BACS2063 Data Structures and Algorithms

ASSIGNMENT 202205

Student Name	:	Lee Wee Harn
Student ID	:	20WMR05673
Programme	:	Bachelor Degree in Software Engineering
Tutorial Group	:	RSFG6
Assignment Title	:	Food Ordering System (Catering Services)

Declaration

- I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University College's plagiarism policy.
- I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

Harn

Student's signature

08/09/2022

Date

Table of Contents

1. Introduction

I’ve chosen to do a food ordering module that falls under the catering (meal) services. The adt that I chose to use is set. The food ordering module will allow customer to add their order, remove their order, show the order and also edit the order. The system will first prompt the menu of the meal. In the menu, it contains the food ID, food name and food price. Therefore, customers can order their meal by looking at the menu as a reference. The system will then prompt a list of option that will let customer to choose whether they want to add order, remove order display order or edit order.

For add order function which is the 1st option in the option list, the system will ask customer to enter the food ID of the food that they want to order. For example, there are 10 food ID in the menu. Hence, customer needs to enter food ID from F001 to F010 in the required field. After entering the food ID, customers are asked to enter the quantity that they wish to order. Next, the system will ask customers whether they want to continue order, if yes, the system will repeat the same step where food ID and quantity will be asked to fill in. If customer choose not to continue order, the system will then show the option list again so that customer can proceed with other functions.

For display order function which is the 2nd option in the option list, the system will show all of the order that had been ordered by the customer. Information such as food ID, food name, food price, quantity and the total price will be shown. The information of the order will be updated also after customer add or edit their orders. Therefore, customer can view and confirm whether the order is correct or incorrect.

For remove order function which is the 3rd option in the option list, customer will ask to enter food ID that they wish to remove. If the food ID is not available or invalid, the system will then prompt message such as “this food is not ordered” to let customer know that they had entered invalid food ID. On the other hand, if the food ID that entered by the customer is valid and available, the food will be removed and message such as “remove successfully” will be prompt.

For edit order function, which is the 4th option in the option list, the system will ask customer to enter the order that they want to edit by entering the food ID. Only valid food ID and orders that are available can be edited, or the system will prompt message such as “Not Ordered this item”. Besides that, after entering valid food ID, the system will the allow customer to edit the order quantity that they wish to change by asking customer to enter the quantity again.

2. Abstract Data Type (ADT) Specification

public String toString ()		
Description	:	Create toString in the list
Precondition	:	-
Postcondition	:	Remain unchanged
Returns	:	The string is successfully created.

public boolean remove (T anElement)		
Description	:	Remove specified element in the list.
Precondition	:	The collection class must no empty
Postcondition	:	The element in the list has been removed
Returns	:	Return True if the element has been removed successfully.

public boolean checkSubset (SetInterface anotherSet)		
Description	:	To check if a set is the subset of another set
Precondition	:	The given set is the subset of another set.
Postcondition	:	Determine whether the list contains given set
Returns	:	Return true if set is the subset of another set

public void add (String newElement)		
Description	:	Add the new element into the specified String
Precondition	:	-
Postcondition	:	The number of entries will increase
Returns	:	The new element is successfully added.

public int findByOrderID (String ID)		
Description	:	Find the String by Order ID
Precondition	:	The collection of String must not be empty
Postcondition	:	The string in the list is found.
Returns	:	The String is successfully found by Order ID.

public boolean contains (T newElement)		
Description	:	Determines whether the list contains newElement
Precondition	:	-
Postcondition	:	The list remains unchanged
Returns	:	Return true if the newElement is in the list.

public int totalSize ()		
Description	:	Show the total of the list
Precondition	:	-
Postcondition	:	Remain unchanged
Returns	:	Return the total size of the list.

3. ADT Implementation

3.1 Overview of ADT

```
package adt;

public interface SetInterface<T> {

    void add (String newElement);

    boolean remove (T anElement);

    boolean checkSubset (SetInterface anotherSet);

          int totalSize ();
      }

```

- **void add (String newElement);**
  This function will add the new element into the specified String. The number of entries will increase and the new element will be successfully added.

- **boolean remove (T anElement);**
  This function will remove specified element in the list. Before removing, the collection class must not be empty and the element in the list will be removed. The function will return true if the element has been removed successfully.

