Cognizant Academy

truYum

C# Specification Document

Version 1.0

	Prepared By / Last Updated By	Reviewed By	Approved By
Name	Ramamoorthy Selvam	Vimalathithan Krishnan	Ramadevanahalli Lingachar, Shashidhara Murthy
Role	Learning Solution Designer	Learning Solution Architect	Learning Solution Lead
Signature			
Date			

Table of Contents

1.0 Introduction

1.1 Purpose of this document

The purpose of this document is to define the C# classes related implementation for truYum project.

1.2 Definitions & Acronyms

Definition / Acronym	Description
ADO.Net	ActiveX Data Objects (ADO.NET) is a framework that contains set of classes to interact with Datasources such as SQL Server and XML.

1.3 Project Overview

Refer truYum-use-case-specification.docx to understand the functionality and features.

1.4 Scope

Creation of model and data access object classes for truYum application

1.5 Intended Audience

- Product Owner
- Scrum Master
- Application Architect
- Project Manager
- Test Manager
- Development Team
- Testing Team

1.6 Hardware and Software Requirement

1. Hardware Requirement:
    a. Developer PC with 8GB RAM
2. Software Requirement

a. IE or Chrome

b. .Net Framework 4.5

c. Visual Studio Professional Edition 2015

d. SQL Server enterprise edition 2014

1.7 Visual Studio 2015 Project Configuration

A new Visual Studio Console project should be created for C# development. Find below the steps to configure Visual Studio for the truYum project.

1. Open Visual Studio 2015
2. File > New > Project > Visual C# > Windows > Console Application
3. Name the project 'TruyumConsole'
4. Right click on the solution and create a Class library project 'TruyumOnline'. Create all the classes listed below in the TruyumOnline class library project.
5. Reference the class library projects in TruYumConsole application
6. Invoke the methods of the class library projects from the TruYumConsole application
7. For C# testing, please set up the TruYumConsole application as the startup project

2.0 Class Diagram

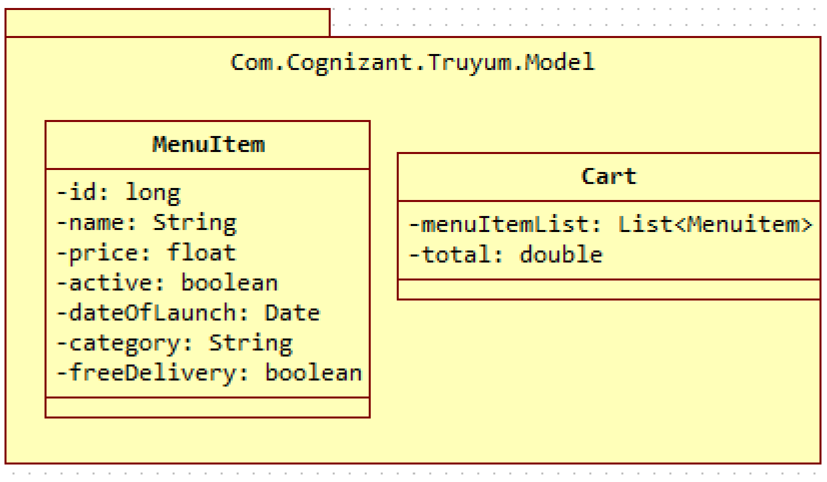
The classes specified in this document are the primary C# classes that are required for implementation of TruYum application. Refer ADO.NET specification to connect to the database.

2.1 **Com.Cognizant.Truyum.Model** Namespace

Following are the real world objects identified for truYum application. Menu Item refers to a menu item available for sale in truYum. Cart will represent customer's cart to hold the selected menu items. Refer the diagram below and create classes accordingly.

2.1.1 Namespace creation

Create class library **Com.Cognizant.Truyum.Model** with the member variables given below



Guidelines for understanding the above class diagram:

1. "`Com.Cognizant.Truyum.Model`" represents the namespace
2. `MenuItem` and `Cart` are classes
3. The content within `MenuItem` are instance variables
4. The hypen in each line represents private access specifier
5. For the sake of simplicity the constructors, Properties are not included in the diagram. But it needs to be implemented in code.
    a. Constructor with option to set all instance variables
    b. Properties for each instance variable
    c. Generate `ToString()` method
    d. Generate `Equals()` method which checks for equality based on the 'id' attribute

2.2 **Utility** Namespace

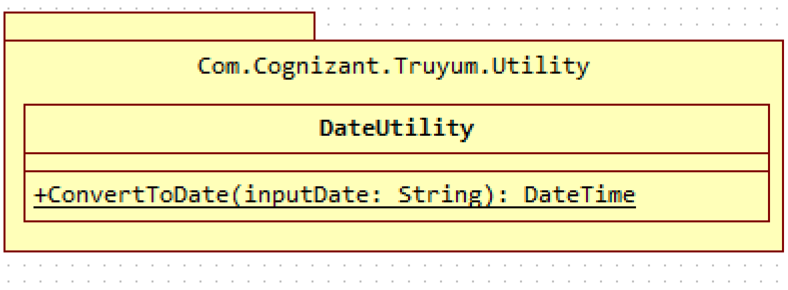
2.2.1 Namespace creation

Create class library **Com.Cognizant.Truyum.Utility** with the method given below.

Common reusable classes and methods across truYum application will be included in this namespace.

2.2.2 DateUtility.cs

As per the below Class diagram, create a method in this class to convert the input date string to shortDateString


Com.Cognizant.Truyum.Utility
DateUtility
+ConvertToDate(inputDate: String): DateTime

ConvertToShortDateString(string inputDate) of return type DateTime

This method is used to convert the input date entered in string format to Datetime.

