Name of the Course : Ultimate Java Masterclass

Level : Hard

Tool Stack : Java8 and Junit5

Problem Statement : Provide a code solution to accept details in menu driven program to make payments through various channels i.e through OnlineBanking, CreditCard and Wallet using Collection(Map, Set) and number formatting.

Description : With all the software systems that are being built, one of the widely used utilities in the financial sector is payments through various channels. The widely used channels are OnlineBanking, CreditCard and Wallet. The banks generate revenue by charging a small margin as part of the usage.

Create five classes one OnlineBanking class with amount as Double and bankName as String types and a parameterized constructor. One CreditCard class with amount as Double type and a parameterized constructor. One Wallet class with amount as Double and and discountPercent as Float types and a parameterized constructor. One PaymentUtil class with three methods

1. public Double makePayment(Map<String,Float> bankTax,String bankName,Double amount). This method is for payment through OnlineBanking and it calculates the total amount (amount + service tax amount). Here the tax percent gets varied depending on the bank. Fetch the corresponding service tax from a map collection by bank name. Map<String bankName, Float serviceTax> . This map is prefilled. Only banks present in the map would be part of input.
2. public Double makePayment1(Double amount). This method is for payment through CreditCard and it calculates the total amount (include service tax and value added tax(VAT). First, calculate the service tax amount and then calculate the VAT amount. VAT % is applied on total amount+service tax
3. public Double makePayment2(Double amount, Float discountPercent). This method is for payment through Wallet and it calculates the total amount, which is a discount from the amount with parameterized discount percentage.

Note it that bank name is case insensitive. The service tax for credit card is 5.2%, VAT for the Credit Card is 2.3% and discount percentage for Wallet is 20.2%.

One MainClass with main method for Input and output operations. It will call various methods from PaymentUtil class to perform payment operation. Place following data inside the static block in the Main to provide payment solution through various channels

onlineBankingMap.put("ICICI", 4.2f);

onlineBankingMap.put("IBRD", 3f);

onlineBankingMap.put("IFC", 4.9f);

onlineBankingMap.put("HSBC", 3.9f);

Code:

**package** yaksha;

**public** **class** OnlineBanking {

Double amount;

String bankName;

**public** OnlineBanking(Double amount, String bankName) {

**super**();

**this**.amount = amount;

**this**.bankName = bankName;

}

**public** Double getAmount() {

**return** amount;

}

**public** **void** setAmount(Double amount) {

**this**.amount = amount;

}

**public** String getBankName() {

**return** bankName;

}

**public** **void** setBankName(String bankName) {

**this**.bankName = bankName;

}

}

**package** yaksha;

**public** **class** CreditCard {

Double amount;

**public** CreditCard(Double amount) {

**super**();

**this**.amount = amount;

}

**public** Double getAmount() {

**return** amount;

}

**public** **void** setAmount(Double amount) {

**this**.amount = amount;

}

}

**package** yaksha;

**public** **class** Wallet {

Double amount;

Float discountPercent;

**public** Wallet(Double amount, Float discountPercent) {

**super**();

**this**.amount = amount;

**this**.discountPercent = discountPercent;

}

**public** Double getAmount() {

**return** amount;

}

**public** **void** setAmount(Double amount) {

**this**.amount = amount;

}

**public** Float getDiscountPercent() {

**return** discountPercent;

}

**public** **void** setDiscountPercent(Float discountPercent) {

**this**.discountPercent = discountPercent;

}

}

**package** yaksha;

**import** java.util.Map;

**import** java.util.Set;

**public** **class** PaymentUtil {

**public** Double makePayment(Map<String, Float> bankTax, String bankName, Double amount) {

Set<String> keys = bankTax.keySet();

**double** amnt = 0;

**for** (String s : keys) {

**if** (s.equalsIgnoreCase(bankName)) {

amnt = bankTax.get(s);

