The Template Design Pattern

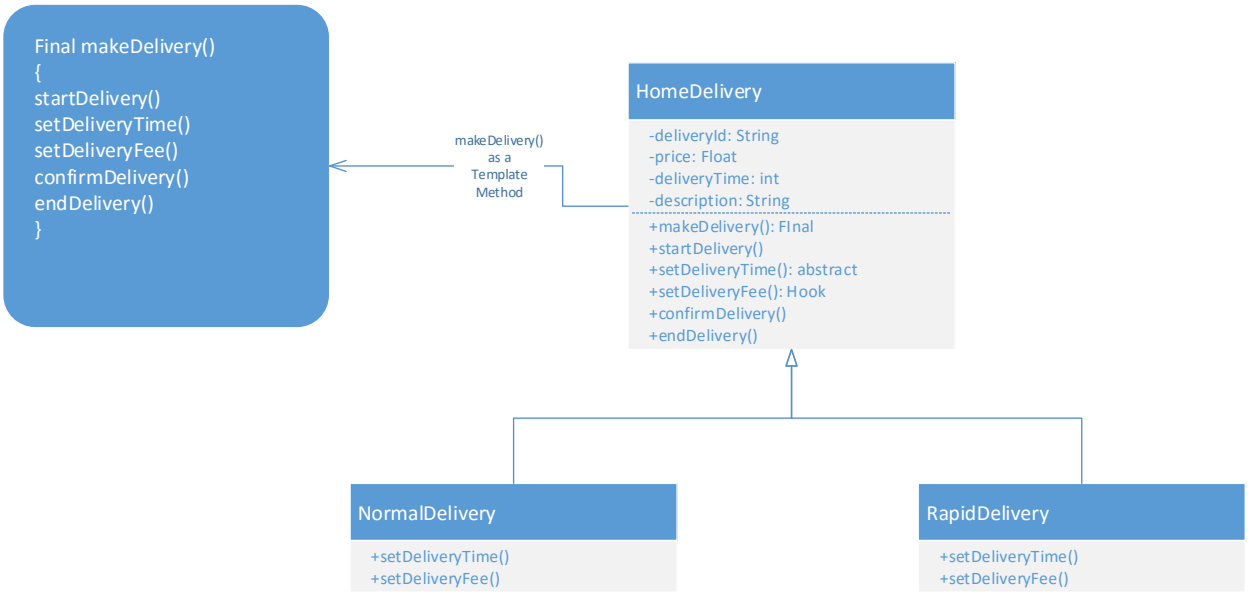
Object Oriented Goal: Code Re-use. Designing to interface and not to implementation.

Definition: The Template Method Pattern 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 algorithm's structure.

Implemented on classes:

1) HomeDelivery
   a) NormalDelivery
   b) RapidDelivery

Design Snapshot:



Design Explanation:

We want our subclasses to implement methods in a certain fashion just like a steps of an algorithm. Moreover, we want to enforce that some methods be implemented by the subclasses itself as a subclass specific implementation. We need a template so that we can make a steps of methods which must be followed by the subclasses

In the delivery options, we have two different methods of delivery but the steps of processing both the types is same. So we define a template method which indicates steps of the algorithm which needs to be implemented by the subclasses.

- The HomeDelivery class has a makeDelivery() method which acts as a template for the subclasses (NormalDelivery and RapidDelivery). The makeDelivery() is declared as final so that the subclasses cannot change the method. Subclass needs to implement the

setDeliveryTime() method and the setDeliveryFee() method because every type of HomeDelivery has different delivery time and fees for the delivery.

- So, the subclass will use the makeDelivery() method. All the other methods are already implemented in the abstract class HomeDelivery. Only setDeliveryTime() and setDeliveryFee() needs to be implemented in the subclasses.
- There is setDeliveryFee() hook method which is implemented in the abstract class. The hook method may or may not be implemented by the subclasses. It depends on the subclasses if they want to implement it or not. So this method is kept as void. Subclasses which implement this method will add behavior to this method.