CSE 460

Programming Assignment Report

Posting ID: [3594-385]

[04/29/2020]

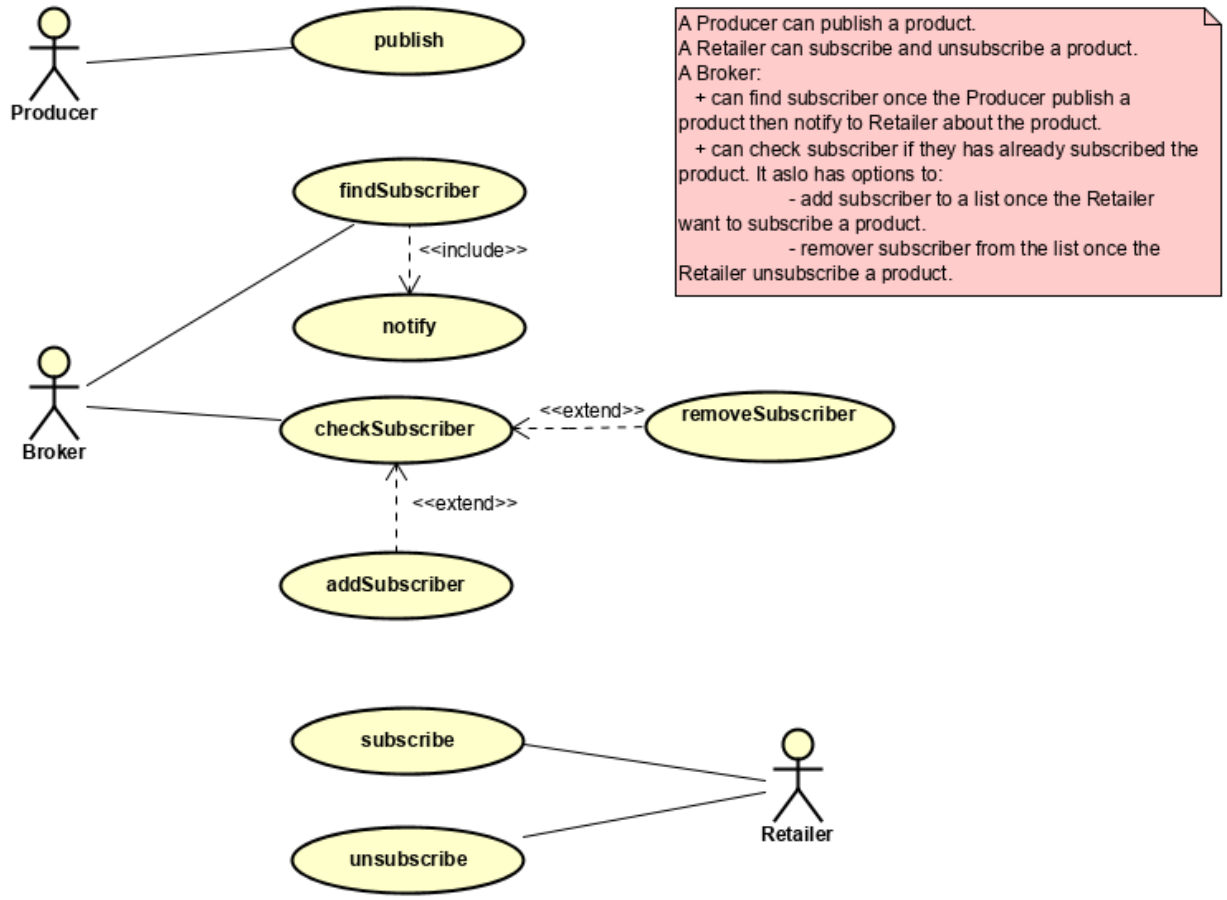
Spring 2020

1. Description and Assumptions

This software implements Publisher-Subscriber pattern. The software uses Supply-Demand class as an interface for the program. It takes inputs to the program and processes the functionalities of Publishers, Subscribers, and Broker.

Publishers publish their products. Subscribers subscribe the products to get notification about the products; Subscribers also can stop subscribing the products by unsubscribe function. Broker is responsible for controlling the function of Publishers and Subscribers. Broker knows when Publishers publish the products and it notifies the products to Subscribers who has subscribed the products. Broker works as a man in the middle of Publishers and Subscribers. Therefore, Publishers and Subscribers are independent on one another.

