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

Level : Difficult

Tool Stack : Java9 and Junit4

Problem Statement : Provide a code solution to calculate electricity bill payment for a group of consumers based on the electric consumption over a period, discount/fine calculation etc, based on Java 9 code solution of List, Stream API, LocalDate etc features.

Description : ***National Electricity Corporation(NEC)*** charges monthly electricity charges based on consumption of it’s consumers. The rate of charge is as follows:

1. Upto 200 unit every consumer must pay flat an amount of Rs 300/- (even no consumption).
2. From 201 to 500 unit rate is Rs 1.25/unit.
3. From 501 to 1000 unit rate is Rs 1.00/unit.
4. From 1001 unit and above rate is 0.75/unit.

Consumption unit must be in whole number. If the consumer pays before the due date of payment, will earn 10% of discount in a round off figure. If the consumer within after 5 days of due date, normal amount. After 5th day fine will be charged Rs.100/day.

The computer operator of NEC normally accepts consumer’s details like consumer number, name, unit consumption and due date in a comma(,) separate String to generate bill (eg: 4743,Steve Jones,976,03-08-2020,05-08-2020). The operator normally generates bill for number of consumers at a time. So before starting operation, the operator asks for number of consumers’ bill to prepare. All bills are stored in a collection. Consumer id cannot be duplicated. Finally display the contains of collection in the ascending order of consumer id, all decimal figures must show 2 decimal places.

You need create

1. class Consumer with private member data

Integer id;

String name;

String unitConsumed;

String dueDate; (dd-mm-yyyy)

String actualPayDate;(dd-mm-yyyy)

String discountFine; (negative figure discount, positive fine, zero nothing)

String billAmount; (amount payable before any fine or discount)

String finalPayment

Create getter/setter methods and constructors.

override toString() in String.format("%-5s %-20s %-10s %-15s %-15s %-10s"). Override equals and hashCode methods.

1. class BillService with member function
2. public static int discountFineCalculation (Consumer): if return is negative then discount, if it is positive then fine, if 0 then no fine nor discount.

public static String billCalcultion(Consumer consumer): It will calculate payament amount based of above formula.

1. class Main with method public static void main(String [] arg): It will asks for number of consumers, If it is 0 or negative number message “invalid input” will be displayed and stop the application, if it is a valid positive number then create an array of Consumer and ask “Enter details of consumer number 1”, “Enter details of consumer number 2”,... so on. After this each of consumer’s bill will be generated then finally display each consumer’s id, name, unit consumed and amount payable.

Code:

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

**public** **class** Consumer **implements** Comparable<Consumer>{

**private** Integer id;

**private** String name;

**private** String unitConsumed;

**private** String dueDate;

**private** String actualPayDate;

**private** String discountFine;

**private** String billAmount;

**private** String finalPayment;

**public** Consumer(Integer id, String name, String unitConsumed,String dueDate,String actualPayDate) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.unitConsumed = unitConsumed;

**this**.dueDate=dueDate;

**this**.actualPayDate=actualPayDate;

}

@Override

**public** String toString() {

String output=String.*format*("%-5s %-20s %-15s %-15s %-15s %-10s %-15s %-10s",id,name,unitConsumed,dueDate,actualPayDate,billAmount,discountFine,finalPayment);

**return** output;

}

**public** **int** compareTo(Consumer other)

{

**return** **this**.id.compareTo(other.id);

}

}

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.time.Period;

import java.util.Calendar;

import java.util.Date;

**public** **class** BillService {

**public** **static** String discountFineCalculation (Consumer consumer) **throws** Exception

{

String discountFine="";

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

Date dueDate=simpleDateFormat.parse(consumer.getDueDate());

Calendar calendar=Calendar.*getInstance*();

calendar.setTime(dueDate);

**int** year=calendar.get(Calendar.***YEAR***);

**int** month=calendar.get(Calendar.***MONTH***)+1;

**int** day=calendar.get(Calendar.***DATE***);

LocalDate dateDue=LocalDate.*of*(year, month, day);

Date actualDate=simpleDateFormat.parse(consumer.getActualPayDate());

calendar.setTime(actualDate);

year=calendar.get(Calendar.***YEAR***);

month=calendar.get(Calendar.***MONTH***)+1;

day=calendar.get(Calendar.***DATE***);

LocalDate dateActual=LocalDate.*of*(year, month, day);

Period difference=Period.*between*(dateDue,dateActual);

**int** dayDifference=difference.getDays();

**if**(dayDifference<0)

{

**double** value=Double.*parseDouble*(consumer.getBillAmount());

value=(value/10.00)\*(-1.00);

discountFine=**new** DecimalFormat("0.00").format(value);

}

**else** **if**(dayDifference<=3)

discountFine="0.00";

**else** **if**(dayDifference>3)

{

discountFine=**new** DecimalFormat("0.00").format(dayDifference\*100.00);

}

**return** discountFine;

}

**public** **static** String billAmountCalcultion(Consumer consumer)

{

**double** billValue=0.0;

**int** consumption=Integer.*parseInt*(consumer.getUnitConsumed());

**if**(consumption<=200)

billValue=300.00;

**else** **if**(consumption<=500)

{

**int** remain=consumption-200;

billValue=300.00+(remain\*1.25);

}

**else** **if**(consumption<=1000)

{

**int** remain=consumption-500;

billValue=300.00+((500-200)\*1.25)+(remain\*1.00);

