Class Diagram Explanation: Design Patterns Used.

In the MoneyTransferFacility class we used the Factory Method design pattern to abstract and encapsulate the process of: Choosing the right provider endpoint that will satisfy the coming user operation.   
It is a little bit of Strategy pattern as well, as each class of the ProviderEndpoint hierarchy handles the coming operation differently, based on the API specification by the company that this class is supposed to call. So different steps (algorithms) to call the different APIs are encapsulated inside each class, and the Factory Method chooses the right Strategy.

The same idea also is applied for paying bills, in the MoneyTransferFacility class, too.

The creator class is MoneyTransferFacility, which is an abstract class. And the InstapayTransferFacility class is the concrete class which provides implementations for both CreateProviderEndpoint and CreateBillingEndpoint.

The abstract strategy classes ProviderEndpoint and BillingEndpoint. And the concrete strategies include: MockupFawryEndpoint, MockupAlhlyEndpoint, MockupCIBEndpoint, MockupVCashEndpoint; for the ProviderEndpoint. And for the BillingEndpoint the concrete strategies are: MockupElectricityBillingEndpoint, MockupWaterBillingEndpoint, MockupGasBillingEndpoint.

The concrete strategies here are the classes responsible for making the HTTP requests to the intended API provided by the Bank/Wallet Provider/Utility.

Now the only part in the code that is concerned about deciding the proper endpoint to use is the Factory Method in the InstapayTransferFacility.