}

}

**return** amount + (amount \* (amnt / 100));

}

**public** Double makePayment1(Double amount) {

Float serviceTax = 5.2f;

Float vat = 2.3f;

**double** d1 = amount + (amount \* serviceTax / 100);

**return** d1 + (d1 \* (vat / 100));

}

**public** Double makePayment2(Double amount, Float discountPercent) {

**return** amount - (amount \* (discountPercent / 100));

}

}

**package** yaksha;

**import** java.io.\*;

**import** java.text.DecimalFormat;

**import** java.util.HashMap;

**import** java.util.Map;

**public** **class** MainClass {

**static** Float *discount* = 20.2f;

**static** Map<String, Float> *onlineBanking*;

**static** {

*onlineBanking* = **new** HashMap<>();

*onlineBanking*.put("ICICI", 4.2f);

*onlineBanking*.put("IBRD", 3f);

*onlineBanking*.put("IFC", 4.9f);

*onlineBanking*.put("HSBC", 3.9f);

}

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

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));

Double amount;

Double totalPayment;

PaymentUtil paymentUtil = **new** PaymentUtil();

String userName, password;

StringBuilder stringBuilder = **new** StringBuilder();

stringBuilder.append("1. Online banking\n").append("2. Credit card\n").append("3. Wallet\n")

.append("Enter the choice:");

System.***out***.println(stringBuilder.toString());

Integer choice = **new** Integer(br.readLine());

DecimalFormat df = **new** DecimalFormat("#.00");

**switch** (choice) {

**case** 1:

System.***out***.println("Enter the user name:");

userName = br.readLine();

System.***out***.println("Enter the password:");

password = br.readLine();

System.***out***.println("Enter the amount:");

amount = **new** Double(br.readLine());

System.***out***.println("Enter the bank name:");

String bankName = br.readLine();

System.***out***.println("Total amount(Inclusive of Service Tax): "

+ df.format(paymentUtil.makePayment(*onlineBanking*, bankName, amount)));

**break**;

**case** 2:

String accNo;

Integer pin;

System.***out***.println("Enter the account number:");

accNo = (br.readLine());

System.***out***.println("Enter the pin:");

pin = **new** Integer(br.readLine());

System.***out***.println("Enter the amount:");

amount = **new** Double(br.readLine());

System.***out***.println(

"Total amount(Inclusive of Service Tax and VAT): " + df.format(paymentUtil.makePayment1(amount)));

**break**;

**case** 3:

System.***out***.println("Enter the user name:");

userName = br.readLine();

System.***out***.println("Enter the password:");

password = br.readLine();

System.***out***.println("Enter the amount:");

amount = **new** Double(br.readLine());

System.***out***.println("Discounted amount: " + df.format(paymentUtil.makePayment2(amount, *discount*)));

**break**;

}

}

}

pom.xml

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>iiht.yaksha.Hq1</groupId>

<artifactId>JavaMasterClassHq1</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>JavaMasterClassHq1</name>

<description>JavaMasterClassHq1</description>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>${maven.compiler.source}</maven.compiler.target>

<junit.jupiter.version>5.5.2</junit.jupiter.version>

<junit.platform.version>1.5.2</junit.platform.version>

</properties>

<dependencies>

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.12</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit.jupiter.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.platform</groupId>

<artifactId>junit-platform-runner</artifactId>

<version>${junit.platform.version}</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

</plugin>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

</plugins>

</build>

</project>

Junit Testing

**package** yaksha;

**import** java.io.File;

**import** java.io.FileWriter;

**import** java.io.IOException;

// boiler-plate code

**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();

}

}

**package** yaksha;

//import static org.junit.jupiter.api.Assertions.\*;

**import** **static** yaksha.TestUtils.*businessTestFile*;

**import** **static** yaksha.TestUtils.*currentTest*;

**import** **static** yaksha.TestUtils.*yakshaAssert*;

**import** java.text.DecimalFormat;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.junit.jupiter.api.Test;

**class** MainClassTest {

**static** Float *discountPercent* = 20.2f;

DecimalFormat df = **new** DecimalFormat("#.00");