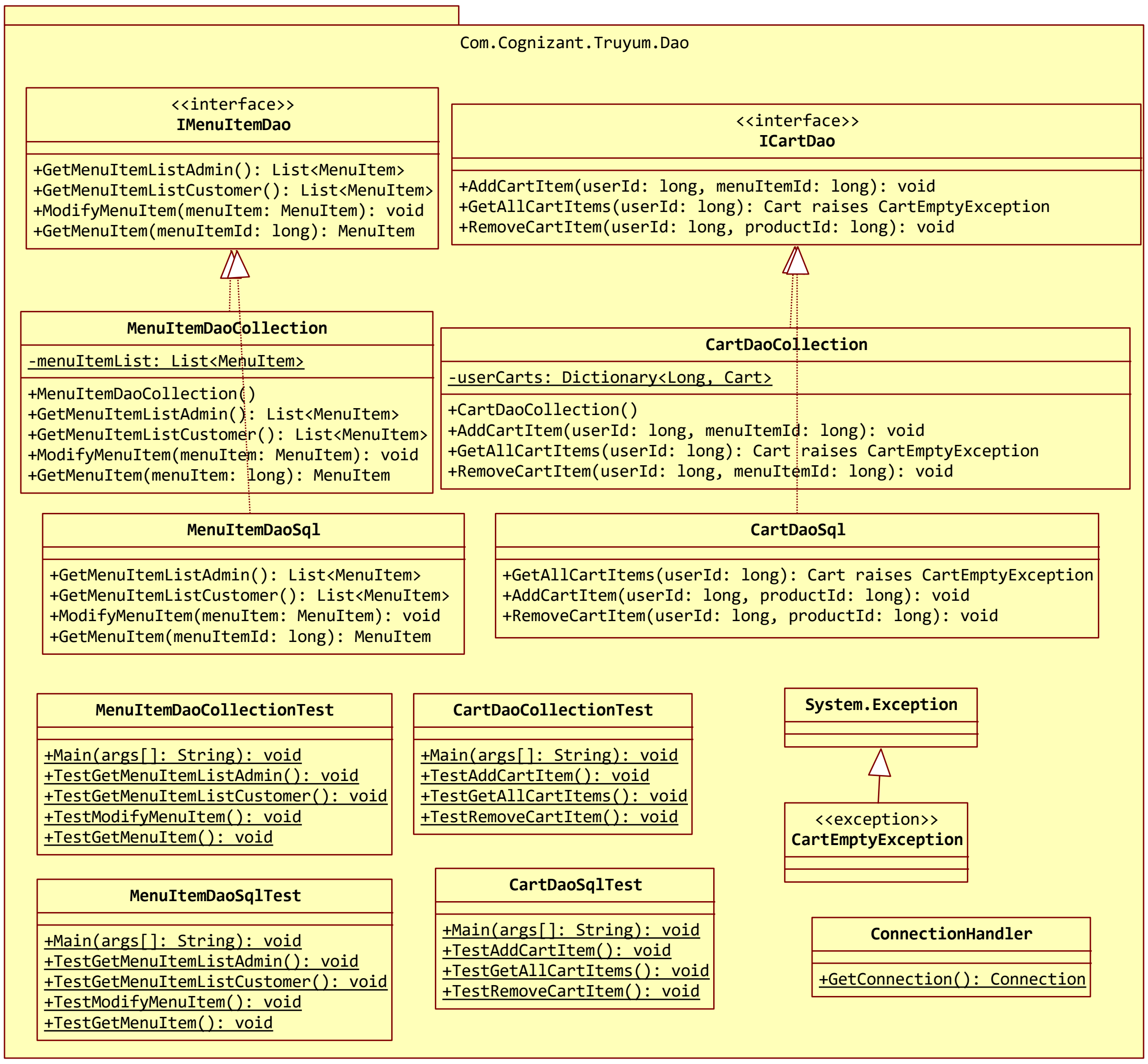
1. Use `DateTime.Parse(inputDate)` to convert the input string to DateTime.

2.3 **Com.Cognizant.Truyum.Dao** namespace

This namespace contains the list of classes that will code to manage the data for truYum application. The methods in Dao classes will be tested using `MenuItemDaoCollectionTest` and `CartDaoCollectionTest` classes. The Dao interface classes will act as a contract for working with any database. In this specification the implementation of `MenuItemDaoCollection` and `CartDaoCollection` will be Collection framework based implementation of Dao interfaces `MenuItemDao` and `CartDao`.

2.3.1 Namespace creation

Create class library **Com.Cognizant.Truyum.Dao** and create the classes listed in the diagram within this namespace


Com.Cognizant.Truyum.Dao
<<interface>>
IMenuItemDao
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
<<interface>>
ICartDao
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollection
-menuItemList: List<MenuItem>
+MenuItemDaoCollection()
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItem: long): MenuItem
CartDaoCollection
-userCarts: Dictionary<Long, Cart>
+CartDaoCollection()
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, menuItemId: long): void
MenuItemDaoSql
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
CartDaoSql
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+AddCartItem(userId: long, productId: long): void
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollectionTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoCollectionTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
System.Exception
<<exception>>
CartEmptyException
MenuItemDaoSqlTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoSqlTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
ConnectionHandler
+GetConnection(): Connection

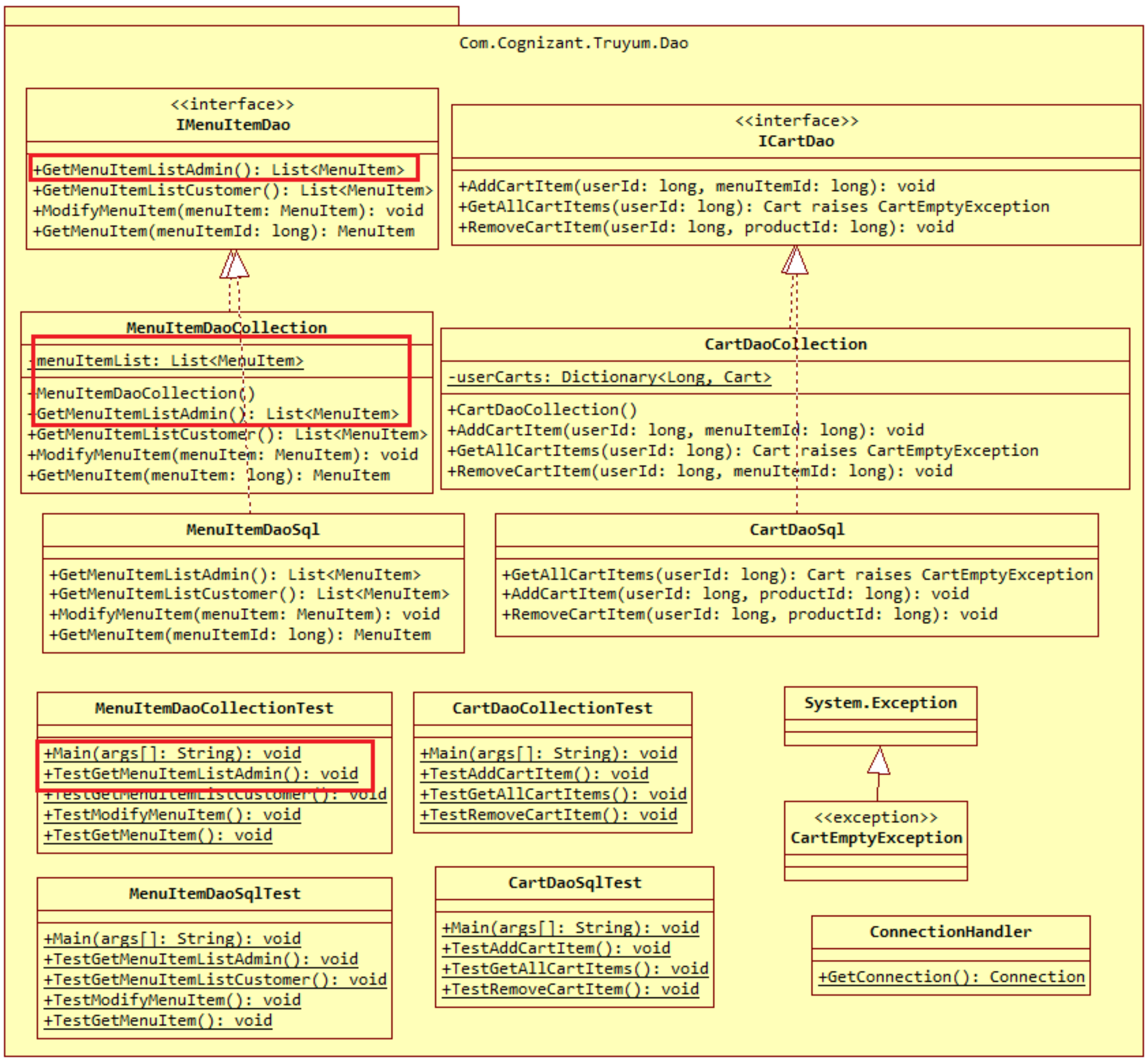
Guidelines for understanding the above class diagram:

