A Petrol Station designed with OCL Language

Junjie He, Heying Cai
jhe654@aucklanduni.ac.nz, hcai665@aucklanduni.ac.nz

Introduction

A petrol company owns a number of petrol stations. Each petrol station consists of a number of petrol pumps. Each pump can be reset by a supervisor, after which it can pump out petrol from a common store, recording at each stage the volume and cost of petrol pumped so far. When pumping is finished, the total cost is recorded by the supervisor and subsequently paid by the customer. Each pump in a petrol station sells petrol at the same price, although this price may vary between stations. The common store can be re-stocked by the petrol company at any time. The company supplies petrol at the same cost to each station. Records are kept of the total volume of petrol supplied to each station, as well as the outstanding amount owed by each station. The amount owed to the company by a station is paid when requested by the company.

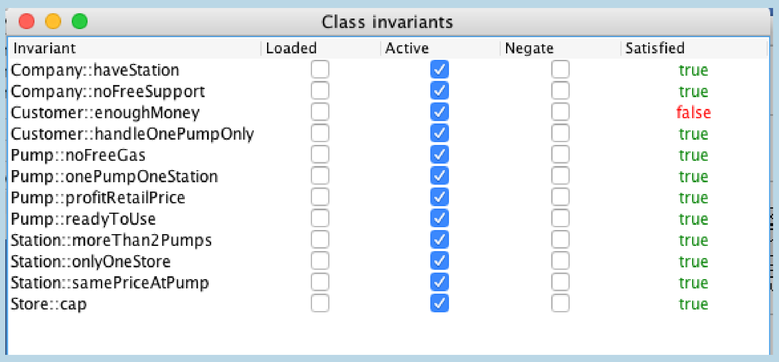
Design Concepts

- *Record* helps logging the transaction system wide. **Station** keeps an instance of *Record* so the amount of money a Station need pays to the Company is clear. *Record* also exists as a note of the supervisor, so they can record the transaction at pump and report to the Station later on.
- *Pumping* is an operation of a **Pump**, it checks prerequisites of both user and pump before it performs the action of providing gas. And after the pumping it will be idle and in a status of waiting for supervisor.
- *reportRecords* is the major design in **Supervisor**, it make sure a supervisor have content to report and the supervisor is working at particular station to avoid chaos. It also ensure the supervisor have logged all pumps before she reports.
- *samePriceAtPump* ensures same petrol price at a station by add up all rtPrice from pumps in a station, and divide by number of pumps. And check if that result of division matches rtPrice from each pump.

