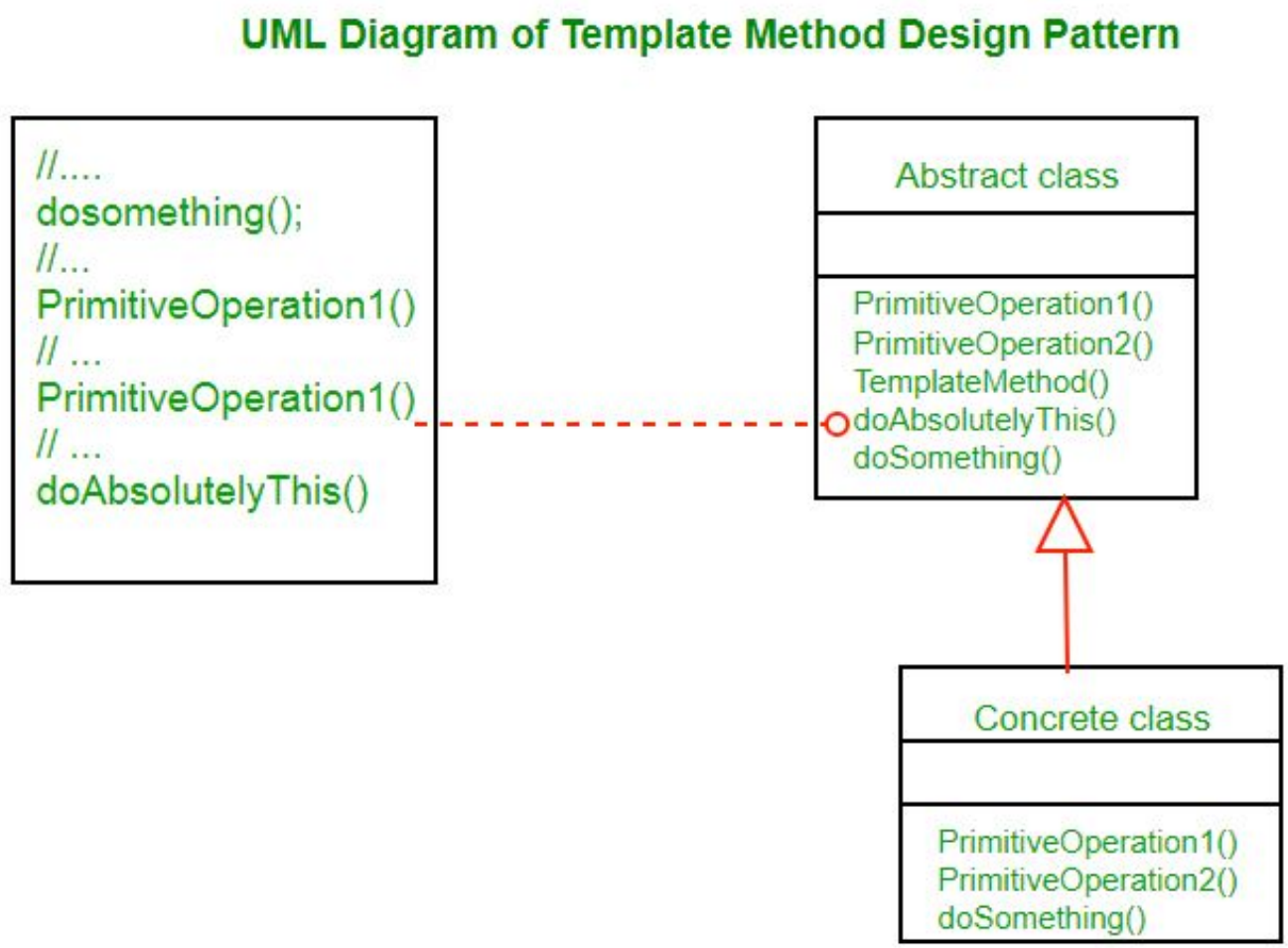
Template Method Design Pattern

Template method design pattern is to define an algorithm as skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm is preserved by the parent class.
Template means Preset format like HTML templates which has fixed preset format.Similarly in template method pattern,we have a preset structure method called template method which consists of steps.This steps can be abstract method which will be implemented by its subclasses.

This behavioral design pattern is one of the easiest to understand and implement. This design pattern is used popularly in framework development. This helps to avoid code duplication also.



