Facility Management System v3.0

1. System Description and Assumption: This system shall support the management of rooms and buildings, their constant use, inspection, and maintenance support when it is needed. The following are the three main functionalities of this system:
1. Facility - this covers the functionalities such as listing all the facilities; adding a new facility; removing a facility; getting and setting detail information such as the name, address, and capacity of facilities; and their current statuses.
2. Facility use - this covers the functionalities such as listing the history of usage; calculating the usage rate of facility; assigning and de-assigning a facility for use.
3. Facility inspect - this covers the functionalities such as inspecting a facility; and listing the inspection history of a facility.
4. Facility maintain - this covers the maintenance of a facility such as making a maintenance request; scheduling a maintenance request; checking maintenance status; listing maintenance requests; calculating down time of a facility.

2. Public interfaces provided by domain layer:

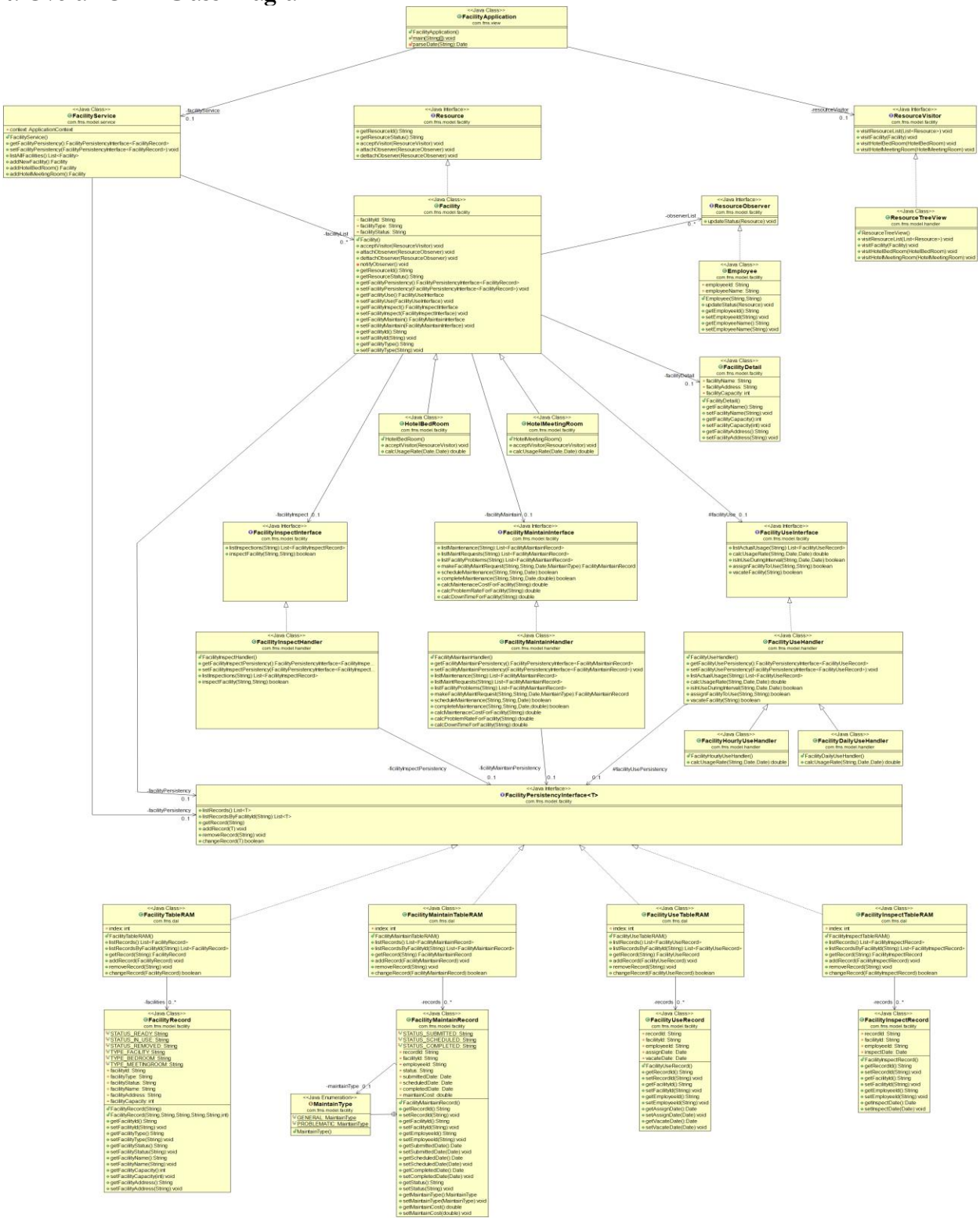
```
class FacilityService
        // List all the facilities
        public List<Facility> listAllFacilities()

        // Add a new facility
        public Facility addNewFacility()

class Facility
        /* Facility public interfaces */
        // Get the detail information of the facility
        public FacilityDetail getFacilityInformation()

        // Request the available capacity of the facility
        public int requestAvailableCapacity()

        // Add or set the detail information of the facility
        public void addFacilityDetail(FacilityDetail facilityDetail)

        // Remove the facility
        public void removeFacility()

        /* Facility use-related public interfaces (call FacilityUseInterface) */
        // List the actual usage of the facility
        public List<FacilityUseRecord> listActualUsage()

        // Calculate the usage rate of the facility