Verification

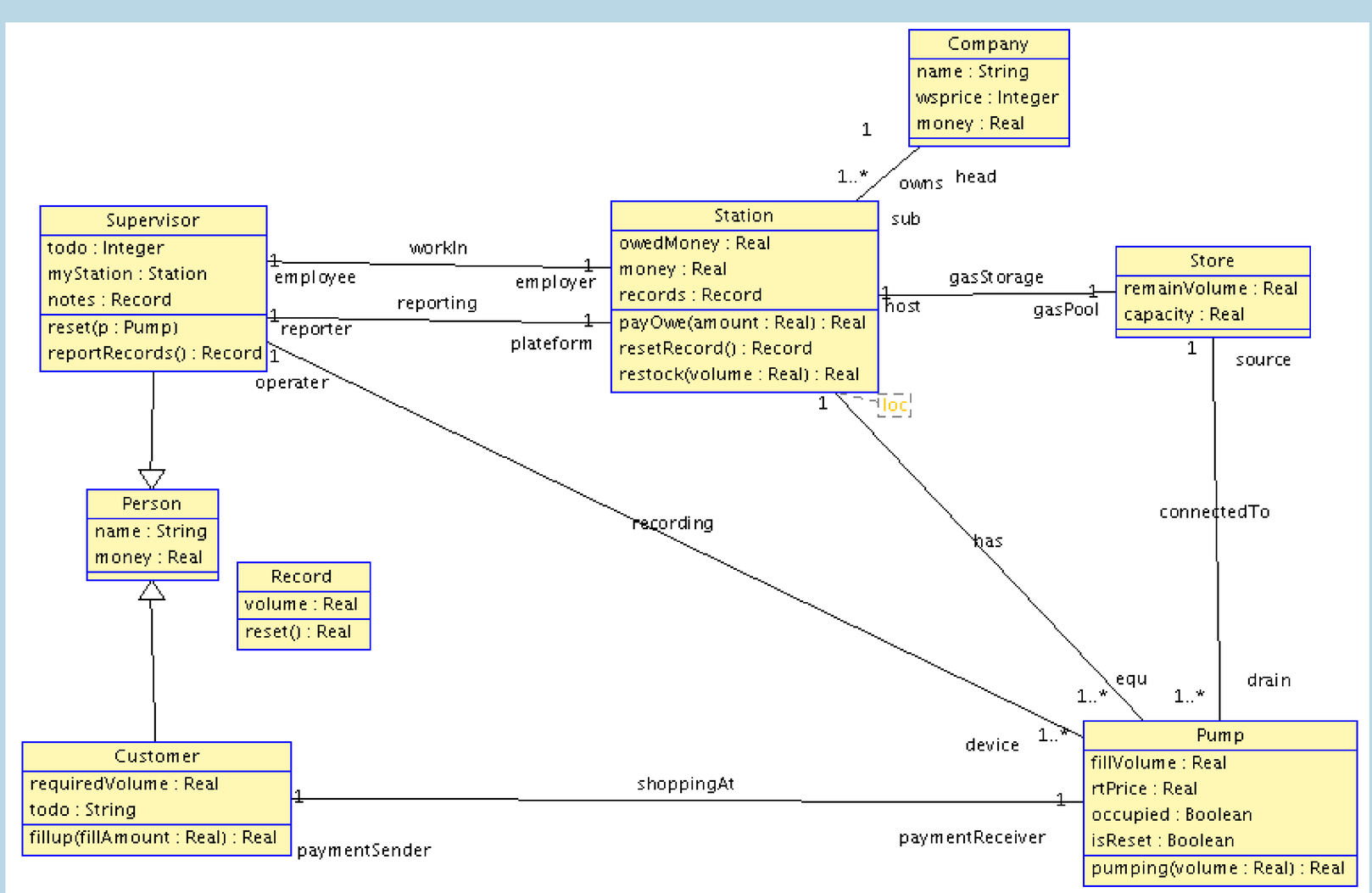
In order to verify the system, Class Invariants were added. While a object have not match the given requirement, it will show in red like the window below.

- *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 engage
- *handleOnePumpOnly* The customer shall only interact with only pump a time
- *noFreeGas* The price on pump shall not be zero
- *onePumpOneStation* No common pumps cross station
- *profitRetailPrice* Check the retail price is profitable
- *readyToUse* Check if all pumps are ready to use
- *moreThan2Pumps* Each station shall have more than two pumps
- *onlyOneStore* Every station shall have and only have one store for storage
- *samePriceAtPump* This checks the sale price at pump station wides
- *cap* Whether the storage is normal



