Name of the Course : Learning Java 9 - Object Oriented Programming

Level : Difficult

Tool Stack : Java9 and Junit4

Problem Statement : Provide a code solution to generate customer bill of snack’s shop using java 9 Stream API, file handling and collection framework.

Description : ***The Great Indian Cafe*** sales varieties of Indian veg snack items both salty and sweet. Every day it displays list of that day’s available items, rate/unit and discount details from an item master file -SnackItem.txt. The File description is given bottom. Only the item with status “A” means Available for the day are uploaded to a collection. The discount rate and purchased quantity for few items are available in file. 12.5% sale tax is up to Rs1000/- , up to 2500/- 10% sales tax after that 7.5% sales tax to be charged on bill amount. You need to develop application for file handling and bill generation.

You need to create

1. class SnackItem with private member data

String itemName

String rate

String status ->A-available,N-not available

String discountQty -> value 10 means 10% discount

String discountRate

Create getter/setter methods and constructors.

1. class OrderedItem with private member function

String itemName,

String rate,

String orderQty,

String discountAmount,

String payAmount

Create getter/setter methods and constructor

1. class ItemData with the following members
2. **private** **static** Map<String,SnackItem> *itemMap*.
3. Static block where everyday available items will be extracted from file and store into itemMap with item name as key and SnackItem object as value.
4. public static Map<String,SnackItem> getAllItems()
5. public static SnackItem getItem(String itemName)
6. public static boolean isAvailable(String itemName)

1. class Main with methods
2. private static String floatToString(double value): It is used convert any floating value to String up to 2 decimal places.
3. public static String salesTaxCalculation(double billAmount): It will calculate sales tax amount as per specification.
4. public static void main(String [] arg): It will accept name of customer, number of items the customer purchases, each item name and quantity purchased in a comma(,) separate format like Samosa,20 etc. It will also accept system date as bill date in the format of dd-mm-yyyy. Finally it generates bill for that customer like shown below(Test Data 1).

**Sample Data for File SnackItem.txt**

**Samosa-15.00-A-0-0**

**Namkin-5.00-A-15-100**

**Dokla-20.00-N-0-0**

**Dosha-65.00-A-10-20**

**Tikia-10.00-N-0-0**

**Idli-50.00-A-0-0**

**Vadapow-25.00-A-15-50**

**CholeBature-75.00-N-0-0**

**Kachauri-25.00-N-0-0**

**Jelebi-25.00-A-0-0**

**Imriti-75.00-A-10-25**

**Perha-10.00-N-0-0**

**Balusahi-45.00-A-0-0**

**Barfi-60.00-A-10-50**

**Seera-75.00-A-15-50**

**MalaiRabri-80.00-N-0-0**

**Rajbhog-85.00-N-0-0**

**Laddu-35.00-A-0-0**

**PuriBhaji-45.00-A-0-0**

**Halwa-55.00-A-10-25**

Code:

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

**public** **class** SnackItem {

**private** String itemName;

**private** String rate;

**private** String status;

**private** String discountRate;

**private** String discountQty;

}

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Set;

**import** java.util.TreeMap;

**import** java.util.stream.Stream;

**public** **class** ItemData {

**private** **static** Map<String,SnackItem> *itemMap*=**new** TreeMap<String, SnackItem>();

**static**

{

**try**

{

FileReader fileReader=**new** FileReader("SnackItem.txt");

BufferedReader bufferedReader=**new** BufferedReader(fileReader);

List<SnackItem> itemList=**new** ArrayList<SnackItem>();

**while**(**true**)

{

String str=bufferedReader.readLine();

**if**(str==**null**)

**break**;

String []arr=str.split("-");

SnackItem snackItem=**new** SnackItem(arr[0],arr[1],arr[2],arr[3],arr[4]);

itemList.add(snackItem);

}// end of loop

Stream<SnackItem> snackStream=itemList.stream().filter(item->item.getStatus().equalsIgnoreCase("A"));

snackStream.forEach((snackItem)->{

*itemMap*.put(snackItem.getItemName().toUpperCase(),snackItem);

}); // end of forEach

}**catch**(Exception ee) {System.***out***.println(ee);}

}// end of static block

**public** **static** Map<String,SnackItem> getAllItems()

{

**return** *itemMap*;

}

**public** **static** SnackItem getItem(String itemName)

{

**return** *itemMap*.get(itemName.toUpperCase());

}

**public** **static** **boolean** isAvailable(String itemName)

{

**return** *itemMap*.containsKey(itemName.toUpperCase());

}

}

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

**public** **class** OrderItem {

**private** String itemName;

**private** String rate;

**private** String orderQty;

**private** String actualAmount;

**private** String discount;

**private** String payAmount;

**public** OrderItem(String itemName, String rate, String orderQty) {

**super**();

**this**.itemName = itemName;

**this**.rate = rate;

**this**.orderQty = orderQty;

}

@Override

**public** String toString() {

String output=String.*format*("%-15s %-10s %-10s %-10s %-10s %-10s ",itemName,rate,orderQty,actualAmount,discount,payAmount);

**return** output;

}

}

**import** java.text.DecimalFormat;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** Main {

**private** **static** String floatToString(**double** value)

{

DecimalFormat decimalFormat=**new** DecimalFormat("0.00");

**return** decimalFormat.format(value);

}

**public** **static** String salesTaxCalculation(**double** billAmount)

{

**double** taxAmount=0.00;

**if**(billAmount<1001)

taxAmount=billAmount\*0.125;

**else** **if**(billAmount<2501)

{

taxAmount=1000.00\*0.125+(billAmount-1000)\*0.1;

