avoid duplication in the code: the general workflow structure is implemented once in the abstract class's algorithm, and necessary variations are implemented in each of the subclasses.



Template Method in LePUS3 (legend)

- Control at what point(s) subclassing is allowed. As opposed to a simple polymorphic override, where the base method would be entirely rewritten allowing radical change to the workflow, only the specific details of the workflow are allowed to change.

4 Example in Java

/** * An abstract class that is common to several games in * which players play against the others, but only one is * playing at a given time. */ abstract class Game { /* Hook methods. Concrete implementation may differ in each subclass*/ protected int playersCount; abstract void initializeGame(); abstract void makePlay(int player); abstract boolean endOfGame(); abstract void printWinner(); /* A template method : */ public final void playOneGame(int playersCount) { this.playersCount = playersCount; initializeGame(); int j = 0; while (!endOfGame()) { makePlay(j); j = (j + 1) % playersCount; } printWinner(); } } //Now we can extend this class in order //to implement actual games: class Monopoly extends Game { /* Implementation of necessary concrete methods */ void initializeGame() { // Initialize players // Initialize money } void makePlay(int player) { // Process one turn of player } boolean endOfGame() { // Return true if game is over // according to Monopoly rules } void printWinner() { // Display who won } /* Specific declarations for the Monopoly game. */ // ... } class Chess extends Game { /* Implementation of necessary concrete methods */ void initializeGame() { // Initialize players // Put the pieces on the board } void makePlay(int player) { // Process a turn for the player } boolean endOfGame() { // Return true if in Checkmate or // Stalemate has been reached } void printWinner() { // Display the winning player } /* Specific declarations for the chess game. */ // ... }

5 See also

- Inheritance (computer science)
- Method overriding (programming)
- GRASP (object-oriented design)
- Adapter pattern

6 References

[1] Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1994). "Template Method". *Design Patterns*. Addison-Wesley. pp. 325–330. ISBN 0-201-63361-2.

[2] Freeman, Eric; Freeman, Elisabeth; Kathy, Sierra; Bert, Bates (2004). Hendrickson, Mike; Loukides, Mike, eds. "Head First Design Patterns" (paperback) **1**. O'REILLY. p. 289. ISBN 978-0-596-00712-6. Retrieved 2012-09-12.

[3] Freeman, Eric; Freeman, Elisabeth; Kathy, Sierra; Bert, Bates (2004). Hendrickson, Mike; Loukides, Mike, eds. "Head First Design Patterns" (paperback) **1**. O'REILLY. p. 311. ISBN 978-0-596-00712-6. Retrieved 2012-09-12.

[4] "Template Method Design Pattern". Source Making - teaching IT professional. Retrieved 2012-09-12. Template Method is used prominently in frameworks.

7 External links

- Working with Template Classes in PHP 5
- Template method pattern in UML and in LePUS3 (a formal modelling language)
- Difference between adapter and template method pattern
- Template Method Design Pattern
- Template Method Example

8 Text and image sources, contributors, and licenses

8.1 Text

- **Template method pattern** *Source:* https://en.wikipedia.org/wiki/Template_method_pattern?oldid=690576535 *Contributors:* Roadrunner, Mrwojo, Michael Hardy, TakuyaMurata, Ronz, Dysprosia, Hao2lian, Fredrik, RedWolf, BenFrantzDale, Chriki, Zhen Lin, T1mmyb, Zarkonnen, Supadawg, Andreas Kaufmann, Remuel, TheParanoidOne, Oleg Alexandrov, Mahanga, Ruud Koot, Rjwilmsi, Vegaswikian, Reinis, FlaBot, Quuxplusone, Rob*, Exe, YurikBot, Wavelength, Zozakral~enwiki, FF2010, LordTorgamus, Eskimbot, GiedrBac, Antonielly, Edenphd, KJS77, Hu12, Iridescent, Pmussler, Cydebot, TJ09, Coconut99 99, Merutak, Dougher, MER-C, Vladimir Bosnjak, Magioladitis, Jsmith030416, AleaB, Abednigo, Allstarecho, AllenDowney, Richard Katz, Mange01, LordAnubisBOT, Dorftrottel, VolkovBot, TXiKiBoT, Liko81, Russell Triplehorn, Andy Dingley, LanceBarber, Mormat, AlleborgoBot, AHMartin, JonatanAndersson, Cp111, Vghuston, XLinkBot, Addbot, Normandomac, Pbyhistorian, Teles, Yobot, Ptbotgourou, Rubinbot, Anjsimmo, KJedi, Sae1962, Jonesey95, NuclearDuckie, Anonymous2420, Ontist, DaveLiu, Klodr, Rafsan 033, Jcodetips, Unnikked and Anonymous: 78

8.2 Images

- **File:Template_Method_UML.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/52/Template_Method_UML.svg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Giacomo Ritucci
- **File:Template_Method_pattern_in_LePUS3.gif** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/2a/Template_Method_pattern_in_LePUS3.gif *License:* Public domain *Contributors:* Own work *Original artist:* Amnon Eden (User:Edenphd)
- **File:Text_document_with_red_question_mark.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a4/Text_document_with_red_question_mark.svg *License:* Public domain *Contributors:* Created by bdesham with Inkscape; based upon Text-x-generic.svg from the Tango project. *Original artist:* Benjamin D. Esham (bdesham)
- **File:Wikibooks-logo-en-noslogan.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/df/Wikibooks-logo-en-noslogan.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* User:Bastique, User:Ramac et al.

8.3 Content license

- Creative Commons Attribution-Share Alike 3.0