- **boolean checkSubset (SetInterface anotherSet);**
  This function is to check if a set is the subset of another set. Before checking, we must make sure that the given set is the subset of another set. After Determining whether the list contains given set, the function will return true if set is the subset of another set.

- **int totalSize ();**
  This function will Show the total of the list. It will remain unchanged and return the total size of the list.

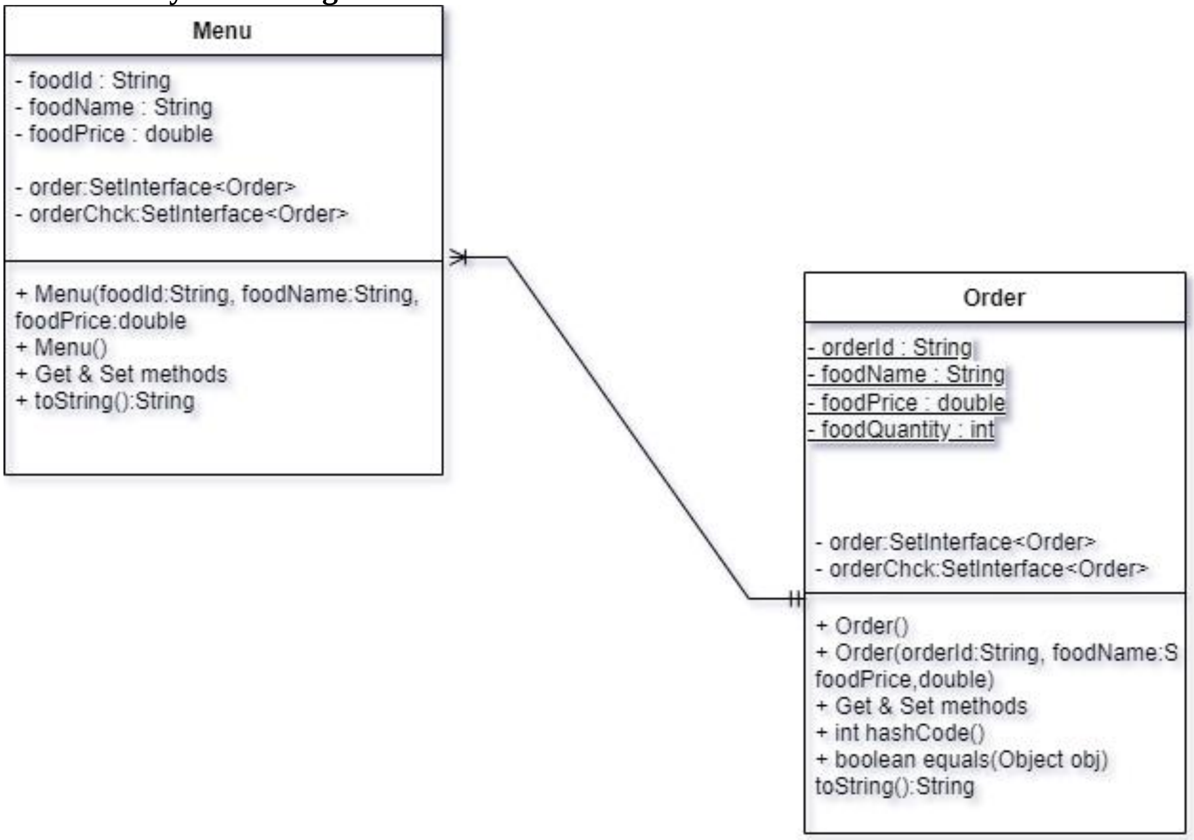
3.2 ADT Implementation

```
package adt;

public class Set<T> implements SetInterface<T> {

    private final static int DEFAULT_CAPACITY = 100;
    private static int numberOfEntries;
    private final T [] array;
    private final String [] array2;

         public Set () {
             array = (T []) new Object [DEFAULT_CAPACITY];
             array2 = new String [DEFAULT_CAPACITY];
             numberOfEntries = 0;
         }

         @Override
         public String toString () {
             String str = "";
             for (int i = 0; i < numberOfEntries; i++) {
                 str += array2[i] + "\n";
             }
             return str;
         }

         @Override
         public boolean remove (T anElement) {
             int i = 0;
             boolean found = false;
             while (numberOfEntries >= i &&! found) {
                 if (array[i]. equals(anElement)) {
                     found = true;
                 } else {
                     i++;
                 }
             }
             if (found) {
                 for (int j = i; j < numberOfEntries; j++) {
                     array[j] = array [j + 1];
                     System.out.println("delete success");
                 }
             } else if (! found) {
                 System.out.println("information not recognized");
             }
             return found;
         }
```

```java
    public void remove (String oId) {
        for (int i = 0, k = 0; i < numberOfEntries; i++) {
            if (array2[i]. contains(oId)) {
                continue;
            }
            array2[k++] = array2[i];
        }
        --numberOfEntries;
    }

    @Override
    public boolean checkSubset (SetInterface anotherSet) {
        Set<T> givenSet = (Set<T>) anotherSet;
        for (int i = 0; Set.numberOfEntries >= i; i++) {
            if (! this. contains (givenSet.array[i])) {
                return false;
            }
        }
        return true;
    }

    @Override
    public void add (String newElement) {
        array2[numberOfEntries] = newElement;
        ++numberOfEntries;
    }

    public int findByOrderID (String ID) {
        for (int i = 0; i < numberOfEntries; i++) {
            if (array2[i]. contains (ID)) {
                return i;
            }
        }
        System.out.println("Not Ordered this Item.");
        return -1;
    }

    public boolean contains (T newElement) {
        boolean exist = false;
        for (int i = 0; i < numberOfEntries &&! exist; i++) {
            if (array[i]. equals(newElement)) {
                exist = true;
            }
        }
        return exist;
    }
    @Override
        public int totalSize () {
            return numberOfEntries;
        }
    }