1. Identify the namespace, classes, access modifiers, methods and static methods from the above diagram.
2. `IMenuItemDao` and `ICartDao` are interfaces
3. `MenuItemDaoCollection` and `CartDaoCollection` are implementation classes for the interfaces as denoted by the dotted arrow line.
4. `MenuItemDaoCollectionTest` and `CartDaoCollectionTest` are implementation classes for testing `MenuItemDaoCollection` and `CartDaoCollection`.
5. MenuItemDaoSql, CartDaoSql, MenuItemDaoSqlTest, CartDaoSqlTest classes will not be implemented in this module. Please ignore these classes for this module.
6. CartEmptyException is an exception class that extends System.Exception.
7. Highlighted classes will be implemented in this module.

3.0 Design for View Menu Item List Admin (EKUC001)

3.1 Class Diagram

The below diagram denotes the methods that needs to be implemented for this use case. Method wise specification is defined after the diagram.



Com.Cognizant.Truyum.Dao
<<interface>>
IMenuItemDao
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
<<interface>>
ICartDao
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollection
-menuItemList: List<MenuItem>
+MenuItemDaoCollection()
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItem: long): MenuItem
CartDaoCollection
-userCarts: Dictionary<Long, Cart>
+CartDaoCollection()
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, menuItemId: long): void
MenuItemDaoSql
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
CartDaoSql
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+AddCartItem(userId: long, productId: long): void
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollectionTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoCollectionTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
System.Exception
<<exception>>
CartEmptyException
MenuItemDaoSqlTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoSqlTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
ConnectionHandler
+GetConnection(): Connection

3.2 IMenuItemDao.cs

Add the method `GetMenuItemListAdmin() with return type List<MenuItem>` in the interface.

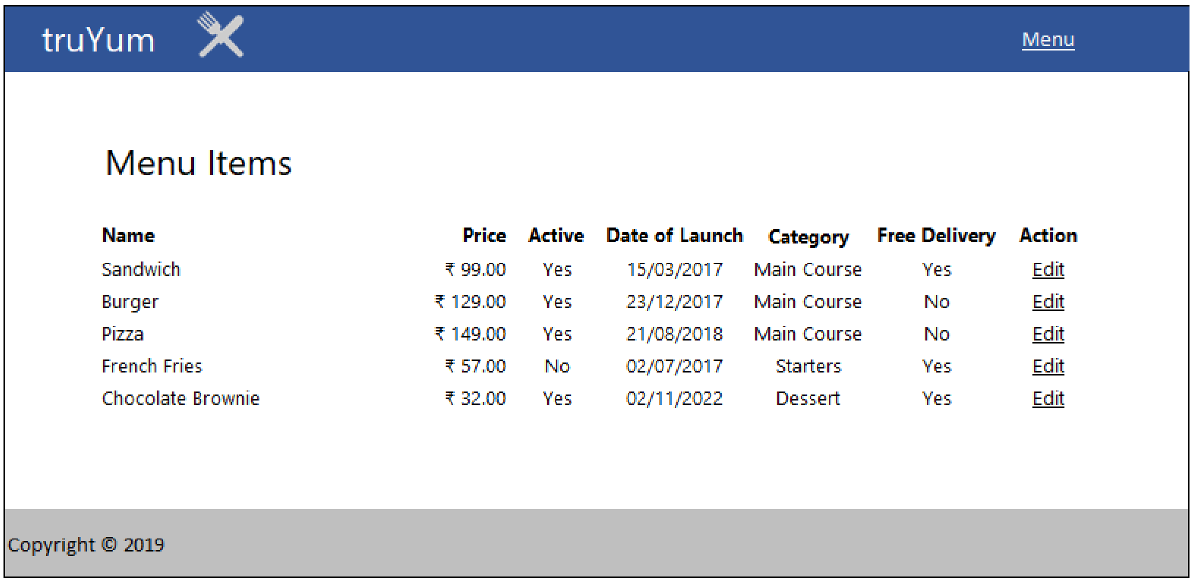
3.3 MenuItemDaoCollection.cs

Class for managing data of menu items using C# Collections Framework.

Constructor

The objective of this constructor is to initialize the menu item data that will be displayed in MenuItem listing screen of Admin.

1. Check if `menuItemList` variable is null or not
2. If it is null perform the steps below:
   a. Create an instance of `List` with `MenuItem` type
   b. Create multiple `MenuItem` instances and add them to `menuItemList`. Refer the screenshot listed below and include sample data for `menuItemList` based on it.



Menu Items

Name	Price	Active	Date of Launch	Category	Free Delivery	Action
Sandwich	₹ 99.00	Yes	15/03/2017	Main Course	Yes	Edit
Burger	₹ 129.00	Yes	23/12/2017	Main Course	No	Edit
Pizza	₹ 149.00	Yes	21/08/2018	Main Course	No	Edit
French Fries	₹ 57.00	No	02/07/2017	Starters	Yes	Edit
Chocolate Brownie	₹ 32.00	Yes	02/11/2022	Dessert	Yes	Edit

Copyright © 2019

GetMenuItemListAdmin() of return type List<MenuItem>

This method returns the list of menu items that will be displayed in the MenuItem listing screen for Admin.

