Cognizant Academy

truYum

JDBC Specification Document

Version 1.0

	Prepared By / Last Updated By	Reviewed By	Approved By
Name	Chandrasekaran Janardhanan	Vimalathithan Krishnan	Ramadevanahalli Lingachar, Shashidhara Murthy
Role	Learning Solution Designer	Learning Solution Architect	Learning Solution Lead
Signature			
Date	23 May 2019	23 May 2019	17 Jun 2019

Table of Contents

1.0 Introduction

1.1 Purpose of this document

The purpose of this document is to define the JDBC module implementation for truYum project.

1.2 Definitions & Acronyms

Definition / Acronym	Description
DAO	Data Access Object
JDBC	Java Database Connectivity

1.3 Project Overview

Refer truYum-use-case-specification.docx for understanding the functionality and features.

1.4 Scope

1. Creation of DAO classes and methods for reading and persisting data of 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 4GB Ram

2. Software Requirement

    a. JDK 1.8

    b. Eclipse IDE for Enterprise Java Developers 2019-03 R

    c. MySQL Community Server 8.0

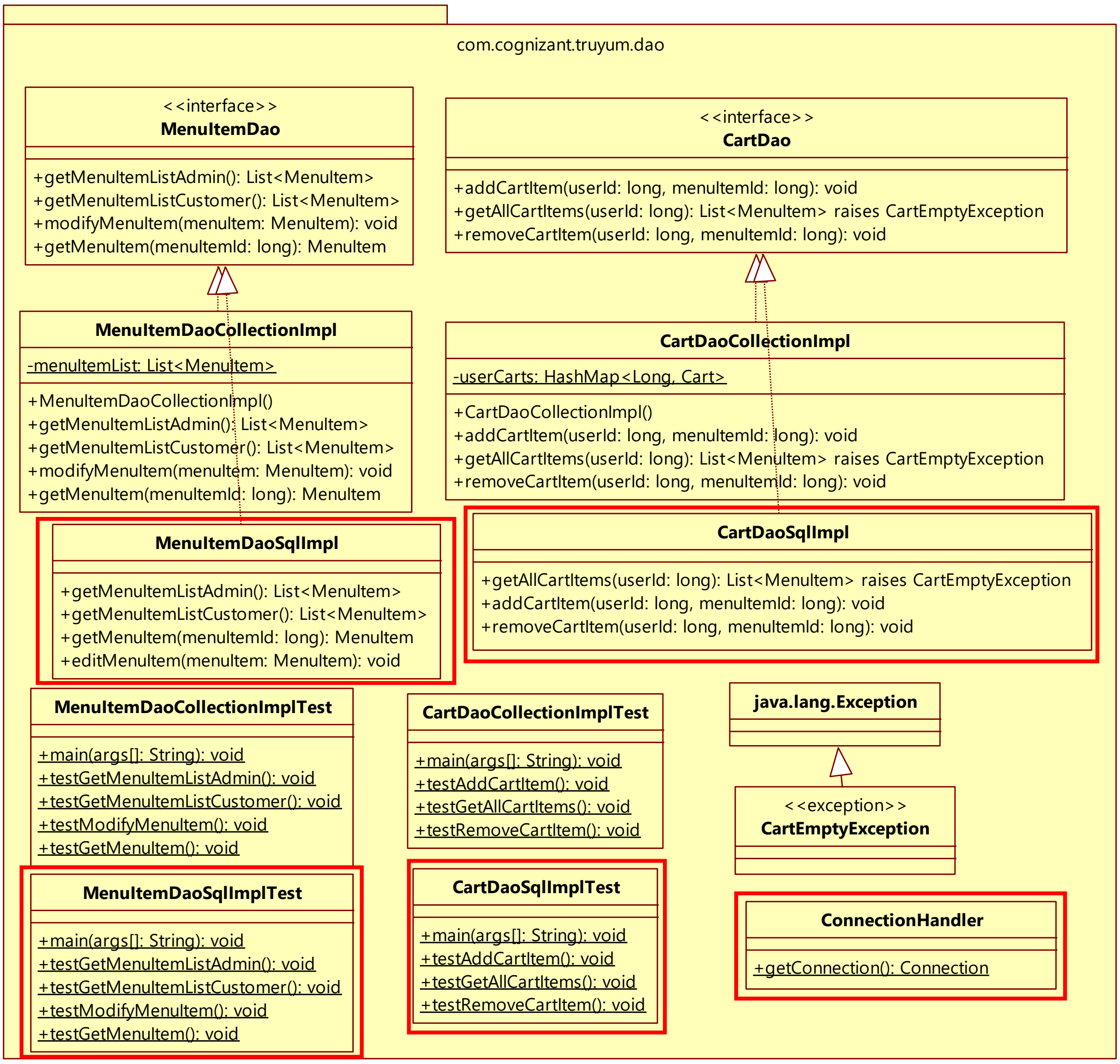
2.0 Project

Use truYum eclipse project created earlier.

3.0 Class Diagram

3.1 Data Access Layer

Refer the diagram below and create classes accordingly.


com.cognizant.truyum.dao
<<interface>>
MenuItemDao
+getMenuItemListAdmin(): List<MenuItem>
+getMenuItemListCustomer(): List<MenuItem>
+modifyMenuItem(menuItem: MenuItem): void
+getMenuItem(menuItemId: long): MenuItem
<<interface>>
CartDao
+addCartItem(userId: long, menuItemId: long): void
+getAllCartItems(userId: long): List<MenuItem> raises CartEmptyException
+removeCartItem(userId: long, menuItemId: long): void
MenuItemDaoCollectionImpl
-menuItemList: List<MenuItem>
+MenuItemDaoCollectionImpl()
+getMenuItemListAdmin(): List<MenuItem>
+getMenuItemListCustomer(): List<MenuItem>
+modifyMenuItem(menuItem: MenuItem): void
