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

Run on IDE

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