Name of the Course : Complete Java SE8 Developer Bootcamp

Level : Difficult

Tool Stack : Java8 and Junit4

Problem Statement : Provide a code solution to generate bill for purchasing various items. The bill includes tax and discount calculation using Map, static block,user defined exception, List, and different display formatting etc features.

Description : ***The People Co-operative Store*** a govt owned store where any person can purchase grocery items at a lower rate then prevailing market rate once in a month. Any consumer, already registered is provided a unique id. You are required to develop an application that will accept: 1. Consumer name and number in a comma(,), separated format e.g. 20203, John Mathew 2. All purchased item names and corresponding quantities in a comma(,) separated format e.g: Soap,3,Sugar,2.5,Rice,10.5.If any not listed in stock item is there then ItemNotFoundException will be raised. Finally display bill in below format(see test data). Date of purchase should be displayed in dd-mmm-yyyy format e.g. 15-Aug-1947. 12.5% sales tax will be added to total purchase amount. If the purchase amount is equal or higher than 5000/- then 10% discount will be deducted.

You need to

1. Create class Item with the following member data

**String itemName,**

**Double rate,**

**Double qtyOrdered,**

**Double amountPayable,**

Create getter and setter method, override toString() in the following format:

in the following format

**String.format("%-20s %-20s %-20s %-20s",....)**

**All floating values should display 2 decimal figures like 0.00,123.45 etc.**

Create 3 constructors Item(), Item(itemName,rate), Item(itemName,rate,qtyOrdered).

1. Create class Stock with following members:
2. private static Map<String,Item> itemMap. Enter at least 6 items with name and price. In map Item name is the key and Item object is the value.
3. public static Map<String,Item> getAllItems();
4. public static Item getItem(String itemName).
5. Create class BillService with the following methods:
6. public String itemPurchasedPayable(Item item): It calculate each purchased item price i.e rate \* quantity.
7. public String saleTaxCalculation(double) : It calculates sales tax payable on total purchase which is 12.5 % of total purchase.
8. public String discountCalculation(double): It calculates 10% discount available on total purchase provided the purchase is 5000/- or greater.
9. Create class ItemNotFoundException inherited from RuntimeException.
10. Create class **Main** withthe method public static void main(String [] args). It will accept the all inputs from the user and then finally displays the bill details including Consumer name, id and date of purchase in proper format(see below test data).

Code:

**import** java.text.DecimalFormat;

**public** **class** Item {

**private** String itemName;

**private** Double rate;

**private** Double qtyOrdered;

**private** Double amountPayable;

**public** Item() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Item(String itemName, Double rate) {

**super**();

**this**.itemName = itemName;

**this**.rate = rate;

**this**.qtyOrdered =0.0;

**this**.amountPayable = 0.0;

}

**public** Item(String itemName, Double rate, Double qtyOrdered) {

**super**();

**this**.itemName = itemName;

**this**.rate = rate;

**this**.qtyOrdered = qtyOrdered;

**this**.amountPayable = 0.0;

}

**public** Item(String itemName, Double rate, Double qtyOrdered, Double amountPayable) {

**super**();

**this**.itemName = itemName;

**this**.rate = rate;

**this**.qtyOrdered = qtyOrdered;

**this**.amountPayable = amountPayable;

}

**public** String getItemName() {

**return** itemName;

}

**public** **void** setItemName(String itemName) {

**this**.itemName = itemName;

}

**public** Double getRate() {

**return** rate;

}

**public** **void** setRate(Double rate) {

**this**.rate = rate;

}

**public** Double getQtyOrdered() {

**return** qtyOrdered;

}

**public** **void** setQtyOrdered(Double qtyOrdered) {

**this**.qtyOrdered = qtyOrdered;

}

**public** Double getAmountPayable() {

**return** amountPayable;

}

**public** **void** setAmountPayable(Double amountPayable) {

**this**.amountPayable = amountPayable;

}

@Override

**public** String toString() {

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

String price=decimalFormat.format(rate);

String qty=decimalFormat.format(qtyOrdered);

String amt=decimalFormat.format(amountPayable);

String output=String.*format*("%-20s %-20s %-20s %-20s",itemName,price,qty,amt);

**return** output;

}

}

**import** java.util.HashMap;

**import** java.util.Map;

**public** **class** Stock {

**private** **static** Map<String,Item> *itemMap*=**new** HashMap<String, Item>();

**static**

{

*itemMap*.put("SOAP",**new** Item("SOAP",37.5));

*itemMap*.put("SUGAR",**new** Item("SUGAR",45.0));

*itemMap*.put("RICE",**new** Item("RICE",67.5));

*itemMap*.put("TEA",**new** Item("TEA",125.50));

*itemMap*.put("COFFEE",**new** Item("COFFEE",150.0));

*itemMap*.put("WHEAT",**new** Item("WHEAT",55.0));

*itemMap*.put("SALT",**new** Item("SALT",17.5));