+getMenuItem(menuItemId: long): MenuItem
CartDaoCollectionImpl
-userCarts: HashMap<Long, Cart>
+CartDaoCollectionImpl()
+addCartItem(userId: long, menuItemId: long): void
+getAllCartItems(userId: long): List<MenuItem> raises CartEmptyException
+removeCartItem(userId: long, menuItemId: long): void
MenuItemDaoSqlImpl
+getMenuItemListAdmin(): List<MenuItem>
+getMenuItemListCustomer(): List<MenuItem>
+getMenuItem(menuItemId: long): MenuItem
+editMenuItem(menuItem: MenuItem): void
CartDaoSqlImpl
+getAllCartItems(userId: long): List<MenuItem> raises CartEmptyException
+addCartItem(userId: long, menuItemId: long): void
+removeCartItem(userId: long, menuItemId: long): void
MenuItemDaoCollectionImplTest
+main(args[]: String): void
+testGetMenuItemListAdmin(): void
+testGetMenuItemListCustomer(): void
+testModifyMenuItem(): void
+testGetMenuItem(): void
CartDaoCollectionImplTest
+main(args[]: String): void
+testAddCartItem(): void
+testGetAllCartItems(): void
+testRemoveCartItem(): void
java.lang.Exception
<<exception>>
CartEmptyException
MenuItemDaoSqlImplTest
+main(args[]: String): void
+testGetMenuItemListAdmin(): void
+testGetMenuItemListCustomer(): void
+testModifyMenuItem(): void
+testGetMenuItem(): void
CartDaoSqlImplTest
+main(args[]: String): void
+testAddCartItem(): void
+testGetAllCartItems(): void
+testRemoveCartItem(): void
ConnectionHandler
+getConnection(): Connection

Dotted arrow represents implementation of an interface.

Make note that getConnection is a static method.

Test method specification is not provided in this document. Refer similar implementation in Java Specification document.

Highlighted classes are the ones that needs to be implemented in this specification.

3.2 ConnectionHandler.java

This class will be used by each Dao implementation class for getting the database connection.