1. Return the `menuItemList`

3.4 MenuItemDaoCollectionTest.cs

1. Create method `TestGetMenuItemListAdmin()`

TestGetMenuItemListAdmin() of return type void

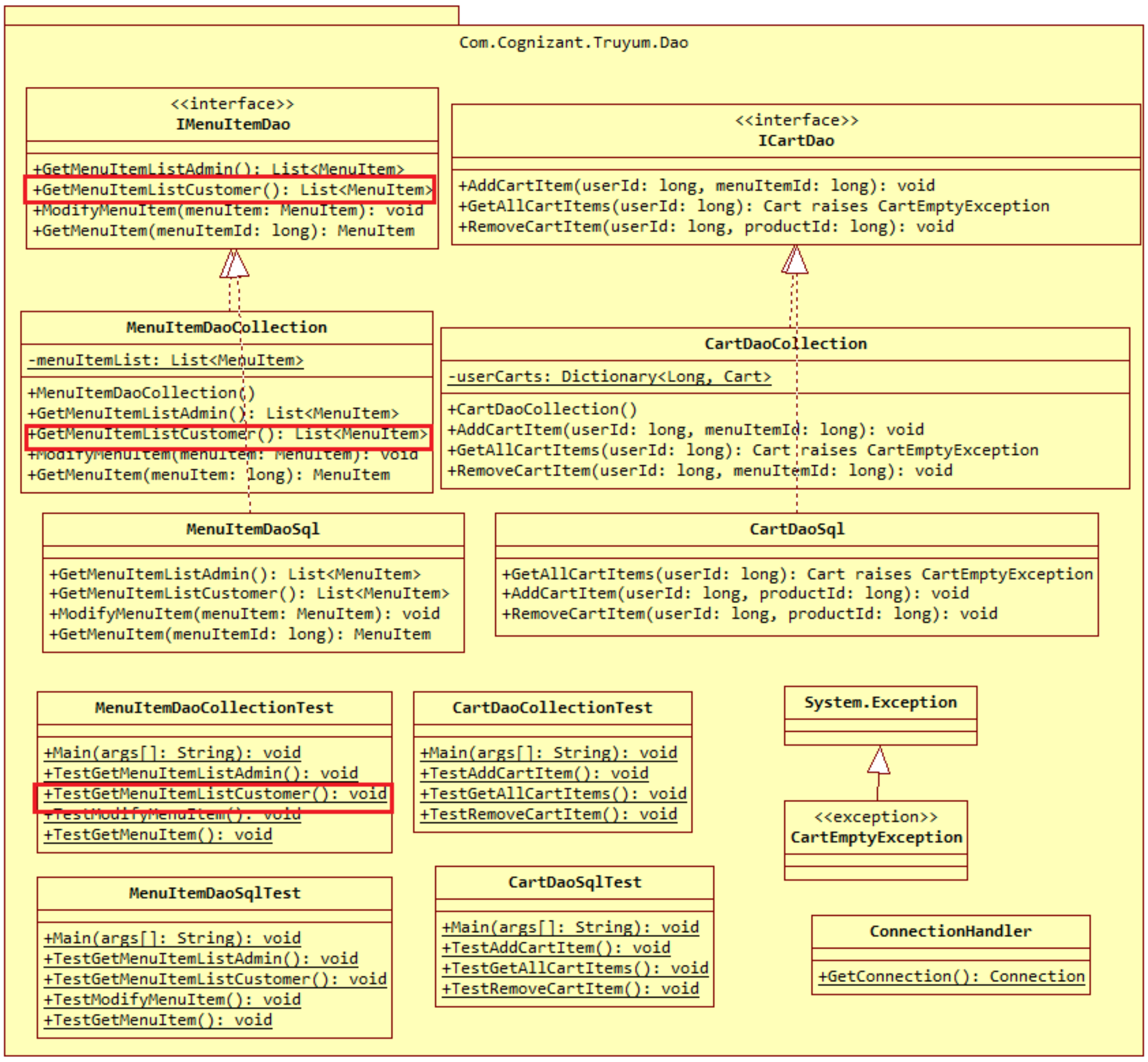
1. Instantiate `MenuItemDaoCollection` and assign `MenuItemDao` reference variable `menuItemDao` in its object instance.

2. Invoke `MenuItemDao.GetMenuItemListAdmin()` and obtain the `menuItemList`
3. Iterate through the `menuItemList` and display all attributes of each menu item.
4. Invoke this method in the Main method of **TruyumConsole** application.

4.0 Design for View Menu Item List Customer (EKUC002)

4.1 Class Diagram

The below diagram denotes the methods that needs to be implemented for this use case. Method wise specification is defined after the diagram.


Com.Cognizant.Truyum.Dao
<<interface>>
IMenuItemDao
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
<<interface>>
ICartDao
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollection
-menuItemList: List<MenuItem>
+MenuItemDaoCollection()
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItem: long): MenuItem
CartDaoCollection
-userCarts: Dictionary<Long, Cart>
+CartDaoCollection()
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, menuItemId: long): void
MenuItemDaoSql
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
CartDaoSql
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+AddCartItem(userId: long, productId: long): void
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollectionTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoCollectionTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
System.Exception
<<exception>>
CartEmptyException
MenuItemDaoSqlTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoSqlTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
ConnectionHandler
+GetConnection(): Connection

4.2 IMenuItemDao.cs

Add the method `GetMenuItemListCustomer() of return type List<MenuItem>` in the interface.

4.3 MenuItemDaoCollection.cs

This class manages the data related to Menu Items of truYum application. A new method needs to be added for this use case.

GetMenuItemListCustomer() of return type List<MenuItem>

This method returns the list of menu items that will be displayed in the Menu Item listing screen for Customer.

1. Initialize a `List` for type `MenuItem`
2. Iterate through `menuItemList` and perform the following steps:
    a. Is the Date of Launch of the menu item is after current date?
    b. Is the menu item available is active?
    c. If the above conditions satisfy, add the menu item into the `List` created in the first step.
3. Return the filtered `List`

