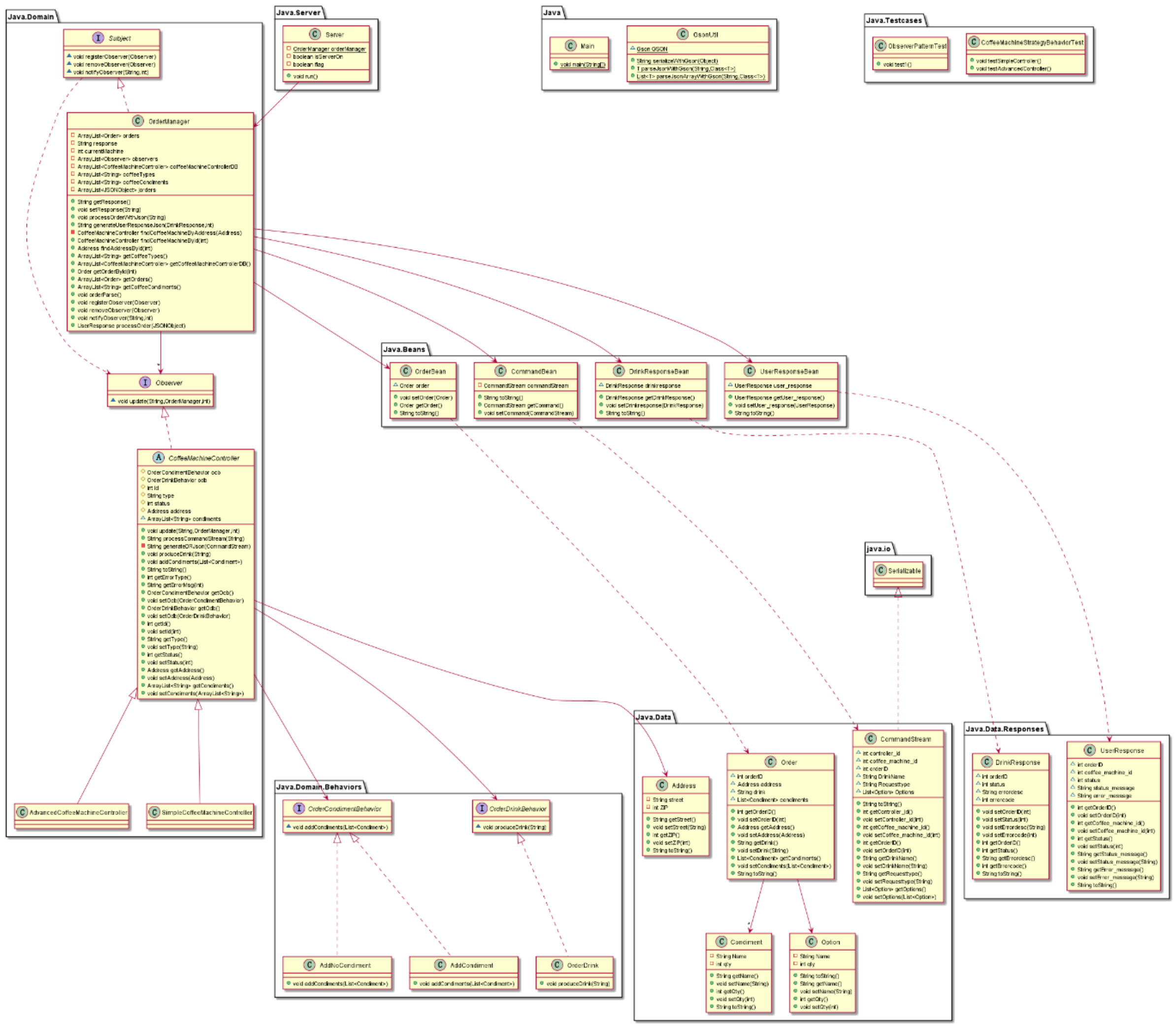
1. Class Diagram

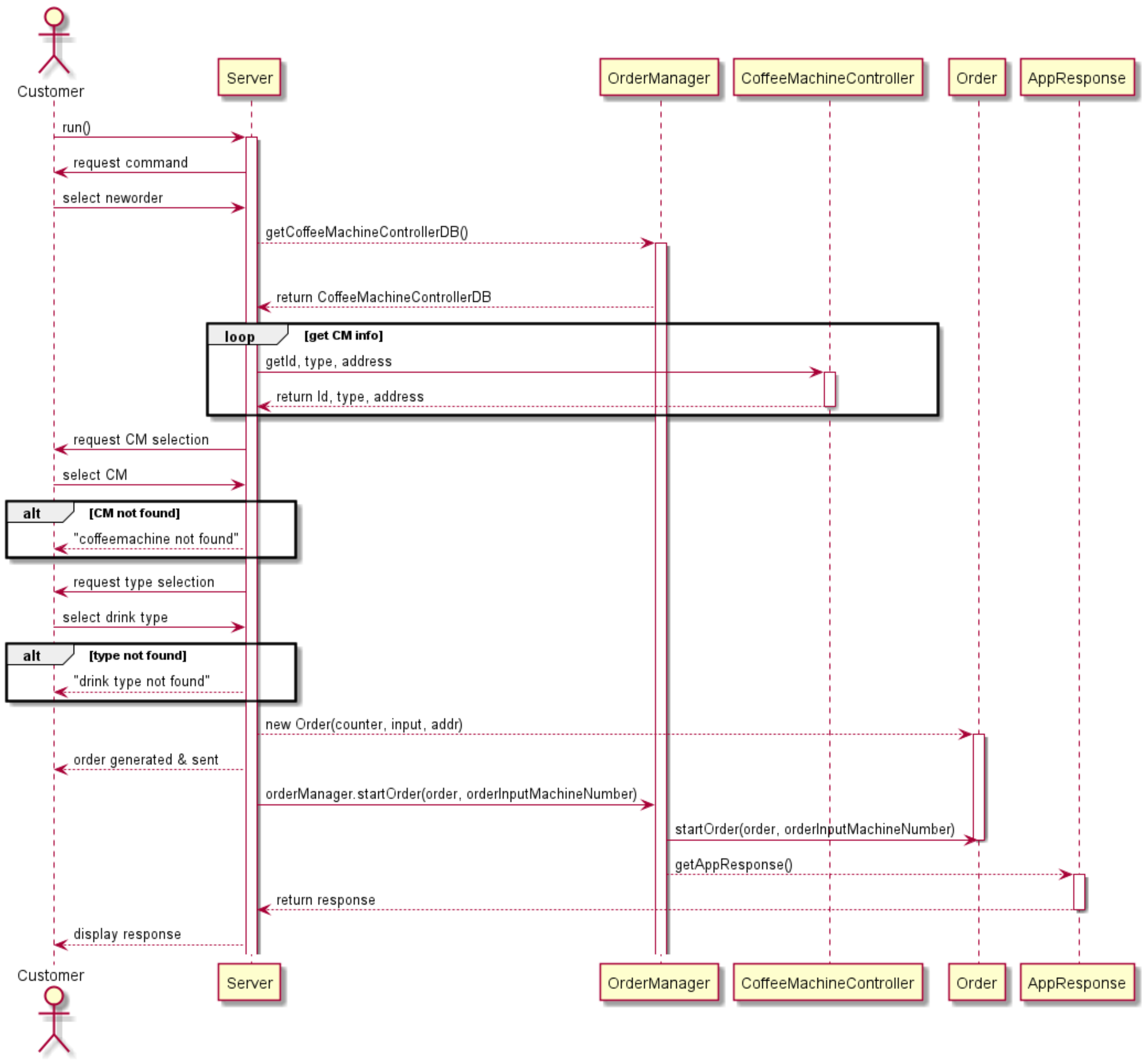


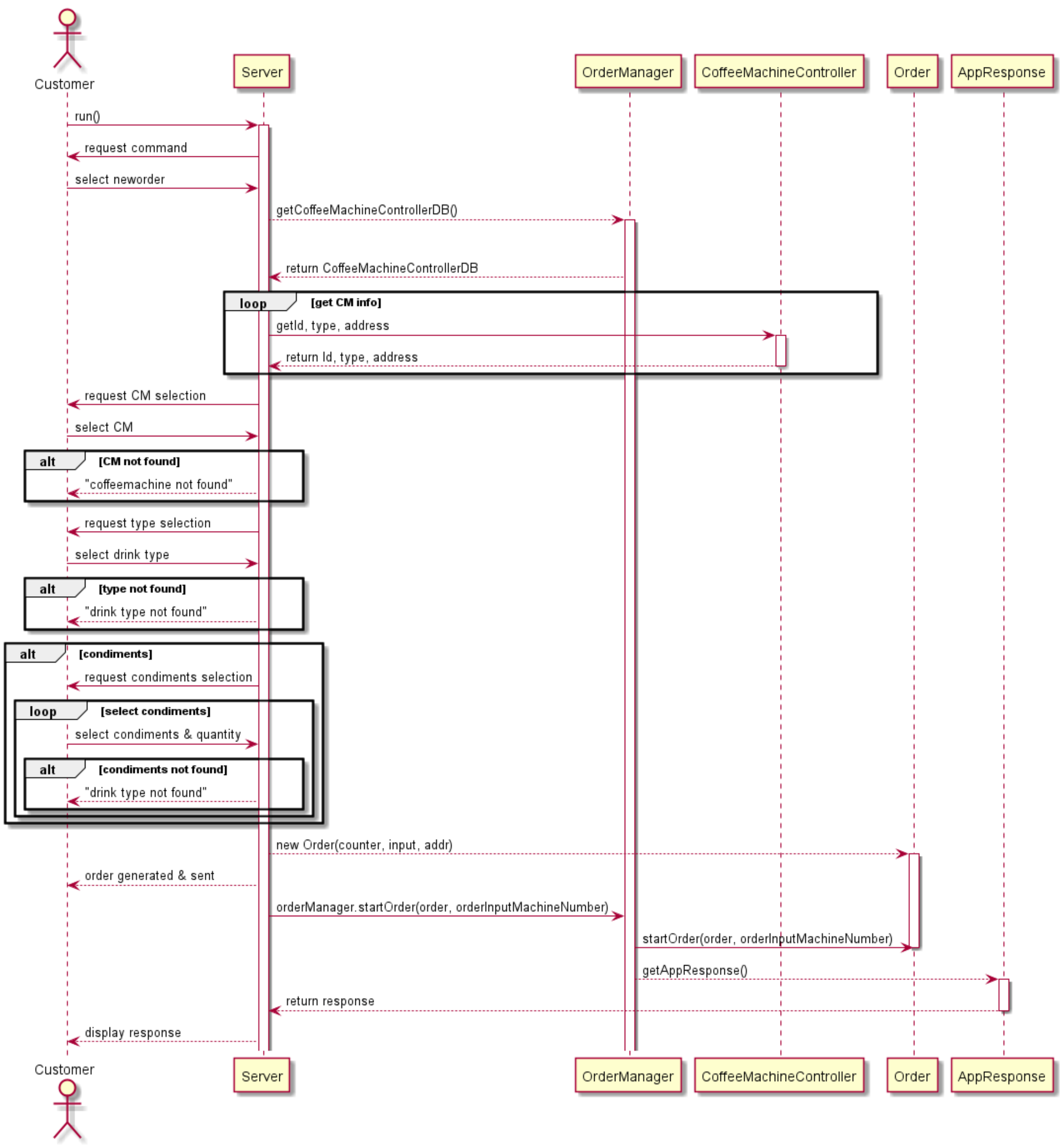
For our class diagram, we divide all the classes into 3 layer: Domain, Data and Server (Besides the Main (to start the program), GsonUtil(for parsing String to Json), and testcases. We implement strategy pattern for the CoffeeMachineController's behaviors, which can improve the reusage of code in this part and it also brings us convenience for implementation new CoffeeMachineController behaviors in the future.

We applied the Observer Pattern for the OrderManager and CoffeeMachineController classes. We take the Observer as subject to be observed and CoffeeMachineController as observer. Once the OrderManger got new command, it will notify all the registered observers to react to the new command.

Also, we applied both inheritance to further code efficiency and reduce duplication to keep our system as clean as possible.

2.Sequence Diagram -UC1&2





Here is the SSD for the Use Case 1. In our system, the custom will only interact with our server class. The server collects all the input from user and send to OrderManager. The OrderManager will check those inputs to make sure they are acceptable for the system. Then, it will begin produce CommadStream for CoffeeMachineControlle and Response for user. This part showed the single-responsibility design principle.