Source : Wikipedia

- **AbstractClass** contains the templateMethod() which should be made final so that it cannot be overridden. This template method makes use of other operations available in order to run the algorithm but is decoupled for the actual implementation of these

methods. All operations used by this template method are made abstract, so their implementation is deferred to subclasses.

- **ConcreteClass** implements all the operations required by the templateMethod that were defined as abstract in the parent class. There can be many different ConcreteClasses.

Lets see an example of the template method pattern.

```
abstract class OrderProcessTemplate
{
    public boolean isGift;

    public abstract void doSelect();

    public abstract void doPayment();

    public final void giftWrap()
    {
        try
        {
            System.out.println("Gift wrap successfull");
        }
        catch (Exception e)
        {
            System.out.println("Gift wrap unsuccessful");
        }
    }

    public abstract void doDelivery();

    public final void processOrder(boolean isGift)
    {
        doSelect();
        doPayment();
        if (isGift) {
            giftWrap();
        }
        doDelivery();
    }
}
```

```
class NetOrder extends OrderProcessTemplate
{
    @Override
    public void doSelect()
    {
        System.out.println("Item added to online shopping cart");
        System.out.println("Get gift wrap preference");
        System.out.println("Get delivery address.");
    }

    @Override
    public void doPayment()
    {
        System.out.println
                   ("Online Payment through Netbanking, card or Paytm");
    }

    @Override
    public void doDelivery()
    {
        System.out.println
                    ("Ship the item through post to delivery address");
    }

}

class StoreOrder extends OrderProcessTemplate
{

    @Override
    public void doSelect()
    {
        System.out.println("Customer chooses the item from shelf.");
    }

    @Override
    public void doPayment()
    {
        System.out.println("Pays at counter through cash/POS");
    }
```

```
    @Override
    public void doDelivery()
    {
        System.out.println("Item deliverd to in delivery counter.");
    }

}

class TemplateMethodPatternClient
{
    public static void main(String[] args)
    {
        OrderProcessTemplate netOrder = new NetOrder();
        netOrder.processOrder(true);
        System.out.println();
        OrderProcessTemplate storeOrder = new StoreOrder();
        storeOrder.processOrder(true);
    }
}
```

Output :

```
Item added to online shopping cart
Get gift wrap preference
Get delivery address.
Online Payment through Netbanking, card or Paytm
Gift wrap successfull
Ship the item through post to delivery address

Customer chooses the item from shelf.
Pays at counter through cash/POS
Gift wrap successfull
Item deliverd to in delivery counter.
```

The above example deals with order processing flow. The OrderProcessTemplate class is an abstract class containing the algorithm skeleton. As shown on note, processOrder() is the method that contains the process steps. We have two subclasses NetOrder and

StoreOrder which has the same order processing steps.

So the overall algorithm used to process an order is defined in the base class and used by the subclasses. But the way individual operations are performed vary depending on the subclass.

When to use template method

The template method is used in frameworks, where each implements the invariant parts of a domain's architecture, leaving "placeholders" for customization options.

The template method is used for the following reasons :

- Let subclasses implement varying behavior (through method overriding)
- Avoid duplication in the code , the general workflow structure is implemented once in the abstract class's algorithm, and necessary variations are implemented in the subclasses.
- Control at what points subclassing is allowed. As opposed to a simple polymorphic override, where the base method would be entirely rewritten allowing radical change to the workflow, only the specific details of the workflow are allowed to change.

Reference :

Wikipedia

This article is contributed by **Saket Kumar**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

Design Patterns | Set 1 (Introduction)

Design Patterns | Set 2 (Factory Method)

Command Pattern

Observer Pattern | Set 1 (Introduction)

Observer Pattern | Set 2 (Implementation)

Singleton Design Pattern | Implementation

Decorator Pattern | Set 1 (Background)

The Decorator Pattern | Set 2 (Introduction and Design)

Decorator Pattern | Set 3 (Coding the Design)

Strategy Pattern | Set 1 (Introduction)

Strategy Pattern | Set 2 (Implementation)

Adapter Pattern

Iterator Pattern

Curiously recurring template pattern (CRTP)

Flyweight Design Pattern

Article Tags : Design Pattern

Be the First to upvote.

☐ To-do ☐ Done

3

Based on **1** vote(s)

Feedback/ Suggest Improvement | Add Notes | Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments | Share this post!

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Company-wise
Topic-wise
Contests
Subjective Questions

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos