Stream API Assignment-9

SetUp:

Create the following classes:

class Fruit{String calories; int price; String color;}

class News{int newsId; String postedByUser; String commentByUser; String comment;}

class Trader {String name; String city;}

class Transaction (Trader trader; int year; int value;}

Class : Fruits.java

package org.stream.app;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.stream.Collectors;

public class Fruits {

private String name;

private Integer calories;

private Integer price;

private String color;

public Fruits(String name, Integer calories, Integer price, String color) {

this.name = name;

this.calories = calories;

this.price=price;

this.color = color;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getCalories() {

return calories;

}

public void setCalories(Integer calories) {

this.calories = calories;

}

public Integer getPrice() {

return price;

}

public void setPrice(Integer price) {

this.price = price;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

@Override

public String toString() {

return "Fruits{" +

"name='" + name + '\'' +

", calories=" + calories +

", color='" + color + '\'' +

'}';

}

}

Class : StreamsTest.java

package org.stream.app;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.stream.Collectors;

public class StreamsTest {

public static void main(String args[]) {

List<Fruits> fruitsList = new ArrayList<>();

fruitsList.add(new Fruits("Apple", 100, 10, "red"));

fruitsList.add(new Fruits("Banana", 100, 20, "Yellow"));

fruitsList.add(new Fruits("PineApple", 80, 30, "Brown"));

fruitsList.add(new Fruits("Guva", 70, 10, "Green"));

fruitsList.add(new Fruits("Watermelon", 100, 50, "Green"));

fruitsList.add(new Fruits("Grapes", 90, 30, "green"));

System.out.println("-----------------------Question1-----------------------------------");

//Display the fruit names of low calories fruits---Question-1

List<String> lowCalories = fruitsList.stream().filter(f -> f.getCalories() < 100).map(t -> t.getName()).sorted(Collections.reverseOrder()).collect(Collectors.toList());

System.out.println("Desending order of calories" + lowCalories);

System.out.println("--------------------------Question2--------------------------------");

//Display color wise fruit names--Question-2

for (Fruits f : fruitsList) {

System.out.println(f.getColor() + " " + f.getName());

}

System.out.println("------------------------Question3----------------------------------");

//Display only RED color fruits sorted as per their price in Ascending Order

List<Fruits> colorList = fruitsList.stream().filter(e -> e.getColor().equalsIgnoreCase("Red")).sorted(Comparator.comparing(Fruits::getPrice)).collect(Collectors.toList());

System.out.println(colorList);

}

}

Class: News.java

**package** org.stream.app;

**public** **class** News

{

**private** Integer newsId;

**private** String postedByUser;

**private** String commentByUser;

**private** String comment;

**public** News(Integer newsId, String postedByUser, String commentByUser, String comment) {

**this**.newsId = newsId;

**this**.postedByUser = postedByUser;

**this**.commentByUser = commentByUser;

**this**.comment = comment;

}

**public** Integer getNewsId() {

**return** newsId;

}

**public** **void** setNewsId(Integer newsId) {

**this**.newsId = newsId;

}

**public** String getPostedByUser() {

**return** postedByUser;

}

**public** **void** setPostedByUser(String postedByUser) {

**this**.postedByUser = postedByUser;

}

**public** String getCommentByUser() {

**return** commentByUser;

}

**public** **void** setCommentByUser(String commentByUser) {

**this**.commentByUser = commentByUser;

}

**public** String getComment() {

**return** comment;

}

**public** **void** setComment(String comment) {

**this**.comment = comment;

}

@Override

**public** String toString() {

**return** "News{" +

"newsId=" + newsId +

", postedByUser='" + postedByUser + '\'' +

", commentByUser='" + commentByUser + '\'' +

", comment='" + comment + '\'' +

'}';

}

}

Class: NewsTest.java

**package** org.stream.app;

**import** java.util.ArrayList;

**import** java.util.Comparator;

**import** java.util.List;

**import** java.util.Optional;

**public** **class** NewsTest {

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

List<News> newsList = **new** ArrayList<>();

newsList.add(**new** News(101, "Ravi", "Raju", "Nice"));

newsList.add(**new** News(102, "priya", "Ravi", "Good"));

newsList.add(**new** News(103, "mani", "mani", "Budget"));

newsList.add(**new** News(104, "vinay", "Vinay", "Budget"));

newsList.add(**new** News(105, "Chandu", "Pranith", "Nice"));

