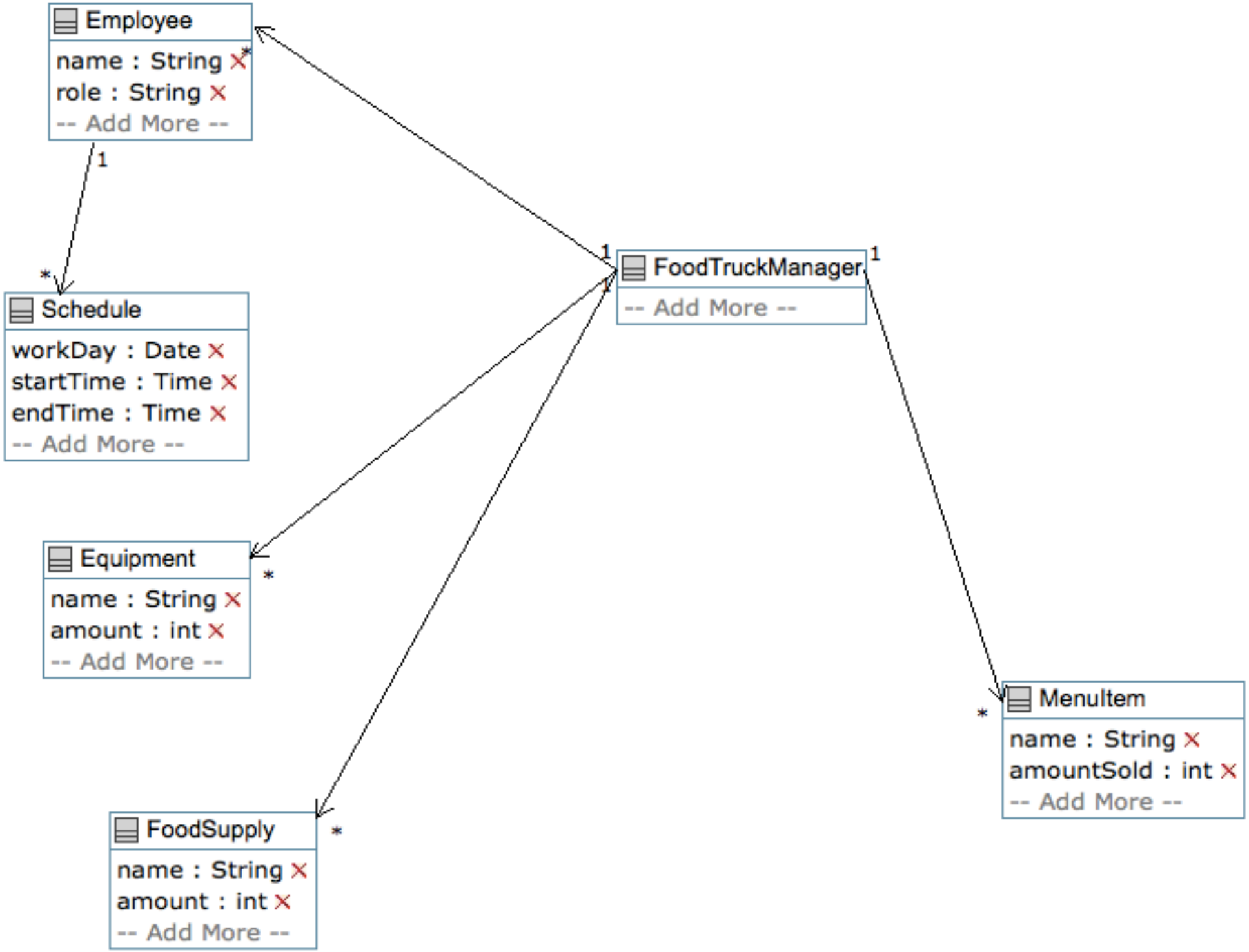
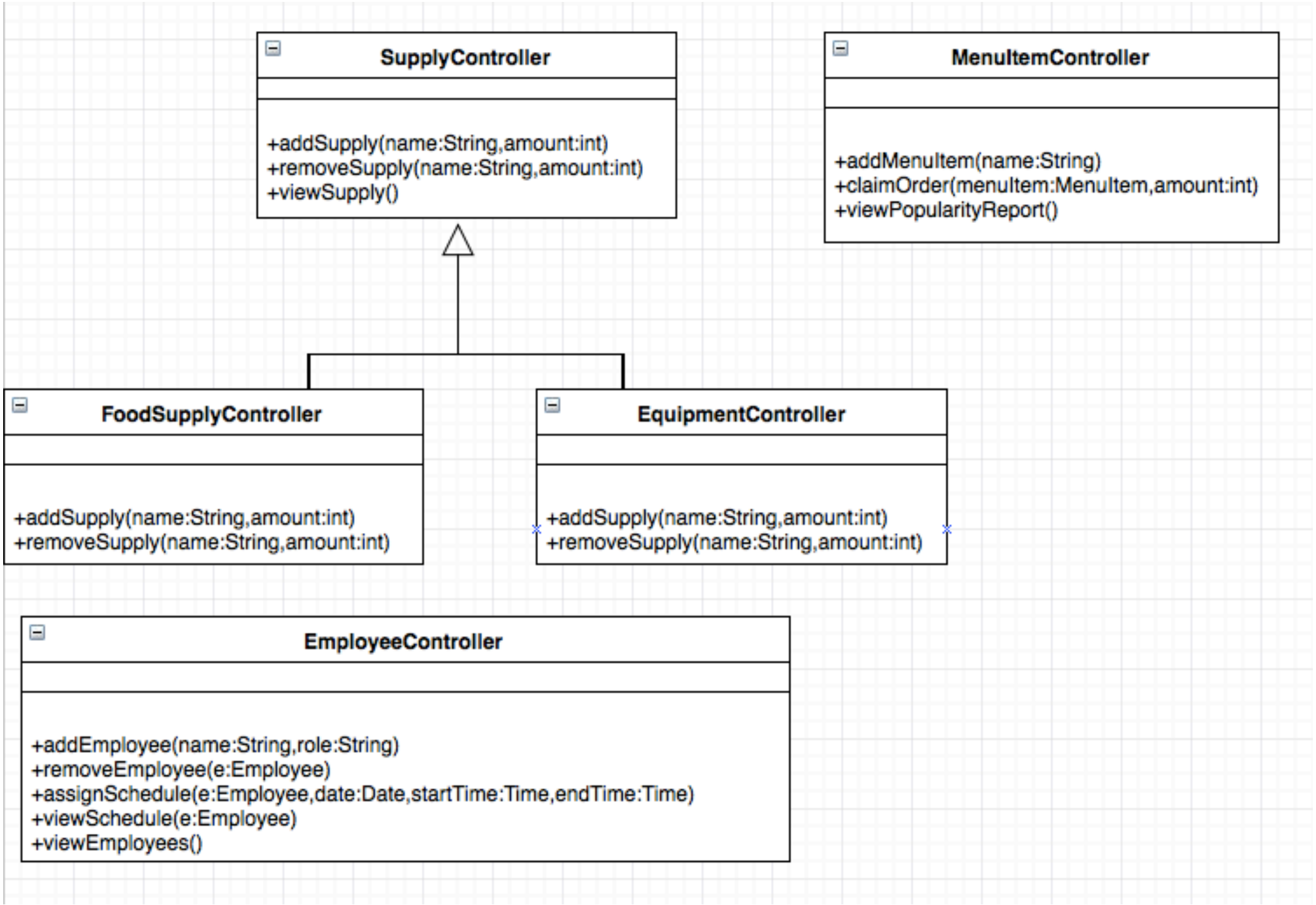

