Name of the Course : Ultimate Java Masterclass

Level : Hard

Tool Stack : Java8 and Junit5

Problem Statement : Provide a code solution to accept list of bookings of a car service center in ArrayList<Booking>. calculate the amount of revenue generated by each service engineer. Print serviceEngineer name and number of bookings for the service engineer in sorted order based on the maximum amount generated by engineer using HashMap<String, List<Booking>>().

Description : Your friend, a mechanical engineer is very passionate about cars and wants to be an entrepreneur. He decides to start his own car service center and with his knowledge about cars has the capability to service all different brands of cars. You are also very keen to help him out in any means possible. One fine day, your friend approaches you to help him setup and automate the process of tracking various customers and their service feedbacks. You decide to quickly build a small system to solve the problem. Given the list of bookings, you would like to calculate the amount of revenue generated by each service engineer. For convenience sake, the customer and car references in the booking class is replaced by their respective ids alone. Print serviceEngineer name and number of bookings for the service engineer in sorted order based on the maximum amount generated by engineer using HashMap<String, List<Booking>>().

Create three classes one Booking class with bookingId Long, dateTimeOfService Date(java.util), paymentMode String, customerId Long, carId Long, amount Double, serviceEngineer String types and a parameterized constructor and one FindingBestServiceEngineer class with two static methods.

1. public static Map<String, List<Booking>> organizeBookings(List<Booking> bookingList), which accepts bookingList of List<Booking> type as input and return a Map<String,List<Booking>>.  The key of the map is "serviceEngineer" and the value is the list of all the bookings handled by the service engineer. This method should loop thorough the list of bookings and create multiple smaller lists of booking for each service engineer (value of the hashmap). Add each of the smaller lists to the map and return the same.
2. public static List<String> findBestServiceEngineer(Map<String, List<Booking>> bookingMap), which accepts Map<String,List<Booking>>  as input and return List<String> type. This method takes up the organized booking for each service engineer and computes the total amount generated by each engineer. The list is sorted based on the maximum amount generated by engineer. The list of the service engineer names are returned from the method.

One MainClass with one static method

1. public static void main method to Read input consists of all booking details separated by comma in the below order  
   (bookingId,dateTimeOfService,paymentMode,customerId,carId,amount,serviceEngineer). In output print the serviceEngineer name and number of bookings for the service engineer in sorted order based on the maximum amount generated by engineer. The output format should be System.out.format("%-15s %s\n","Name","No of Booking"); to generate the solution.

Code:

**package** yaksha;

**import** java.util.Date;

**public** **class** Booking {

**private** Long bookingId;

**private** Date dateTimeOfService;

**private** String paymentMode;

**private** Long customerId;

**private** Long carId;

**private** Double amount;

String serviceEngineer;

