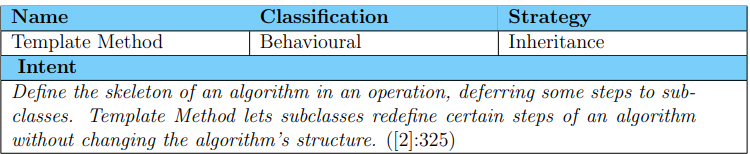
**Background**

* Template uses public inheritance (is-a relationship)
* Subclasses(derived), superclasses(parent)
* Avoid if-then-else and switch statements
* Non-virtual
  + Implementation happens in class that defines them
* Virtual
  + May be overridden in derived classes
* Pure-virtual
  + Not implemented in parent class, have to be implemented in child
* Pull up method refactoring
  + changing existing code in such a way that its behaviour is not altered, yet the internal structure is improved

**Template Method**



Diagram

Description automatically generated

* problems
  + If a change common to both components becomes necessary, duplicate effort must be expended
  + Promoted a false is-a relation
  + Added complexity
* Participants
  + AbstractClass
  + ConcreteClass
* Description: an algorithm with pluggable steps (virtual functions)
* Invariant operations:
  + steps in the algorithm that are the same for all (non-virtual)
* operational operations:
  + may be skipped under certain conditions (virtual)
* variant operations:
  + pluggable parts of the algorithm (pure virtual)
* code improvement?
  + Code duplication reduced
  + Easier to maintain
  + System is easier to extend
  + Coupling is reduced because classes that call the template method is separated from the concrete classes.
* Misconceptions:
  + Inheritance does not necessarily mean Template method
  + Templates != template method
  + Factory method != specialized version of Template method
* Related patterns
  + Strategy
  + Factory method
  + Adapter
  + Builder

Example

Graphical user interface, text, table

Description automatically generated with medium confidence