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

Level : Easy

Tool Stack : Java9 and Junit4

Problem Statement : Provide a code solution to declare star performer of Salespersons based on Java 9 stream API, collection etc features.

Description : ***The Great Fancy Store*** sales various types of consumable goods. During the festive season of Dussera and Diwali, it sets target of minimum sale of Rs 100000/- per salesperson. Whenever any salesperson crosses target, his/her id, name is be placed in a collection. So collection must maintain the order of salesperson details. After Diwali each salesperson’s total sale amount for that season to be added to that collection. Based on this data each salesperson’s appraisal will be taken care. Id of the salesperson cannot be duplicated. The authority declares first five persons as star performers. Among them who’s sale is the highest called 5star,next 4star and so on. You are provided with numbers of salesperson’s id, name and performances in a sequence. You need to develop an application so that it will display the list of star performers along with grade and other details.

You need create

1. class Salesperson with private member data

String id;

String name;

Integer saleAmount,

String grade;

Create getter/setter methods and constructors.

override toString() in String.format("%-5s %-20s %-10s %-10s").

1. class Main with method public static void main(String [] arg): It will asks for number of salesperson, If it is 0 or negative number message “invalid input” will be displayed and stop the application, if it is a valid positive number it will accept the each saleperson’s information as a string in a comma(,) separate format (eg: 7024,Tom Matt,175000).Finally displays all five the star performer details along with grade in the order of grade (i.e. 5star, 4star, 3star, 2star and 1star).

Code:

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@Data

@AllArgsConstructor

@NoArgsConstructor

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

**private** Integer id;

**private** String name;

**private** Integer saleAmount;

**private** String grade;

@Override

**public** String toString() {

String output=String.*format*("%-5s %-20s %-10s %-10s",id,name,saleAmount,grade);

**return** output;

}

**public** **int** compareTo(SalesPerson salesPerson)

{

**return** **this**.saleAmount.compareTo(salesPerson.saleAmount);

}

@Override

**public** **boolean** equals(Object obj) {

SalesPerson other = (SalesPerson) obj;

**if**(**this**.hashCode()==other.hashCode())

**return** **true**;

**else**

**return** **false**;

}

@Override

**public** **int** hashCode() {

**return** id;

}

**public** SalesPerson(Integer id, String name, Integer saleAmount) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.saleAmount = saleAmount;

}

}

**import** java.util.Collections;

**import** java.util.LinkedHashSet;

**import** java.util.List;

**import** java.util.Scanner;

**import** java.util.Set;

**import** java.util.stream.Collectors;

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

**public** **class** Main

{

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

{

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

System.***out***.println("Enter number of Sales Persons:");

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

Set<SalesPerson> salesPersonSet=**new** LinkedHashSet<SalesPerson>();

System.***out***.println("Enter Sales Person's details one line at a time");

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

{

String str=scanner.nextLine();

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

SalesPerson salesPerson=**new** SalesPerson(Integer.*parseInt*(arr[0]),arr[1],Integer.*parseInt*(arr[2]));

salesPersonSet.add(salesPerson);

}

Stream<SalesPerson> salesStarStream=salesPersonSet.stream().filter(salesPerson->salesPerson.getSaleAmount()>=100000).limit(5);

List<SalesPerson> salesStarList=salesStarStream.collect(Collectors.*toList*());

Collections.*sort*(salesStarList);

**int** x=0;

**for**(SalesPerson salesPerson:salesStarList)

{

x++;

salesPerson.setGrade(""+x+"Star");

}

Collections.*reverse*(salesStarList);

System.***out***.println("\nThe Star Performers List");

System.***out***.println(String.*format*("%-5s %-20s %-10s %-10s","ID","NAME","SALE AMOUNT","GRADE"));

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

}

}

Junit Testing

Test Data1

Enter Number of consumers bill to prepare:

-3

invalid input

Test Data2

Enter number of Sales Persons:

10

Enter Sales Person's details one line at a time

753,Ruben Cohen,175000

923,Anna,135000

1001,Julie,25000

1245,Isabel,195000

1321,Rob,16500

1743,Bob,165000

1263,Rio,145000

1724,Lou,125000

1921,Ria,124000

182,Mia,115000

The Star Performers List

ID NAME SALE AMOUNT GRADE

1245 Isabel 195000 5Star

753 Ruben Cohen 175000 4Star

1743 Bob 165000 3Star

1263 Rio 145000 2Star

923 Anna 135000 1Star

Learning outcome: Participant could able to use Stream API.