Design:

Some unique designs have been implemented in this petrol system. First, the pump will wait for a supervisor to reset before it can online again. Second, the supervisor will record the transaction every time a customer pumped gas from a pump. While he had reset every pump in the station, he will have a chance to report the Record he has to the Station. And the Record at station side will be updated base on this. The Record at Station side is the one used to calculate how much does a station need pay to the Company. Since the company can collect bill anytime they want, the stations have been given operation resetRecord(), pay-Owe(amount:Real) and restock(volume:Real) to simulate the real scenario.

Verification:

To verify the system invariant was add into UML specification to help with design. The invariant has been added into UML specification as verification. Notice the ‘false‘ in red is intentionally left.

* haveStation: The entire system shall at least have one station by a Company
* noFreeSupport: The wholesale price a Company provide shall not be free
* enoughMoney: A customer shall have enough money before engaging
* handleOnePumpOnly: The customer shall only interact with one pump at one time
* noFreeGas:The price on the pumps shall not be zero
* onePumpOneStation: No common pumps cross stations
* profitRetailPrice: Check the retail price is profitable
* readyToUse: Check if all pumps are ready to use
* moreThan2PumpsEach: station shall have more than two pumps
* onlyOneStoreEvery: station shall have and only have one store for storage
* samePriceAtPump: This checks the sale price at pump station wides
* capwhether: the storage is normal

Class Diagram:

Class diagram demonstrates the relationship between classes, and it can also provide the role name for easy observation. Start from the head - company; they own stations and sale gas to stations with a wholesale price, they collect the bill from station whenever they want. While that happens, Station::payOwe handles the situation. Pre/Post-conditions are used to ensure Station have money to pay, and Station holds a Record. This Record at Station gets cleared after they pay off the bill.

Object Diagram:

Partial view of object diagram is displayed here to demonstrate the idea implementation of this petrol station. In this diagram, we create instances for each class to explain how the petrol station work.

Taking slices of the cumulative 3D charts shows us how the degree distribution changes. The log-log charts below show the progression of these changes as the aggregation window gets larger.