2. Use Case Diagrams



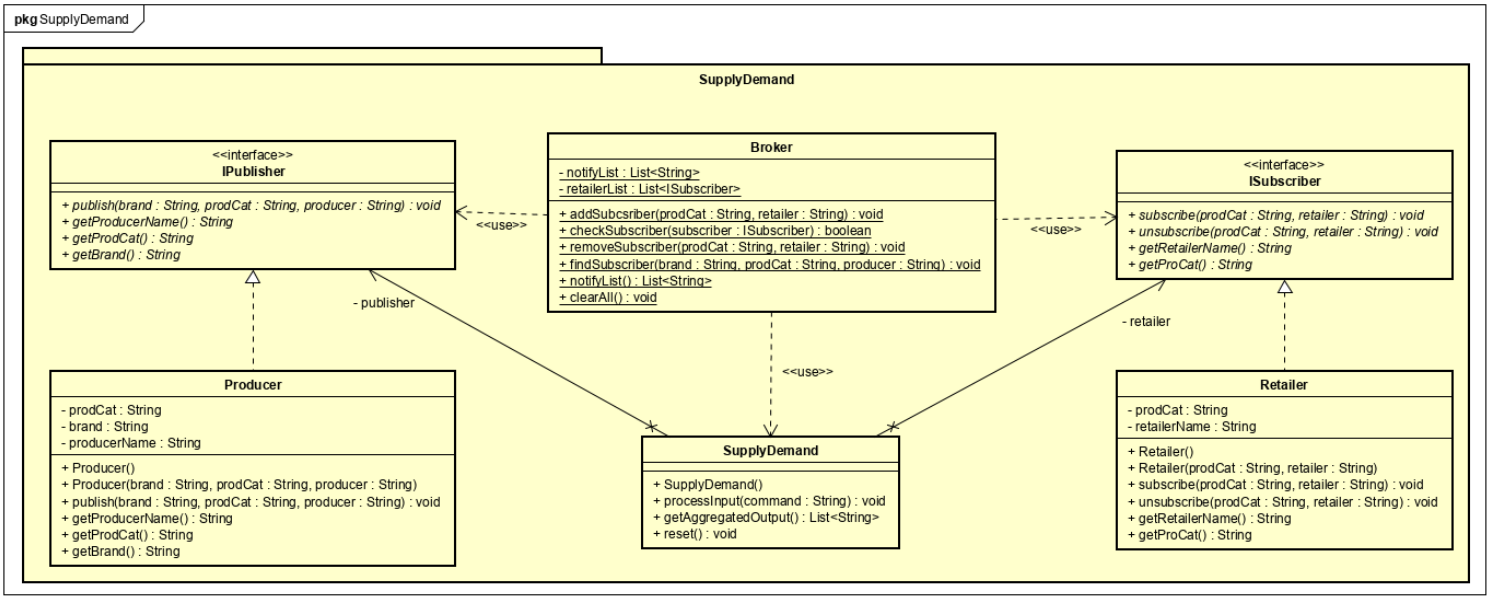
2.1 Use case diagram of the Supply-Demand software system.

- Normal Case 1:
    1. Retailer A subscribes product “banana” via subscribe.
    2. Broker checks Retailer A via checkSubscriber:
        a. Broker checks Retailer A not in the list.
        b. Broker adds Retailer A to the list of subscribers via addSubscriber.

3. Producer B publishes product “banana” from brand “C” via publish.
4. Broker finds subscribers who subscribe product “banana” in the subscriber list via findSubscriber.
    a. Broker finds Retailer A in the list.
    b. Broker notifies the product from Producer B to Retailer A.