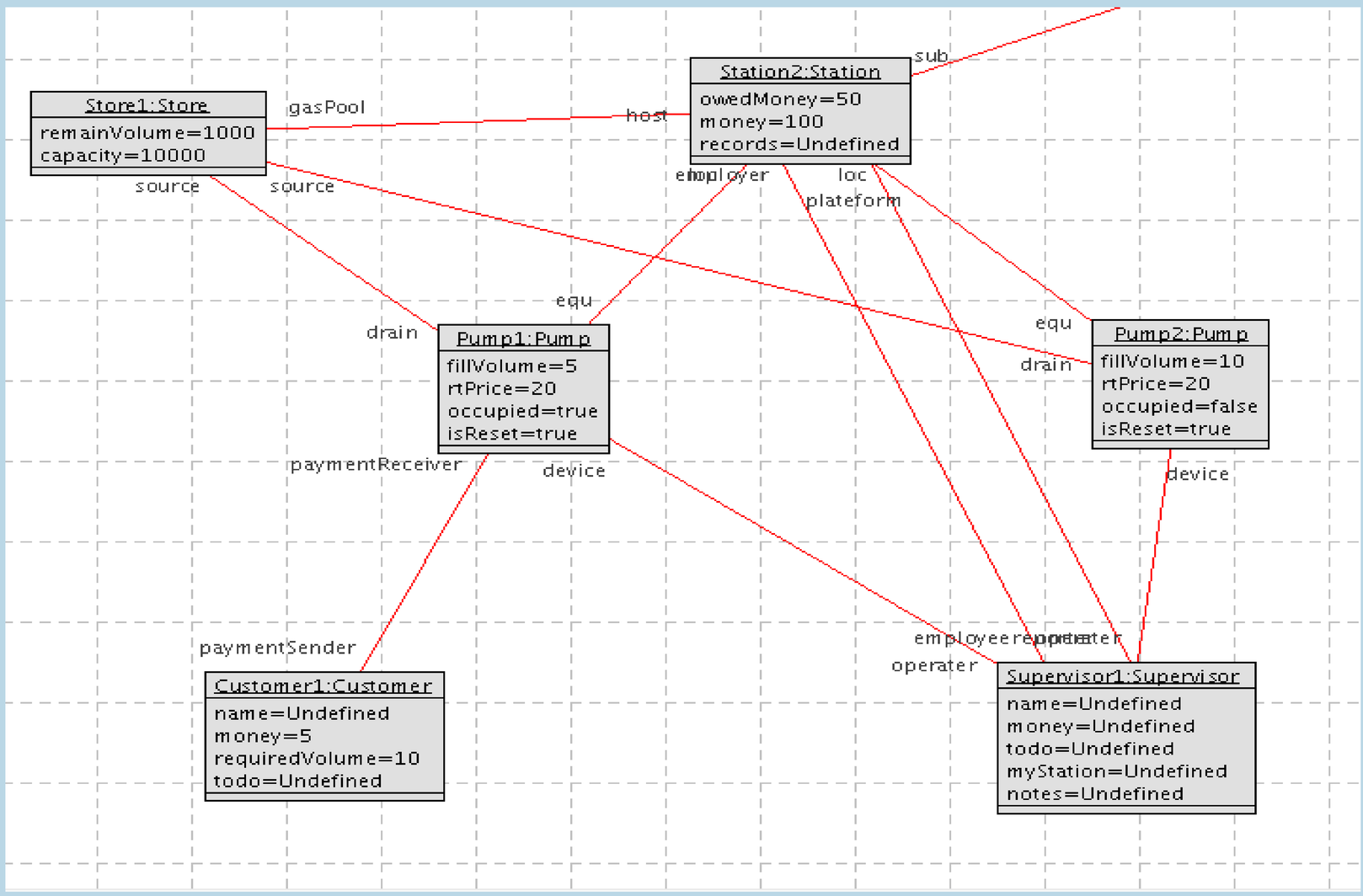
Class Diagram

Taking benefit of USE and OCL, the Class diagram demonstrate the relationship between classes and it can also provide the rolename for easy observation. Start from the head - company, they own stations and sale gas to stations with a wholesale price, they can collect bill from station whenever they want. This feature was implemented in Station class, with operation 'payOwe'. Pre/Post conditions were specifying used to ensure Station have money process the payment and Station holds a Record. Record class is create for the sake of recording either money or volume of gas involves in this system. This Record at Station get cleared after they pay off the bill. The supervisor also have a instance of Record, that is designed for her to record the amount of gas pumped by customer.



Object Diagram

Partial view of object diagram is display here to demonstrate idea implementation of this petrol station. In this diagram we create instances for each class to demonstrate 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.



Reference

Object Management Group, "Unified Modeling Language (OMG UML), Version 2.5.1", December 2017.
Kyas, Marcel. (2018). Verifying OCL specifications of UML models : tool support and compositionality.