Memo

Status: Project is complete.

The project Delectable is a Catering application that primarily deals with Menu and Ordering functionalities. I have incorporated all the use cases as outlined in the API and as discussed in the class. Additionally I have also integrated data persistence, for this I have used csv files to store and retrieve data. I particularly chose csv and not Json/xml because I come from a Quant Finance background and we have been predominantly using csv files as a data-source in the research codes we write. However I realize that using Json/xml as data files for this project would have been better, but I got to know this at the very end, when we were discussing about data persistence, hence would have been difficult to change my data source. Also there is just one other thing that is not programed as instructed; one of the Reports where the user requests to see all the orders between certain dates is not functioning as desired.

* Link to code repo: <https://github.com/jimeetns/Project445>
* Lines of code: Around 6500, Please see appendix
* Lines of code in unit tests: Around 1240, Please see appendix
* Unit test coverage: 83%

[EMMA v2.0.5312 report, generated Fri Apr 22 17:13:48 CDT 2016]

-------------------------------------------------------------------------------

OVERALL COVERAGE SUMMARY:

[class, %] [method, %] [block, %] [line, %] [name]

71% (10/14)! 82% (91/111) 83% (5225/6320) 77% (963.3/1244)! all classes

OVERALL STATS SUMMARY:

total packages: 1

total classes: 14

total methods: 111

total executable files: 14

total executable lines: 1244

COVERAGE BREAKDOWN BY PACKAGE:

[class, %] [method, %] [block, %] [line, %] [name]

71% (10/14)! 82% (91/111) 83% (5225/6320) 77% (963.3/1244)! default package

-------------------------------------------------------------------------------

* Cyclomatic complexity: Max: 15 Std Dev: 2.681 Mean: 2.433
* A list of challenges you faced while working on this assignment and the solutions you found to those challenges.

I spent a little more than 100 hours to develop this application and get the code working. I may have spent more hours than those spent by my class mates because of my unfamiliarity with Java and a lot of processes around the project (like building the rest controller etc). And I spent a couple of hours preparing for submission which basically included writing the script, uploading to git, documenting this memo, read me file etc.

Coming from a non-Computers background (I have some programming experience and knowledge but no core Computers' knowledge), this project served to polish my programming skills immensely. In the sense that this really gave me an opportunity to structure my program in an object oriented fashion, understanding the design and programming implementation in a detailed manner.

The most challenging part for me was to enable the Rest Controller part and writing the script. Had to do a lot of research online and took a friends help to prepare that.

Appendix:

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

import java.io.FileWriter;

import java.text.\*;

public class Admin {

private String surchargeFile = "Surcharge.csv";

public Admin()

{

}

public int Create\_Menu(List<String> itemsToAdd)

{

int idCreated = 0;

Menu m = new Menu();

idCreated = m.Create\_Menu(itemsToAdd);

return idCreated;

}

public void Modify\_MenuPrice(int idToModify, int modifiedPrice)

{

Menu m = new Menu();

int idModified = m.Modify\_Menu(idToModify, modifiedPrice);

System.out.println(idModified);

}

public double Get\_Surcharge()

{

double surcharge = 0;

try{

BufferedReader br = new BufferedReader(new FileReader(surchargeFile));

br.readLine();

String temp = br.readLine();

surcharge = Double.parseDouble(temp);

br.close();

} catch(FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return surcharge;

}

public void Modify\_Surcharge(double surchargeToMod)

{

try{

FileWriter writer = new FileWriter(surchargeFile);

writer.append("Surcharge");

writer.append('\n');

writer.append(String.valueOf(surchargeToMod));

writer.append('\n');

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

}

public void Order\_Delivered(int oidToModify)

{

Order o = new Order();

int oidModified = o.Cancel\_Delivered\_Order(oidToModify, "Delivered");

System.out.println(oidModified);

}

}

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)

@SuiteClasses({ TestCustomerClass.class, TestMenuClass.class, TestOrderClass.class })

public class AllTests {

public static void main(String[] args) {

org.junit.runner.JUnitCore.main("AllTests");

}

}

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

import java.io.FileWriter;

import java.text.\*;

public class Customer {

private List<Integer> id;

private List<String> nameL;

private List<String> emailL;

private List<String> phoneL;

private String name;

private String email;

private String phone;

//private String customerFile = "/home/jimeet/workspace/Project445/Customer.csv";

private String customerFile;

public Customer(){

//customerFile = "/home/jimeet/workspace/Project445/Customer.csv";

customerFile = "Customer.csv";

SetParameters();

ReadFile();

}

public Customer(String frmOrderManager){

}

public String getNameTemp() {

return name;

}

public void setNameTemp(String nameTemp) {

this.name = nameTemp;

}

public String getEmailTemp() {

return email;

}

public void setEmailTemp(String emailTemp) {

this.email = emailTemp;

}

public String getMobileTemp() {

return phone;

}

public void setMobileTemp(String mobileTemp) {

this.phone = mobileTemp;

}

public void Create\_Customer(List<String> itemsToAdd)

{

int idCreated = id.get(id.size() - 1) + 1;

List<String> customerToAdd = new ArrayList<String>();

String email\_ = itemsToAdd.get(8);

int found = 0;

for(int i = 0; i < id.size(); i++)

if(emailL.get(i).equals(email\_))

found=1;

if(found == 0)

{

customerToAdd.add(String.valueOf(idCreated));

customerToAdd.add(itemsToAdd.get(7));

customerToAdd.add(itemsToAdd.get(8));

customerToAdd.add(itemsToAdd.get(9));

try{

FileWriter writer = new FileWriter(customerFile, true);

for(int i = 0; i < customerToAdd.size(); i++)

{

if (customerToAdd.get(i).contains(","))

customerToAdd.set(i, "\"" + itemsToAdd.get(i) + "\"");

writer.append(customerToAdd.get(i));

writer.append(",");

}

writer.append('\n');

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

}

}

public List<List<String>> Get\_Customer()

{

List<List<String>> customerList = new ArrayList<List<String>>();

int found = 0;

for(int i = 0; i < id.size(); i++)

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

tempList.add(nameL.get(i));

tempList.add(emailL.get(i));

tempList.add(phoneL.get(i));

customerList.add(tempList);

found = 1;

}

if(found == 0)

customerList = null;

return customerList;

}

public List<List<String>> Get\_Customer(String nameStr)

{

List<List<String>> customerList = new ArrayList<List<String>>();

int found = 0;

for(int i = 0; i < id.size(); i++)

{

if(nameL.get(i).contains(nameStr))

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

tempList.add(nameL.get(i));

tempList.add(emailL.get(i));

tempList.add(phoneL.get(i));

customerList.add(tempList);

found = 1;

}

}

if(found == 0)

customerList = null;

return customerList;

}

public List<String> Get\_Customer(Integer cid)

{

List<String> customerList = new ArrayList<String>();

int found = 0;

for(int i = 0; i < id.size(); i++)

{

if(id.get(i) == cid)

{

customerList.add(String.valueOf(id.get(i)));

customerList.add(nameL.get(i));

customerList.add(emailL.get(i));

customerList.add(phoneL.get(i));

found = 1;

break;

}

}

if(found == 0)

customerList = null;

return customerList;

}

private void SetParameters()

{

id = new ArrayList<Integer>();

nameL = new ArrayList<String>();

emailL = new ArrayList<String>();

phoneL = new ArrayList<String>();

}

private void ReadFile()

{

String line = null;

try{

BufferedReader br = new BufferedReader(new FileReader(customerFile));

br.readLine();

while((line = br.readLine()) != null){

String[] customerItems = line.split(",");

id.add(Integer.parseInt(customerItems[0]));

nameL.add(customerItems[1]);

emailL.add(customerItems[2]);

phoneL.add(customerItems[3]);

}

br.close();

} catch(FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

import java.io.FileWriter;

import java.text.\*;

public class CustomerManager {

public String id;

public String name;

public String email;

public String phone;

public List<CustomerManager> DisplayCustomers()

{

List<CustomerManager> cmList = new ArrayList<CustomerManager>();

List<List<String>> customerList = new ArrayList<List<String>>();

Customer c = new Customer();

customerList = c.Get\_Customer();

if(customerList == null)

{

cmList = null;

return cmList;

}

for(int i = 0; i < customerList.size(); i++)

{

CustomerManager cmTemp = new CustomerManager();

cmTemp.id = customerList.get(i).get(0);

cmTemp.name = customerList.get(i).get(1);

cmTemp.email = customerList.get(i).get(2);

cmTemp.phone = customerList.get(i).get(3);

cmList.add(cmTemp);

}

return cmList;

}

public List<CustomerManager> DisplayCustomers(String nameStr)

{

List<CustomerManager> cmList = new ArrayList<CustomerManager>();

List<List<String>> customerList = new ArrayList<List<String>>();

Customer c = new Customer();

customerList = c.Get\_Customer(nameStr);

if(customerList == null)

{

cmList = null;

return cmList;

}

for(int i = 0; i < customerList.size(); i++)

{

CustomerManager cmTemp = new CustomerManager();

cmTemp.id = customerList.get(i).get(0);

cmTemp.name = customerList.get(i).get(1);

cmTemp.email = customerList.get(i).get(2);

cmTemp.phone = customerList.get(i).get(3);

cmList.add(cmTemp);

