**Design Principles**

+ S: Each class is responsible for a particular task (Controller for logics, Handlers for users’ interactions…)

+ O: open for extension & close for modification

* Handlers and controllers class extended from a base class
* Use interface and abstract classes
* On the other hand, the class PlaceOrderController has the methodcalculateShippingFee to calculate shipping fees. Clearly, this design violates the principle since we must modify the old codes when changing the formula of shipping fees. We could deal with this issue by using abstraction: create an interfaceShippingFeeCalculatorand let the controllerdepends on it instead of a concrete implementaion