```

4. Entity Classes

4.1 Entity Class Diagram


Menu
- foodId : String
- foodName : String
- foodPrice : double
- order:SetInterface<Order>
- orderChck:SetInterface<Order>
+ Menu(foodId:String, foodName:String, foodPrice:double
+ Menu()
+ Get & Set methods
+ toString():String
Order
- orderId : String
- foodName : String
- foodPrice : double
- foodQuantity : int
- order:SetInterface<Order>
- orderChck:SetInterface<Order>
+ Order()
+ Order(orderId:String, foodName:S foodPrice,double)
+ Get & Set methods
+ int hashCode()
+ boolean equals(Object obj)
toString():String

4.2 Entity Class Implementation

- Order.java

```
package adt;

import java. util. Objects;

public class Order {

    static Set<Order> order = new Set<Order> ();
    static Set<Order> orderChck = new Set<Order> ();
    private static String orderId;
          private static String foodName;
          private static double foodPrice;
          private static int foodQuantity;
          int no;

          public Order () {
          }

          public Order (String orderId) {
              Order.orderId = orderId;
              no++;
          }

          public Order (String orderId, String foodName, double
                  foodPrice) {
              Order.orderId = orderId;
              Order.foodName = foodName;
              Order.foodPrice = foodPrice;
              no++;
          }

          public String orderId () {
              return orderId;
          }

          public String getOrderId () {
              return orderId;
          }

          public void setOrderId (String orderId) {
              Order.orderId = orderId;
          }

```

```
        public String getFoodName () {
            return foodName;
        }

        public void setFoodName (String foodName) {
            Order.foodName = foodName;
        }

        public double getFoodPrice () {
            return Order.foodPrice;
        }

        public void setFoodPrice (double foodPrice) {
            Order.foodPrice = foodPrice;
        }

        public void setFoodQuantity (int foodQuantity) {
            Order.foodQuantity = foodQuantity;
        }

        @Override
        public int hashCode () {
            int hash = 3;
            hash = 17 * hash + Objects.hashCode(Order.orderId);
            hash = 17 * hash + Objects.hashCode(Order.foodName);
            hash = 17 * hash + Objects.hashCode(Order.foodPrice);
            return hash;
        }

        @Override
        public boolean equals (Object obj) {
            if (this == obj) {
                return true;
            }
            if (obj == null) {
                return false;
            }
            if (getClass ()! = obj. getClass ()) {
                return false;
            }
            final Order other = (Order) obj;
            if (! Objects.equals(Order.orderId, Order.orderId)) {
                return false;
            }
            if (! Objects.equals(Order.foodName, Order.foodName))
{
                return false;
```

```java
    return Objects.equals(Order.foodPrice, Order.foodPrice);
        }

        @Override
        public String toString () {
            return "|" + " " + Order.orderId + " " + "\t" + "\t" + "|" + " " + Order.foodName + "\t" + "\t" + "|" + " " + Order.foodQuantity + "\t" + "\t" + " " + " " + " " + "|" + " " + "RM " + Order.foodPrice + "\t" + " |";
        }
    }