}

return cmList;

}

public CustomerManager DisplayCustomers(int cId)

{

CustomerManager cmList = new CustomerManager();

List<String> customerList = new ArrayList<String>();

Customer c = new Customer();

customerList = c.Get\_Customer(cId);

if(customerList == null)

{

cmList = null;

return cmList;

}

cmList.id = customerList.get(0);

cmList.name = customerList.get(1);

cmList.email = customerList.get(2);

cmList.phone = customerList.get(3);

return cmList;

}

}

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

import java.io.FileWriter;

import java.text.\*;

public class LineItems {

public String id;

public String name;

public String count;

public LineItems()

{

}

public LineItems(List<String> orderDetailsTemp) {

// TODO Auto-generated constructor stub

this.id = orderDetailsTemp.get(0);

this.name = orderDetailsTemp.get(1);

this.count = orderDetailsTemp.get(2);

}

}

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.util.\*;

import java.text.\*;

public class Menu{

public List<String> menuHeader;

public List<Integer> id;

public List<String> food;

public List<List<String>> category;

public List<Double> price;

public List<Integer> minOrder;

public List<String> createdDate;

public List<String> modifiedDate;

//public String menuFile = "/home/jimeet/workspace/Project445/Menu.csv";

public String menuFile = new File("Menu.csv").getAbsolutePath();

public Menu(){

SetParameters();

ReadFile();

}

public int Create\_Menu(List<String> itemsToAdd){

int idCreated = id.get(id.size() - 1) + 1;

itemsToAdd.add(0, String.valueOf(idCreated));

Date date1 = new Date();

Date date2 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");

itemsToAdd.add(formatter.format(date1));

itemsToAdd.add(formatter.format(date2));

WriteFile(itemsToAdd, 0);

return idCreated;

}

public List<List<String>> Display\_Menu(){

SetParameters();

ReadFile();

List<List<String>> menuDisplay = new ArrayList<List<String>>();

for (int i = 0; i < id.size(); i++)

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

tempList.add(food.get(i));

tempList.add(String.valueOf(price.get(i)));

tempList.add(String.valueOf(minOrder.get(i)));

String tempCategory = category.get(i).get(0);

for(int j = 1; j < category.get(i).size(); j++)

tempCategory = tempCategory + "\_" + category.get(i).get(j);

tempList.add(tempCategory);

tempList.add(createdDate.get(i));

tempList.add(modifiedDate.get(i));

menuDisplay.add(tempList);

}

return menuDisplay;

}

public List<List<String>> Display\_Menu(int menuId){

SetParameters();

ReadFile();

List<List<String>> menuDisplay = new ArrayList<List<String>>();

int found = 0;

for (int i = 0; i < id.size(); i++)

{

if(id.get(i) == menuId)

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

tempList.add(food.get(i));

tempList.add(String.valueOf(price.get(i)));

tempList.add(String.valueOf(minOrder.get(i)));

String tempCategory = category.get(i).get(0);

for(int j = 1; j < category.get(i).size(); j++)

tempCategory = tempCategory + "\_" + category.get(i).get(j);

tempList.add(tempCategory);

tempList.add(createdDate.get(i));

tempList.add(modifiedDate.get(i));

menuDisplay.add(tempList);

found = 1;

break;

}

}

if(found == 1)

return menuDisplay;

else

{

menuDisplay = null;

return menuDisplay;

}

}

/\*public int Modify\_Menu(int idToModify, List<String> itemsToModify){

SetParameters();

ReadFile();

List<List<String>> itemsReAdd = new ArrayList<List<String>>();

int found = 0;

for (int i = 0; i < id.size(); i++)

{

List<String> tempList = new ArrayList<String>();

if(id.get(i) == idToModify)

{

tempList.add(String.valueOf(id.get(i)));

tempList.add(itemsToModify.get(0));

tempList.add(itemsToModify.get(1));

tempList.add(itemsToModify.get(2));

tempList.add(itemsToModify.get(3));

Date date1 = new Date();

Date date2 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");

try

{date1 = formatter.parse(createdDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter1 = new SimpleDateFormat("MM/dd/yyyy");

tempList.add(String.valueOf(formatter1.format(date1)));

tempList.add(String.valueOf(formatter1.format(date2)));

itemsReAdd.add(tempList);

found = 1;

}

else

{

tempList.add(String.valueOf(id.get(i)));

tempList.add(food.get(i));

tempList.add(String.valueOf(price.get(i)));

tempList.add(String.valueOf(minOrder.get(i)));

String tempCategory = category.get(i).get(0);

for(int j = 1; j < category.get(i).size(); j++)

tempCategory = tempCategory + "\_" + category.get(i).get(j);

tempList.add(tempCategory);

Date date1 = new Date();

Date date2 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");

try

{date1 = formatter.parse(createdDate.get(i));

date2 = formatter.parse(modifiedDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter1 = new SimpleDateFormat("MM/dd/yyyy");

tempList.add(String.valueOf(formatter1.format(date1)));

tempList.add(String.valueOf(formatter1.format(date2)));

itemsReAdd.add(tempList);

}

}

if(found == 1)

{

try{

FileWriter writer = new FileWriter(menuFile);

for(int i = 0; i < menuHeader.size(); i++)

{

writer.append(menuHeader.get(i));

writer.append(",");

}

writer.append('\n');

for(int j = 0; j < itemsReAdd.size(); j++)

{

for(int i = 0; i < itemsReAdd.get(j).size(); i++)

{

writer.append(itemsReAdd.get(j).get(i));

writer.append(",");

}

writer.append('\n');

}

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

return idToModify;

}

else

{

return 0;

}

}\*/

public int Modify\_Menu(int idToModify, double modifiedPrice){

SetParameters();

ReadFile();

List<List<String>> itemsReAdd = new ArrayList<List<String>>();

int found = 0;

for (int i = 0; i < id.size(); i++)

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

tempList.add(food.get(i));

//tempList.add(String.valueOf(price.get(i)));

tempList.add(String.valueOf(minOrder.get(i)));

String tempCategory = category.get(i).get(0);

for(int j = 1; j < category.get(i).size(); j++)

tempCategory = tempCategory + "\_" + category.get(i).get(j);

tempList.add(tempCategory);

if(id.get(i) == idToModify)

{

tempList.add(2, String.valueOf(modifiedPrice));

Date date1 = new Date();

Date date2 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");

try

{date1 = formatter.parse(createdDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter1 = new SimpleDateFormat("MM/dd/yyyy");

tempList.add(String.valueOf(formatter1.format(date1)));

tempList.add(String.valueOf(formatter1.format(date2)));

found = 1;

}

else

{

tempList.add(2, String.valueOf(price.get(i)));

Date date1 = new Date();

Date date2 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");

try

{date1 = formatter.parse(createdDate.get(i));

date2 = formatter.parse(modifiedDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter1 = new SimpleDateFormat("MM/dd/yyyy");

tempList.add(String.valueOf(formatter1.format(date1)));

tempList.add(String.valueOf(formatter1.format(date2)));

}

itemsReAdd.add(tempList);

}

if(found == 1)

{

try{

FileWriter writer = new FileWriter(menuFile);

for(int i = 0; i < menuHeader.size(); i++)

{

writer.append(menuHeader.get(i));

writer.append(",");

}

writer.append('\n');

for(int j = 0; j < itemsReAdd.size(); j++)

{

for(int i = 0; i < itemsReAdd.get(j).size(); i++)

{

writer.append(itemsReAdd.get(j).get(i));

writer.append(",");

}

writer.append('\n');

}

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

return idToModify;

}

else

{

return 0;

}

}

public List<String> Get\_ItemNames(String[] mID)

{

List<String> itemNames = new ArrayList<String>();

for(int j = 0; j < mID.length; j++)

for(int i = 0; i < id.size(); i++)

if(Integer.parseInt(mID[j]) == id.get(i))

{

itemNames.add(food.get(i));

break;

}

return itemNames;

}

private void SetParameters()

{

menuHeader = new ArrayList<String>();

id = new ArrayList<Integer>();

food = new ArrayList<String>();

category = new ArrayList<List<String>>();

price = new ArrayList<Double>();

minOrder = new ArrayList<Integer>();

createdDate = new ArrayList<String>();

modifiedDate = new ArrayList<String>();

}

private void ReadFile()

{

String line = null;

try{

BufferedReader br = new BufferedReader(new FileReader(menuFile));

int tempCount = 0;

while((line = br.readLine()) != null){

String[] menuItems = line.split(",");

if(tempCount == 0)

{

menuHeader = Arrays.asList(menuItems);

}

else

{

id.add(Integer.parseInt(menuItems[0]));

food.add(menuItems[1]);

price.add(Double.parseDouble(menuItems[2]));

minOrder.add(Integer.parseInt(menuItems[3]));

String[] tempCategory = menuItems[4].split("\_");

List<String> tempCategoryList = new ArrayList<String>();

for(int i = 0; i < tempCategory.length; i++)

tempCategoryList.add(tempCategory[i]);

category.add(tempCategoryList);

Date date1 = new Date();