4.4 MenuItemDaoTest.cs

1. Create `TestGetMenuItemListCustomer()`

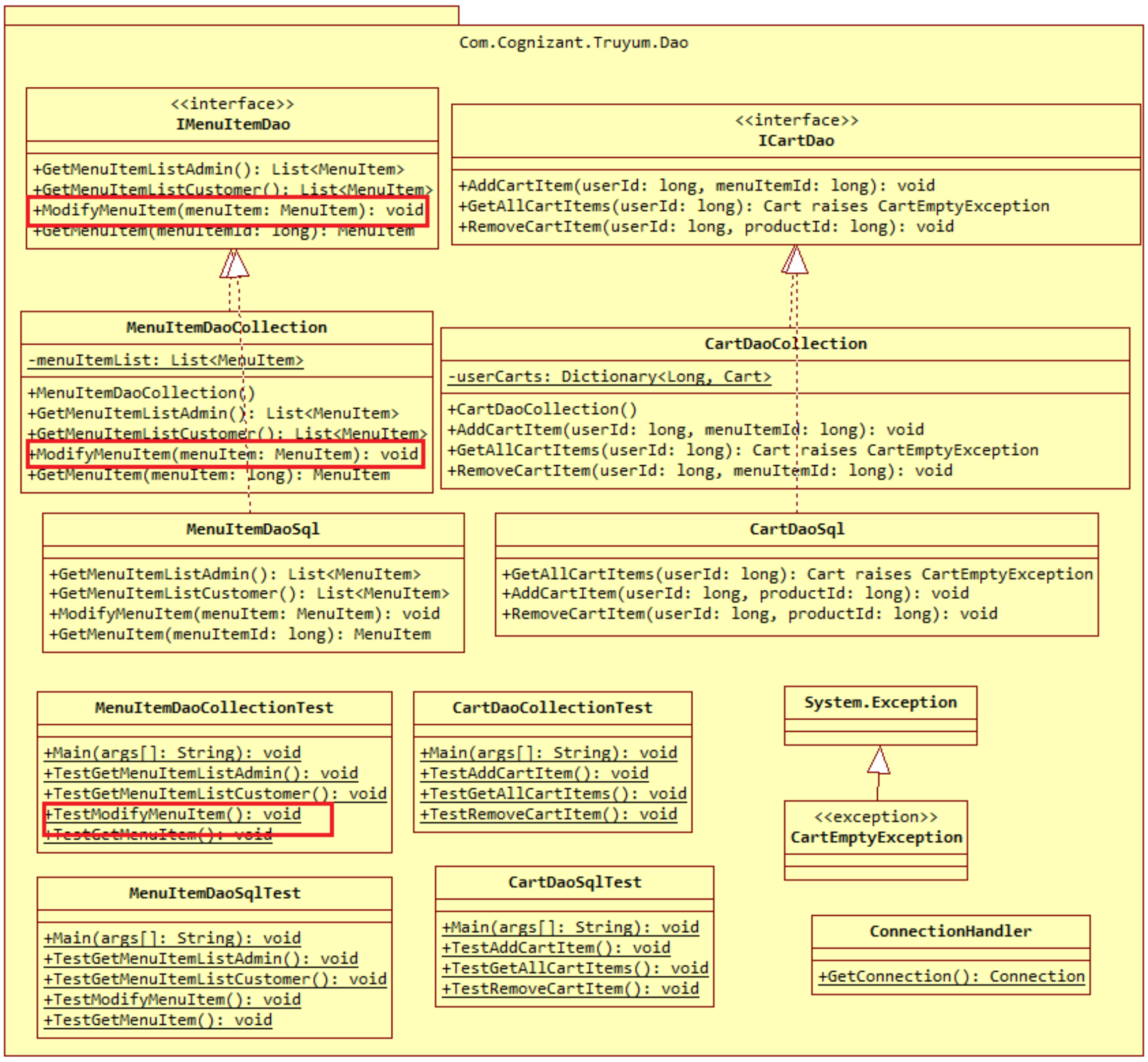
TestGetMenuItemListCustomer() of return type void

1. Instantiate `MenuItemDaoCollection` and assign `MenuItemDao` reference variable `menuItemDao` to its object.
2. Invoke `menuItemDao.GetMenuItemListCustomer()` and obtain the `menuItemList`
3. Iterate through the `menuItemList` and display all attributes of each menu item.
4. Invoke this method in the Main method of **TruyumConsole** application.

5.0 Design for Edit Menu Item (EKUC003)

5.1 Class Diagram

The below diagram denotes the methods that needs to be implemented for this use case. Method wise specification is defined after the diagram.


Com.Cognizant.Truyum.Dao
<<interface>>
IMenuItemDao
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
<<interface>>
ICartDao
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollection
-menuItemList: List<MenuItem>
+MenuItemDaoCollection()
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItem: long): MenuItem
CartDaoCollection
-userCarts: Dictionary<Long, Cart>
+CartDaoCollection()
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, menuItemId: long): void
MenuItemDaoSql
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
CartDaoSql
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+AddCartItem(userId: long, productId: long): void
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollectionTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoCollectionTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
System.Exception
<<exception>>
CartEmptyException
MenuItemDaoSqlTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoSqlTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
ConnectionHandler
+GetConnection(): Connection

5.2 IMenuItemDao.cs

1. Add method `ModifyMenuItem(MenuItem menuItem)` **of return type** `void` in the interface.

2. Add method `GetMenuItem(long menuItemId)` **of return type** `MenuItem` in the interface.

5.3 MenuItemDaoCollection.cs

This class manages the data related to Menu Items of truYum application. A new method needs to be added for this use case.

ModifyMenuItem(MenuItem menuItem) of return type void

This method will be used to change the menu item data in the list of menu items. This method will be invoked when Customer submits the user form.

1. Iterate through the `menuItemList` and find the matching menu item
2. Update the matching `menuItem` in the `List`

GetMenuItem(long menuItemId) of return type MenuItem

This method is used to retrieve a particular menu item's detail from the menu item list. This method will be invoked when user click on Edit link in menu item listing screen of Admin.

1. Iterate through `menuItemList` and find the matching menu item
2. Return the matching `menuItem` from the `menuItemList`

