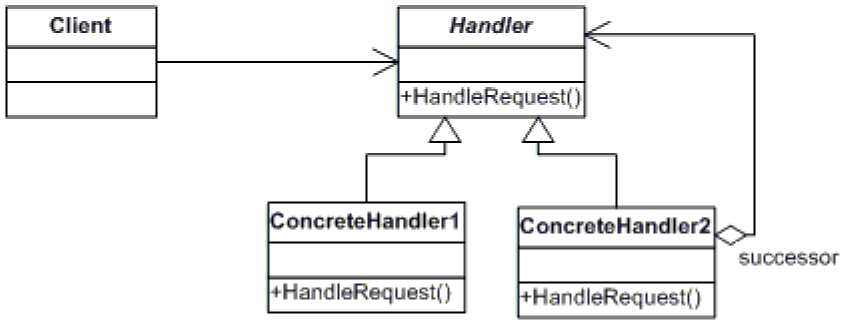
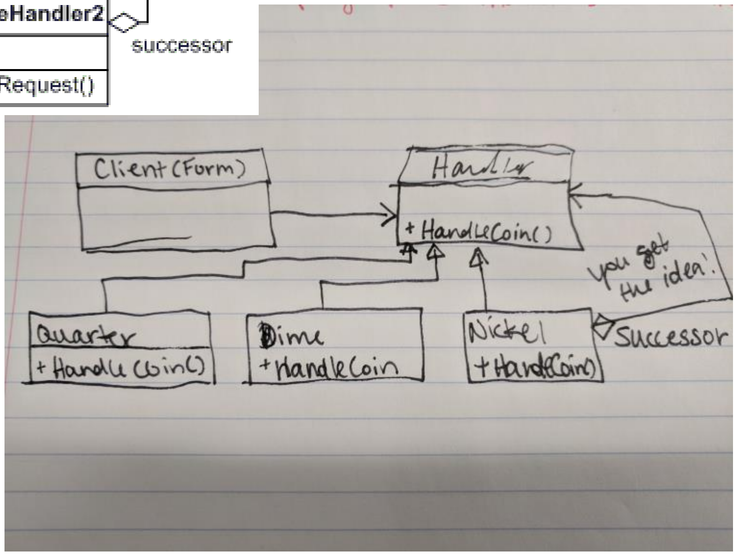
The Chain-Of-Responsibility Pattern

The Chain of Responsibility design pattern is described as a pattern to “Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.” (dofactory.com) The pattern chains the receiving objects together and passes any requests from object to object until it is able to find one capable of handling it.

The first thing that this made me think of is a vending machine where all coins are accepted through the same slot instead of having multiple slots for each coin. The coin is then routed through the machine based on its size, and it’s given a value so that the machine knows how much money has been inserted. I think that this would be an okay example to implement with code, based on the UML diagram I would basically just need the form, the handler interface, and classes for each type of coin. I would probably have a combobox with each type of coin and when you selected one it would be handled by whichever concrete handler corresponded to the type of coin selected. The handle request method might look something like if (coin == quarter) {do stuff}, etc.





I’m not particularly sure if I fully understand the successor part of it, but from what I’ve been reading it just makes sure that if the first thing cant handle it, it goes to the successor and so on, but I’m not sure if I did that correctly in my UML diagram. I like this pattern I think that even though it was labeled as low use there are a lot of different things it could be used for, like apps where levels of authentication are needed to do things, or different things need to be done based on what the user selects. I think it seems very useful and has a lot of real world applications.