Date date2 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");

try

{date1 = formatter.parse(menuItems[5]);

date2 = formatter.parse(menuItems[6]);}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter1 = new SimpleDateFormat("yyyyMMdd");

createdDate.add(formatter1.format(date1));

modifiedDate.add(formatter1.format(date2));

}

tempCount++;

}

br.close();

} catch(FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

private void WriteFile(List<String> itemsToAdd, int writeHeader)

{

try{

FileWriter writer = new FileWriter(menuFile, true);

// writer.append(String.valueOf(idCreated));

// writer.append(",");

for(int i = 0; i < itemsToAdd.size(); i++)

{

if (itemsToAdd.get(i).contains(","))

itemsToAdd.set(i, "\"" + itemsToAdd.get(i) + "\"");

writer.append(itemsToAdd.get(i));

writer.append(",");

}

writer.append('\n');

//System.out.println(String.join(",", itemsToAdd));

//writer.append(String.join(",", itemsToAdd));v

//writer.append(itemsToAdd.get(i));

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

}

public String test()

{

return "Helloooo world";

}

}

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

import java.io.FileWriter;

import java.text.\*;

public class MenuManager {

public String id;

public String name;

public String price\_per\_person;

public String minimum\_order;

public List<Map<String, String>> categories;

public String create\_date;

public String last\_modified\_date;

public MenuManager()

{

}

public MenuManager(List<String> s)

{

this.name = s.get(0);

this.price\_per\_person = s.get(1);

this.minimum\_order = s.get(2);

this.categories = new ArrayList<Map<String, String>>();

String[] categoryTemp = s.get(3).split("\_");

for(int i = 0; i < categoryTemp.length; i++)

{

Map<String, String> mapTemp = new HashMap<String, String>();

mapTemp.put("name", categoryTemp[i]);

this.categories.add(mapTemp);

}

}

public boolean isMatch\_Name(String nameStr)

{

return (nameStr.equals(this.name));

}

public boolean isMatch\_Price(String PriceStr)

{

return (PriceStr.equals(this.price\_per\_person));

}

public boolean isMatch\_MinQty(String MinQtyStr)

{

return (MinQtyStr.equals(this.minimum\_order));

}

public boolean isMatch\_Categories(String[] cat)

{

boolean val = false;

for(int i =0; i < cat.length; i++)

val = (cat[i].equals(this.categories.get(i).get("name")));

return val;

}

public int CreateMenu(MenuManager mmTemp)

{

List<String> itemsToAdd = new ArrayList<String>();

itemsToAdd.add(mmTemp.name);

itemsToAdd.add(mmTemp.price\_per\_person);

itemsToAdd.add(mmTemp.minimum\_order);

String tempStr = mmTemp.categories.get(0).get("name");

for(int i = 1; i < mmTemp.categories.size(); i++)

tempStr = tempStr + "\_" + mmTemp.categories.get(i).get("name");

itemsToAdd.add(tempStr);

Menu m = new Menu();

int id = m.Create\_Menu(itemsToAdd);

return id;

}

public List<MenuManager> DisplayMenu()

{

List<MenuManager> mmList = new ArrayList<MenuManager>();

Menu m = new Menu();

List<List<String>> menuDisplay = m.Display\_Menu();

if(menuDisplay == null)

{

mmList = null;

return mmList;

}

for(List<String> menuRowOne : menuDisplay)

{

MenuManager mmTemp = new MenuManager();

mmTemp.categories = new ArrayList<Map<String, String>>();

mmTemp.id = menuRowOne.get(0);

mmTemp.name = menuRowOne.get(1);

mmTemp.price\_per\_person = menuRowOne.get(2);

mmTemp.minimum\_order = menuRowOne.get(3);

String[] categoryTemp = menuRowOne.get(4).split("\_");

for(int i = 0; i < categoryTemp.length; i++)

{

Map<String, String> mapTemp = new HashMap<String, String>();

mapTemp.put("name", categoryTemp[i]);

mmTemp.categories.add(mapTemp);

}

mmList.add(mmTemp);

}

return mmList;

}

public List<MenuManager> DisplayMenu(int mId)

{

List<MenuManager> mmList = new ArrayList<MenuManager>();

Menu m = new Menu();

List<List<String>> menuDisplay = m.Display\_Menu(mId);

if(menuDisplay == null)

{

mmList = null;

return mmList;

}

for(List<String> menuRowOne : menuDisplay)

{

MenuManager mmTemp = new MenuManager();

mmTemp.categories = new ArrayList<Map<String, String>>();

mmTemp.id = menuRowOne.get(0);

mmTemp.name = menuRowOne.get(1);

mmTemp.price\_per\_person = menuRowOne.get(2);

mmTemp.minimum\_order = menuRowOne.get(3);

String[] categoryTemp = menuRowOne.get(4).split("\_");

for(int i = 0; i < categoryTemp.length; i++)

{

Map<String, String> mapTemp = new HashMap<String, String>();

mapTemp.put("name", categoryTemp[i]);

mmTemp.categories.add(mapTemp);

}

mmTemp.create\_date = menuRowOne.get(5);

mmTemp.last\_modified\_date = menuRowOne.get(6);

mmList.add(mmTemp);

}

return mmList;

}

public Integer ModifyMenuPrice(MenuManager mm1, int mid)

{

Menu m1 = new Menu();

//int mid = Integer.parseInt(mm1.id);

double price = Double.parseDouble(mm1.price\_per\_person);

int modId = m1.Modify\_Menu(mid, price);

return modId;

}

}

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

import java.io.FileWriter;

import java.text.\*;

public class Order{

private List<String> orderHeader;

public List<Integer> id;

public List<String> orderDate;

public List<String> deliveryDate;

public List<Double> amount;

public List<Double> surcharge;

public List<String> status;

public List<String> name;

public List<String> email;

public List<List<Integer>> count;

public List<List<Integer>> menuId;

public List<List<String>> menuItem;

public List<String> deliveryAddress;

public List<String> mobile;

public List<String> note;

private List<String> holidays;

//private String orderFile = "/home/jimeet/workspace/Project445/Order.csv";

//private String holidaysFile = "/home/jimeet/workspace/Project445/Holidays.csv";

//private String surchargeFile = "/home/jimeet/workspace/Project445/Surcharge.csv";

private String orderFile = "Order.csv";

private String holidaysFile = "Holidays.csv";

private String surchargeFile = "Surcharge.csv";

public Order(){

SetParameters();

ReadFile();

}

public int Create\_Order(Map<String, String> orderToAdd)

{

int idCreated = id.get(id.size() - 1) + 1;

List<String> itemsToAdd = new ArrayList<String>();

itemsToAdd.add(String.valueOf(idCreated));

Date date1 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");

String dateString = formatter.format(date1);

itemsToAdd.add(dateString);

SimpleDateFormat formatter1 = new SimpleDateFormat("yyyyMMdd");

Date date2 = new Date();

try

{date2 = formatter1.parse(orderToAdd.get("deliveryDate"));}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter2 = new SimpleDateFormat("MM/dd/yyyy");

itemsToAdd.add(formatter2.format(date2));

if(date1.after(date2) || date1.equals(date2))

return -1;

//if(date1.compareTo(date2) > -1)

//return -1;

//if(date2 < date1)

//return -1;

List<Double> tempAmount = new ArrayList<Double>();

tempAmount = getAmount(orderToAdd);

if(tempAmount == null)

{

return -1;

}

itemsToAdd.add(String.valueOf(tempAmount.get(0)));

itemsToAdd.add(String.valueOf(tempAmount.get(1)));

itemsToAdd.add("Open"); //Status

itemsToAdd.add(orderToAdd.get("deliveryAddress"));

itemsToAdd.add(orderToAdd.get("name"));

itemsToAdd.add(orderToAdd.get("email"));

itemsToAdd.add(orderToAdd.get("mobile"));

itemsToAdd.add(orderToAdd.get("note"));

String[] menuIdTemp = orderToAdd.get("menuId").split(",");

String mIdStr = menuIdTemp[0];

for (int i = 1; i < menuIdTemp.length; i++)

mIdStr = mIdStr + "\_" + menuIdTemp[i];

itemsToAdd.add(mIdStr);

Menu m = new Menu();

List<String> itemNames = new ArrayList<String>();

itemNames = m.Get\_ItemNames(menuIdTemp);

String itemNamesTemp = itemNames.get(0);

for(int i =1; i < itemNames.size(); i++)

itemNamesTemp = itemNamesTemp + "\_" + itemNames.get(i);

itemsToAdd.add(itemNamesTemp);

String[] countTemp = orderToAdd.get("count").split(",");

String countStr = countTemp[0];

for (int i = 1; i < countTemp.length; i++)

countStr = countStr + "\_" + countTemp[i];

itemsToAdd.add(countStr);

try{

FileWriter writer = new FileWriter(orderFile, true);

for(int i = 0; i < itemsToAdd.size(); i++)

{

if (itemsToAdd.get(i).contains(","))

itemsToAdd.set(i, "\"" + itemsToAdd.get(i) + "\"");

writer.append(itemsToAdd.get(i));

writer.append(",");

}

writer.append('\n');

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

Customer c = new Customer();

c.Create\_Customer(itemsToAdd);

return idCreated;

