**Core Java Assignment 9**

**Streams Assignment**

**Question**

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.concurrent.ConcurrentHashMap;

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

**import** java.util.Comparator;

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

**import** java.util.function.Function;

**import** java.util.function.Predicate;

**class** DistinctByCustomPropertyExample{

**public** **static** <T> Predicate<T> distinctByKey(Function<? **super** T,Object> keyExtractor){

Map<Object, Boolean> uniqueMap = **new** ConcurrentHashMap<>();

**return** t -> uniqueMap.putIfAbsent(keyExtractor.apply(t), Boolean.***TRUE***) == **null**;

}

}

**class** Fruit{

String name;

**int** calories;

**int** price;

String colour;

**public** Fruit(String name, **int** calories, **int** price, String colour) {

**super**();

**this**.name = name;

**this**.calories = calories;

**this**.price = price;

**this**.colour = colour;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** **int** getCalories() {

**return** calories;

}

**public** **void** setCalories(**int** calories) {

**this**.calories = calories;

}

**public** **int** getPrice() {

**return** price;

}

**public** **void** setPrice(**int** price) {

**this**.price = price;

}

**public** String getColour() {

**return** colour;

}

**public** **void** setColour(String colour) {

**this**.colour = colour;

}

@Override

**public** String toString() {

**return** "Fruit [name=" + name + ", calories=" + calories + ", price=" + price + ", colour=" + colour + "]";

}

}

**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;

}

**public** **int** getNewsID() {

**return** newsID;

}

**public** **void** setNewsID(**int** 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** Trader{

String name;

String city;

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

**super**();

**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** 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;

}

**public** Trader getTrader() {

**return** trader;

}

**public** **void** setTrader(Trader trader) {

**this**.trader = trader;

}

**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 [trader=" + trader + ", year=" + year + ", value=" + value + "]";

}

}

**public** **class** Stream\_API\_Assignment {

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

// Display the fruit names of low calories fruits i.e. calories<100 in decreasing order

List<Fruit> list = Arrays.*asList*(

**new** Fruit("Mango", 50, 100, "Yellow"),

**new** Fruit("Orange", 100, 200, "Orang"),

**new** Fruit("Watermelon",150, 170, "Red"),

**new** Fruit("Grape", 40, 300, "Green"),

**new** Fruit("Fig",70, 150, "Purple"),

**new** Fruit("Apple",190, 270, "Red")

);

Stream<Fruit> filtered\_data = list.stream().filter(p -> p.calories<100);

Stream<Fruit> decreasingOrder = list.stream().sorted(Comparator.*comparingInt*(Fruit::getCalories)).filter(p -> p.calories<100);

System.***out***.println("Display the fruit names of low calories fruits i.e. calories<100 in decreasing order:");

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

// Display color wise list of fruit name

System.***out***.println("\nDisplay color wise list of fruit name");

list.forEach((Fruit) -> System.***out***.println("Fruit Name: "+Fruit.getName()+" Fruit Colour: "+Fruit.getColour()));

// Display RED colour fruits in ascending order

System.***out***.println("\nDisplay RED colour fruits in ascending order:");

list.stream().filter(f -> f.getColour().matches("Red"))

.sorted(Comparator.*comparing*(Fruit::getPrice))

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

//List of News

List<News> list1 = Arrays.*asList*(

**new** News(1, "A", "abc", "Hii"),

**new** News(2, "B", "bcd", "Report submitted to college"),

**new** News(3, "C", "www", "strike is made by farmers"),

**new** News(4, "D", "tyu", "Gold medal achived by Neeraj"),

**new** News(5, "E", "mno", "kashmir is now union territory")

);

// NewsID which recevied maximum comment

System.***out***.println("\nNewsId which received max comments:");

**int** l=0;

**for**(News i:list1) {

**if**(i.getComment().length()>l) {

l=i.getComment().length();

}

}

**for**(News i:list1) {

**if**(i.getComment().length()==l) {

System.***out***.println(i.getNewsID());

}

}

//Find out how many times 'is' word arrived in users coment

Long count=list1.stream().filter(P->P.getComment().contains("is")).count();

System.***out***.println("\nis appeared in comments "+count+" times\n");

//Display comment by user wise no of comments

list1.forEach((News)-> {

System.***out***.println("usercomments: " + News.getCommentByUser() + "," + " Number of comments: "+ News.getComment());

});

//List of Trader

Trader t1=**new** Trader("Chetana","Sangli");

Trader t2=**new** Trader("Anjali","Delhi");

Trader t3=**new** Trader("Sonali","Indore");

Trader t4=**new** Trader("Snehal","Delhi");

Trader t5=**new** Trader("Yogita","Pune");

Trader t6=**new** Trader("Vedika","Delhi");

Trader t7=**new** Trader("Manali","Pune");

List<Transaction> trans = Arrays.*asList*(

**new** Transaction(t1, 2015, 10000),

**new** Transaction(t2, 2011, 9000),

**new** Transaction(t3, 2012, 12000),

**new** Transaction(t4, 2011, 3000),

**new** Transaction(t5, 2018, 19000),

**new** Transaction(t6, 2019, 5000),

**new** Transaction(t7, 2012, 8000)

);

//Transaction in year 2011 in sorted form

System.***out***.println("\nTransaction in year 2011");

List<Transaction> Year2011=trans.stream().filter(P -> P.year==2011).sorted(Comparator.*comparing*(Transaction::getValue)).collect(Collectors.*toList*());

Year2011.forEach(System.***out***::println);

//unique cities where traders work

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

trader.add(t1);

trader.add(t2);

trader.add(t3);

trader.add(t4);

trader.add(t5);

trader.add(t6);

trader.add(t7);

System.***out***.println("\nUnique Cities:");

List<Trader> UniqueCity=trader.stream().filter(DistinctByCustomPropertyExample.*distinctByKey*(P -> P.getCity())).collect(Collectors.*toList*());

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

//Traders from pune and sorted by name

System.***out***.println("\nTrader from pune in sorting order by their name:");

List<Trader> FromPune=trader.stream().filter(P -> P.city=="Pune").sorted(Comparator.*comparing*(Trader::getName)).collect(Collectors.*toList*());

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

//Sorted trader name

System.***out***.println("\nSorted trader name:");

trader.stream().sorted(Comparator.*comparing*(Trader::getName)).forEach(P ->System.***out***.println(P.name));

//Trader from Indore

System.***out***.println("\nTrader from Indore:");

List<Trader> FromIndore=trader.stream().filter(P -> P.city=="Indore").collect(Collectors.*toList*());

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

//Print all transaction values from trader living in delhi

System.***out***.println("\nTransaction list from Delhi Trader:");

List<Transaction> FromDelhi=trans.stream().filter(P -> P.trader.getCity().equals("Delhi")).collect(Collectors.*toList*());

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

//Highest value from transaction

System.***out***.println("\nHighest Transaction Value:");

Transaction Max=trans.stream().max(Comparator.*comparing*(Transaction::getValue)).get();

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

//Lowest value from transaction

System.***out***.println("\nLowest Transaction Value:");

Transaction Min=trans.stream().min(Comparator.*comparing*(Transaction::getValue)).get();

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

}

}