}

**else**

{

**int** remain=consumption-1000;

billValue=300.00+((500-200)\*1.25)+((1000-500)\*1.00)+(remain\*0.75);

}

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

String billPayment=decimalFormat.format(billValue);

**return** billPayment;

}

**public** **static** String totalPaymentCalculation(Consumer consumer)

{

**double** value=Double.*parseDouble*(consumer.getBillAmount());

value=value+Double.*parseDouble*(consumer.getDiscountFine());

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

String finalPayment=decimalFormat.format(value);

**return** finalPayment;

}

}

**import** java.util.Scanner;

**import** java.util.Set;

**import** java.util.TreeSet;

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

**public** **class** Main

{

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

{

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

System.***out***.println("Enter Number of consumers bill to prepare:");

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

**if**(no<=0)

{

System.***out***.println("invalid input");

System.*exit*(0);

}

Set<Consumer> consumerSet=**new** TreeSet<Consumer>();

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

{

**int** j=i+1;

System.***out***.println("Enter details of consumer number "+j+":");

String input=scanner.nextLine();

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

Consumer consumer=**new** Consumer(Integer.*parseInt*(arr[0]),arr[1],arr[2],arr[3],arr[4]);

String value=BillService.*billAmountCalcultion*(consumer);

consumer.setBillAmount(value);

String discountFine=BillService.*discountFineCalculation*(consumer);

consumer.setDiscountFine(discountFine);

consumer.setFinalPayment(BillService.*totalPaymentCalculation*(consumer));

consumerSet.add(consumer);

}

System.***out***.println(String.*format*("%-5s %-20s %-15s %-15s %-15s %-10s %-10s %-10s","ID","NAME","UNIT-CONSUME","DUE-DATE","PAY-DATE","BILL-AMOUNT","DISCOUNT/FINE","FINAL AMOUNT"));

Stream<Consumer> consumers=consumerSet.stream();

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

}

}

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** org.junit.Test;

**import** **static** java9.diff.app2.TestUtils.\*;

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

@Test

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

Consumer consumer1=**new** Consumer(3214,"Ryan Stewert","1010","03-08-2020","08-08-2020");

Consumer consumer2=**new** Consumer(4321,"Lucy Sarley","990","05-08-2020","03-08-2020");

Consumer consumer3=**new** Consumer(4767,"John Miller","950","03-08-2020","05-08-2020");

String value1=BillService.*discountFineCalculation*(consumer1);

String value=BillService.*billAmountCalcultion*(consumer2);

consumer2.setBillAmount(value);

String value2=BillService.*discountFineCalculation*(consumer2);

String value3=BillService.*discountFineCalculation*(consumer3);

*yakshaAssert*(*currentTest*(),(value1.equals("500.00")?"true":"false"),*businessTestFile*);

*yakshaAssert*(*currentTest*(),(value2.equals("-116.50")?"true":"false"),*businessTestFile*);

*yakshaAssert*(*currentTest*(),(value3.equals("0.00")?"true":"false"),*businessTestFile*);

}

@Test

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

Consumer consumer1=**new** Consumer(3214,"Ryan Stewert","1010","03-08-2020","08-08-2020");

Consumer consumer2=**new** Consumer(4321,"Lucy Sarley","990","05-08-2020","03-08-2020");

Consumer consumer3=**new** Consumer(4767,"John Miller","950","03-08-2020","05-08-2020");

String value1=BillService.*billAmountCalcultion*(consumer1);

String value2=BillService.*billAmountCalcultion*(consumer2);

String value3=BillService.*billAmountCalcultion*(consumer3);

*yakshaAssert*(*currentTest*(),(value1.equals("1182.00")?"true":"false"),*businessTestFile*);

*yakshaAssert*(*currentTest*(),(value2.equals("1165.00")?"true":"false"),*businessTestFile*);

*yakshaAssert*(*currentTest*(),(value3.equals("1125.00")?"true":"false"),*businessTestFile*);

}

}

Test Data1

Enter Number of consumers bill to prepare:

-3

invalid input

Test Data2

**Enter Number of consumers bill to prepare:**

**5**

**Enter details of consumer number 1:**

**4767,John Miller,950,03-08-2020,05-08-2020**

**Enter details of consumer number 2:**

**5621,Mary Ann,850,05-08-2020,12-08-2020**

**Enter details of consumer number 3:**

**4321,Lucy Sarley,990,05-08-2020,03-08-2020**

**Enter details of consumer number 4:**

**3214,Ryan Stewert,1010,03-08-2020,08-08-2020**

**Enter details of consumer number 5:**

**5337,Erica Jones,1250,06-08-2020,06-08-2020**

**ID NAME UNIT-CONSUME DUE-DATE PAY-DATE BILL-AMOUNT DISCOUNT/FINE FINAL AMOUNT**

**3214 Ryan Stewert 1010 03-08-2020 08-08-2020 1182.50 500.00 1682.50**

**4321 Lucy Sarley 990 05-08-2020 03-08-2020 1165.00 -116.50 1049.00**

**4767 John Miller 950 03-08-2020 05-08-2020 1125.00 0.00 1125.00**

**5337 Erica Jones 1250 06-08-2020 06-08-2020 1362.50 0.00 1362.50**

**5621 Mary Ann 850 05-08-2020 12-08-2020 1025.00 700.00 1725.00**

Learning outcome: Participant could able to use Stream API ,Set, forEach() functions.