}

/\*public int Modify\_Order(int oidToModify, Map<String, String> orderToModify)

{

SetParameters();

ReadFile();

List<List<String>> itemsReAdd = new ArrayList<List<String>>();

int found = 0;

for(int i = 0; i < id.size(); i++)

{

List<String> tempList = new ArrayList<String>();

if(id.get(i) == oidToModify)

{

tempList.add(String.valueOf(id.get(i)));

SimpleDateFormat formatter1 = new SimpleDateFormat("yyyyMMdd");

Date date2 = new Date();

try

{date2 = formatter1.parse(orderDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter2 = new SimpleDateFormat("MM/dd/yyyy");

tempList.add(formatter2.format(date2));

Date date3 = new Date();

try

{date3 = formatter1.parse(orderToModify.get("deliveryDate"));}

catch(ParseException e)

{e.printStackTrace();}

tempList.add(formatter2.format(date3));

List<Double> tempAmount = new ArrayList<Double>();

tempAmount = getAmount(orderToModify);

tempList.add(String.valueOf(tempAmount.get(0)));

tempList.add(String.valueOf(tempAmount.get(1)));

tempList.add(status.get(i));

tempList.add(orderToModify.get("deliveryAddress"));

tempList.add(orderToModify.get("name"));

tempList.add(orderToModify.get("email"));

tempList.add(orderToModify.get("mobile"));

tempList.add(orderToModify.get("note"));

String[] menuIdTemp = orderToModify.get("menuId").split(",");

String mIdStr = menuIdTemp[0];

for (int j = 1; j < menuIdTemp.length; j++)

mIdStr = mIdStr + "\_" + menuIdTemp[j];

tempList.add(mIdStr);

Menu m = new Menu();

List<String> itemNames = new ArrayList<String>();

itemNames = m.Get\_ItemNames(menuIdTemp);

String itemNamesTemp = itemNames.get(0);

for(int j =1; j < itemNames.size(); j++)

itemNamesTemp = itemNamesTemp + "\_" + itemNames.get(j);

tempList.add(itemNamesTemp);

String[] countTemp = orderToModify.get("count").split(",");

String countStr = countTemp[0];

for (int j = 1; j < countTemp.length; j++)

countStr = countStr + "\_" + countTemp[j];

tempList.add(countStr);

found = 1;

}

else

{

tempList.add(String.valueOf(id.get(i)));

SimpleDateFormat formatter1 = new SimpleDateFormat("yyyyMMdd");

Date date2 = new Date();

try

{date2 = formatter1.parse(orderDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter2 = new SimpleDateFormat("MM/dd/yyyy");

tempList.add(formatter2.format(date2));

Date date3 = new Date();

try

{date3 = formatter1.parse(deliveryDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

tempList.add(formatter2.format(date3));

tempList.add(String.valueOf(amount.get(i)));

tempList.add(status.get(i));

tempList.add(deliveryAddress.get(i));

tempList.add(name.get(i));

tempList.add(email.get(i));

tempList.add(mobile.get(i));

tempList.add(note.get(i));

List<Integer> menuIdsTemp = new ArrayList<Integer>();

String mIdStr = String.valueOf(menuId.get(i).get(0));

for(int j = 1; j < menuId.get(i).size(); j++)

mIdStr = mIdStr + "\_" + menuId.get(i).get(j);

tempList.add(mIdStr);

List<Integer> countTemp = new ArrayList<Integer>();

String countStr = String.valueOf(count.get(i).get(0));

for(int j = 1; j < count.get(i).size(); j++)

countStr = countStr + "\_" + count.get(i).get(j);

tempList.add(countStr);

}

itemsReAdd.add(tempList);

}

if(found == 1)

{

try{

FileWriter writer = new FileWriter(orderFile);

for(int i = 0; i < orderHeader.size(); i++)

{

writer.append(orderHeader.get(i));

writer.append(",");

}

writer.append('\n');

for(int j = 0; j < itemsReAdd.size(); j++)

{

for(int i = 0; i < itemsReAdd.get(j).size(); i++)

{

writer.append(itemsReAdd.get(j).get(i));

writer.append(",");

}

writer.append('\n');

}

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

return oidToModify;

}

else

{

return 0;

}

}

\*/

public List<List<String>> Display\_Orders()

{

SetParameters();

ReadFile();

List<List<String>> orderDisplay = new ArrayList<List<String>>();

for (int i = 0; i < id.size(); i++)

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

tempList.add(orderDate.get(i));

tempList.add(deliveryDate.get(i));

tempList.add(String.valueOf(amount.get(i)));

tempList.add(String.valueOf(surcharge.get(i)));

tempList.add(status.get(i));

tempList.add(email.get(i));

orderDisplay.add(tempList);

}

return orderDisplay;

}

public List<List<String>> Display\_Orders(String byDeliveryDate)

{

SetParameters();

ReadFile();

List<List<String>> orderDisplay = new ArrayList<List<String>>();

int found = 0;

for (int i = 0; i < id.size(); i++)

{

if(deliveryDate.get(i).equals(byDeliveryDate))

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

tempList.add(orderDate.get(i));

tempList.add(deliveryDate.get(i));

tempList.add(String.valueOf(amount.get(i)));

tempList.add(String.valueOf(surcharge.get(i)));

tempList.add(status.get(i));

tempList.add(email.get(i));

orderDisplay.add(tempList);

found = 1;

}

}

if(found == 1)

return orderDisplay;

else

{

orderDisplay = null;

return orderDisplay;

}

}

public List<List<String>> Display\_Orders(int orderId)

{

SetParameters();

ReadFile();

List<List<String>> orderDisplay = new ArrayList<List<String>>();

int found = 0;

for (int i = 0; i < id.size(); i++)

{

if(id.get(i) == orderId)

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

tempList.add(String.valueOf(amount.get(i)));

tempList.add(String.valueOf(surcharge.get(i)));

tempList.add(status.get(i));

tempList.add(orderDate.get(i));

tempList.add(deliveryDate.get(i));

tempList.add(name.get(i));

tempList.add(email.get(i));

tempList.add(mobile.get(i));

tempList.add(deliveryAddress.get(i));

tempList.add(note.get(i));

String menuIdStr = String.valueOf(menuId.get(i).get(0));

for(int j = 1; j < menuId.get(i).size(); j++)

menuIdStr = menuIdStr + "\_" + menuId.get(i).get(j);

tempList.add(menuIdStr);

String menuItemStr = String.valueOf(menuItem.get(i).get(0));

for(int j = 1; j < menuItem.get(i).size(); j++)

menuItemStr = menuItemStr + "\_" + menuItem.get(i).get(j);

tempList.add(menuItemStr);

String countStr = String.valueOf(count.get(i).get(0));

for(int j = 1; j < count.get(i).size(); j++)

countStr = countStr + "\_" + count.get(i).get(j);

tempList.add(countStr);

orderDisplay.add(tempList);

found = 1;

break;

}

}

if(found == 1)

return orderDisplay;

else

{

System.out.println("No order by that ID");

return null;

}

}

public int Cancel\_Delivered\_Order(int oidToCancel, String orderModStr)

{

SetParameters();

ReadFile();

List<List<String>> itemsReAdd = new ArrayList<List<String>>();

int found = 0;

for(int i = 0; i < id.size(); i++)

{

List<String> tempList = new ArrayList<String>();

tempList.add(String.valueOf(id.get(i)));

SimpleDateFormat formatter1 = new SimpleDateFormat("yyyyMMdd");

Date date2 = new Date();

try

{date2 = formatter1.parse(orderDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter2 = new SimpleDateFormat("MM/dd/yyyy");

tempList.add(formatter2.format(date2));

Date date3 = new Date();

try

{date3 = formatter1.parse(deliveryDate.get(i));}

catch(ParseException e)

{e.printStackTrace();}

tempList.add(formatter2.format(date3));

tempList.add(String.valueOf(amount.get(i)));

tempList.add(String.valueOf(surcharge.get(i)));

//tempList.add(status.get(i));

tempList.add(deliveryAddress.get(i));

tempList.add(name.get(i));

tempList.add(email.get(i));

tempList.add(mobile.get(i));

tempList.add(note.get(i));

String mIdStr = String.valueOf(menuId.get(i).get(0));

for(int j = 1; j < menuId.get(i).size(); j++)

mIdStr = mIdStr + "\_" + menuId.get(i).get(j);

tempList.add(mIdStr);

String mItemStr = String.valueOf(menuItem.get(i).get(0));

for(int j = 1; j < menuItem.get(i).size(); j++)

mItemStr = mItemStr + "\_" + menuItem.get(i).get(j);

tempList.add(mItemStr);

String countStr = String.valueOf(count.get(i).get(0));

for(int j = 1; j < count.get(i).size(); j++)

countStr = countStr + "\_" + count.get(i).get(j);

tempList.add(countStr);

if(id.get(i) == oidToCancel)

{

tempList.add(5, orderModStr);

found = 1;

}

else

{

tempList.add(5, status.get(i));

}

itemsReAdd.add(tempList);

}

if(found == 1)

{

try{

FileWriter writer = new FileWriter(orderFile);

for(int i = 0; i < orderHeader.size(); i++)

{

writer.append(orderHeader.get(i));

writer.append(",");