**static** Map<String, Float> *onlineBanking*;

**static** {

*onlineBanking* = **new** HashMap<>();

*onlineBanking*.put("ICICI", 4.2f);

*onlineBanking*.put("IBRD", 3f);

*onlineBanking*.put("IFC", 4.9f);

*onlineBanking*.put("HSBC", 3.9f);

}

@Test

**void** testMakePayment() **throws** Exception {

// Test will pass

OnlineBanking onlineBanking1 = **new** OnlineBanking(12560.00, "icici");

PaymentUtil paymentUtil = **new** PaymentUtil();

String result = df.format(

paymentUtil.makePayment(*onlineBanking*, onlineBanking1.getBankName(), onlineBanking1.getAmount()));

*yakshaAssert*(*currentTest*(), (result.contentEquals("13087.52") ? "true" : "false"), *businessTestFile*);

}

@Test

**void** testMakePayment1() **throws** Exception {

PaymentUtil paymentUtil = **new** PaymentUtil();

CreditCard creditCard1 = **new** CreditCard(16300.00);

String result = df.format((paymentUtil.makePayment1(creditCard1.getAmount())));

*yakshaAssert*(*currentTest*(), (result.equals("17541.99") ? "true" : "false"), *businessTestFile*);

}

@Test

**void** testMakePayment2() **throws** Exception {

PaymentUtil paymentUtil = **new** PaymentUtil();

Wallet wallet1 = **new** Wallet(28500.00, 20.2f);

String result = df.format(paymentUtil.makePayment2(wallet1.getAmount(), wallet1.getDiscountPercent()));

*yakshaAssert*(*currentTest*(), (result.contentEquals("22743.00") ? "true" : "false"), *businessTestFile*);

}

}

output\_revised.txt

testMakePayment1=true

testMakePayment2=true

testMakePayment=true

testing-JavaMasterClassHq1.xml

<test-cases>

<cases xsi:type="java:com.assessment.data.TestCase">

<test-case-type>Functional</test-case-type>

<expected-ouput>true</expected-ouput>

<name>testMakePayment</name>

<weight>4</weight>

<mandatory>true</mandatory>

<desc>Testing payment through Online Banking</desc>

</cases>

<cases xsi:type="java:com.assessment.data.TestCase">

<test-case-type>Functional</test-case-type>

<expected-ouput>true</expected-ouput>

<name>testMakePayment1</name>

<weight>3</weight>

<mandatory>true</mandatory>

<desc>Testing payment through Credit Card</desc>

</cases>

<cases xsi:type="java:com.assessment.data.TestCase">

<test-case-type>Functional</test-case-type>

<expected-ouput>true</expected-ouput>

<name>testMakePayment2</name>

<weight>3</weight>

<mandatory>true</mandatory>

<desc>Testing payment through Wallet</desc>

</cases>

</test-cases>

Test Data1

1. Online banking

2. Credit card

3. Wallet

Enter the choice:

1

Enter the user name:

ICICI74484

Enter the password:

Mu7485

Enter the amount:

12560

Enter the bank name:

icici

Total amount(Inclusive of Service Tax): 13087.52

Test Data2

1. Online banking

2. Credit card

3. Wallet

Enter the choice:

2

Enter the account number:

8458 9665 7485 2256

Enter the pin:

8451

Enter the amount:

16300

Total amount(Inclusive of Service Tax and VAT): 17541.99

Test Data3

1. Online banking

2. Credit card

3. Wallet

Enter the choice:

3

Enter the user name:

HSBC7457

Enter the password:

Yu67488

Enter the amount:

28500

Discounted amount: 22743.00

Learning outcome: Participant could able to learn Collection (Map and Set), number format and menu driven programming.