**public** Long getBookingId() {

**return** bookingId;

}

**public** **void** setBookingId(Long bookingId) {

**this**.bookingId = bookingId;

}

**public** Date getDateTimeOfService() {

**return** dateTimeOfService;

}

**public** **void** setDateTimeOfService(Date dateTimeOfService) {

**this**.dateTimeOfService = dateTimeOfService;

}

**public** String getPaymentMode() {

**return** paymentMode;

}

**public** **void** setPaymentMode(String paymentMode) {

**this**.paymentMode = paymentMode;

}

**public** Long getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(Long customerId) {

**this**.customerId = customerId;

}

**public** Long getCarId() {

**return** carId;

}

**public** **void** setCarId(Long carId) {

**this**.carId = carId;

}

**public** Double getAmount() {

**return** amount;

}

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

**this**.amount = amount;

}

**public** String getServiceEngineer() {

**return** serviceEngineer;

}

**public** **void** setServiceEngineer(String serviceEngineer) {

**this**.serviceEngineer = serviceEngineer;

}

**public** Booking(Long bookingId, Date dateTimeOfService, String paymentMode, Long customerId, Long carId,

Double amount, String serviceEngineer) {

**super**();

**this**.bookingId = bookingId;

**this**.dateTimeOfService = dateTimeOfService;

**this**.paymentMode = paymentMode;

**this**.customerId = customerId;

**this**.carId = carId;

**this**.amount = amount;

**this**.serviceEngineer = serviceEngineer;

}

}

**package** yaksha;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.TreeMap;

**public** **class** FindingBestServiceEngineer {

**public** **static** Map<String, List<Booking>> organizeBookings(List<Booking> bookingList) {

Map<String, List<Booking>> ser = **new** HashMap<String, List<Booking>>();

List<String> service = **new** ArrayList<String>();

Iterator<Booking> itr = bookingList.iterator();

**while** (itr.hasNext()) {

**boolean** flag = **true**;

Booking b = itr.next();

Iterator<String> itr1 = service.iterator();

**while** (itr1.hasNext()) {

**if** (itr1.next().equals(b.getServiceEngineer()))

flag = **false**;

}

**if** (flag)

service.add(b.getServiceEngineer());

}

Iterator<String> itr1 = service.iterator();

**while** (itr1.hasNext()) {

List<Booking> service1 = **new** ArrayList<Booking>();

String s = itr1.next();

Iterator<Booking> itr2 = bookingList.iterator();

**while** (itr2.hasNext()) {

Booking b = itr2.next();

**if** (s.equals(b.getServiceEngineer())) {

service1.add(b);

}

}

ser.put(s, service1);

}

**return** ser;

}

**public** **static** List<String> findBestServiceEngineer(Map<String, List<Booking>> bookingMap) {

List<String> service = **new** ArrayList<String>();

Map<Double, String> bookingSort = **new** TreeMap<Double, String>();

Iterator<String> itr = bookingMap.keySet().iterator();

**while** (itr.hasNext()) {

String key = itr.next();

Double sum = 0.0;

List<Booking> booking = bookingMap.get(key);

Iterator<Booking> itr1 = booking.iterator();

**while** (itr1.hasNext()) {

Booking b = itr1.next();

sum = sum + b.getAmount();

}

bookingSort.put(sum, key);

}

Iterator<Double> itr2 = bookingSort.keySet().iterator();

**while** (itr2.hasNext()) {

service.add(bookingSort.get(itr2.next()));

}

**return** service;

}

}

**package** yaksha;

**import** java.io.BufferedReader;

**import** java.io.InputStreamReader;

**import** java.text.SimpleDateFormat;

**import** java.util.ArrayList;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.ListIterator;

**import** java.util.Map;

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

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

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

List<Booking> booking = **new** ArrayList<Booking>();

Long bookingId;

Date dateTimeOfService;

String paymentMode;

Long customerId;

Long carId;

Double amount;

String serviceEngineer;

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

String ch = "yes";

**do** {

System.***out***.println("Enter a booking detail:");

String details = br.readLine();

String a[] = details.split(",");

bookingId = Long.*parseLong*(a[0]);

dateTimeOfService = sdf.parse(a[1]);

paymentMode = a[2];

customerId = Long.*parseLong*(a[3]);

carId = Long.*parseLong*(a[4]);

amount = Double.*parseDouble*(a[5]);

serviceEngineer = a[6];

Booking b = **new** Booking(bookingId, dateTimeOfService, paymentMode, customerId, carId, amount,

serviceEngineer);

booking.add(b);

System.***out***.println("Do you want to add another booking detail:");

ch = br.readLine();

} **while** (ch.equalsIgnoreCase("yes"));

Map<String, List<Booking>> sername = FindingBestServiceEngineer.*organizeBookings*(booking);

System.***out***.format("%-15s %s\n", "Name", "No of Booking");

List<String> service = FindingBestServiceEngineer.*findBestServiceEngineer*(sername);

ListIterator<String> ltr = service.listIterator(service.size());