}

writer.append('\n');

for(int j = 0; j < itemsReAdd.size(); j++)

{

for(int i = 0; i < itemsReAdd.get(j).size(); i++)

{

writer.append(itemsReAdd.get(j).get(i));

writer.append(",");

}

writer.append('\n');

}

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

return oidToCancel;

}

else

{

return -1;

}

}

private void SetParameters()

{

orderHeader = new ArrayList<String>();

id = new ArrayList<Integer>();

orderDate = new ArrayList<String>();

deliveryDate = new ArrayList<String>();

amount = new ArrayList<Double>();

surcharge = new ArrayList<Double>();

status = new ArrayList<String>();

deliveryAddress = new ArrayList<String>();

name = new ArrayList<String>();

email = new ArrayList<String>();

mobile = new ArrayList<String>();

note = new ArrayList<String>();

menuId = new ArrayList<List<Integer>>();

menuItem = new ArrayList<List<String>>();

count = new ArrayList<List<Integer>>();

holidays = new ArrayList<String>();

}

private void ReadFile()

{

String line = null;

try{

BufferedReader br = new BufferedReader(new FileReader(orderFile));

int tempCount = 0;

while((line = br.readLine()) != null){

String[] orderItems = line.split(",");

if(tempCount == 0)

{

orderHeader = Arrays.asList(orderItems);

}

else

{

id.add(Integer.parseInt(orderItems[0]));

Date date1 = new Date();

Date date2 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");

try

{date1 = formatter.parse(orderItems[1]);

date2 = formatter.parse(orderItems[2]);}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter1 = new SimpleDateFormat("yyyyMMdd");

orderDate.add(formatter1.format(date1));

deliveryDate.add(formatter1.format(date2));

amount.add(Double.parseDouble(orderItems[3]));

surcharge.add(Double.parseDouble(orderItems[4]));

status.add(orderItems[5]);

deliveryAddress.add(orderItems[6]);

name.add(orderItems[7]);

email.add(orderItems[8]);

mobile.add(orderItems[9]);

note.add(orderItems[10]);

String[] menuIds = orderItems[11].split("\_");

List<Integer> menuIdsTemp = new ArrayList<Integer>();

for(int i = 0; i < menuIds.length; i++)

menuIdsTemp.add(Integer.parseInt(menuIds[i]));

menuId.add(menuIdsTemp);

String[] menuItems = orderItems[12].split("\_");

List<String> menuItemsTemp = new ArrayList<String>();

for(int i = 0; i < menuItems.length; i++)

menuItemsTemp.add(menuItems[i]);

menuItem.add(menuItemsTemp);

String[] counts = orderItems[13].split("\_");

List<Integer> countsTemp = new ArrayList<Integer>();

for(int i = 0; i < counts.length; i++)

countsTemp.add(Integer.parseInt(counts[i]));

count.add(countsTemp);

}

tempCount++;

}

br.close();

} catch(FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

private List<Double> getAmount(Map<String, String> orderToAdd)

{

double amount = 0;

double surcharge\_ = 0;

List<Double> returnTemp = new ArrayList<Double>();

Menu m1 = new Menu();

String line = null;

try{

BufferedReader br = new BufferedReader(new FileReader(holidaysFile));

br.readLine();

while((line = br.readLine()) != null){

Date date1 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");

try

{date1 = formatter.parse(line);}

catch(ParseException e)

{e.printStackTrace();}

SimpleDateFormat formatter1 = new SimpleDateFormat("yyyyMMdd");

holidays.add(formatter1.format(date1));

}

}catch(FileNotFoundException e) {

e.printStackTrace();

}catch (IOException e) {

e.printStackTrace();

}

String[] menuId = orderToAdd.get("menuId").split(",");

String[] count = orderToAdd.get("count").split(",");

for(int i = 0; i < menuId.length; i++)

{

int menuIdTemp = Integer.parseInt(menuId[i]);

int countTemp = Integer.parseInt(count[i]);

int success = 0;

for(int j = 0; j < m1.id.size(); j++)

{

if(m1.id.get(j) == menuIdTemp)

{

if(countTemp >= m1.minOrder.get(j))

{

amount = amount + m1.price.get(j)\*countTemp;

success = 1;

}

}

}

if(success != 1)

{

returnTemp = null;

return returnTemp;

}

}

Date date2 = new Date();

SimpleDateFormat formatter1 = new SimpleDateFormat("yyyyMMdd");

try

{date2 = formatter1.parse(orderToAdd.get("deliveryDate"));}

catch(ParseException e)

{e.printStackTrace();}

Date date1 = new Date();

SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");

date1 = date2;

Calendar c = Calendar.getInstance();

c.set(date1.getYear() + 1900, date1.getMonth(), date1.getDate());

if(c.get(Calendar.DAY\_OF\_WEEK) == Calendar.SUNDAY || c.get(Calendar.DAY\_OF\_WEEK) == Calendar.SATURDAY)

{

surcharge\_ = Get\_Surcharge();

//amount = amount + surcharge\_;

}

else

{

for(int i = 0; i < holidays.size(); i++)

if(orderToAdd.get("deliveryDate").equals(holidays.get(i)))

{

surcharge\_ = Get\_Surcharge();

//amount = amount + surcharge\_;

}

}

returnTemp.add(amount);

returnTemp.add(surcharge\_);

return returnTemp;

}

public double Get\_Surcharge()

{

double surcharge\_ = 0;

try{

BufferedReader br = new BufferedReader(new FileReader(surchargeFile));

br.readLine();

String temp = br.readLine();

surcharge\_ = Double.parseDouble(temp);

br.close();

} catch(FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return surcharge\_;

}

public void Modify\_Surcharge(double surcharge\_)

{

try{

FileWriter writer = new FileWriter(surchargeFile);

writer.append("Surcharge");

writer.append('\n');

writer.append(String.valueOf(surcharge\_));

writer.append('\n');

writer.flush();

writer.close();

} catch(IOException e){

e.printStackTrace();

}

}

}

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

import java.io.FileWriter;

import java.text.\*;

public class OrderManager {

public String id;

public String orderDate;

public String delivery\_date;

public String amount;

public String surcharge;

public String status;

public String delivery\_address;

public String name;

public String email;

public String mobile;

public String note;

public Customer personal\_info;

public List<LineItems> order\_detail;

public OrderManager()

{

}

public OrderManager(List<String> menuRowOne)

{

this.delivery\_date = menuRowOne.get(0);

this.delivery\_address = menuRowOne.get(1);

this.personal\_info = new Customer();

this.personal\_info.setNameTemp(menuRowOne.get(2));

this.personal\_info.setEmailTemp(menuRowOne.get(3));

this.personal\_info.setMobileTemp(menuRowOne.get(4));

this.note = menuRowOne.get(5);

String[] menuIdTemp = menuRowOne.get(6).split(",");

String[] menuCountTemp = menuRowOne.get(7).split(",");

this.order\_detail = new ArrayList<LineItems>();

for(int i = 0; i < menuIdTemp.length; i++)

{

LineItems lTemp = new LineItems();

lTemp.id = menuIdTemp[i];

lTemp.count = menuCountTemp[i];

this.order\_detail.add(lTemp);

}

}

public boolean isMatch\_DeliveryDate(String dateStr)

{

return (dateStr.equals(this.delivery\_date));

}

public boolean isMatch\_DeliveryAddress(String addrStr)

{

return (addrStr.equals(this.delivery\_address));

}

public boolean isMatch\_name(String nameStr)

{

return (nameStr.equals(this.personal\_info.getNameTemp()));

}

public boolean isMatch\_email(String emailStr)

{

return (emailStr.equals(this.personal\_info.getEmailTemp()));

}

public boolean isMatch\_mobile(String mobileStr)

{

return (mobileStr.equals(this.personal\_info.getMobileTemp()));

}

public boolean isMatch\_note(String noteStr)

{

return (noteStr.equals(this.note));

}

public boolean isMatch\_MenuIds(String menuIds)

{

boolean val = false;

String[] menuIdTemp = menuIds.split(",");

for(int i =0; i < menuIdTemp.length; i++)

val = (menuIdTemp[i].equals(this.order\_detail.get(i).id));

return val;

}

public boolean isMatch\_MenuCount(String menuCounts)

{

boolean val = false;

String[] menuCountTemp = menuCounts.split(",");

for(int i =0; i < menuCountTemp.length; i++)

val = (menuCountTemp[i].equals(this.order\_detail.get(i).count));

return val;

}

public int CreateOrder(OrderManager omTemp)

{

Order oMain = new Order();

Map<String, String> itemsToAdd = new HashMap<String, String>();

itemsToAdd.put("deliveryDate", omTemp.delivery\_date);

if(omTemp.delivery\_address.contains(","))

{

String[] tempStrings = omTemp.delivery\_address.split(",");

String tempStr = tempStrings[0];

for(int i = 1; i < tempStrings.length; i++)

tempStr = tempStr + " " + tempStrings[i];

itemsToAdd.put("deliveryAddress", tempStr);

}

else

itemsToAdd.put("deliveryAddress", omTemp.delivery\_address);

itemsToAdd.put("name", omTemp.personal\_info.getNameTemp());

itemsToAdd.put("email", omTemp.personal\_info.getEmailTemp());

itemsToAdd.put("mobile", omTemp.personal\_info.getMobileTemp());

itemsToAdd.put("note", omTemp.note);

