# Assignment Description

The goal for assignment five is to review the development steps in chapter eight, so that I can construct a model, create a façade and run test on these components using JUnit. The example used by the book in chapter eight is to create a model that represents a sports league. The book first describes the object model and has the user create the different java objects that will be used in the model. Next is the creation of the service layer. The service layer contains the business logic that will be used by the application. This layer contains the Java interfaces and the implementation of those interfaces. The interfaces are also known as façades. Next is the creation of the data access layer. This layer will contain logic to do the calls to the database or the location where the data is stored and serve the content to the service layer. Once these three layers are designed and implemented, the user can create JUnit test and test the logic of the code.

# Logic Employed

For the assignment I decided to develop a model for a food store. Following the process outlined in chapter eight, the first step is to create the objects that will represent the model of the food store. The three objects are Employee, Item and Store. The Employee object contains attributes for first name, last name, start date of employment, employee identification number and store identification number where the employee works. The item object is used to represent anything that is sold in the store. The item object contains attributes for the name of the object, the department the item belongs to (i.e. Dairy, Meat), the universal price code (upc) and the price. The store object is used to represent an individual store. The store object contains attributes for the name of the store, the city the store is located in and the identification number of the store. The object classes contain getter and setter methods for all of the attributes in the classes. The object classes are located in the “foodstore.model” java package.

After creating the objects that represent the data model, the next step was to create the service layer. The service layer is where the application logic will be defined. The “FoodStoreFacade” interface will contain methods that represent the functionality of the food store model. The interface contains methods to get information about an employee, item or store, create an employee item or store, check if an employee is currently employed, get the department that an item belongs to and get the name of a store. The methods definitions contained in “FoodStoreFacade” interface are implemented in the “FoodStoreFacadeImpl” class. The implemented methods will be used to build the application’s functionality. The “FoodStoreFacade” class is located in the “foodstore.services” java package and the “FoodStoreFacadeImpl” class is located in the “foodstore.services.impl” java package.

Next was to create and implementing the interface for the data access layer. The methods implemented in the data access layer will be used by the service layer to create, update and retrieve data from the data source. An example of a data source could be a database. The interface is called “FoodStoreDAO” and is located in the “foodstore.dao” java package. The methods in the “FoodStoreDAO” interface contains methods to create, update, remove and get information for the employee, item or store objects. The “FoodStoreDAOImpl” class implements the “FoodStoreDAO” interface. The methods are implemented and contain the logic to manage the data of the data access level.

The final step is to use JUnit to test the methods defined in the “FoodStoreFacade” interface. I did not use a database to store the data for this assignment. Test data will be generated using the “init” method of the “FoodStoreDAOImpl” class. The “init” method will create three store objects, three employee objects (one for each store) and three item objects (one for each store). The “assertTrue” method will be used to test that the methods are returning the proper data from each method. For example, the JUnit test for the “testCurrentlyEmployed” method would be to assert that the “getEmployee” method does not return null when a valid employee identification number is passed in as an argument to the “getEmployee” method.

# Snapshots

# Conclusion

After developing the objects for the food store model I realized that the objects I created could apply to any kind of store. Employee, item and store do not contain and attributes that would only relate to a food store. The model could be renamed to “Store Franchise” model or “Store Chain” model or something of that kind. This would allow the application to be more marketable.

This was the first time I have created and implemented interfaces. I did like developing this way, the project felt more organized when I was working. This assignment was the first time that I used JUnit to test as well. I was a salesforce.com developer in my last job and it seems that a lot of salesforce.com’s development platform is based on chapter 8. Test classes are very common in salesforce.com and are required to implement and deploy code in a production environment. Interfaces weren’t as common at my last consulting company because none of the project managers or lead developers used them.

My previous reports I believe I have gone into too much detail. To the point where I start talking about all the individual variables of the class. I have decided to give a broader overview of the codes components.