Core Java Assignment 9

|  |
| --- |
|  |
|  | import java.util.ArrayList;  import java.util.Arrays; |
|  | import java.util.Collections; |
|  | import java.util.Comparator; |
|  | import java.util.HashMap; |
|  | import java.util.HashSet; |
|  | import java.util.Iterator; |
|  | import java.util.List; |
|  | import java.util.Map; |
|  | import java.util.Set; |
|  | import java.util.TreeSet; |
|  | import java.util.function.Consumer; |
|  | import java.util.function.Function; |
|  | import java.util.stream.Collector; |
|  | import java.util.stream.Collectors; |
|  |  |
|  | import javax.swing.plaf.nimbus.NimbusLookAndFeel; |
|  |  |
|  | public class StreamDemo { |
|  |  |
|  | public static void main(String[] args) { |
|  |  |
|  | List<Fruit> fruitList = Arrays.asList( |
|  | new Fruit("A", 150 , 10, "Red"), |
|  | new Fruit("B", 60 , 30, "Blue"), |
|  | new Fruit("C", 30 , 20, "Red"), |
|  | new Fruit("D", 180 , 50, "Blue") |
|  | ); |
|  |  |
|  | List<News> newsList = Arrays.asList( |
|  | new News(1, "E" , "I", "Hello"), |
|  | new News(2, "F" , "J", "Hiii"), |
|  | new News(1, "F" , "K", "Thankyou"), |
|  | new News(4, "H" , "I", "Welcome") |
|  |  |
|  | ); |
|  |  |
|  | List<Trader> traderList = Arrays.asList( |
|  | new Trader("O", "Pune"), |
|  | new Trader("N", "Mumbai"), |
|  | new Trader("M", "pune"), |
|  | new Trader("P", "Delhi") |
|  | ); |
|  |  |
|  | List<Transaction> transactionList = Arrays.asList( |
|  | new Transaction(traderList.get(0), 2000, 1000), |
|  | new Transaction(traderList.get(1), 2011, 8000), |
|  | new Transaction(traderList.get(2), 2011, 3000), |
|  | new Transaction(traderList.get(3), 2003, 6000) |
|  | ); |
|  |  |
|  |  |
|  | // 1st Question |
|  |  |
|  | System.out.println("Stream First Question output"); |
|  |  |
|  | fruitList.stream().filter(l -> l.calories<100).forEach(l -> System.out.println(l.name)); |
|  |  |
|  |  |
|  | // 2nd Question |
|  | System.out.println("\n"+"Stream Second Question output"); |
|  | fruitList.stream().sorted(Comparator.comparing(l -> l.color)).forEach( l-> System.out.println(l)); |
|  |  |
|  |  |
|  |  |
|  | //4th Question |
|  |  |
|  | System.out.println("\n"+"Stream 4th Question output"); |
|  | newsList.stream().collect(Collectors.groupingBy(l -> l.newsId, Collectors.counting())) |
|  | .entrySet() |
|  | .stream() |
|  | .max(Map.Entry.comparingByValue()) |
|  | .ifPresent(l-> System.out.println("News Id : "+ l.getKey() + " has the maxium comment i.e. :" + l.getValue())); |
|  |  |
|  |  |
|  |  |
|  | //6th Question |
|  | System.out.println("\n"+"Stream six Question output"); |
|  | newsList.stream().collect(Collectors.groupingBy(l->l.commentByUser, Collectors.counting())) |
|  | .entrySet() |
|  | .stream() |
|  | .max(Map.Entry.comparingByValue()) |
|  | .ifPresent(l-> System.out.println("User Id : "+ l.getKey() + " has did the maximum comment i.e. :" + l.getValue())); |
|  |  |
|  |  |
|  | // 8th Question |
|  | System.out.println("\n"+"Stream 8th Question output"); |
|  |  |
|  | transactionList.stream().filter(l -> l.year == 2011).sorted(Comparator.comparingInt(l-> l.value)) |
|  | .forEach(l -> System.out.println(l)); |
|  |  |
|  | //9th |
|  | System.out.println("\n"+"Stream 9th Question output"); |
|  | traderList.stream().map(l-> l.city.toLowerCase()).distinct().forEach(l -> System.out.println(l)); |
|  |  |
|  | // 10th Question |
|  | System.out.println("\n"+"Stream 10th Question output"); |
|  | traderList.stream().filter(l -> l.city.equalsIgnoreCase("Pune")).sorted(Comparator.comparing(l -> l.name)) |
|  | .forEach(l -> System.out.println(l)); |
|  |  |
|  | //13 |
|  | System.out.println("\n"+"Stream 13th Question output"); |
|  | transactionList.stream().filter(l-> l.trader.city.equalsIgnoreCase("Delhi")).forEach(System.out::println); |
|  |  |
|  | //14th question |
|  | System.out.println("\n"+"Stream 14th Question output"); |
|  | transactionList.stream().max(Comparator.comparingInt(l-> l.value)).ifPresent(System.out::println);; |
|  |  |
|  | } |
|  |  |
|  |  |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  | class Fruit{ |
|  | String name; |
|  | int calories; |
|  | int price; |
|  | String color; |
|  | public Fruit(String name, int calories, int price, String color) { |
|  | super(); |
|  | this.name = name; |
|  | this.calories = calories; |
|  | this.price = price; |
|  | this.color = color; |
|  | } |
|  | @Override |
|  | public String toString() { |
|  | return "Fruit [name=" + name + ", calories=" + calories + ", price=" + price + ", color=" + color + "]"; |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |
|  |  |
|  | class News{ |
|  | int newsId; |
|  | String postedByUser; |
|  | String commentByUser; |
|  | String comment; |
|  | public News(int newsId, String postedByUser, String commentByUser, String comment) { |
|  | super(); |
|  | this.newsId = newsId; |
|  | this.postedByUser = postedByUser; |
|  | this.commentByUser = commentByUser; |
|  | this.comment = comment; |
|  | } |
|  |  |
|  | } |
|  |  |
|  | class Trader{ |
|  | String name; |
|  | String city; |
|  | public Trader(String name, String city) { |
|  | super(); |
|  | this.name = name; |
|  | this.city = city; |
|  | } |
|  |  |
|  | @Override |
|  | public String toString() { |
|  |  |
|  | return name+" "+ city; |
|  | } |
|  |  |
|  | } |
|  |  |
|  | class Transaction{ |
|  | Trader trader; |
|  | int year; |
|  | int value; |
|  | public Transaction(Trader trader, int year, int value) { |
|  | super(); |
|  | this.trader = trader; |
|  | this.year = year; |
|  | this.value = value; |
|  | } |
|  | @Override |
|  | public String toString() { |
|  | return trader +" "+year+ " " +value ; |
|  | } |
|  |  |
|  | } |