**while** (ltr.hasPrevious()) {

String key = ltr.previous();

System.***out***.format("%-15s %s\n", key, sername.get(key).size());

}

}

}

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.Hq2</groupId>

<artifactId>JavaMasterClassHq2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>JavaMasterClassHq2</name>

<description>JavaMasterClassHq2</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** yaksha.TestUtils.*businessTestFile*;

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

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

**import** java.text.SimpleDateFormat;

**import** java.util.\*;

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

**class** MainClassTest {

@Test

**void** testfindBestServiceEngineer() **throws** Exception {

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

List<Booking> booking = **new** ArrayList<Booking>();

Booking booking1 = **new** Booking((**long**) 1, sdf.parse("06-01-2015"), "CC", (**long**) 11, (**long**) 1001, 100.00, "John");

Booking booking2 = **new** Booking((**long**) 2, sdf.parse("01-06-2017"), "CC", (**long**) 12, (**long**) 1002, 200.00,

"Peter");

Booking booking3 = **new** Booking((**long**) 3, sdf.parse("28-02-2015"), "DC", (**long**) 13, (**long**) 1003, 150.00,

"Peter");

Booking booking4 = **new** Booking((**long**) 4, sdf.parse("24-06-2015"), "CC", (**long**) 14, (**long**) 1004, 250.00, "John");

Booking booking5 = **new** Booking((**long**) 5, sdf.parse("14-06-2016"), "DC", (**long**) 15, (**long**) 1005, 75.00, "John");

Booking booking6 = **new** Booking((**long**) 6, sdf.parse("12-05-2017"), "DC", (**long**) 16, (**long**) 1006, 125.00,

"Peter");

Booking booking7 = **new** Booking((**long**) 7, sdf.parse("24-12-2011"), "CC", (**long**) 17, (**long**) 1007, 185.00,

"Peter");

booking.add(booking1);

booking.add(booking2);

booking.add(booking3);

booking.add(booking4);

booking.add(booking5);

booking.add(booking6);

booking.add(booking7);

Map<String, List<Booking>> sername = FindingBestServiceEngineer.*organizeBookings*(booking);

String expectedResult[] = { "John", "Peter" };

List<String> service1 = Arrays.*asList*(expectedResult);

List<String> service = FindingBestServiceEngineer.*findBestServiceEngineer*(sername);

*yakshaAssert*(*currentTest*(), (service.containsAll(service1) ? "true" : "false"), *businessTestFile*);

}

}

output\_revised.txt

testfindBestServiceEngineer=true

testing-JavaMasterClassMqTwo.xml

<test-cases>

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

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

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

<name>testFindBestServiceEngineer</name>

<weight>10</weight>

<mandatory>true</mandatory>

<desc>Test to findout best service engineer from given collection data</desc>

</cases>

</test-cases>

Test Data1

Enter a booking detail:

1,06-01-2015,CC,11,1001,100,John

Do you want to add another booking detail:

yes

Enter a booking detail:

2,01-06-2017,CC,12,1002,200,Peter

Do you want to add another booking detail:

yes

Enter a booking detail:

3,28-02-2015,DC,13,1003,150,Peter

Do you want to add another booking detail:

yes

Enter a booking detail:

4,24-06-2015,CC,14,1004,250,John

Do you want to add another booking detail:

yes

Enter a booking detail:

5,14-06-2016,DC,15,1005,75,John

Do you want to add another booking detail:

yes

Enter a booking detail:

6,12-05-2017,DC,16,1006,125,Peter

Do you want to add another booking detail:

yes

Enter a booking detail:

7,24-12-2011,CC,17,1007,185,Peter

Do you want to add another booking detail:

no

Name No of Booking

Peter 4

John 3

Learning outcome: Participant could able to learn Iterator, ListIterator, TreeMap, HashMap, ArrayList in collection, Date, Date Format, and IO using BufferedReader.