String menuIdStr = omTemp.order\_detail.get(0).id;

String menuCountStr = omTemp.order\_detail.get(0).count;

for(int i = 1; i < omTemp.order\_detail.size(); i++)

{

menuIdStr = menuIdStr + "," + omTemp.order\_detail.get(i).id;

menuCountStr = menuCountStr + "," + omTemp.order\_detail.get(i).count;

}

itemsToAdd.put("menuId", menuIdStr);

itemsToAdd.put("count", menuCountStr);

/\*itemsToAdd.put("menuId", "12,15");

itemsToAdd.put("count", "50,50");

\*/

int oId = oMain.Create\_Order(itemsToAdd);

return oId;

}

public List<OrderManager> DisplayOrder()

{

List<OrderManager> om = new ArrayList<OrderManager>();

List<List<String>> orderDetails = new ArrayList<List<String>>();

Order oMain = new Order();

orderDetails = oMain.Display\_Orders();

for(List<String> menuRowOne : orderDetails)

{

OrderManager tempO = new OrderManager();

tempO.id = menuRowOne.get(0);

tempO.orderDate = menuRowOne.get(1);

tempO.delivery\_date = menuRowOne.get(2);

tempO.amount = menuRowOne.get(3);

tempO.surcharge = menuRowOne.get(4);

tempO.status = menuRowOne.get(5);

tempO.email = menuRowOne.get(6);

om.add(tempO);

}

return om;

}

public OrderManager DisplayOrder(int oid)

{

OrderManager om = new OrderManager();

List<List<String>> orderDetails = new ArrayList<List<String>>();

Order oMain = new Order();

personal\_info = new Customer();

orderDetails = oMain.Display\_Orders(oid);

if(orderDetails == null)

{

om = null;

return om;

}

for(List<String> menuRowOne : orderDetails)

{

om.id = menuRowOne.get(0);

om.amount = menuRowOne.get(1);

om.surcharge = menuRowOne.get(2);

om.status = menuRowOne.get(3);

om.orderDate = menuRowOne.get(4);

om.delivery\_date = menuRowOne.get(5);

/\*om.name = menuRowOne.get(6);

om.email = menuRowOne.get(7);

om.mobile = menuRowOne.get(8);

\*/

Customer c1 = new Customer("from OrderManager");

c1.setNameTemp(menuRowOne.get(6));

c1.setEmailTemp(menuRowOne.get(7));

c1.setMobileTemp(menuRowOne.get(8));

om.personal\_info = c1;

/\*om.ordered\_by.setNameTemp(menuRowOne.get(6));

om.ordered\_by.setEmailTemp(menuRowOne.get(6));

om.ordered\_by.setMobileTemp(menuRowOne.get(6));

\*/

om.delivery\_address = menuRowOne.get(9);

om.note = menuRowOne.get(10);

om.order\_detail = new ArrayList<LineItems>();

String[] menuIds = menuRowOne.get(11).split("\_");

String[] menuItems = menuRowOne.get(12).split("\_");

String[] menuCount = menuRowOne.get(13).split("\_");

for(int i = 0; i < menuIds.length; i++)

{

List<String> orderDetailsTemp = new ArrayList<String>();

orderDetailsTemp.add(menuIds[i]);

orderDetailsTemp.add(menuItems[i]);

orderDetailsTemp.add(menuCount[i]);

LineItems l1 = new LineItems(orderDetailsTemp);

om.order\_detail.add(l1);

}

/\*om.menuId = menuRowOne.get(11);

om.menuItem = menuRowOne.get(12);

om.count = menuRowOne.get(13);\*/

//System.out.print(om.id);

}

return om;

}

public List<OrderManager> DisplayOrder(String date)

{

List<OrderManager> om = new ArrayList<OrderManager>();

List<List<String>> orderDetails = new ArrayList<List<String>>();

Order oMain = new Order();

orderDetails = oMain.Display\_Orders(date);

if(orderDetails == null)

{

om = null;

return om;

}

for(List<String> menuRowOne : orderDetails)

{

OrderManager tempO = new OrderManager();

tempO.id = menuRowOne.get(0);

tempO.orderDate = menuRowOne.get(1);

tempO.delivery\_date = menuRowOne.get(2);

tempO.amount = menuRowOne.get(3);

tempO.surcharge = menuRowOne.get(4);

tempO.status = menuRowOne.get(5);

tempO.email = menuRowOne.get(6);

om.add(tempO);

}

return om;

}

public Integer CancelOrder(int oId)

{

Order oMain = new Order();

int idCancelled = -1;

idCancelled = oMain.Cancel\_Delivered\_Order(oId, "Cancel");

return idCancelled;

}

public double GetSurcharge()

{

Order o = new Order();

double surcharge\_ = o.Get\_Surcharge();

return surcharge\_;

}

public void SetSurcharge(OrderManager om1)

{

Order o = new Order();

double surcharge\_ = Double.parseDouble(om1.surcharge);

o.Modify\_Surcharge(surcharge\_);

}

}

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.util.\*;

import java.io.FileWriter;

import java.text.\*;

public class Reports {

public String \_startDate;

public String \_endDate;

public String \_orderCount;

public String \_amountTotal;

public String \_surchargeTotal;

public List<Map<String, String>> reportNames;

public List<OrderManager> omList;

public Reports()

{

reportNames = new ArrayList<Map<String, String>>();

Map<String, String>tempMap = new HashMap<String, String>();

tempMap.put("id", "801");

tempMap.put("name", "Orders to deliver today");

reportNames.add(tempMap);

Map<String, String>tempMap1 = new HashMap<String, String>();

tempMap1.put("id", "802");

tempMap1.put("name", "Orders to deliver tomorrow");

reportNames.add(tempMap1);

Map<String, String>tempMap2 = new HashMap<String, String>();

tempMap2.put("id", "803");

tempMap2.put("name", "Revenue Report");

reportNames.add(tempMap2);

Map<String, String>tempMap3 = new HashMap<String, String>();

tempMap3.put("id", "804");

tempMap3.put("name", "Orders Delivery Report");

reportNames.add(tempMap3);

}

public Reports(int rId)

{

}

public List<Reports> Get\_Report(int rId)

{

List<Reports> r1 = new ArrayList<Reports>();

List<String> tempList = new ArrayList<String>();

if(rId == 803)

{

tempList = Get\_Report\_803();

Reports rTemp = new Reports(rId);

rTemp.\_startDate = tempList.get(0);

rTemp.\_endDate = tempList.get(1);

rTemp.\_orderCount = tempList.get(2);

rTemp.\_amountTotal = tempList.get(3);

rTemp.\_surchargeTotal = tempList.get(4);

r1.add(rTemp);

}

if(rId == 801 || rId == 802)

{

Calendar calendar = Calendar.getInstance();

if(rId == 802)

calendar.add(calendar.DAY\_OF\_YEAR, 1);

Date date1 = calendar.getTime();

SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");

String dateString = formatter.format(date1);

Reports rTemp = new Reports(rId);

OrderManager om = new OrderManager();

rTemp.omList = om.DisplayOrder(dateString);

r1.add(rTemp);

}

return r1;

}

public List<String> Get\_Report\_803()

{

List<String> reportDetails = new ArrayList<String>();

List<List<String>> reportDetailsTemp = new ArrayList<List<String>>();

Order o = new Order();

reportDetailsTemp = o.Display\_Orders();

int orderCount = 0;

double amount = 0;

double surcharge = 0;

for(int i = 0; i < reportDetailsTemp.size(); i++)

{

if(!reportDetailsTemp.get(i).get(5).equals("Cancel"))

{

orderCount++;

amount = amount + Double.parseDouble(reportDetailsTemp.get(i).get(3));

surcharge = surcharge + Double.parseDouble(reportDetailsTemp.get(i).get(4));

}

}

String startDate = "20160301";

reportDetails.add(startDate);

Integer endDate = 0;

for(int i = 0; i < reportDetailsTemp.size(); i++)

if(Integer.parseInt(reportDetailsTemp.get(i).get(1)) > endDate)

endDate = Integer.parseInt(reportDetailsTemp.get(i).get(1));

reportDetails.add(String.valueOf(endDate));

reportDetails.add(String.valueOf(orderCount));

reportDetails.add(String.valueOf(amount));

reportDetails.add(String.valueOf(surcharge));

return reportDetails;

}

public List<Reports> Get\_Report(int rId, String deliveryDate)

{

List<Reports> r1 = new ArrayList<Reports>();

List<String> tempList = new ArrayList<String>();

Reports rTemp = new Reports(rId);

OrderManager om = new OrderManager();

rTemp.omList = om.DisplayOrder(deliveryDate);

r1.add(rTemp);

return r1;

}

}

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import com.google.gson.JsonObject;

import javax.annotation.PostConstruct;

import javax.ws.rs.\*;

import javax.ws.rs.core.\*;

import java.util.\*;

@Path("/delectable")

public class REST\_Controller {

OrderManager om = new OrderManager();

MenuManager mm = new MenuManager();

CustomerManager cm = new CustomerManager();