5.4 IMenuItemDao.cs

1. Create `TestModifyMenuItem()`

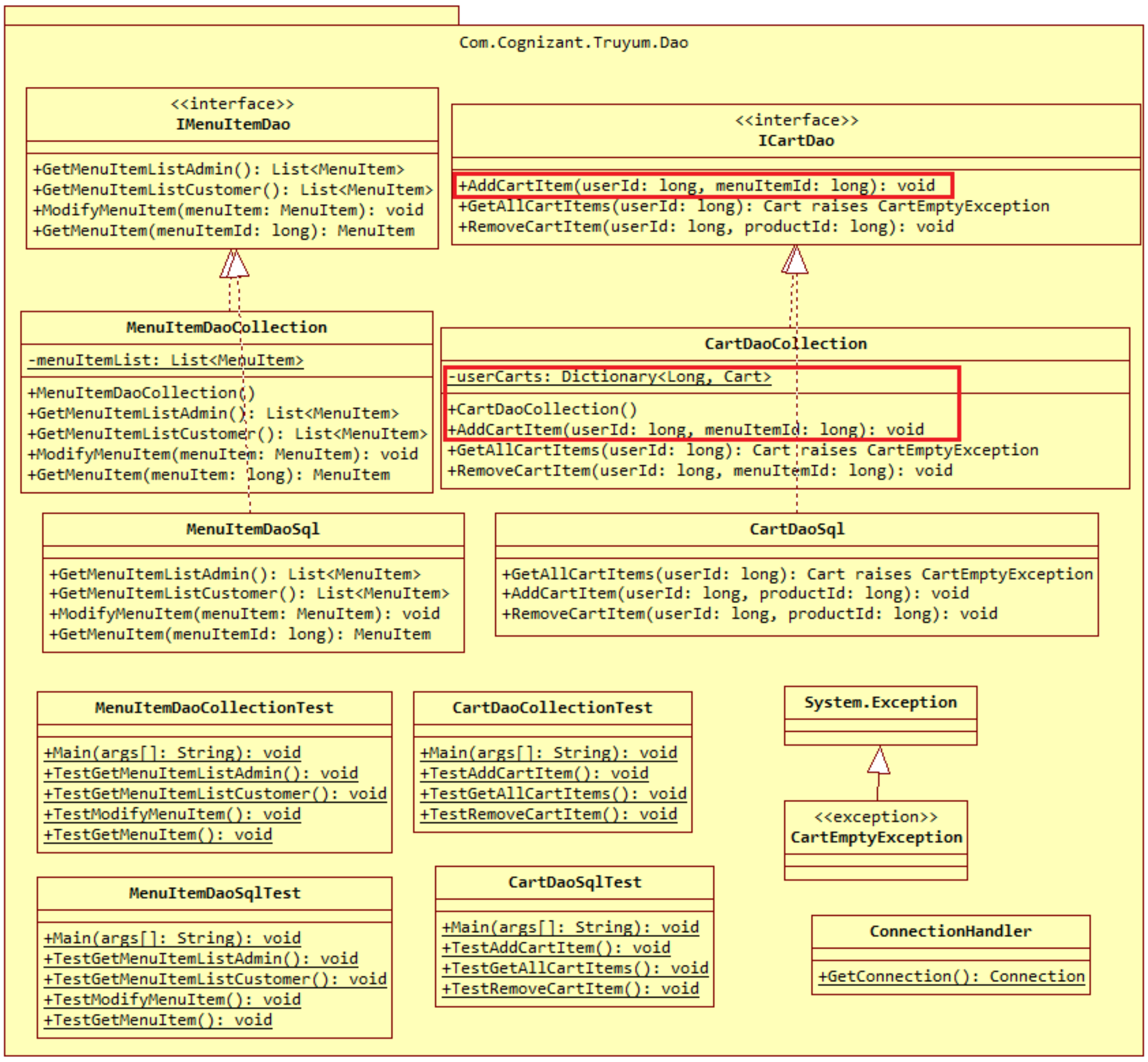
TestModifyMenuItem() of return type void

1. Create an instance for Menu Item with id matching with one of the menu item already added to the menuItemList.
2. Instantiate `MenuItemDaoCollection` and assign `MenuItemDao` reference variable `menuItemDao to its object`.
3. Invoke `MenuItemDao.ModifyMenuItem(menuItem)` by passing the menu item created in the first step.
4. Invoke `MenuItemDao.GetMenuItem(producId)` to read and check if the menu item details are modified.
5. Invoke this method in the Main method of **TruyumConsole** application.

6.0 Design for Add to Cart (EKUC004)

6.1 Class Diagram

The below diagram denotes the methods that needs to be implemented for this use case. Method wise specification is defined after the diagram.


Com.Cognizant.Truyum.Dao
<<interface>>
IMenuItemDao
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
<<interface>>
ICartDao
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollection
-menuItemList: List<MenuItem>
+MenuItemDaoCollection()
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItem: long): MenuItem
CartDaoCollection
-userCarts: Dictionary<Long, Cart>
+CartDaoCollection()
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, menuItemId: long): void
MenuItemDaoSql
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
CartDaoSql
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+AddCartItem(userId: long, productId: long): void
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollectionTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoCollectionTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
System.Exception
<<exception>>
CartEmptyException
MenuItemDaoSqlTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoSqlTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
ConnectionHandler
+GetConnection(): Connection

6.2 ICartDao.cs

1. Add method `AddCartItem(long userId, long menuItemId)` **of return type** `void` in the interface.

6.3 CartDaoCollection.cs

This class manages the data related to Cart of all users of truYum application. A new method needs to be added for this use case.

Constructor CartDaoCollection()

Data for all users will be stored in the Dictionary available in Cart instance. This constructor initialized the Cart as well as the Dictionary within the Cart, so that the class instance is ready to store values in the Dictionary when Customer adds items into the Cart.

1. Check if the `userCart` instance variable is null or not

2. If userCart is null then create a new instance of `Dictionary` with type `Long` and `Cart` and assign it to userCart instance variable.

3. The userCart instance variable will hold the cart details for each user in a `Dictionary`. The key of this `Dictionary` will have the `userId`. Each value in the `Dictionary` will be the Cart item that internally contains the list of MenuItems.

AddCartItem(long userId, long menuItemId) of return type void

This method is invoked when Customer clicks Add to Cart link in menu item listing screen. This method gets the Cart from the Dictionary for the specific user and adds the menu item into the cart's menu item list. If there is no such user in the Dictionary, then a new entry needs to be added in the Dictionary with userId as key and new Cart item with a List of Menu Items as value.