}

**else**

{

taxAmount=1000.00\*0.125+2500.00\*0.1+(billAmount-2500)\*0.075;

}

**return** *floatToString*(taxAmount);

}

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

Scanner scanner=**new** Scanner(System.***in***);

List<OrderItem> orderList=**new** ArrayList<OrderItem>();

System.***out***.println("Enter Customer Name:");

String customerName=scanner.nextLine();

Date date=**new** Date();

SimpleDateFormat dateFormat=**new** SimpleDateFormat("dd-MM-yyyy");

String today=dateFormat.format(date);

System.***out***.println("Enter number of items the customer purchases:");

**int** no=Integer.*parseInt*(scanner.nextLine());

**for**(**int** i=0;i<no;i++)

{

**int** j=i+1;

System.***out***.println("Enter name and quantity (comma separate format) of purchased item no "+j);

String detail=scanner.nextLine();

String arr[]=detail.split(",");

SnackItem snackItem=ItemData.*getItem*(arr[0]);

String rate=snackItem.getRate();

OrderItem orderItem=**new** OrderItem(arr[0], rate, arr[1]);

**double** actualAmount=Double.*parseDouble*(rate)\*Double.*parseDouble*(arr[1]);

orderItem.setActualAmount(*floatToString*(actualAmount));

**double** purchaseQty=Double.*parseDouble*(arr[1]);

**double** discountQty=Double.*parseDouble*(snackItem.getDiscountQty());

**if**(purchaseQty>=discountQty)

{

**double** discountRate=Double.*parseDouble*(snackItem.getDiscountRate());

**double** discountAmount=0.00;

**if**(discountRate!=0)

{

discountAmount=actualAmount/discountRate;

}

**double** payAmount=actualAmount-discountAmount;

orderItem.setDiscount(*floatToString*(discountAmount));

orderItem.setPayAmount(*floatToString*(payAmount));

}

**else**

{

orderItem.setDiscount("0.00");

orderItem.setPayAmount(*floatToString*(actualAmount));

}

orderList.add(orderItem);

}// end of loop

**double** billAmount=0.00;

**for**(OrderItem orderItem:orderList){

billAmount=billAmount+Double.*parseDouble*(orderItem.getPayAmount());

}

String salesTax=*salesTaxCalculation*(billAmount);

String finalAmount=*floatToString*(billAmount+Double.*parseDouble*(salesTax));

System.***out***.println("\n\nCustomer Name:"+customerName+"\t\t\t\tDate:"+today);

System.***out***.println("\n"+String.*format*("%-15s %-10s %-10s %-10s %-10s %-10s ","ITEM","RATE","QUANTITY","PRICE","DISCOUNT","AMOUNT"));

orderList.forEach(System.***out***::println);

System.***out***.println("\n\t\t\t\t\t Bill Amount:-"+*floatToString*(billAmount));

System.***out***.println("\t\t\t\t\t Add: Sales Tax:-"+salesTax);

System.***out***.println("\t\t\t\t\t Final Amount:-"+finalAmount);

}

}

Junit Testing

**import** java.io.File;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** TestUtils {

**public** **static** File *businessTestFile*;

**public** **static** File *boundaryTestFile*;

**public** **static** File *exceptionTestFile*;

**static** {

*businessTestFile* = **new** File("./output\_revised.txt");

*businessTestFile*.delete();

*boundaryTestFile* = **new** File("./output\_boundary\_revised.txt");

*boundaryTestFile*.delete();

*exceptionTestFile* = **new** File("./output\_exception\_revised.txt");

*exceptionTestFile*.delete();

}

**public** **static** **void** yakshaAssert(String testName, Object result, File file) **throws** IOException {

System.***out***.println("\n" + testName + "=" + result);

FileWriter writer = **new** FileWriter(file,**true**);

writer.append("\n" + testName + "=" + result);

writer.flush();

writer.close();

}

**public** **static** String currentTest() {

**return** Thread.*currentThread*().getStackTrace()[2].getMethodName();

}

}

**import** **static** org.junit.Assert.\*;

**import** org.junit.Test;

**import** **static** java9.diff.app1.TestUtils.\*;

**public** **class** MainTest {

@Test

**public** **void** testSalesTaxCalculation() **throws** Exception{

String value=Main.*salesTaxCalculation*(9315.00);

*yakshaAssert*(*currentTest*(),(value.equals("886.12")?"true":"false"),*businessTestFile*);

}

}

Test Data1

Enter Customer Name:

Zubin Kumar

Enter number of items the customer purchases:

5

Enter name and quantity (comma separate format) of purchased item no 1

Samosa,25

Enter name and quantity (comma separate format) of purchased item no 2

Dosha,30

Enter name and quantity (comma separate format) of purchased item no 3

Imriti,20

Enter name and quantity (comma separate format) of purchased item no 4

Seera,60

Enter name and quantity (comma separate format) of purchased item no 5

Halwa,30

Customer Name:Zubin Kumar Date:05-09-2020

ITEM RATE QUANTITY PRICE DISCOUNT AMOUNT

Samosa 15.00 25 375.00 0.00 375.00

Dosha 65.00 30 1950.00 195.00 1755.00

Imriti 75.00 20 1500.00 0.00 1500.00

Seera 75.00 60 4500.00 300.00 4200.00

Halwa 55.00 30 1650.00 165.00 1485.00

Bill Amount:9315.00

Add: Sales Tax:886.12

Final Amount:10201.12

Learning outcome: Participant could able to use file handling with collection and stream api in java, forEach() function etc.