System.***out***.println("------------------------Question5-----------------------------------------------");

//BudgetCount

Long count = newsList.stream().filter(e -> e.getComment().equalsIgnoreCase("budget")).count();

System.***out***.println("the count of budet are:" + count);

System.***out***.println("------------------------Question6-----------------------------------------------");

//Max posted by user

Optional<String> user = newsList.stream().max(Comparator.*comparing*(News::getPostedByUser)).map(News::getPostedByUser);

System.***out***.println(user.get());

System.***out***.println("------------------------Question7-----------------------------------------------");

}

}

Class: Trader.java

**package** org.stream.app;

**public** **class** Trader

{

**private** String name;

**private** String city;

**public** Trader(String name, String city) {

**this**.name = name;

**this**.city = city;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

@Override

**public** String toString() {

**return** "Trader{" +

"name='" + name + '\'' +

", city='" + city + '\'' +

'}';

}

}

Class: TradersTest.java

**package** org.stream.app;

**import** java.util.ArrayList;

**import** java.util.Comparator;

**import** java.util.List;

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

**public** **class** TradersTest {

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

List<Trader> traderList = **new** ArrayList<>();

traderList.add(**new** Trader("Walmart", "Banglore"));

traderList.add(**new** Trader("Jim", "Pune"));

traderList.add(**new** Trader("John", "indore"));

traderList.add(**new** Trader("Steven", "chennai"));

traderList.add(**new** Trader("Richad", "Banglore"));

traderList.add(**new** Trader("Richad", "Pune"));

System.***out***.println("---------------------------Question10-------------------------------------");

//sort them by name

List<Trader> sortNames = traderList.stream().filter(e -> e.getCity().equalsIgnoreCase("pune")).sorted(Comparator.*comparing*(Trader::getName)).collect(Collectors.*toList*());

System.***out***.println(sortNames);

System.***out***.println("---------------------------Question11-------------------------------------");

List<String> stringList = traderList.stream().map(Trader::getName).sorted().collect(Collectors.*toList*());

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

System.***out***.println("---------------------------Question12-------------------------------------");

traderList.stream().filter(e -> e.getCity().equalsIgnoreCase("indore")).forEach(System.***out***::println);

System.***out***.println("---------------------------Question9-------------------------------------");

traderList.stream().map(Trader::getCity).distinct().forEach(System.***out***::println);

}

}

Class: Transaction.java

**package** org.stream.app;

**public** **class** Transaction

{

String name;

String city;

**int** year;

**int** value;

**public** Transaction(String name, String city, **int** year, **int** value) {

**super**();

**this**.name = name;

**this**.city = city;

**this**.year = year;

**this**.value = value;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** **int** getYear() {

**return** year;

}

**public** **void** setYear(**int** year) {

**this**.year = year;

}

**public** **int** getValue() {

**return** value;

}

**public** **void** setValue(**int** value) {

**this**.value = value;

}

@Override

**public** String toString() {

**return** "transaction [name=" + name + ", city=" + city + ", year=" + year + ", value=" + value + "]";

}

}

Class:TransactionTest.java

**package** org.stream.app;

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

**public** **class** TransactionTest

{

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

{

List<Transaction> m1=**new** ArrayList<Transaction>();

m1.add(**new** Transaction("priya","hyd", 2011,250));

m1.add(**new** Transaction("mani", "srcl", 2018,1200));

m1.add(**new** Transaction("madhu", "Hyd", 2011,950));

m1.add(**new** Transaction("chintu", "mumbai", 1999,450));

m1.add(**new** Transaction("Anusha","Delhi", 2011,350));

// Print all transaction values from the traders living in Delhi ----->Question13

System.***out***.println("printing all the transcation value living in delhi");

m1.stream().filter(n->n.getCity().contains("Delhi"))

.forEach(pr->System.***out***.println(pr.getName()+" "+pr.getValue()+" "+pr.getYear()));

