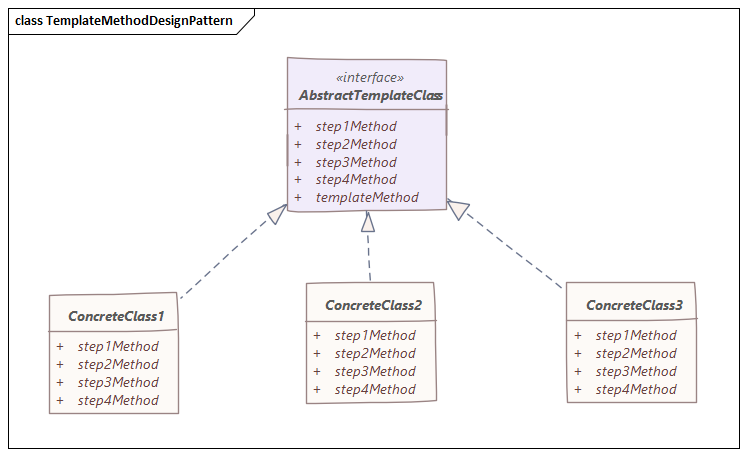
Definition :

**It makes it easier to implement complex algorithms by encapsulating logic in a single method.**

* The **Template Method** pattern is one of the behavioral design patterns identified by Gamma et al. in the book [*Design Patterns*](https://en.wikipedia.org/wiki/Design_Patterns).
* The **Template Method** pattern provides a method in a super-class, usually an abstract super-class, and defines the skeleton of an operation in terms of several high-level steps.
* Generally, these steps are implemented by additional *helper methods* in the same class as the template method.
* The helper methods may be either created as an *abstract method*, for which sub-classes are required to provide concrete implementations, or *hook methods,* which have empty bodies in the super-class.
* The **Template Method** design pattern is used to define an algorithm as a skeleton of operations and leave the details to be implemented by the child classes.
* In this way of implementation, the overall structure and sequence of the algorithm are preserved by the parent class.
* The **Template Method** pattern defines the sequential steps to execute a multi-step algorithm. We can provide a default implementation as well.
* In the **Template Method** pattern, we define a preset structure method called *template method* which consists of steps.
* These steps can be created as an *abstract method* which will be implemented by its sub-classes.
* In the **Template Method** pattern, an abstract class exposes defined way(s)/template(s) to execute its methods.
* The *template method* uses and defines the sequence of steps to perform the algorithm.



Advantage :

No code Duplication

As it uses Inheritance and not composition. Only a few methods need to be overridden

Let subclasses decide how to implement step in algorithm

Disadvantage:

Debugging and understanding the flow will become confusing at times