1. Instantiate `MenuItemDaoCollection` and assign `MenuItemDao` reference variable `menuItemDao to its object`.

2. Get the `menuItem` using `menuItemDao.GetMenuItem(menuItemId)` method

3. Check existence of user in `userCart` based on `userId`

4. If user exists in `userCart`, perform the steps below:

    a. Get the `menuItemList` from the userCart

    b. Add the `menuItem` obtained in previous step into `menuItemList`

5. If user does not exist in userCart, perform the steps below:

    a. Create a new `Cart` instance with new `List`

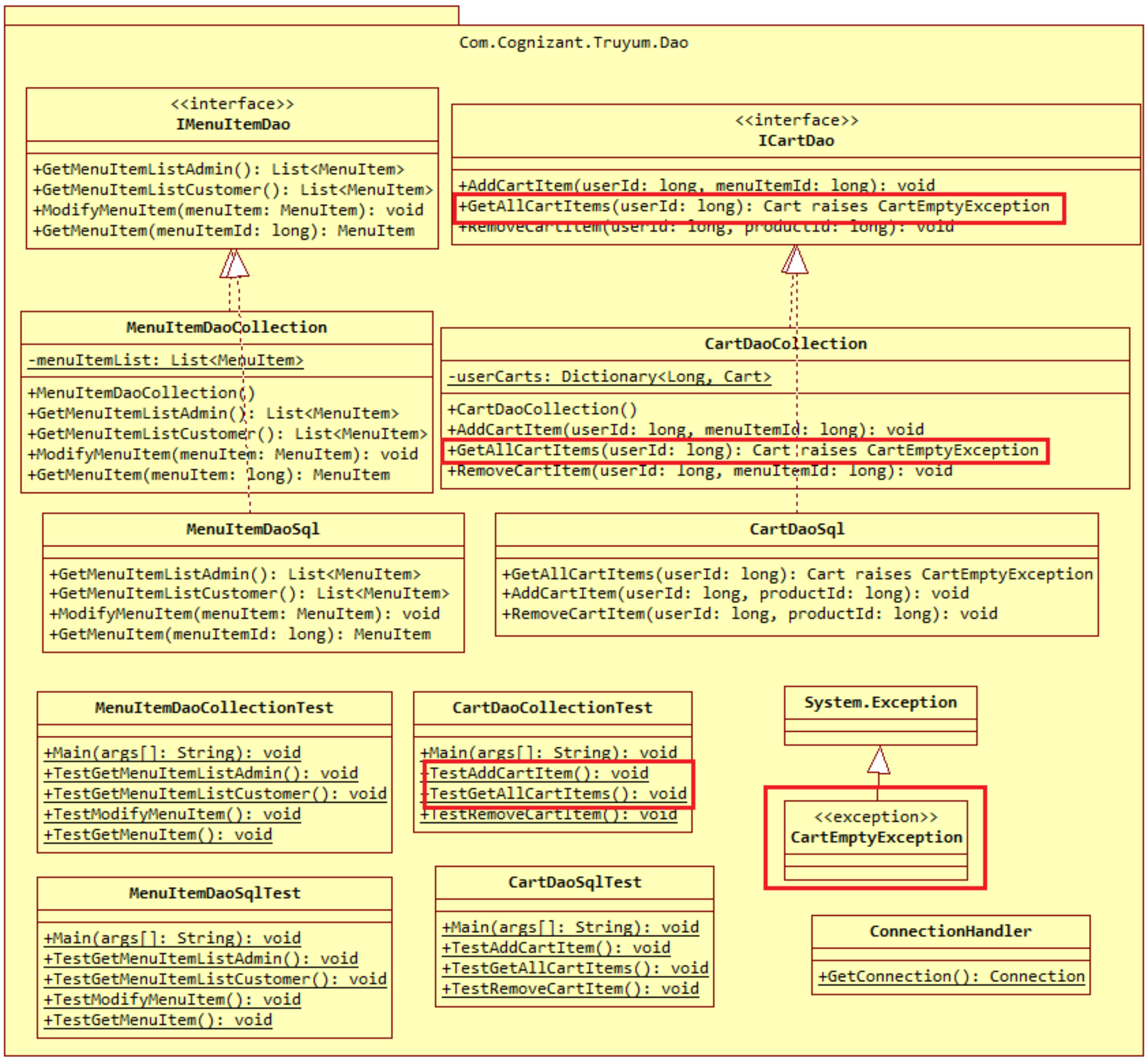
    b. Add the menu item obtained in step one and add it to `menuItemList` created in previous step

    c. Put the `userId` and `List` of `MenuItem` into the userCart

7.0 Design for View Cart (EKUC005)

7.1 Class Diagram

The below diagram denotes the methods that needs to be implemented for this use case. Method wise specification is defined after the diagram.


Com.Cognizant.Truyum.Dao
<<interface>> IMenuItemDao
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
<<interface>> ICartDao
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollection
-menuItemList: List<MenuItem>
+MenuItemDaoCollection()
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItem: long): MenuItem
CartDaoCollection
-userCarts: Dictionary<Long, Cart>
+CartDaoCollection()
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, menuItemId: long): void
MenuItemDaoSql
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
CartDaoSql
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+AddCartItem(userId: long, productId: long): void
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollectionTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoCollectionTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
System.Exception
<<exception>> CartEmptyException
MenuItemDaoSqlTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoSqlTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
ConnectionHandler
+GetConnection(): Connection

7.2 ICartDao.cs

1. Add method `GetAllCartItems(long userId)` **of return type** `void` in the interface.

7.3 CartEmptyException.cs

1. Extend this class from `System.Exception` and include an empty constructor.

7.4 CartDaoCollection.cs

This class manages the data related to Cart of all users of truYum application. A new method needs to be added for this use case.

GetAllCartItems(long userId) of return type Cart throws CartEmptyException

Method to get list of menu items added by a customer to Cart.

1. Get the menuItemList based on `userId` from the `Dictionary` of `userCart`
2. If the returned list is empty
   a. Create new `CartEmptyException` and throw it
3. If the returned list is not empty
   a. Iterate through the `menuItemList` and add up the prices.
   b. Set the `total` instance variable of `cart` with the added up menu item prices.
   c. return cart