```

- Menu.java

```
package adt;

public class Menu {

    static Set<Order> order = new Set<Order> ();
    static Set<Order> orderChck = new Set<Order> ();

    private String foodId;
    private String foodName;
        private double foodprice;

        public String getFoodId () {
            return foodId;
        }

        public void setFoodId (String foodId) {
            this. foodId = foodId;
        }

        public String getFoodName () {
            return foodName;
        }

        public void setFoodName (String foodName) {
            this. foodName = foodName;
        }

        public double getFoodPrice () {
            return foodprice;
        }

        public void setFoodPrice (double foodprice) {
            this. foodprice = foodprice;
        }

        public Menu () {

        }

        public Menu (String foodId, String foodName, double foodprice) {
            this. foodId = foodId;
            this. foodName = foodName;
            this. foodprice = foodprice;
        }

    @Override
    public String toString () {
        return "|" + " " + foodId + " " + "\t" + "\t" + "|" + " " + foodName + "\t" + "\t" + "|" + " " + "RM " + foodprice + "\t" + "|";
    }
}
```

5. Client Program

In this assignment, I’ve selected set as my collection ADT. The reason of using set in this project is because it can extend the collection that unordered collection of objects in which duplicate values cannot be stored. The collection ADT that I used allow the new element into the specified String, remove specified element in the list and also check if a set is the subset of another set.

- Console Based Prototype (OrderManager.java)

```java
package adt;

import static adt. MainClient.displayOption;
import java. util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

    public class OrderManager {

        static Scanner myObj = new Scanner(System.in); // Create a Scanner object
        static Set<Order> order = new Set<Order>();
        static Set<Order> orderChck = new Set<Order>();
        static int[] Quan_new;
        static Order o = new Order();
        static Menu m = new Menu();
        static int menuIdx = 0;
        static int orders_no = 0;
        static float orders_price = 0;

        static void addOrder(Menu[] menu, int orderTotal) throws IOException {
            char pick;
            do {
                System.out.println("Enter the Food ID to order:");
                String oId = myObj.next();
                for (int i = 0; i < menuIdx; i++) {

                    if (oId.compareToIgnoreCase(menu[i].getFoodId()) == 0) {
                        System.out.println("Enter the Quantity: ");
                        int quantity = myObj.nextInt();
                        o.setOrderId(oId);

    o.setFoodName(menu[i].getFoodName());
                        o.setFoodPrice(menu[i].getFoodPrice());
                        o.setFoodQuantity(quantity);
                        Quan_new[i] = quantity;
                        orders_price += (menu[i].getFoodPrice() * quantity);
                        orders_no++;
                        break;
                    }
                }
                order.add(o.toString());
                System.out.printf("Do you wish to continue add order ?\n");
                System.out.printf("Y = Yes,N = No\n");
                pick = myObj.next().charAt(0);
                if (pick == 'n') {
                    displayOption(menu, orderTotal);
                }
            } while (pick == 'y');
        }

        static void displayOrder(Menu[] menu, int orderTotal) throws IOException {

            try {
                FileWriter writer;
                writer = new FileWriter("writeFile.txt");
                writer.write(order.toString());

                writer.close();
            } catch (IOException e) {
                System.out.println("An Error occurred while writing into the file");
            }
            System.out.println("*==============================================================================*");
            System.out.println("|     ORDER ID    | FOOD NAME           |      QUANTITY     |     PRICE     |");
            System.out.println("*==============================================================================*");
            BufferedReader reader;
            try {
                reader = new BufferedReader(new
String line = reader.readLine();
            while (line != null) {

                System.out.println(line);

                // read next line
                line = reader.readLine();
            }
            reader.close();
        } catch (IOException e) {
        }


        System.out.println("*================================================================================*");
        System.out.println("|TOTAL NUMBER OF ORDERS" + "\t\t\t\t" + "|" + " " + orders_no + "\t\t" + "   " + "|" + "\t\t" + " |");
        System.out.printf("|TOTAL PRICE" + "\t\t\t\t\t" + "|" + "\t\t" + "   " + "|" + " " + "RM " + orders_price + "\t" + " |" + "\n");
        System.out.println("*================================================================================*");
    }

    static void editOrder(Menu[] menu, int orderTotal) {
        System.out.println("Enter the Order ID to edit order: ");
        String oId = myObj.next();
        if (order.findByOrderID(oId) != -1) {
            System.out.println("Enter the Quantity again: ");
            int quan = myObj.nextInt();
            for (int i = 0; i < menuIdx; i++) {
                if (oId.compareToIgnoreCase(menu[i].getFoodId()) == 0) {
                    o.setOrderId(oId);
                    o.setFoodName(menu[i].getFoodName());
                    o.setFoodPrice(menu[i].getFoodPrice());
                    o.setFoodQuantity(quan);
                    orders_price -= Quan_new[i] * menu[i].getFoodPrice();
    orders_price += (menu[i].getFoodPrice() * quan);
                        Quan_new[i] = quan;
                        break;
                    }
                }
                order.remove(oId);
                order.add(o.toString());

            } else {
                System.out.println("Order Not Found\n");
            }
        }

        static void removeOrder(Menu[] menu, int orderTotal) {
            System.out.printf("Enter the Food ID that you wish to remove :\n");
            String oId = myObj.next();
            if (order.findByOrderID(oId) != -1) {
                int quan = 0;
                for (int i = 0; i < menuIdx; i++) {
                    if (oId.compareToIgnoreCase(menu[i].getFoodId()) == 0) {
                        o.setOrderId(oId);
                        o.setFoodName(menu[i].getFoodName());
                        o.setFoodPrice(menu[i].getFoodPrice());
                        o.setFoodQuantity(quan);
                        orders_price -= Quan_new[i] * menu[i].getFoodPrice();
                        Quan_new[i] = quan;
                        orders_no--;
                        break;
                    }
                }
                order.remove(oId);
                System.out.println("Remove Succesful");
            }
        }

    }
```