@Path("/menu")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response DisplayMenu(){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

if(mm.DisplayMenu() != null)

{

String s = gson.toJson(mm.DisplayMenu());

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("Menu Not Created yet").build();

}

@Path("/menu/{mid}")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response DisplayMenu(@PathParam("mid") int mid){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

if(mm.DisplayMenu(mid) != null)

{

String s = gson.toJson(mm.DisplayMenu(mid));

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("Menu ID " + mid + " Not Found").build();

}

@Path("/admin/menu")

@PUT

@Produces(MediaType.APPLICATION\_JSON)

public Response CreateMenu(String json, @Context UriInfo uriInfo){

Gson gson = new Gson();

MenuManager mm1 = gson.fromJson(json, MenuManager.class);

int id = mm.CreateMenu(mm1);

JsonObject jo = new JsonObject();

if(id == -1)

return Response.status(Response.Status.NOT\_FOUND).entity("Something wrong with Count or MenuID").build();

jo.addProperty("id", id);

String s = gson.toJson(jo);

UriBuilder builder = uriInfo.getAbsolutePathBuilder();

builder.path(Integer.toString(id));

return Response.created(builder.build()).entity(s).build();

}

@Path("/admin/menu/{mid}")

@POST

@Produces(MediaType.APPLICATION\_JSON)

public Response ModifyMenu(String json, @PathParam("mid") int mid){

Gson gson = new Gson();

MenuManager mm1 = gson.fromJson(json, MenuManager.class);

int id = mm.ModifyMenuPrice(mm1, mid);

JsonObject jo = new JsonObject();

if(id == 0)

return Response.status(Response.Status.BAD\_REQUEST).entity("Something wrong with MenuID/ MenuID not found").build();

jo.addProperty("id", id);

String s = gson.toJson(jo);

return Response.status(Response.Status.NO\_CONTENT).entity(s).build();

}

@Path("/admin/surcharge")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response DisplaySurcharge(){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

double surcharge\_ = om.GetSurcharge();

JsonObject jo = new JsonObject();

jo.addProperty("surcharge", surcharge\_);

String s = gson.toJson(jo);

return Response.status(Response.Status.OK).entity(s).build();

}

@Path("/admin/surcharge")

@POST

@Produces(MediaType.APPLICATION\_JSON)

public Response PostSurcharge(String json){

Gson gson = new Gson();

OrderManager om1 = gson.fromJson(json, OrderManager.class);

om.SetSurcharge(om1);

return Response.status(Response.Status.OK).entity("Surcharge Modified").build();

}

@Path("/order")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response DisplayOrder(@DefaultValue("") @QueryParam("date") String date){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

if(date.equals(""))

{

if(om.DisplayOrder() != null)

{

String s = gson.toJson(om.DisplayOrder());

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("No Orders Created Yet").build();

}

else

{

if(om.DisplayOrder(date) != null)

{

String s = gson.toJson(om.DisplayOrder(date));

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("No Orders Created Yet").build();

}

}

@Path("/order/{oid}")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response DisplayOrderByOid(@PathParam("oid") int oid){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

if(om.DisplayOrder(oid) != null)

{

String s = gson.toJson(om.DisplayOrder(oid));

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("Order ID " + oid + " Not Found").build();

}

@Path("/order")

@PUT

@Produces(MediaType.APPLICATION\_JSON)

public Response CreateOrder(String json, @Context UriInfo uriInfo){

Gson gson = new Gson();

OrderManager om1 = gson.fromJson(json, OrderManager.class);

int id = om.CreateOrder(om1);

JsonObject jo = new JsonObject();

if(id == -1)

return Response.status(Response.Status.NOT\_FOUND).entity("Something wrong with Count or MenuID").build();

jo.addProperty("id", id);

String s = gson.toJson(jo);

UriBuilder builder = uriInfo.getAbsolutePathBuilder();

builder.path(Integer.toString(id));

return Response.created(builder.build()).entity(s).build();

}

@Path("/order/cancel/{oid}")

@POST

@Produces(MediaType.APPLICATION\_JSON)

public Response CancelOrder(@PathParam("oid") int oid){

Gson gson = new GsonBuilder().setPrettyPrinting().create(); if(om.CancelOrder(oid) != -1)

{

String s = gson.toJson(om.CancelOrder(oid));

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("Order ID " + oid + " Not Found").build();

}

@Path("/customer")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response Displaycustomers(@DefaultValue("") @QueryParam("key") String nameStr){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

if(nameStr.equals(""))

{

if(cm.DisplayCustomers() != null)

{

String s = gson.toJson(cm.DisplayCustomers());

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("No Customers have ordered yet").build();

}

else

{

if(cm.DisplayCustomers(nameStr) != null)

{

String s = gson.toJson(cm.DisplayCustomers(nameStr));

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("No Customers with name having: " + nameStr).build();

}

}

@Path("/customer/{cid}")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response DisplaycustomersByCid(@PathParam("cid") int cid){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

if(cm.DisplayCustomers(cid) != null)

{

String s = gson.toJson(cm.DisplayCustomers(cid));

return Response.status(Response.Status.OK).entity(s).build();

}

return Response.status(Response.Status.NOT\_FOUND).entity("No Customers with cId: " + cid).build();

}

@Path("/report")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response DisplayReportsName(){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

Reports r = new Reports();

String s = gson.toJson(r);

return Response.status(Response.Status.OK).entity(s).build();

}

@Path("/report/{rid}")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Response DisplayReportsNameByRId(@PathParam("rid") int rid, @DefaultValue("") @QueryParam("start\_date") String start\_date, @DefaultValue("") @QueryParam("end\_date") String end\_date){

Gson gson = new GsonBuilder().setPrettyPrinting().create();

if(start\_date.equals(""))

{

Reports r = new Reports(rid);

List<Reports> rList = r.Get\_Report(rid);

String s = gson.toJson(rList);

return Response.status(Response.Status.OK).entity(s).build();

}

else

{

Reports r = new Reports(rid);

List<Reports> rList = r.Get\_Report(804, start\_date);

String s = gson.toJson(rList);

return Response.status(Response.Status.OK).entity(s).build();

}

}

}

import junit.framework.TestCase;

import java.io.\*;

import java.text.SimpleDateFormat;

import org.junit.\*;

import java.util.\*;

public class TestCustomerClass extends TestCase{

private List<String> s = Arrays.asList("20160501", "10 West 31st ST, Chicago IL 60616", "Virgil B", "virgil@example.com", "312-456-7890", "Room SB-214", "12,15", "25,25");

@Test

public void testDisplayCustomer() {

OrderManager omTest = new OrderManager(s);

int id1 = omTest.CreateOrder(omTest);

Customer c= new Customer();

List<List<String>> customerList = new ArrayList<List<String>>();

customerList = c.Get\_Customer();

int match = 0;

for(int i = 0; i < customerList.size(); i++)

{

if(customerList.get(i).get(2).equals(s.get(3)))

{

for(int j = 1; j < customerList.get(i).size(); j++)

{

if(customerList.get(i).get(j).equals(s.get(j+1)))

match = 1;

else

{

match = 0;

break;

}

}

}

}

assertTrue(match == 1);

List<String> s1 = Arrays.asList("20160501", "10 West 31st ST, Chicago IL 60616", "Virgil", "virgil@example.com", "312-456-7890", "Room SB-214", "12,15", "25,25");

omTest = new OrderManager(s1);

id1 = omTest.CreateOrder(omTest);

Customer c1= new Customer();

List<List<String>> customerList1 = new ArrayList<List<String>>();

customerList1 = c1.Get\_Customer();

match = 0;

for(int i = 0; i < customerList1.size(); i++)

{

if(customerList1.get(i).get(2).equals(s1.get(3)))

{

for(int j = 1; j < customerList1.get(i).size(); j++)

{

if(customerList1.get(i).get(j).equals(s1.get(j+1)))

match = 1;

else

{

match = 0;

break;

}

}

}

}

assertFalse(match == 1);

}

int globalCustId = 0;

@Test

public void testDisplayCustomerByName() {

OrderManager omTest = new OrderManager(s);

int id1 = omTest.CreateOrder(omTest);

Customer c= new Customer();

List<List<String>> customerList = new ArrayList<List<String>>();

customerList = c.Get\_Customer(s.get(2));

int match = 0;

for(int i = 0; i < customerList.size(); i++)

{

if(customerList.get(i).get(2).equals(s.get(3)))

{

for(int j = 1; j < customerList.get(i).size(); j++)

{

if(customerList.get(i).get(j).equals(s.get(j+1)))

match = 1;

else

{

match = 0;

break;

}

}

} }

assertTrue(match == 1);

}

@Test

public void testDisplayCustomerByCId() {

OrderManager omTest = new OrderManager(s);

int id1 = omTest.CreateOrder(omTest);

Customer c= new Customer();

List<String> customerList = new ArrayList<String>();

List<List<String>> customerList1 = new ArrayList<List<String>>();

customerList1 = c.Get\_Customer(s.get(2));

int match = 0;

for(int i = 0; i < customerList1.size(); i++)

{

if(customerList1.get(i).get(2).equals(s.get(3)))

{

globalCustId = Integer.parseInt(customerList1.get(i).get(0));

break;