        public double calcUsageRate(Date startDate, Date endDate)

        // Check if the facility is in-use or not from start date to end date
        public boolean isInUseDuringInterval(Date startDate, Date endDate)

        // Assign the facility to use
        public boolean assignFacilityToUse(String employeeId)

        // Vacate the facility
        public boolean vacateFacility()
```

```
/* Facility inspect-related public interfaces (call FacilityInspecInterface) */
// List all the inspection records of the facility
public List<FacilityInspectRecord> listInspections()

// Inspect the facility
public boolean inspectFacility(String employeeId)

/* Facility maintain-related public interfaces (call FacilityMaintainInterface) */
// List all the maintain records of the facility
public List<FacilityMaintainRecord> listMaintenance()

// List the maintain records of the facility with submitted status
public List<FacilityMaintainRecord> listMaintRequests()

// List the maintain records of the facility with problematic type
public List<FacilityMaintainRecord> listFacilityProblems()

// Submit a maintain record for the facility
public FacilityMaintainRecord makeFacilityMaintRequest(String employeeId, Date submittedDate,
FacilityMaintainRecord.MaintainType maintainType)

// Schedule a maintain record for the facility with scheduled date
public boolean scheduleMaintenance(String recordId, Date scheduledDate)

// Complete a maintain record for the facility with completed date and maintain cost
public boolean completeMaintenance(String recordId, Date completedDate, double maintainCost)

// Calculate the total maintain cost for the facility
public double calcMaintenaceCostForFacility()

// Calculate the problem rate for the facility (number of problematic records by number of total records)
public double calcProblemRateForFacility()

// Calculate the down time for the facility (days between submitted date and completed date of the
problematic records)
public double calcDownTimeForFacility()
```