7.5 CartDaoCollectionTest.cs

1. Create `TestAddCartItem()`

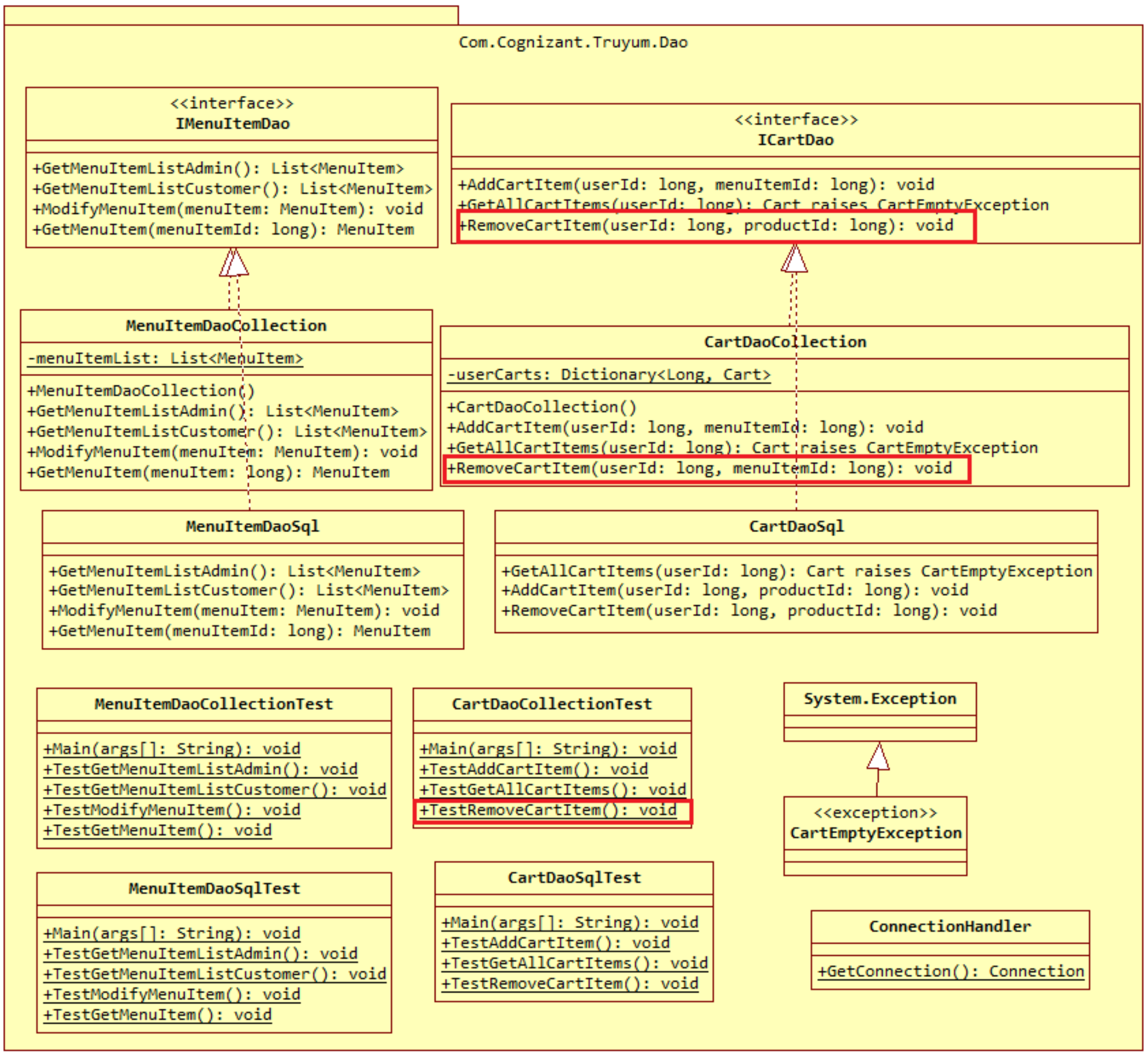
TestAddCartItem() of return type void

1. Instantiate `CartDaoCollection` and assign `CartDao` reference variable `cartDao to its object`.
2. Invoke `CartDao`.AddCart`Item()` method with following parameters
   a. userId: 1
   b. menuItemId: one of existing `menuItemId` in `MenuItemDaoCollection`
3. Invoke `CartDao.GetAllCartItems()` with userId as 1
4. Display the contents of `MenuItemList` returned in previous step and check if the added cart item is present or not.
5. Invoke this method in the Main method of **TruyumConsole** application.

8.0 Design for Remove Cart Item (EKUC006)

8.1 Class Diagram

The below diagram denotes the methods that needs to be implemented for this use case. Method wise specification is defined after the diagram.


Com.Cognizant.Truyum.Dao
<<interface>> IMenuItemDao
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
<<interface>> ICartDao
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollection
-menuItemList: List<MenuItem>
+MenuItemDaoCollection()
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItem: long): MenuItem
CartDaoCollection
-userCarts: Dictionary<Long, Cart>
+CartDaoCollection()
+AddCartItem(userId: long, menuItemId: long): void
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+RemoveCartItem(userId: long, menuItemId: long): void
MenuItemDaoSql
+GetMenuItemListAdmin(): List<MenuItem>
+GetMenuItemListCustomer(): List<MenuItem>
+ModifyMenuItem(menuItem: MenuItem): void
+GetMenuItem(menuItemId: long): MenuItem
CartDaoSql
+GetAllCartItems(userId: long): Cart raises CartEmptyException
+AddCartItem(userId: long, productId: long): void
+RemoveCartItem(userId: long, productId: long): void
MenuItemDaoCollectionTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoCollectionTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
System.Exception
<<exception>> CartEmptyException
MenuItemDaoSqlTest
+Main(args[]: String): void
+TestGetMenuItemListAdmin(): void
+TestGetMenuItemListCustomer(): void
+TestModifyMenuItem(): void
+TestGetMenuItem(): void
CartDaoSqlTest
+Main(args[]: String): void
+TestAddCartItem(): void
+TestGetAllCartItems(): void
+TestRemoveCartItem(): void
ConnectionHandler
+GetConnection(): Connection

8.2 ICartDao.cs

1. Add method `RemoveCartItem(long userId, long menuItemId) of return type void` in the interface.

8.3 CartDaoCollection.cs

This class manages the data related to Cart of all users of truYum application. A new method needs to be added for this use case.

RemoveCartItem(long userId, long menuItemId) of return type void

Method to remove a menu item from the cart. This will be invoked when Customer clicks Delete link in the Cart screen.

1. Get the `List<MenuItem>` from userCart based on `userId`
2. Iterate through the `List` of `MenuItem` and perform the below steps
   a. Check if the `menuItemId` of each menu item from the list matches with this methods input parameter
   b. If `menuItemId` matches then remove the menu item from the list

8.4 CartDaoCollectionTest.cs

1. Create `TestRemoveCartItem()`

TestRemoveCartItem() of return type void

1. Instantiate `CartDaoCollection` and assign it `CartDao` reference variable `cartDao`.
2. Invoke `CartDao.RemoveCartItem()` method with following parameters
   a. `userId`: 1
   b. `menuItemId`: Same `menuItemId` as what was provided when testing add cart item.
3. Invoke `CartDao.GetAllCartItems()` with userId as 1
4. Enclose the above method within `try catch` block with catch block handling `CartEmptyException`. Check if the catch block is executed, which means that the cart item added during `TestAddCartItem()` is removed now and the cart is empty, due to which the `CartEmptyException` is thrown.
5. Invoke this method in the Main method of **TruyumConsole** application.

9.0 Standards and Guidelines

9.1 C#

1. Naming standards to be followed:
   a. Variable/parameter:
      i. Camel casing should be followed for naming variables/parameters

ii. Variable names should be short, but meaningful

iii. Single character variable names should be avoided except for temporary variables

iv. Avoid using single character for naming variables/parameters

b. Class

i. Pascal casing should be followed for naming classes

ii. Class name should be a noun

iii. Must use whole words and should not have acronyms or abbreviations

Examples: Employee, TaxCalculator

c. Method

i. Pascal casing should be followed for naming methods

ii. Must use whole words and should not have acronyms or abbreviations

2. Code Formatting

a. Class Structure

i. Place the elements of a class in the following order:

1. Static variables
2. Instance variables
3. Constructors
4. Methods and Getter/Setters

b. Spacing

i. A space before and after an operator is required

ii. A space before curly braces is required

iii. A space after a comma is required

iv. A space after semicolon in for loop is required

v. A single line space after a method is required

c. Curly braces position

i. Opening curly braces should be in the same line

ii. Closing curly braces should always be in a new line

d. Tab spacing

i. Use 4 spaces instead of tab character

ii. Increase a tab character in the lines after opening curly braces

iii. Reduce a tab character on the of closing curly braces

iv. Include one more tab in the wrapped line

e. Line Width

i. Width of a line should not exceed 100 characters

10.0 Console application to test C# class library (Com.Cognizant.Truyum.Model & Com.Cognizant.Truyum.Dao)

1. Create a Console application project – **TruYumConsole**
2. Reference the **Com.Cognizant.Truyum.Model & Com.Cognizant.Truyum.Dao** class library projects in TruYumConsole application
3. Invoke the methods of the Com.Cognizant.Truyum.Model & Com.Cognizant.Truyum.Dao from the TruYumConsole application
4. For C# testing, please set up the TruYumConsole application as the startup project

11.0 Change Log

	Changes Made			
V1.0.0	Initial baseline created on 20-May-19 by Ramamoorthy Selvam			
Vx.y.z	<Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed>			
	Section No.	**Changed By**	**Effective Date**	**Changes Effected**