**Restaurant Model**

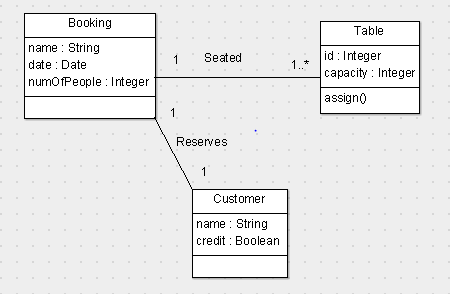
Sean Hartnett C12425762

**Introduction:**

The objective of this assignment is to design and implement both UML and USE models for the Restaurant case study. The main part of the system is allowing the customer reservations of tables and all that entails, such as is the table capacity great enough for the customer’s party.

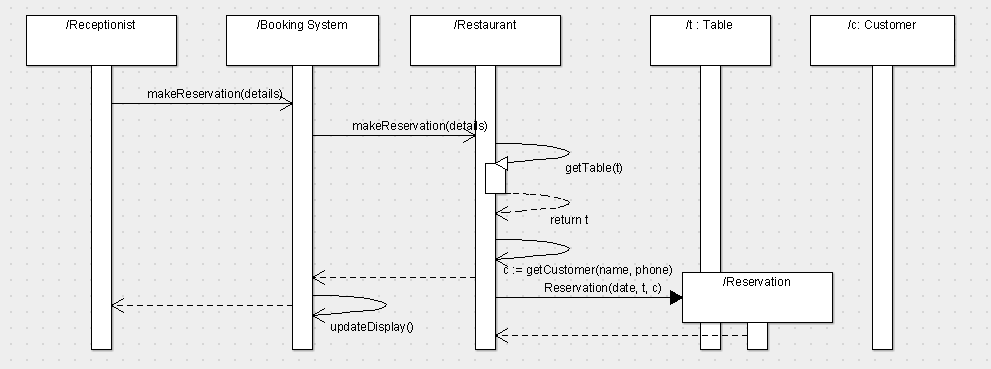
**UML Models:**

The following models were designed using Argo UML



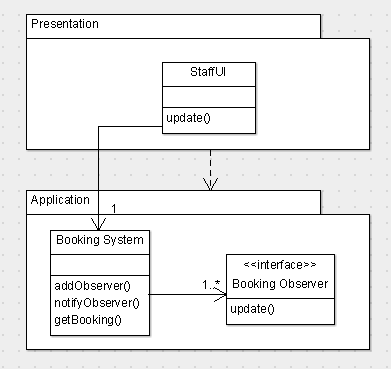
**Class Diagram:**

This basic class diagram above shows the three different types of objects that will be in the USE model, Customer, Booking and Table. A customer can have one booking at a time, as shown by its one to one multiplicity, and that booking made by the customer can have multiple tables assigned to it, if the number of people in the booking requires it. An Object Orientated design pattern is being used here as the relationship between objects is being shown



**Sequence Diagram:**

This sequence diagram above shows how a reservation is made, starting with the receptionist. The receptionist actor sends the makeReservation (details) message which from there uses the details about the customer and table to make the reservation and then it finishes by updating the display for the receptionist actor.

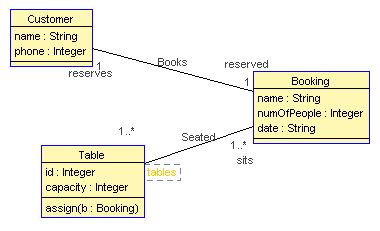


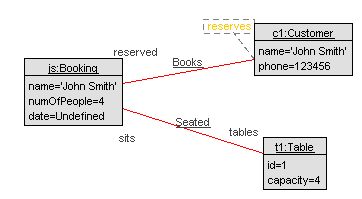
**Observer Pattern:**

The Observer Pattern above shows us the changes that are communicated one object and another, in this case the Application and the Presentation. The interface class implements the Booking Observer. The other classes realise this interface class. The booking system has a list of all registered observers, but there is no upwards dependency.

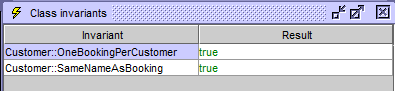
**USE Model:**

There are three classes in the model, Booking, Customer and Table. Booking and Table have the association Seated between them, and Booking and Customer have the association Books between them.



The object diagram below shows all the different objects and their relationship. The Customer c2 called John Smith makes a Booking, called js which then is assigned to a table, t1.

The image below shows the class invariants, both true. The OneBookingPerCustomer means that a customer can only have one active booking at a time. The SameNameAsBooking means that the name of Customer must match the Customer name.



The image below is of the USE command line with the openter and opexit commands being used. All the pre-conditions are returned as true and when the booking is assigned a table the post-condition is true.