3. Public interfaces provided by data access layer:

```
interface FacilityPersistencyInterface<T> {
        public List<T> listRecords();                              // List all the records
        public List<T> listRecordsByFacilityId(String facilityId); // List the records with the specific facility ID
        public T getRecord(String recordId);                       // Get a record with the specific record ID
        public void addRecord(T record);                           // Add a record
        public void removeRecord(String recordId);                 // Remove a record with the specific record ID
        public boolean changeRecord(T record);                     // Change a record
}

public class FacilityTableRAM implements FacilityPersistencyInterface<FacilityRecord>

public class FacilityUseTableRAM implements FacilityPersistencyInterface<FacilityUseRecord>

public class FacilityInspectTableRAM implements FacilityPersistencyInterface<FacilityInspectRecord>

public class FacilityMaintainTableRAM  implements FacilityPersistencyInterface<FacilityMaintainRecord>
```

4. Overall UML Class Diagram

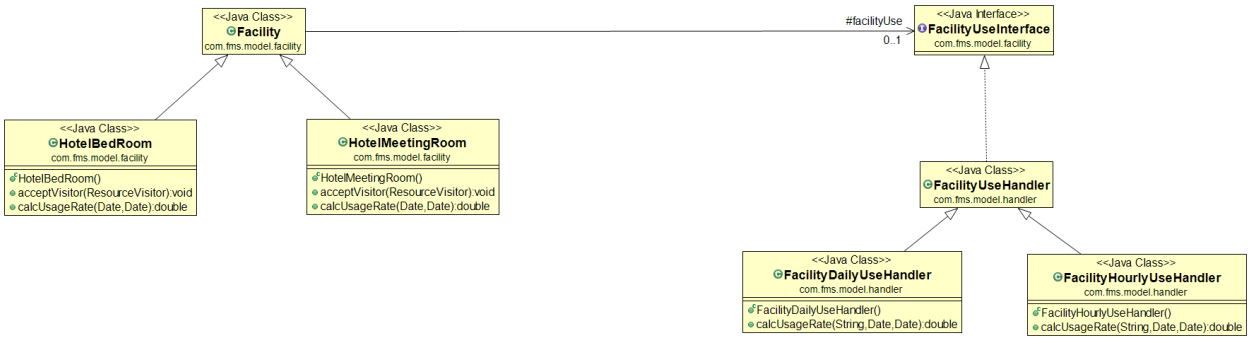


Code:

