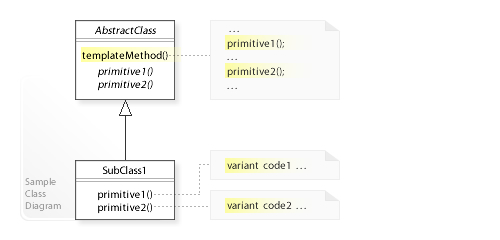
The Template Method pattern is known as a **behavioral**pattern, as it's used to manage algorithms, relationships and responsibilities between objects.

GOF definition:

**Defines the skeleton of an algorithm in a method, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithms structure.**

Class Diagram:



A high level blueprint for an algorithm to be completed by inheritors

Overall algorithm makes use of abstract member. Parent template method is invoked.

Strategy patter does this through composition while Template Method does this through inheritance.

**Strategy Vs Template Method:**

The key **difference** is that **Strategy Pattern** is about modifying a behavior of a context in runtime using **strategies**, while **Template Method Pattern** is about following a skeleton implementation of an algorithm and modifying its behavior by overriding methods of the skeleton class **in the** subclasses.

The main difference between the two is when the concrete algorithm is chosen.

With the **Template method pattern** this happens at compile-time by subclassing the template. Each subclass provides a different concrete algorithm by implementing the template's abstract methods. When a client invokes methods of the template's external interface the template calls its abstract methods (its internal interface) as required to invoke the algorithm.

class ConcreteAlgorithm : AbstractTemplate

{

void DoAlgorithm(int datum) {...}

}

class AbstractTemplate

{

void run(int datum) { DoAlgorithm(datum); }

virtual void DoAlgorithm() = 0; // abstract

}

In contrast, the **Strategy pattern** allows an algorithm to be chosen at runtime by containment. The concrete algorithms are implemented by separate classes or functions which are passed to the strategy as a parameter to its constructor or to a setter method. Which algorithm is chosen for this parameter can vary dynamically based on the program's state or inputs.

class ConcreteAlgorithm : IAlgorithm

{

void DoAlgorithm(int datum) {...}

}

class Strategy

{

Strategy(IAlgorithm algo) {...}

void run(int datum) { this->algo.DoAlgorithm(datum); }

}

## In summary:

* Template method pattern: **compile-time** algorithm selection by **subclassing**
* Strategy pattern: **run-time algorithm** selection by **containment**