Contents

[Template Method: 1](#_Toc82074564)

[Intent: 1](#_Toc82074565)

[Participants: 1](#_Toc82074566)

[Application: 1](#_Toc82074567)

[Memento Pattern 1](#_Toc82074568)

[Motivation: 1](#_Toc82074569)

[Intent: 1](#_Toc82074570)

[Implementation: 1](#_Toc82074571)

[Factory Method 1](#_Toc82074572)

[Intent: 1](#_Toc82074573)

[Participants 2](#_Toc82074574)

[Application: 2](#_Toc82074575)

[Prototype Pattern 2](#_Toc82074576)

[Intent: 2](#_Toc82074577)

[Implementation: 2](#_Toc82074578)

[Participants: 3](#_Toc82074579)

[Abstract Factory Pattern 3](#_Toc82074580)

[Intent: 3](#_Toc82074581)

[Implementation: 3](#_Toc82074582)

[Participants: 3](#_Toc82074583)

[Strategy Pattern 4](#_Toc82074584)

[Intent: 4](#_Toc82074585)

[Implementation 4](#_Toc82074586)

[Participants: 4](#_Toc82074587)

[State 5](#_Toc82074588)

[Intent: 5](#_Toc82074589)

[Implementation: 5](#_Toc82074590)

[Participants: 5](#_Toc82074591)

# Template Method:

## Intent:

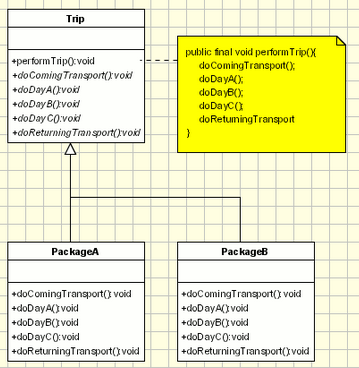
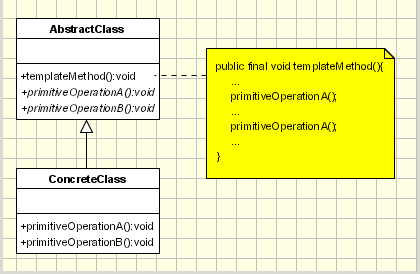
* Define a skeleton of an algorithm in an operation, deferring some steps to subclasses.
* Allows subclasses to redefine certain steps of an algorithm without letting them change the algorithm’s structure.

## Participants:

* Abstract Class
  + Define methods that concrete class must define to implement steps of an algorithm
* Concrete Class
  + Implements the methods to carry out subclass-specific steps of the algorithm
* Operation Note
  + Define the steps in the Template Method

## Application:

* When common behaviour is identified among classes
* Abstract class containing all common code



# Memento Pattern

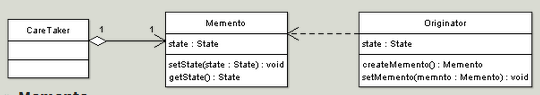
## Motivation:

* Sometimes necessary to store the internal state of an object and can restore it at a later stage. Example, error or failure.

## Intent:

* Capture internal state of an object without violating encapsulation and thus providing a mean for restoring the object to initial state when needed.

## Implementation:



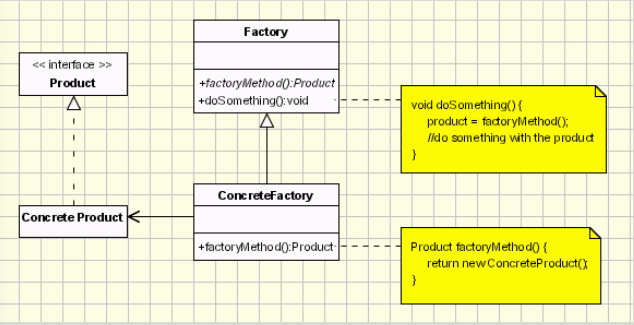
* Care Taker:
  + Responsible for keeping the Memento
* Originator:
  + Creates memento object capturing the originator’s internal state
  + Use Memento to restore to previous state
* Memento:
  + Stores Internal state of Originator
  + Must have interface to care taker:
    - Not allow any operations or any access to internal state stored by memento.
  + Originator Interface:
    - Allow the originator to access any state

# Factory Method

## Intent:

* It defines an interface for creating an object but leaves the choice of its type to the subclasses.

Implementation:



## Participants

* AbstractFactory:
  + Interface for ConcreteFactories
* ConcreteFactories:
  + Produce the concrete Products
* ConcreteProduct:
  + Creates the object
* AbstractProduct:
  + Interface for defining concreteProducts

## Application:

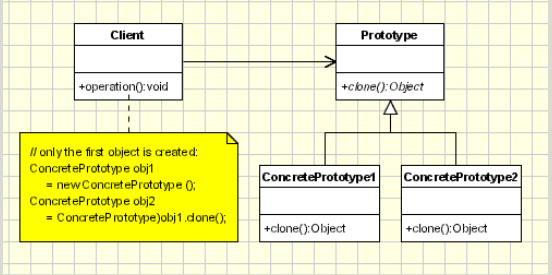
* When class can’t anticipate the type of objects it is supposed to create

# Prototype Pattern

## Intent:

* Specifying the kind of objects to create using a prototypical instance
* Creating new objects by copying this prototype.