- Normal Case 2:
    1. Retailer A subscribes product “banana” via subscribe.
    2. Broker checks Retailer A via checkSubscriber:
        a. Broker checks Retailer A not in the list.
        b. Broker adds Retailer A to the list of subscribers via addSubscriber.
    3. Producer B publishes product “banana” from brand “C” via publish.
    4. Broker finds subscribers who subscribe product “banana” in the subscriber list via findSubscriber.
        a. Broker finds Retailer A in the list.
        b. Broker notifies the product from Producer B to Retailer A.
    5. Retailer A unsubscribes product “banana” via unsubscribe.
    6. Broker checks Retailer A via checkSubscriber:
        a. Broker checks Retailer A in the list.
        b. Broker removes Retailer A from the list via removeSubscriber.
    7. Retailer D subscribes product “banana” via subscribe
    8. Producer B publishes product “banana” from brand “C” via publish.
    9. Broker finds subscribers who subscribe product “banana” in the subscriber list via findSubscriber.
        a. Broker finds Retailer D in the list.
        b. Broker notifies the product from Producer B to Retailer D.
- Abnormal Case 1:
    1. Retailer A unsubscribes product “apple juice” via unsubscribe.
    2. Broker checks Retailer A via checkSubscriber:
        a. Broker checks Retailer A not in the list.
- Abnormal Case 2:
    1. Retailer A subscribes product “banana” via subscribe.
    2. Broker checks Retailer A via checkSubscriber:
        a. Broker checks Retailer A not in the list.
        b. Broker adds Retailer A to the list of subscribers via addSubscriber.
    3. Retailer A subscribes product “banana” via subscribe.
    4. Broker checks Retailer A via checkSubscriber:
        a. Broker checks Retailer A in the list.
    5. Retailer A subscribes product “banana” via subscribe.
    6. Broker checks Retailer A via checkSubscriber:
        a. Broker checks Retailer A in the list.

3. Class Diagrams

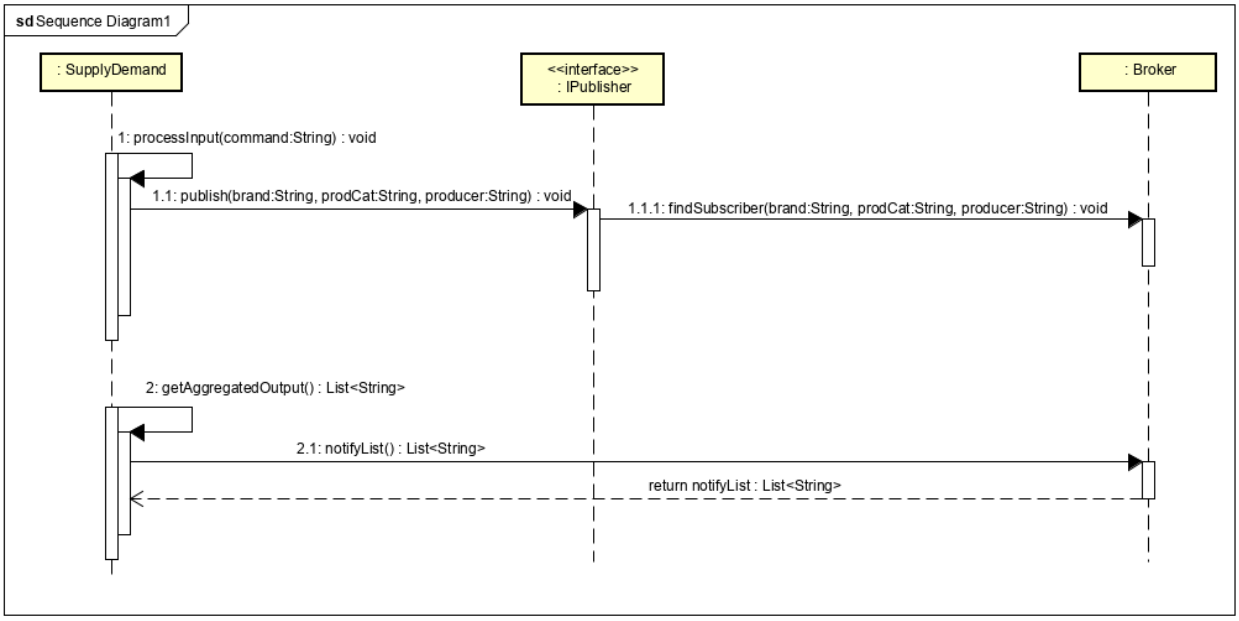


3.1 Class Diagram of Supply-Demand software system.

- Broker class: has responsibilities for communication between Producer class and Retailer class via IPublisher and ISubscriber interfaces. Therefore, Producer and Retailer are not directly contacting each other or exchanging information/data. Broker has dependency relationships with IPublisher, ISubscriber, and SupplyDemand classes.
- SupplyDemand class: working as an interface of the program. It takes input and sends signals to Broker, Producer, and Subscriber via processInput, getAggregatedOutput, and reset operations.
- Producer class: is a concrete class which implements IPublisher interface. It has a generalization relationship with IPublisher interface. It provides details of a producer such as brand of the product, product category, and the producer's name.
- IPublisher interface: is an interface of Producer that Broker can use (depend on) without knowing its implementation.
- ISubscriber interface: is an interface of Subscriber that Broker can use (depend on) without knowing its implementation.