https://github.com/gaobibo/FacilityManagement

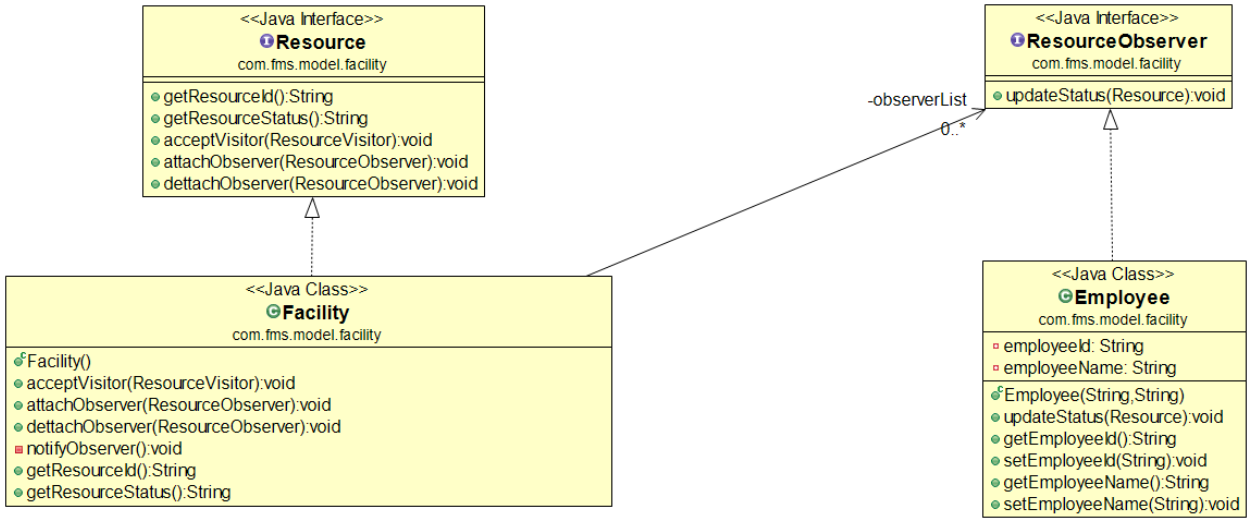
5. Design Patterns

5.1 Bridge Pattern



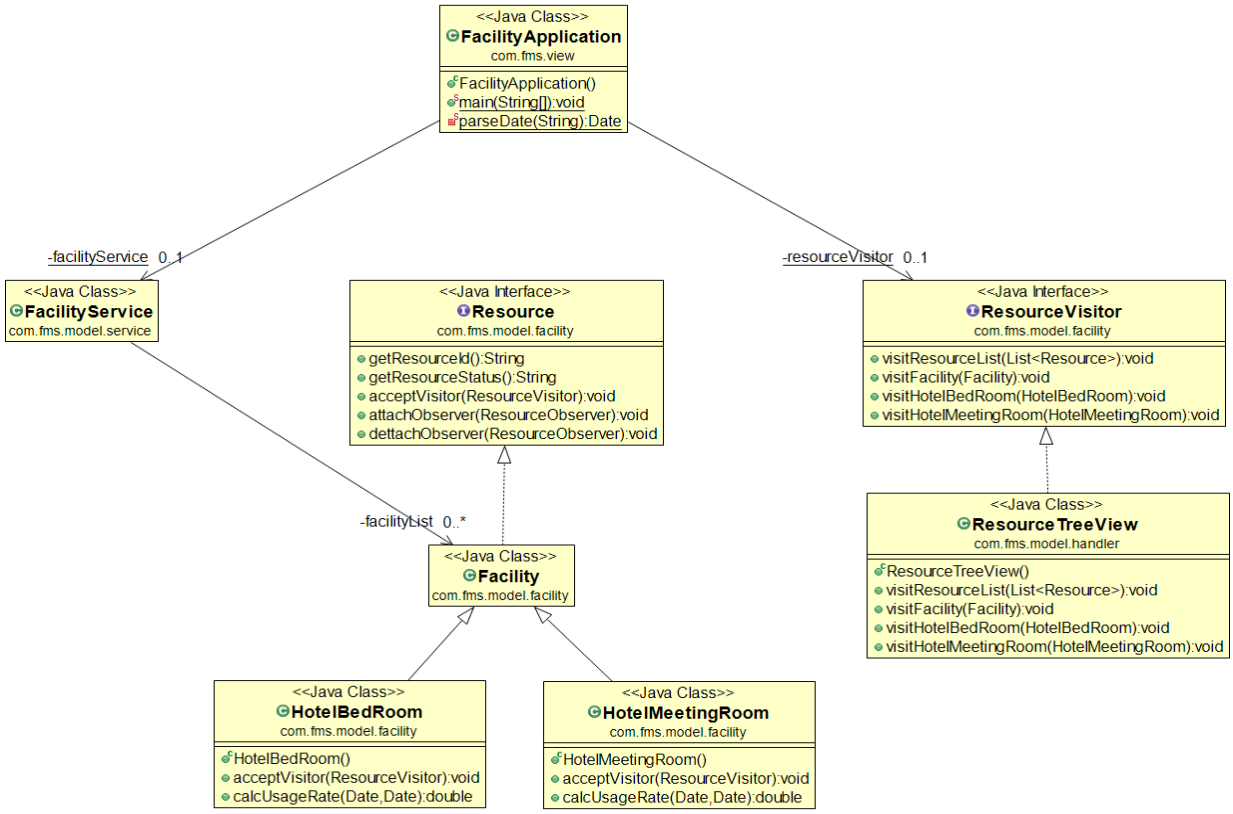
Abstraction Facility has two refined abstractions: HotelBedRoom and HotelMeetingRoom, and they have different usage calculation strategies: one is based on days, and the other is based on hours. The implementor of FacilityUseInterface is decoupled from abstraction Facility, so that the two implementations of daily-based usage and hourly-based usage can vary independently. The two concrete implementors are FacilityDailyUseHandler and FacilityHourlyUseHandler.

5.2 Observer Pattern



Resource provides interface for ResourceObserver to attach and detach themselves with the resource, so the resource knows who its subscribers are. Class Facility is a concreate Resource, and Facility implements the attach and detach methods, and maintain a list of observers (instances of Employee), and class Employee is a concreate ResourceObserver. Whenever Facility's status change, Facility will notify all the subscribed/attached Employees. Once the notification is received from Facility, Employee calls the getResourceStatus() method to get the latest status.

5.2 Visitor Pattern



ResourceVisitor provides interface for the visitor of resource, and ResourceTreeView is a concrete ResourceVisitor, and it represents a tree view operation to be performed on the elements of a resource list. FacilityApplication creates a ResourceTreeView object and then traverse a resource list, and visit each resource with the ResourceTreeView object. When a resource (either Facility, HotelBedRoom, or HotelMeetingRoom) is visited, it calls the visitor's operation with the corresponding class. The resource supplies itself as an argument to that operation, and the ResourceTreeView visitor will access the resource's state to conduct the tree view operation.