- GUI Based Prototype (MainClient.java)

```java
package adt;

import java.io.IOException;

public class MainClient {

    static Set<Order> order = new Set<Order>();
    static Set<Order> orderChck = new Set<Order>();

        public static void main(String[] args) throws IOException {

            ReadFile read = new ReadFile();
            Menu[] menu = read.readFile();
            OrderManager.menuIdx = read.actualSize();
            OrderManager.Quan_new = new int[OrderManager.menuIdx];

            int orderTotal = 0;

System.out.println("*=============================================================*");
System.out.println("|/////////////////////////////MENU//////////////////////////////|");
System.out.println("*=============================================================*");
            System.out.println("|    ORDER ID   |   FOOD NAME          |    PRICE     |");
System.out.println("*=============================================================*");
            for (orderTotal = 0; menu.length >= orderTotal; orderTotal++) {
                if (menu[orderTotal] != null) {
                    System.out.println(menu[orderTotal]);
                } else {
                    break;
                }
            }

        System.out.println("*=============================================================*");

            displayOption(menu, orderTotal);
        }

    static void displayOption(Menu[] menu, int orderTotal) throws IOException {
        boolean exit = false;

        while (!exit) {
            System.out.printf("\n*======================*\n");
            System.out.printf("|    DISPLAY OPTION    |\n");
            System.out.printf("*======================*\n");
            System.out.printf("|1. | Add Order        |\n");
            System.out.printf("|2. | Display order    |\n");
            System.out.printf("|3. | Remove order     |\n");
            System.out.printf("|4. | Edit order       |\n");
            System.out.printf("|5. | Exit             |\n");
            System.out.printf("*======================* \n");

            System.out.printf("Please choose: \n");
            int choose = OrderManager.myObj.nextInt(); // Read user input

            switch (choose) {
                case 1 ->
                    OrderManager.addOrder(menu, orderTotal);
                case 2 ->
                    OrderManager.displayOrder(menu, orderTotal);
                case 3 ->
                    OrderManager.removeOrder(menu, orderTotal);
                case 4 ->
                    OrderManager.editOrder(menu,
     case 5 ->
                                  System.exit(0);
                              default ->
                                  System.out.println("Invalid
Option ! Please Choose Option from 1 - 4");
                          }
                      }
                  }
              }
```