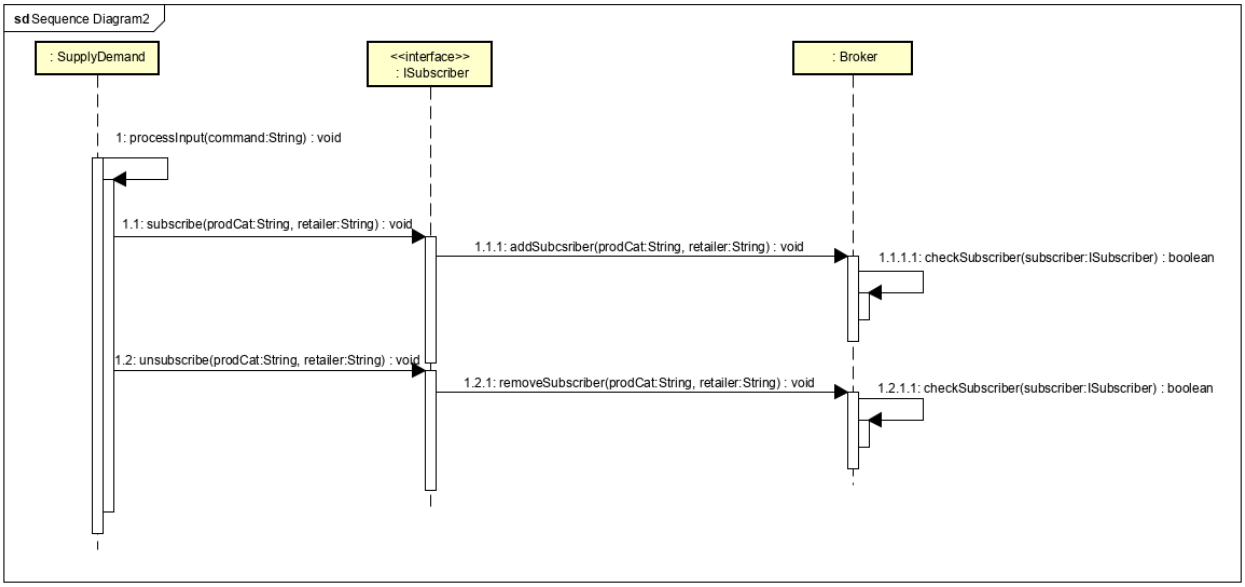
Note: The details of each class are in the Astah file.

