Template Method Design Pattern [Gamma et al]

This pattern is a preset format used as starting point in order to avoid recreation each time when required. This preset format is basically the algorithm. Applications provide an opportunity within its domain for a certain algorithm to perform a task with minimum requirements, therefore some additional steps can be added later to provide an extensible solution. This pattern defines a program skeleton of an algorithm in a method called template method and facilities the subclasses to add additional steps to redefine the algorithm if needed in future. Following are the variants of template method design pattern:

Template Method – Factory Method Compound Pattern [72]

The template method-factory method compound pattern consists of a combination of one instance of Template method pattern and one instance of Factory method pattern. The idea of this variant is to separate fixed and variable parts of a method: the fixed parts are implemented in the superclass as template methods while the variable parts are implemented in subclasses as primitive operations. However, the template pattern does not place any restrictions on the functionality of its primitive operations. Following is the UML diagram of the compound implementation:

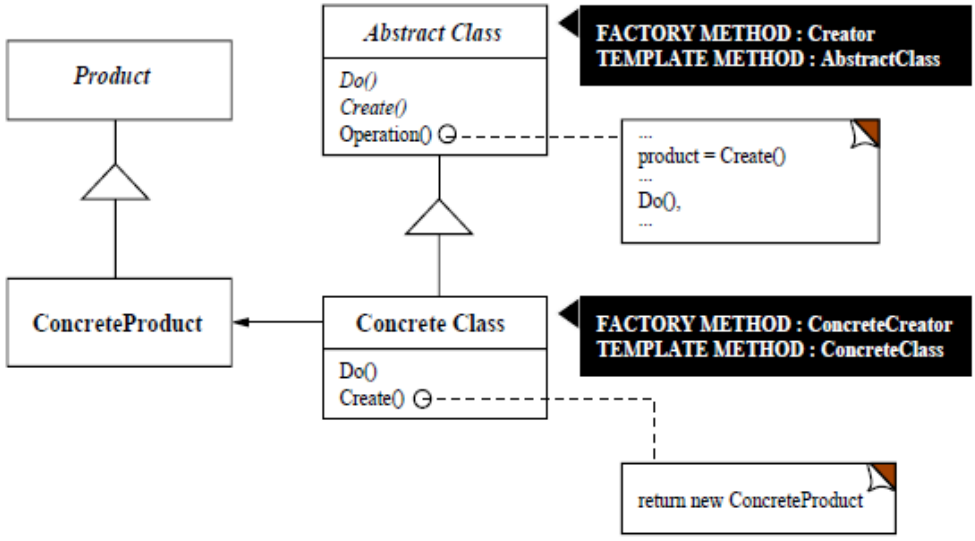

Figure 1.27 UML Diagram of Template Pattern variant

Enhanced Template Design Pattern [73]

The purpose of this variant is to develop a design, based on the Template Method design pattern, for testing object equality that simultaneously maximizes code reuse, encapsulation, data hiding, and minimizes semantic errors. We begin by considering a typical, but flawed, beginner's implementation of object equality. Following is the UML diagram:

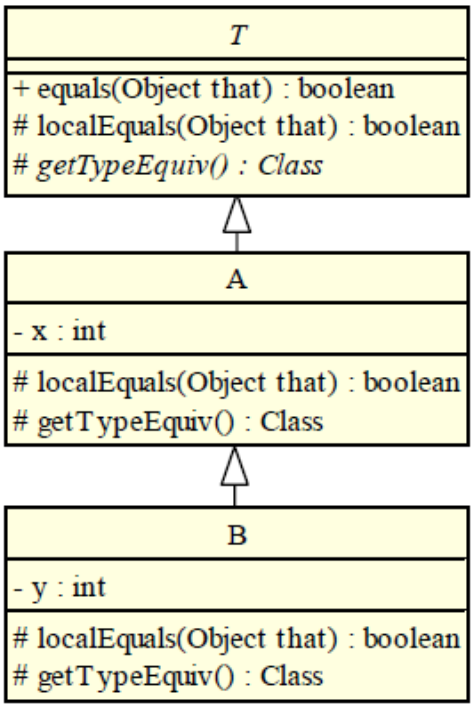

Figure 1.28 Template Pattern variant