// What's the highest value of all the transactions ------>Question14

System.***out***.println("highest value of all transcation");

Optional<Integer> max=m1.stream().map(n->n.getValue()).max((v1,v2) -> v1.compareTo(v2));

System.***out***.println(max.get());

// Find the transaction with the smallest value ------->Question15

System.***out***.println("smallest value of all transcation");

Optional<Integer> min=m1.stream().map(n->n.getValue()).min((v1,v2) -> v1.compareTo(v2));

System.***out***.println(min.get());

//Find all transactions in the year 2011 and sort them by value(Small to high)------>Question8

System.***out***.println("transaction in 2011 and sort by value small to large");

m1.stream().filter(n->n.getYear()==(2011)).sorted((p1,p2) -> p1.getValue()-p2.getValue()).forEach(p-> System.***out***.println(p.getName()+" "+p.getValue()));

}

}

1. Display the fruit names of low calories fruits i.e calories < 100 sorted in descending order of

calories.

List<String> lowCalories = fruitsList.stream().filter(f -> f.getCalories() < 100).map(t -> t.getName()).sorted(Collections.*reverseOrder*()).collect(Collectors.*toList*());

System.***out***.println("Desending order of calories" + lowCalories);

1. Display color wise list of fruit names.

**for** (Fruits f : fruitsList) {

System.***out***.println(f.getColor() + " " + f.getName());

1. Display only RED color fruits sorted as per their price in ascending order.

List<Fruits> colorList = fruitsList.stream().filter(e -> e.getColor().equalsIgnoreCase("Red")).sorted(Comparator.comparing(Fruits::getPrice)).collect(Collectors.*toList*());

System.***out***.println(colorList);

1. Find out the newsId which has received maximum comments.
2. Find out how many times the word ‘budget’ arrived in user comments all news.

Long count = newsList.stream().filter(e -> e.getComment().equalsIgnoreCase("budget")).count();

System.***out***.println("the count of budet are:" + count);

1. Find out which user has posted maximum comments.

Optional<String> user = newsList.stream().max(Comparator.*comparing*(News::getPostedByUser)).map(News::getPostedByUser);

System.***out***.println(user.get());

1. Display commentByUser wise number of comments.
2. Find all transactions in the year 2011 and sort them by value(small to high).

System.***out***.println("transaction in 2011 and sort by value small to large");

m1.stream().filter(n->n.getYear()==(2011)).sorted((p1,p2) -> p1.getValue()-p2.getValue()).forEach(p-> System.***out***.println(p.getName()+" "+p.getValue()));

1. What are all the unique cities where the traders work?

traderList.stream().map(Trader::getCity).distinct().forEach(System.***out***::println);

1. Find all traders from Pune and sort them by name.

List<Trader> sortNames = traderList.stream().filter(e -> e.getCity().equalsIgnoreCase("pune")).sorted(Comparator.*comparing*(Trader::getName)).collect(Collectors.*toList*());

System.***out***.println(sortNames);

1. Return a String of all traders’ names sorted alphabetically.

List<String> stringList = traderList.stream().map(Trader::getName).sorted().collect(Collectors.*toList*()); stringList.forEach(System.***out***::println);

1. Are any traders based in Indore?

traderList.stream().filter(e -> e.getCity().equalsIgnoreCase("indore")).forEach(System.***out***::println);

1. Print all transactions value from the traders living in Delhi.

System.***out***.println("printing all the transcation value living in delhi");

m1.stream().filter(n->n.getCity().contains("Delhi")).forEach(pr- >System.***out***.println(pr.getName()+" "+pr.getValue()+" "+pr.getYear()));

1. What’s the highest value of all the transactions?

System.***out***.println("highest value of all transcation");

Optional<Integer> max=m1.stream().map(n->n.getValue()).max((v1,v2) -> v1.compareTo(v2));

System.***out***.println(max.get());

1. Find the transaction with the smallest value.

System.***out***.println("smallest value of all transcation");

Optional<Integer> min=m1.stream().map(n->n.getValue()).min((v1,v2) -> v1.compareTo(v2));

System.***out***.println(min.get());