**Template Method Design Pattern:** The Template Method pattern is a [behavioral design pattern](https://www.geeksforgeeks.org/behavioral-design-patterns/) that defines the skeleton of an algorithm or operations in a superclass (often abstract) and leaves the details to be implemented by the child classes. It allows subclasses to customize specific parts of the algorithm without altering its overall structure.

* The overall structure and sequence of the algorithm are preserved by the parent class.
* Template means Preset format like HTML templates which has a fixed preset format. Similarly in the template method pattern, we have a preset structure method called template method which consists of steps.
* These steps can be an abstract method that will be implemented by its subclasses.
* This is one of the easiest to understand and implement. This design pattern is used popularly in framework development and helps to avoid code duplication.

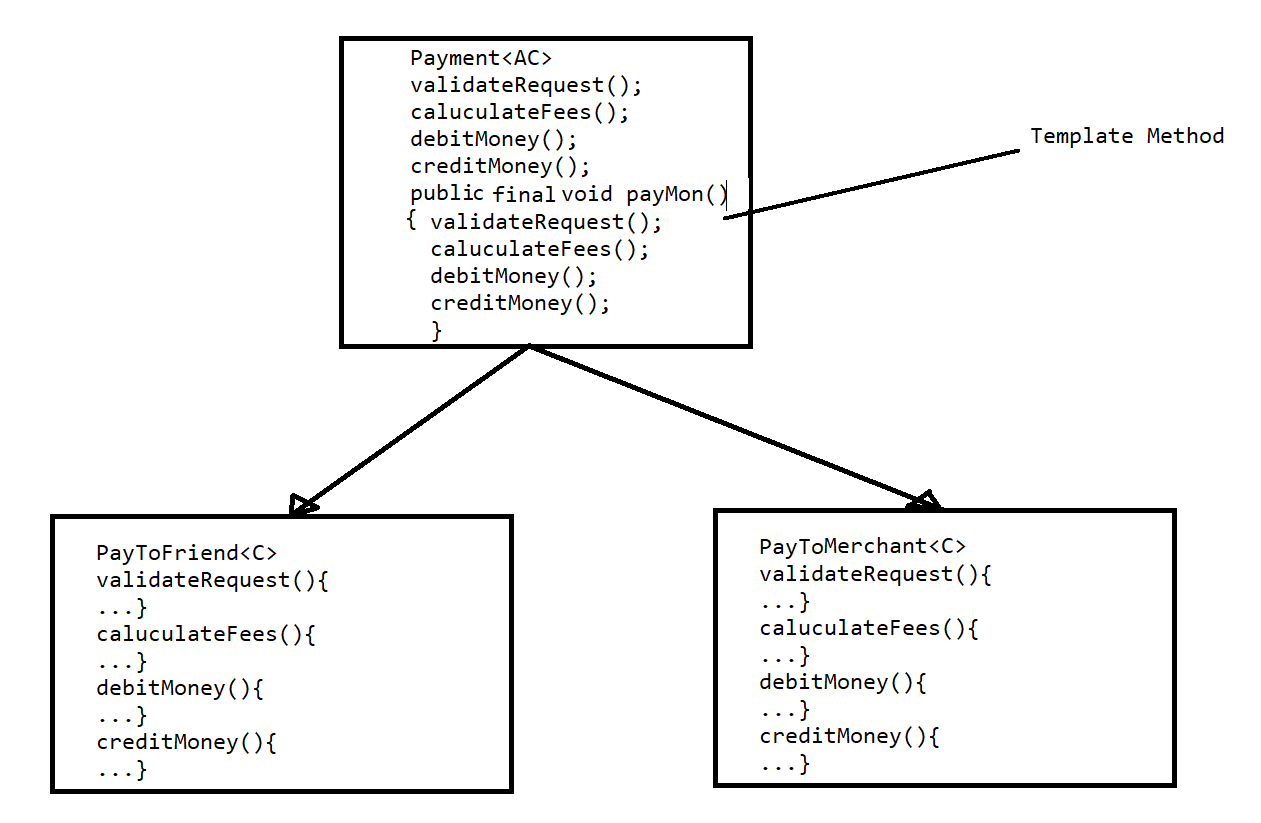
Example: Let’s consider a scenario where we have a process for making different types of beverages, such as tea and coffee. While the overall process of making beverages is similar (e.g., boiling water, adding ingredients), the specific steps and ingredients vary for each type of beverage.

**When to use the Template Method Design Pattern?**

* **Common Algorithm with Variations**: When you have an algorithm with a common structure but with some varying steps or implementations, the Template Method pattern helps to encapsulate the common steps in a superclass while allowing subclasses to override specific steps.
* **Code Reusability**: If you have similar tasks or processes that need to be performed in different contexts, the Template Method pattern promotes code reuse by defining the common steps in one place.
* **Enforcing Structure**: It’s beneficial when you want to enforce a specific structure or sequence of steps in an algorithm while allowing for flexibility in certain parts.
* **Reducing Duplication**: By centralizing common behavior in the abstract class and avoiding duplication of code in subclasses, the Template Method pattern helps in maintaining a clean and organized codebase.

**When not to use the Template Method Design Pattern?**

* **When Algorithms are Highly Variable**: If the algorithms you’re working with vary greatly in their structure and steps, and there’s minimal commonality between them, using the Template Method pattern might not be appropriate as it may lead to excessive complexity or unnecessary abstraction.
* **Tight Coupling Between Steps**: If there’s tight coupling between the steps of the algorithm, such that changes in one step necessitate changes in other steps, the Template Method pattern may not provide sufficient flexibility.
* **Inflexibility with Runtime Changes**: If you anticipate frequent changes in the algorithm structure or steps at runtime, using the Template Method pattern might not be the best choice, as it relies on predefined structure and behavior.
* **Overhead in Abstraction**: If the cost of abstraction and inheritance outweighs the benefits of code reuse and structure enforcement, it’s better to avoid using the Template Method pattern and opt for simpler solutions.
  + When you want all class to follow the specific steps to process the tasks but also need to provide the flexibility that each class can have their own logic in that specific steps.



* + In the above diagram we showed that doing payment steps are common for friend & merchant.
  + So we set the common template in a method and made it has final.
  + Sub classes like PayToFriend & PayToMerchant can implement those steps with their own logic.