**OOAD - Project Requirements for Hotel Management System (HMS)**

**Functional Requirements**

1. The system shall provide staff with reservation handling for customers.
2. The system shall allow management to set the base rate.
3. The system shall allow management to change the base rate.
4. The system shall allow staff to make reservations.
   1. The system shall allow staff to make *prepaid reservations*.
   2. The system shall allow staff to make *60-days in advance reservations*.
   3. The system shall allow staff to make *conventional reservations*.
   4. The system shall allow staff to make *incentive reservations*.
5. The system shall allow staff to cancel reservations.
6. The system shall allow staff to change reservations.
7. The system shall allow staff to check in guests.
8. The system shall allow staff to check out guests.
   1. The system shall allow staff to print *accommodation bills*.
9. The system shall allow staff to accept payments from guests.
10. The system shall allow staff to classify a guest as a no show.
11. The system shall allow staff to make a backup copy of all files.
12. The system shall allow staff to print reports.
    1. The system shall allow management to print *expected occupancy reports.*
    2. The system shall allow management to print *expected room income reports.*
    3. The system shall allow management to print *incentive reports.*
    4. The system shall allow employees to print *daily arrivals reports.*
    5. The system shall allow management to print *daily occupancy reports.*

**Non-functional Requirements**

1. Reservation information must be stored securely.
2. The system must ensure it does not overbook.
3. The system must ensure secure payment transaction.
4. The system must have some default cost specified if no manually cost is set.
5. Employees must be allowed taco Tuesday.

**Weights and Priorities**

1. Class structure and components
2. Class interaction and framework
3. Accurate reservation type interaction
4. Date IO, i.e. saving reservation information to file
5. Printing out report orders
6. Saving reservation information to a database

**OOAD – Domain Modeling**

**Business Description**

**Overview**

Ophelia’s Oasis in the Amlet desert is a hotel in located within the Amlet desert. The business needs a reservation management system to improve user interactivity and reduce the need for manual reservation management. By using a system like this, the business can increase its efficiency and reservation processing capabilities

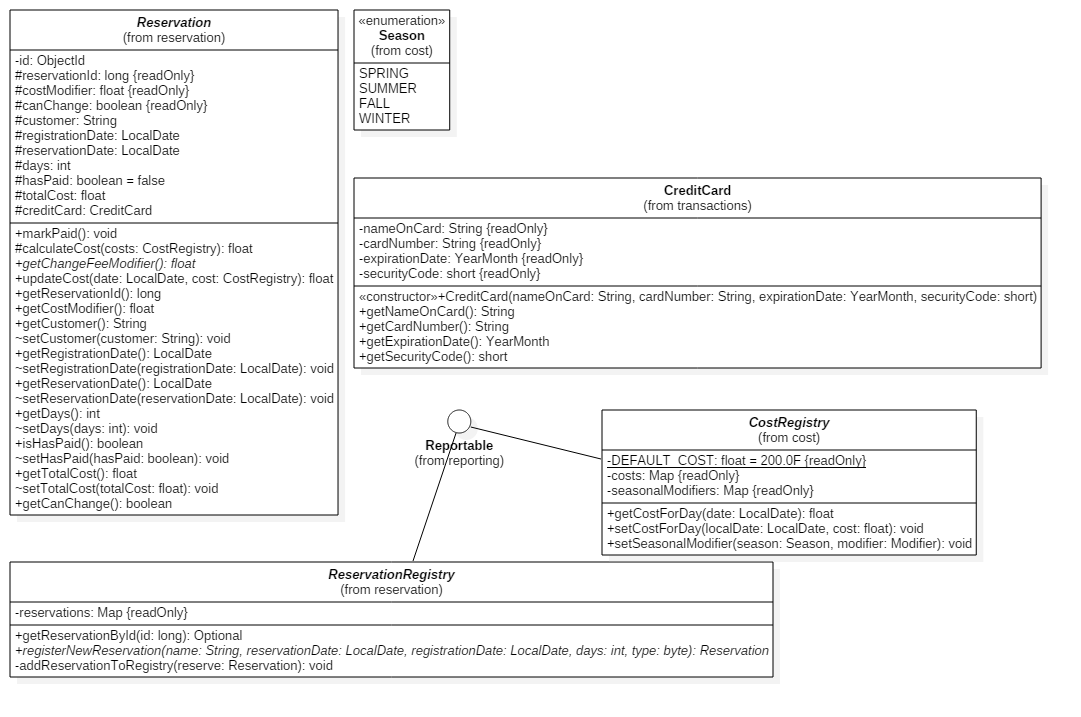
**Business Operations**

This system will, ideally, include a web interface that allows users to access the reservation system at any time. Additionally, there might be a desire to include an automated answering service that can handle these reservations as well.

These requests are passed the central reservation system, which will validate the request, and added it to the central registry is it is approved. Exact process may vary with the type of reservation. For certain reservations, guests will receive emails notifying them of the coming reservation date, as well as the need to provide a payment to ensure their reservation won’t be canceled. Room numbers will not be assigned until the morning of the arrival. Obviously, the system must prevent overbooking, while striving to achieve full capacity every day. The guests, if they have not prepaid, must provide a credit card before they are allowed to occupy a room. On checkout, they will pay for the room, if not already done, and receive a receipt for their stay.

Daily, the system will generate different types of management reports, such as vacancy reports, income, as well as expected income and occupancy. All reservations records must be kept, whether canceled or not. These will be stored locally after each transaction, and will be backed up elsewhere at the end of the day.

**UML Class Diagram**



1

1

**Domain Model Review Checklist**

1. This design model contains the bulk of all significant classes that will be required for this system to work.
2. This model design does show important relationships where they exist.
3. These models are designed similar to real-world concepts, and accurately reflect relationships these real world items would have.
4. This model shows multiplicity constraints, such as the registries of which we only need one instance.
5. The classes shown are all enumerated/abstract classes intended to show relationships, without showing the particular implementations thereof.
6. All classes show the important and relevant attributes that are necessary for that class to function properly within the application context.
7. The class naming and attributes are simple and clear to understand.
8. The classes aren’t used as attributes types. Instead, a composite part relationship is indicated between them.
9. Classes do show operations, since the designs were ported straight from the Java classes. However, due the accelerated schedule of this project, this can be accepted.