}

}

customerList = c.Get\_Customer(globalCustId);

match = 0;

for(int i = 1; i < customerList.size(); i++)

{

if(customerList.get(i).equals(s.get(i+1)))

match = 1;

else

{

match = 0;

break;

}

}

assertTrue(match == 1);

}

}

import junit.framework.TestCase;

import java.io.\*;

import org.junit.\*;

import java.util.\*;

public class TestMenuClass extends TestCase{

/\*OutputStream outContent = new ByteArrayOutputStream();

OutputStream errContent = new ByteArrayOutputStream();

\*/ private List<String> s = Arrays.asList("Soup2", "99", "3", "Veg\_InOrganic");

/\*@Before

public void setUpStreams() {

System.setOut(new PrintStream(outContent));

System.setErr(new PrintStream(errContent));

}

@After

public void cleanUpStreams() {

System.setOut(null);

System.setErr(null);

} \*/

@Test

public void testIsMatchName() {

MenuManager mmTest = new MenuManager(s);

assertTrue(mmTest.isMatch\_Name("Soup2"));

}

@Test

public void testIsMatchPrice() {

MenuManager mmTest = new MenuManager(s);

assertTrue(mmTest.isMatch\_Price("99"));

}

@Test

public void testIsMatchMinQty() {

MenuManager mmTest = new MenuManager(s);

assertTrue(mmTest.isMatch\_MinQty("3"));

}

@Test

public void testIsMatchCategory() {

MenuManager mmTest = new MenuManager(s);

String[] cat = {"Veg", "InOrganic"};

assertTrue(mmTest.isMatch\_Categories(cat));

}

@Test

public void testCreateMenu() {

MenuManager mmTest = new MenuManager(s);

int id1 = mmTest.CreateMenu(mmTest);

MenuManager mmTest1 = new MenuManager(s);

int id2 = mmTest.CreateMenu(mmTest1);

assertTrue(id2 == id1 + 1);

}

@Test

public void testDisplayMenu() {

MenuManager mmTest = new MenuManager(s);

int id1 = mmTest.CreateMenu(mmTest);

List<MenuManager> mmList = new ArrayList<MenuManager>();

mmList = mmTest.DisplayMenu();

assertEquals(id1, Integer.parseInt(mmList.get(mmList.size()-1).id));

assertTrue(s.get(0).equals(mmList.get(mmList.size()-1).name));

assertTrue(Double.parseDouble(s.get(1)) == Double.parseDouble((mmList.get(mmList.size()-1).price\_per\_person)));

}

@Test

public void testDisplayMenuById() {

MenuManager mmTest = new MenuManager(s);

int id1 = mmTest.CreateMenu(mmTest);

List<MenuManager> mmList = new ArrayList<MenuManager>();

mmList = mmTest.DisplayMenu(id1);

assertEquals(id1, Integer.parseInt(mmList.get(mmList.size()-1).id));

assertTrue(s.get(0).equals(mmList.get(mmList.size()-1).name));

assertTrue(Double.parseDouble(s.get(1)) == Double.parseDouble((mmList.get(mmList.size()-1).price\_per\_person)));

}

@Test

public void testModifyMenuPrice() {

MenuManager mmTest = new MenuManager(s);

int id1 = mmTest.CreateMenu(mmTest);

String modPrice = "200";

mmTest.price\_per\_person = modPrice;

int modId = mmTest.ModifyMenuPrice(mmTest, id1);

List<MenuManager> mmList = new ArrayList<MenuManager>();

mmList = mmTest.DisplayMenu(id1);

assertEquals(Double.parseDouble(modPrice), Double.parseDouble((mmList.get(mmList.size()-1).price\_per\_person)));

}

}

import junit.framework.TestCase;

import java.io.\*;

import java.text.SimpleDateFormat;

import org.junit.\*;

import java.util.\*;

public class TestOrderClass extends TestCase{

private List<String> s = Arrays.asList("20160501", "10 West 31st ST, Chicago IL 60616", "Virgil B", "virgil@example.com", "312-456-7890", "Room SB-214", "12,15", "25,25");

@Test

public void testIsMatchDeliveryDate() {

OrderManager omTest = new OrderManager(s);

assertTrue(omTest.isMatch\_DeliveryDate(s.get(0)));

}

@Test

public void testIsMatchDeliveryAddress() {

OrderManager omTest = new OrderManager(s);

assertTrue(omTest.isMatch\_DeliveryAddress(s.get(1)));

}

@Test

public void testIsMatchDeliveryName() {

OrderManager omTest = new OrderManager(s);

assertTrue(omTest.isMatch\_name(s.get(2)));

}

@Test

public void testIsMatchDeliveryEmail() {

OrderManager omTest = new OrderManager(s);

assertTrue(omTest.isMatch\_email(s.get(3)));

}

@Test

public void testIsMatchDeliveryMobile() {

OrderManager omTest = new OrderManager(s);

assertTrue(omTest.isMatch\_mobile(s.get(4)));

}

@Test

public void testIsMatchDeliveryNote() {

OrderManager omTest = new OrderManager(s);

assertTrue(omTest.isMatch\_note(s.get(5)));

}

@Test

public void testIsMatchDeliveryMenuIds() {

OrderManager omTest = new OrderManager(s);

assertTrue(omTest.isMatch\_MenuIds(s.get(6)));

}

@Test

public void testIsMatchDeliveryMenuCounts() {

OrderManager omTest = new OrderManager(s);

assertTrue(omTest.isMatch\_MenuCount(s.get(7)));

}

@Test

public void testCreateOrder() {

OrderManager omTest = new OrderManager(s);

int id1 = omTest.CreateOrder(omTest);

OrderManager omTest1 = new OrderManager(s);

int id2 = omTest.CreateOrder(omTest1);

assertTrue(id2 == id1 + 1);

}

@Test

public void testDisplayOrder() {

OrderManager omTest = new OrderManager(s);

int id1 = omTest.CreateOrder(omTest);

List<OrderManager> omList = new ArrayList<OrderManager>();

omList = omTest.DisplayOrder();

assertEquals(id1, Integer.parseInt(omList.get(omList.size()-1).id));

assertTrue(s.get(0).equals(omList.get(omList.size()-1).delivery\_date));

Calendar calendar = Calendar.getInstance();

Date date1 = calendar.getTime();

SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");

String dateString = formatter.format(date1);

assertTrue(dateString.equals(omList.get(omList.size()-1).orderDate));

//int x = omList.size()-1;

String status\_ = "Open";

assertTrue(status\_.equals(omList.get(omList.size()-1).status));

}

@Test

public void testDisplayOrderByDate() {

OrderManager omTest = new OrderManager(s);

int id1 = omTest.CreateOrder(omTest);

List<OrderManager> omList = new ArrayList<OrderManager>();

omList = omTest.DisplayOrder(s.get(0));

assertEquals(id1, Integer.parseInt(omList.get(omList.size()-1).id));

assertTrue(s.get(0).equals(omList.get(omList.size()-1).delivery\_date));

Calendar calendar = Calendar.getInstance();

Date date1 = calendar.getTime();

SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");

String dateString = formatter.format(date1);

assertTrue(dateString.equals(omList.get(omList.size()-1).orderDate));

String status\_ = "Open";

assertTrue(status\_.equals(omList.get(omList.size()-1).status));

}

@Test

public void testDisplayOrderByOid() {

OrderManager omTest = new OrderManager(s);

int id1 = omTest.CreateOrder(omTest);

OrderManager omList = new OrderManager();

omList = omTest.DisplayOrder(id1);

assertEquals(id1, Integer.parseInt(omList.id));

assertTrue(s.get(0).equals(omList.delivery\_date));

Calendar calendar = Calendar.getInstance();

Date date1 = calendar.getTime();

SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMdd");

String dateString = formatter.format(date1);

assertTrue(dateString.equals(omList.orderDate));

String status\_ = "Open";

assertTrue(status\_.equals(omList.status));

String[] menuIdTemp = s.get(6).split(",");

for(int i = 0; i < menuIdTemp.length; i++)

{

assertTrue(menuIdTemp[i].equals(omList.order\_detail.get(i).id));

}

String[] menuCountTemp = s.get(7).split(",");

for(int i = 0; i < menuCountTemp.length; i++)

{

assertTrue(menuCountTemp[i].equals(omList.order\_detail.get(i).count));

}

}

@Test

public void testCancelOrder() {

OrderManager omTest = new OrderManager(s);

int id1 = omTest.CreateOrder(omTest);

OrderManager omList = new OrderManager();

omList = omTest.DisplayOrder(id1);

String status\_ = "Open";

assertTrue(status\_.equals(omList.status));

omTest.CancelOrder(id1);

omList = omTest.DisplayOrder(id1);

status\_ = "Cancel";

assertTrue(status\_.equals(omList.status));

}

@Test

public void testSurcharge() {

OrderManager omTest = new OrderManager();

double surcharge\_ = omTest.GetSurcharge();

double newSurcharge = surcharge\_ + 100;

omTest.surcharge = String.valueOf(newSurcharge);

omTest.SetSurcharge(omTest);

surcharge\_ = omTest.GetSurcharge();

assertEquals(newSurcharge, surcharge\_);

}

}