**Template Method Design Pattern**

**Overview**

This design pattern defines barebone skeleton of algorithm in a method defined in core parent class that defers slightly from subclasses in some steps. Subclasses then inherit parent class methods and redefine certain steps differently without changing algorithms structure. Parent class defines mandatory steps ( methods that cannot be overridden ) some optional methods that can be overridden by subclasses inheriting parent classes. We can make common sense of template in code where we pick a template and then make some changes leaving some part intact.

A simple example would be making of tea and coffee where processes are similar such as boiling of water, mixing milk, and then there are steps such as coffee versus tea that is added in mixture which shows some activities carried over from parent, and some activities not shared at all.

**Benefits**

* It can lock number of steps that objects can do in shared situation and can do its own activity as well without duplicating process ( method duplication) by inheriting characters of parent.
* It provides flexibility to some of activities which are allowed to overridden.
* One object or one template means there is control in what can be done in subclasses and what cannot be done; it is clearly defined by parent.

**Problem/Scenario**

As an example of this would be two processes that are similar yet different in some way. There are lot of them where I have chosen cooking of pasta and cooking of noodle. Both of these processes involve following processes which are similar.

* Heating the pan
* Adding oil on pan
* Adding vegetable
* Adding pasta/noodle and stirring

Process that is not similar

In noodle cooking, you would add water and make soup but with pasta, you would eat it dry so adding water is not necessary.

**Before Refactor**

Before refactor, when making these two, they could be done in two objects which would have same processes repeated, and extra method to handle adding of water in noodle versus not adding on pasta stir fry.

**After Refactor**

After refactor, we can use inheritance data structure to carry out activities in parent which can be on children objects as well such as pasta and noodle objects can inherit method, then have an extra method to do additional process on noodle object.

Before/after link

**UML Diagram**

Please find it inside relevant project UML in pattern folder

/TemplateMethodPattern.png

**Reference**

Freeman Eric., Freeman Elisa. 2004. 1st ed. O’Reily Media Inc. 1995, Sebasttopo, CA