4. Sequence Diagrams



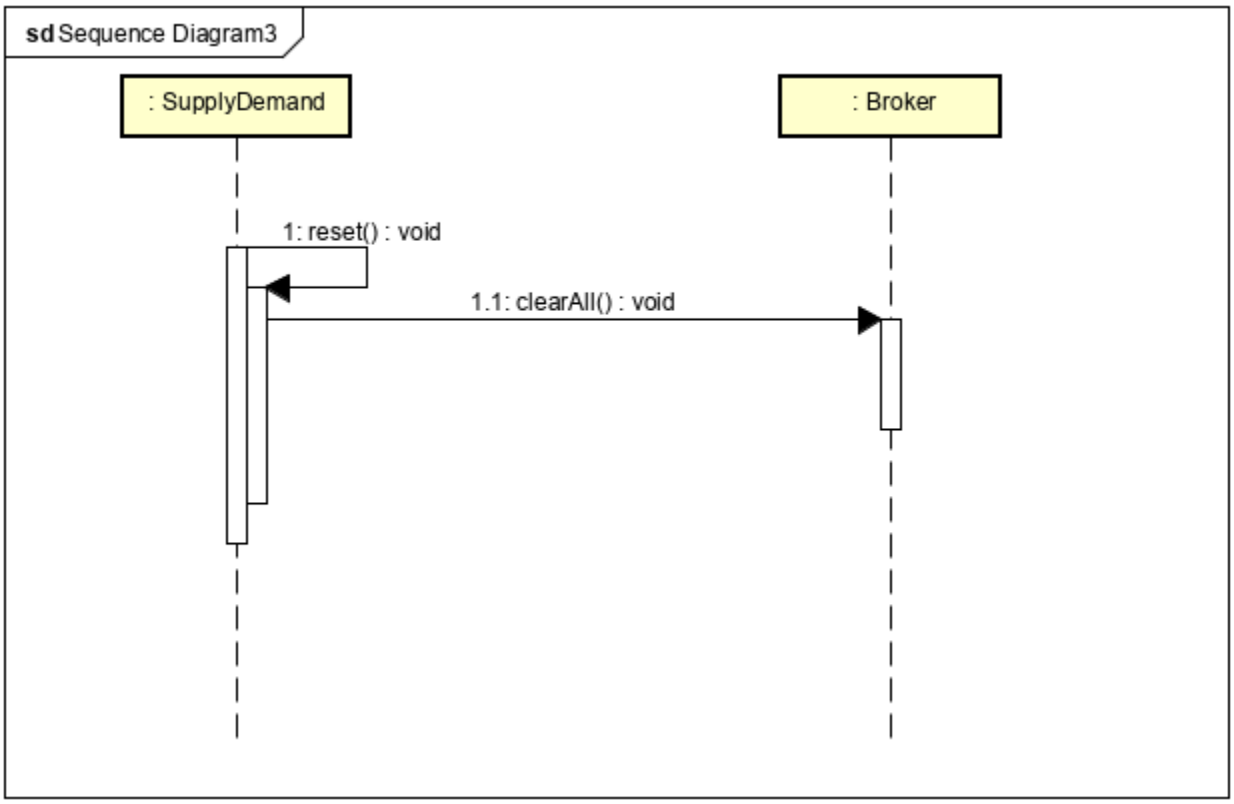
4.1 Sequence Diagram of Publishing and Notifying Processes.

The sequence diagram 4.1 describes the process of publishing and getting notification. When a SupplyDemand object receives input “publish”, it signals to Producer that they can publish their product with the given input via IPublisher interface. The Broker gets a signal message from Publisher via IPublisher interface, it can start to find subscribers who subscribe the given product. Once Broker has done its job, nothing needs to be returned.
SupplyDemand object triggers getAggregatedOuput and messages to the Broker to get the notification list. The Broker returns the notification list to the SupplyDemand.