*itemMap*.put("TOOTHPASTE",**new** Item("TOOTHPASTE",75.0));

*itemMap*.put("DAL",**new** Item("DAL",85.0));

*itemMap*.put("SPICE",**new** Item("SPICE",95.0));

}

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

{

**return** *itemMap*;

}

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

{

itemName=itemName.toUpperCase();

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

}

}

**public** **class** ItemNotFoundException **extends** RuntimeException {

}

**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 {

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

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

System.***out***.println("Enter Consumer Number and name in a comma separate format:");

String consumer=scanner.nextLine();

String [] consumerDetails=consumer.split(",");

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

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

String today=dateFormat.format(date);

System.***out***.println("Enter purchased item names and quantities in comma separate format:");

String purchaseItems=scanner.nextLine();

purchaseItems=purchaseItems.toUpperCase();

List<Item> purchaseList=**new** ArrayList<Item>();

String[] purchaseDetails=purchaseItems.split(",");

**double** totalPurchase=0.0;

**for**(**int** i=0;i<purchaseDetails.length-1;i=i+2)

{

**try**

{

Item item=Stock.*getItem*(purchaseDetails[i]);

**if**(item==**null**)

**throw** **new** ItemNotFoundException();

**double** qtyOrdered=Double.*parseDouble*(purchaseDetails[i+1]);

item.setQtyOrdered(qtyOrdered);

**double** amountPayable=Double.*parseDouble*(**new** BillService().itemPurchasedPayable(item));

item.setAmountPayable(amountPayable);

totalPurchase=totalPurchase+amountPayable;

purchaseList.add(item);

}**catch**(ItemNotFoundException exp){}

}// end of loop

String tax=**new** BillService().saleTaxCalculation(totalPurchase);

String discount=**new** BillService().discountCalculation(totalPurchase);

**double** finalPayment=totalPurchase+Double.*parseDouble*(tax)-Double.*parseDouble*(discount);

System.***out***.println(today);

System.***out***.println("Consumer Number:"+consumerDetails[0]);

System.***out***.println("Consumer Name:"+consumerDetails[1]);

String output=String.*format*("%-20s %-20s %-20s %-20s","Item Name","Price","Quantity","Amount");

System.***out***.println(output);

**for**(Item item:purchaseList)

System.***out***.println(item);

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

System.***out***.println("Total. "+df.format(totalPurchase));

System.***out***.println("Add:Sales Tax@12.5%. "+tax);

System.***out***.println("Less:Discount. "+discount);

System.***out***.println("Final Bill Payment. "+df.format(finalPayment)+"\n");

}

}

Junit Testing

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

**import** org.junit.Test;

**public** **class** BillServiceTest {

@Test

**public** **void** testItemPurchasedPayable() {

Item item=**new** Item("Oil",37.87,5.00);

*assertEquals*("189.35",**new** BillService().itemPurchasedPayable(item));

}

@Test

**public** **void** testSaleTaxCalculation() {

*assertEquals*("23.67",**new** BillService().saleTaxCalculation(189.35));

}

@Test

**public** **void** testDiscountCalculation() {

*assertEquals*("500.00",**new** BillService().discountCalculation(5000.00));

*assertEquals*("0.00",**new** BillService().discountCalculation(4999.00));

}

}

Test Data1

Enter Consumer Number and name in a comma separate format:

7521,Stella Williams

Enter purchased item names and quantities in comma separate format:

Rice,25,Wheat,20,Sugar,10,Coffee,10,Tea,15,Toothpaste,5,dal,35

21-Aug-2020

Consumer Number:7521

Consumer Name:Stella Williams

Item Name Price Quantity Amount

RICE 67.50 25.00 1687.50

WHEAT 55.00 20.00 1100.00

SUGAR 45.00 10.00 450.00

COFFEE 150.00 10.00 1500.00

TEA 125.50 15.00 1882.50

TOOTHPASTE 75.00 5.00 375.00

DAL 85.00 35.00 2975.00

Total. 9970.00

Add:Sales Tax@12.5%. 1246.25

Less:Discount. 997.00

Final Bill Payment. 10219.25

Test Data2

Enter Consumer Number and name in a comma separate format:

2145,Mike Johnson

Enter purchased item names and quantities in comma separate format:

Sugar,5,Rice,15,Wheat,10,Butter,2,Salt,5,Dal,10

21-Aug-2020

Consumer Number:2145

Consumer Name:Mike Johnson

Item Name Price Quantity Amount

SUGAR 45.00 5.00 225.00

RICE 67.50 15.00 1012.50

WHEAT 55.00 10.00 550.00

SALT 17.50 5.00 87.50

DAL 85.00 10.00 850.00

Total. 2725.00

Add:Sales Tax@12.5%. 340.62

Less:Discount. 0.00

Final Bill Payment. 3065.62

Learning outcome: Participant could able to learn how to use static block, java.util.Map, User defined Exception, Date and String formatting.