## Implementation:



## Participants:

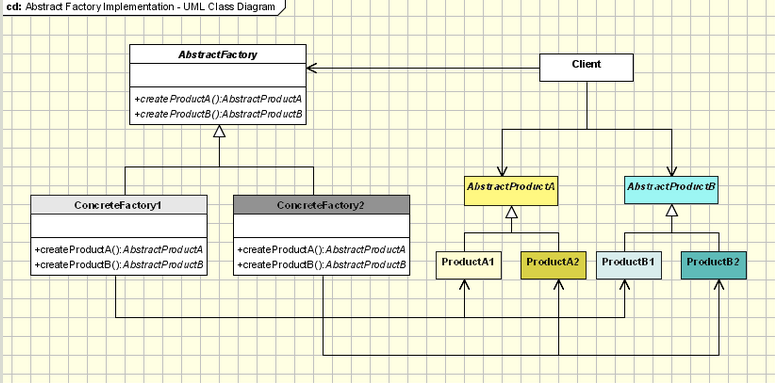
* Client:
  + Creates new object by asking a prototype to clone itself
* Prototype:
  + Declares an interface for cloning itself
* ConcretePrototype:
  + Implements the operations to clone itself

# Abstract Factory Pattern

## Intent:

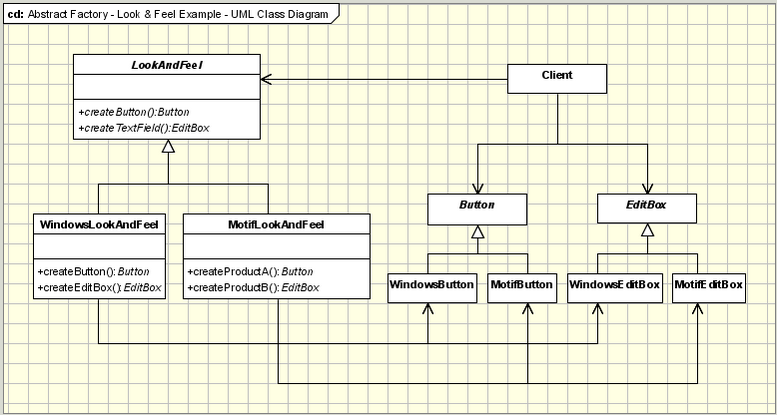
* Offers the interface for creating a family of related objects, without explicitly specifying their classes.

## Implementation:



## Participants:

* Abstract factory:
  + Declares interface for operations that create abstract products
* Concrete Factory:
  + Implmenets operations to create concrete products
* Abstract Products:
  + Declares an interface for a type of product object
* Concrete Product:
  + Defines a product to be created by a corresponding Concrete Factory
* Client:
  + Uses interface declared by Abstract Factory and Abstract Product classes

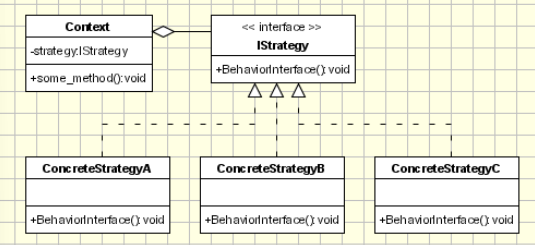


# Strategy Pattern

## Intent:

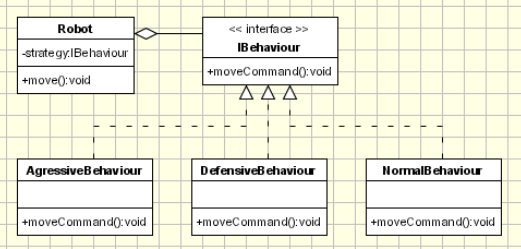
* Define a family of algorithms, encapsulate each one, and make them interchangeable.

## Implementation



## Participants:

* Strategy:
  + Defines common interface to all supported algorithms
  + Context uses Strategy to call algorithm defined by a ConcreteStrategy
* ConcreteStrategy
  + Each concrete strategy implements an algorithm
* Context:
  + Contains reference to strategy object
  + May define interface that lets strategy accessing its data

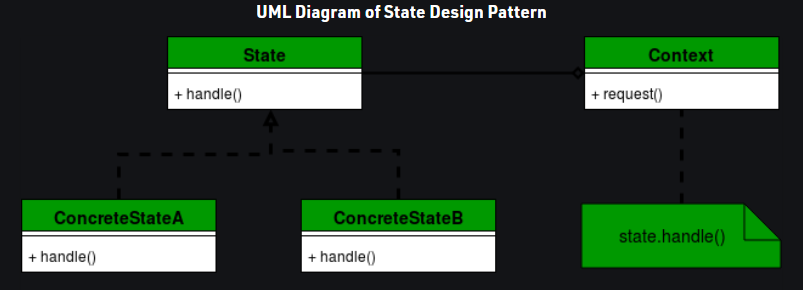


# State

## Intent:

* Object changes its behaviour based on its internal state

## Implementation:



## Participants:

* State:
  + Defines interface for declaring what each concrete state should od
* Context:
  + Defines interface to client
  + Maintains reference to concrete object
* ConcreteState:
  + Provides implementation for methods defined in state