4.2 Sequence Diagram of Subscribing and Unsubscribing Processes.

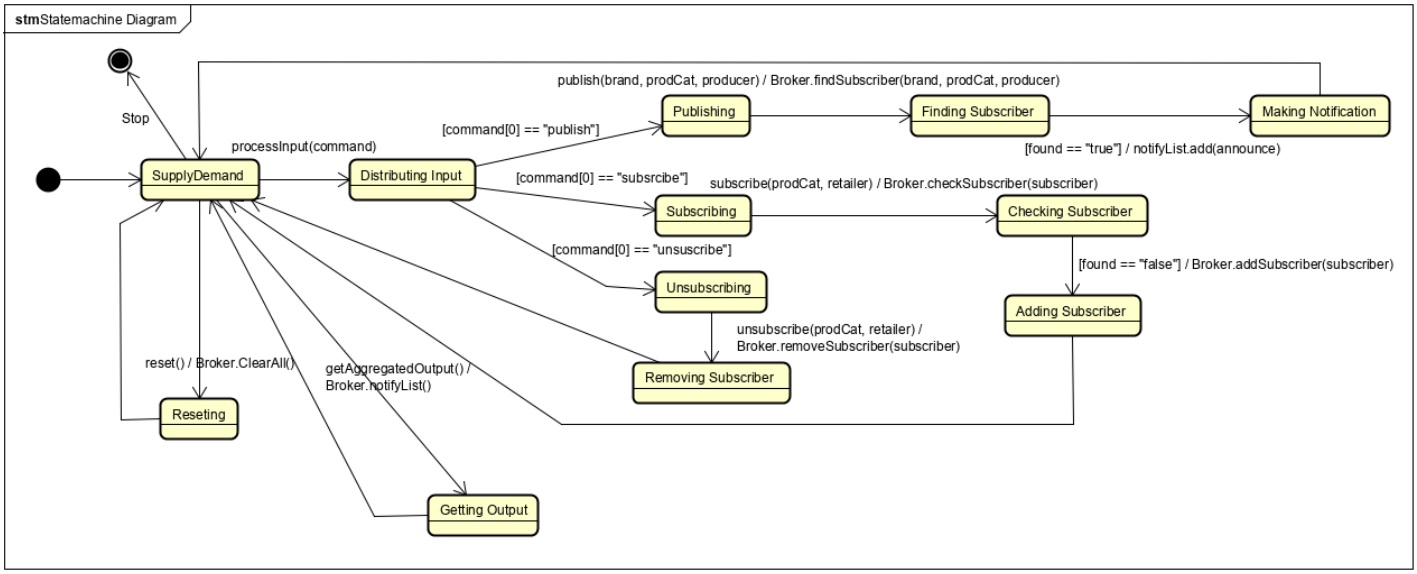
The sequence diagram 4.2 describes the processes of subscribing and unsubscribing. When the SupplyDemand object receives input “subscribe”, it messages to Retailer that they can subscribe the product via ISubsciber interface. The Broker gets message addSubscriber from ISubsciber and starts to check if the subscriber has not subscribed the product earlier, then it can add the subscriber to the list. The Broker does not return any message once it has done the job.
Once SupplyDemand object receives input “unsubscribe”, it messages to Retailer that they can unsubscribe the product via ISubscriber interface. The Broker gets message removeSubscriber from ISubscriber and starts to check if the subscriber has subscribed the product earlier, then it can remove the subscriber from the list. The Broker does not return any message once it has done the job.



4.3 Sequence Diagram of Resetting Process.

The sequence diagram 4.3 describes the process of resetting. When SupplyDemand object triggers the reset event, it messages to the Broker to delete all the data in the notification list as well as the subscriber list. The Broker returns nothing when the process is finished.

5. State Machine Diagrams



5.1 State Machine Diagram of the Supply-Demand Software System.

The state machine 5.1 describes the states and transitions of how the system behaves and works between SupplyDemand-Subscriber, SupplyDemand-Publisher, SupplyDemand-Broker, Broker-Subscriber, and Broker-Publisher.

The program starts and also ends with the SupplyDemand state.

1. When processInput is fired, it transitions to the state Distributing. From here, the state detects the input and decides which states should be transitioned to by the guard condition.
    - SupplyDemand-Publisher-Broker: from [command[0] == “publish”], it will transition to the state Publishshing which belongs to Publisher. The Publisher can now fire the trigger “publish” and the state transitions to Find Subscriber which belongs to Broker. From here Broker will find subscriber, and with the guard condition [found == “true”] it transitions to the state Making Notification. After that, it will come back the SupplyDemand state, and the program keeps running until it stops.
    - SupplyDemand-Subscriber-Broker: from [command[0] == “subscribe”], the state will be transitioned to Subscribing state which belongs to Subscriber. Now Subscriber can fires “subscribe” to transition to Checking Subscriber which belongs to Broker. From here, Broker checks whether the subscriber subscribes the product. After the guard [found== “false”], the state is being transitioned to Add Subscriber, the Broker now can add subscriber to the list. It transitions to the SupplyDemand state, and the program keeps running until it stops.
    - SupplyDemand-Subscriber-Broker: from [command[0] == “unsubscribe”], the state transitions to Unsubscribing which belongs to Subscriber; the subscriber can fire unsubscribe trigger and transition to Removing Subscriber which belongs to Broker. It tells Broker to withdraw the subscriber from the list. It transitions to the SupplyDemand state, and the program keeps running until it stops.

2. When getAggregatedOutput is fired, it transitions to Getting Output from Broker. The Broker can now return the list of subscribers. After that, it transitions to the SupplyDemand state, and the program keeps running until it stops.
3. When reset is fired, it transitions to Resetting state, it tells Broker to delete all the data from the subscriber list and notification list. After that, it transitions to the SupplyDemand state, and the program keeps running until it stops.

6. Appendix & Credits

All of the diagrams in this project are created from Astah.