Employee
name : String
role : String
-- Add More --
Schedule
workDay : Date
startTime : Time
endTime : Time
-- Add More --
FoodTruckManager
-- Add More --
Equipment
name : String
amount : int
-- Add More --
FoodSupply
name : String
amount : int
-- Add More --
MenuItem
name : String
amountSold : int
-- Add More --
1
*
1
*
1
*
1
*
1
*


SupplyController
+addSupply(name:String,amount:int)
+removeSupply(name:String,amount:int)
+viewSupply()
MenuItemController
+addMenuItem(name:String)
+claimOrder(menuItem:MenuItem,amount:int)
+viewPopularityReport()
FoodSupplyController
+addSupply(name:String,amount:int)
+removeSupply(name:String,amount:int)
EquipmentController
+addSupply(name:String,amount:int)
+removeSupply(name:String,amount:int)
EmployeeController
+addEmployee(name:String,role:String)
+removeEmployee(e:Employee)
+assignSchedule(e:Employee,date:Date,startTime:Time,endTime:Time)
+viewSchedule(e:Employee)
+viewEmployees()

Description of Design

Discussion/Rationale:

The key classes for the model subsystem are Employee, Schedule, Equipment, FoodSupply, MenuItem and FoodTruckManager. All of these classes were auto-generated using Umple. The model uses the singleton design pattern, as only one instance of the FoodTruckManager can be instantiated. The FoodTruckManager class stores all of the data in the system. It is advantageous to use the singleton pattern, as all of the data only needs to be stored in this one instance of the class.

The controller subsystem was divided into several classes so as to avoid having a blob class. The controller classes include an abstract SupplyController that extends the concrete FoodSupplyController and the concrete EquipmentController. In this case, the template design pattern was used. The template pattern was used because the two supply controllers are very similar. These controllers use the same viewSupply() method but have different implementations for the addSupply(String name, int amount) and removeSupply(String name, int amount) methods. The other controller classes include the EmployeeController and the MenuItemController. The EmployeeController manages adding/removing/viewing employees as well as assigning/viewing an employee's schedule. The MenuItemController manages adding items, claiming that an item was ordered and viewing a popularity report of the most ordered items. No design patterns were used for these controllers.

The view subsystem was implemented using Java Swing and a single FoodTruckManagementPage class. The view class provides the outline of the interface that the user uses to manipulate the controller.

The persistence layer was implemented using the provided PersistenceXStream class from assignment #1. The persistence layer is used in order to store the data.