The connection details has to be stored in a properties file. Refer details below:

1. File: `truYum/src/connection.properties`

2. Content for `connection.properties`:

```
driver= com.mysql.jdbc.Driver
connection-url= jdbc:mysql://localhost:3306/lch_marketplace
user=root
password=password123
```

static getConnection(): Connection

1. Using `java.io.FileInputStream` and `java.util.Properties` read the properties from `connection.properties` file.

2. Gets connection using `ConnectionManager` based on properties from previous step and return the connection.

4.0 DAO for View Menu Item List Admin (TYUC001)

4.1 MenuItemDaoSqlImpl.java

getMenuItemListAdmin(): List<MenuItem>

1. Get `connection` using `ConnectionHandler`

2. Initialize an `ArrayList` of `MenuItem`

3. Using `PreparedStatement` execute the select query that retrieves all the records from menu_item table

4. Iterate through the `ResultSet`

5. For each item in the `ResultSet` create a new `MenuItem` instance and add it to the `ArrayList` created in the step 2 and return the `ArrayList`

5.0 DAO for View Menu Item List Customer

(TYUC002)

5.1 MenuItemDaoSqlImpl.java

getMenuItemListCustomer(): List<MenuItem>

1. Get connection using `ConnectionHandler`
2. Initialize an `ArrayList` of `MenuItem`
3. Using `PreparedStatement` execute the select query that retrieves the records from menu_item table applying the following filters:
    a. The menu item is active and
    b. The menu item is past the launch date
4. Iterate through the `ResultSet`
5. For each item in the `ResultSet` create a new `MenuItem` instance and add it to the `ArrayList` created in the step 2 and return the `ArrayList`

6.0 DAO for Edit Menu Item (TYUC003)

6.1 MenuItemDaoSqlImpl.java

getMenuItem(menuItemId: long): MenuItem

1. Get connection using `ConnectionHandler`
2. Execute select query using `PreparedStatement` that retrieves an item from `menuItem` table based on `menuItemId`.
3. Create a `MenuItem` instance and set the values for this menuItem instance from the first item of the `ResultSet`
4. Return the `menuItem` created in the previous step

editMenuItem(menuItem: MenuItem): void

1. Get connection using `ConnectionHandler`
2. Execute update statement using `PreparedStatement` that modifies the values of menuItem table based on `menuItemId`.
3. Set the parameters of the `PreparedStatement` and execute the statement.

7.0 DAO for Add a Menu Item to Cart (TYUC004)

7.1 CartDaoSqlImpl.java

addCartItem(userId: long, menuItemId: long): void

1. Get connection using `ConnectionHandler`
2. Execute insert statement using `PreparedStatement` for inserting data into `cart` table with `userId` and `menuItemId`.

8.0 DAO for View Cart (TYUC005)

8.1 CartDaoSqlImpl.java

getAllCartItems(userId: long): List<MenuItem>

1. Get connection using `ConnectionHandler`
2. Create a new instance of `Cart` with new `ArrayList<MenuItem>` and price as 0 in `Cart` constructor.
3. Execute select statement using `PreparedStatement` that joins `Cart` and `MenuItem` table to retrieve the list of products filtered based on the input `userId`.
4. Iterate through the `ResultSet`
5. For each item in the `ResultSet` create a new `Product` instance and add it to the `ArrayList` created in the step 2 and return the `ArrayList`
6. Execute select statement using `PreparedStatement` that gets sum of price after joining `Cart` and `MenuItem` filtered based on input `userId`.
7. Set the `totalPrice` of the `Cart` based on the result from above query.

9.0 DAO for Remove Item from Cart (TYUC006)

9.1 CartDaoSqlImpl.java

removeCartItem(userId: long, menuItemId: long): void

1. Get connection using `ConnectionHandler`

2. Execute delete statement using `PreparedStatement` for delete data into cart table based on `userId` and `menuItemId`.

10.0 Standards and Guidelines

10.1 DAO

1. All Java coding standards are applicable
2. Read database connection details from properties file
3. Closure of connection should be done within finally block

11.0 Change Log

<table>
<tr><th></th><th colspan="4">Changes Made</th></tr>
<tr><td>V1.0.0</td><td colspan="4">Initial baseline created on <dd-Mon-yy> by <Name of Author></td></tr>
<tr><td rowspan="3">Vx.y.z</td><td colspan="4"><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></td></tr>
<tr><th>Section No.</th><th>Changed By</th><th>Effective Date</th><th>Changes Effected</th></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>