CORE JAVA

STREAMS ASSIGNMENT

Setup:

Create the following classes:

class Fruit { String name; int 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; }

1. Display the fruit names of low calories fruits i.e. calories < 100 sorted in descending order of calories.
2. Display color wise list of fruit names.
3. Display only RED color fruits sorted as per their price in ascending order.
4. Find out the newsId which has received maximum comments.
5. Find out how many times the word 'budget' arrived in user comments all news.
6. Find out which user has posted maximum comments.
7. Display commentByUser wise number of comments.
8. Find all transactions in the year 2011 and sort them by value (small to high).
9. What are all the unique cities where the traders work?
10. Find all traders from Pune and sort them by name.
11. Return a string of all traders' names sorted alphabetically.
12. Are any traders based in Indore?
13. Print all transactions' values from the traders living in Delhi.
14. What's the highest value of all the transactions?
15. Find the transaction with the smallest value.

Code: -

*StreamDemo.java

```
1  package stream;
2  import java.util.*;
3  import java.util.stream.Collectors;
4  public class StreamDemo
5  {
6  public static void main(String[] args)
7  {
8           List<Fruit> fruitList = Arrays.asList(
9                    new Fruit("Mango", 150 , 10, "Yellow"),
10                   new Fruit("Apple", 60 , 30, "Red"),
11                   new Fruit("Orange", 30 , 20, "Orange"),
12                   new Fruit("Banana", 50 , 50, "Yellow")
13                   );
14
15          List<News> newsList = Arrays.asList(
16                   new News(1, "N1" , "I", "Hello"),
17                   new News(2, "N2" , "J", "budget"),
18                   new News(1, "N3" , "K", "budget"),
19                   new News(4, "N4" , "I", "Welcome")
20
21                   );
22
23          List<Trader> traderList = Arrays.asList(
24                   new Trader("Siddhi", "Pune"),
25                   new Trader("Mrunal", "Mumbai"),
26                   new Trader("Yash", "Indore"),
27                   new Trader("Myra", "Delhi")
28                   );
29
30          List<Transaction> transactionList = Arrays.asList(
31                   new Transaction(traderList.get(0), 2000, 1000),
32                   new Transaction(traderList.get(1), 2011, 8000),
```

*StreamDemo.java

```
33                   new Transaction(traderList.get(2), 2011, 3000),
34                   new Transaction(traderList.get(3), 2003, 6000)
35                   );
36
37
38         // 1st Question
39         System.out.println("Stream 1st Question output-Fruit names of low calories:");
40         fruitList.stream().filter(l -> l.calories<100).forEach(l -> System.out.println(l.name));
41
42         // 2nd Question
43         System.out.println("\n"+"Stream 2nd Question output-Colorwise list of fruit names:");
44         fruitList.stream().sorted(Comparator.comparing(l -> l.color)).forEach( l-> System.out.println(l));
45
46         //3rd Question
47         System.out.println("\n"+"Stream 3rd Question output-Only red color fruits:");
48         fruitList.stream().filter(l -> l.color=="Red").forEach(l -> System.out.println(l.name));;
49
50          //4th Question
51
52         System.out.println("\n"+"Stream 4th Question output - news id with max content:");
53         newsList.stream().collect(Collectors.groupingBy(l -> l.newsId, Collectors.counting()))
54           .entrySet()
55           .stream()
56           .max(Map.Entry.comparingByValue())
57           .ifPresent(l-> System.out.println("News Id : "+ l.getKey() + " has the maxium comment i.e. :" + l.getValue()));
58
59         //5th Question
60         System.out.println("\n"+"Stream 5th Question output-no. of times word 'budget' arrived:");
61         newsList.stream().filter(l-> l.comment.equalsIgnoreCase("budget")).collect(Collectors.groupingBy(l -> l.comment,
62         Collectors.counting()))
63         .entrySet().stream().max(Map.Entry.comparingByValue())
64         .ifPresent(l-> System.out.println( l.getKey() + " arrived " + l.getValue() + " times"));
```

*StreamDemo.java

```
66
67          //6th Question
68          System.out.println("\n"+"Stream 6th Question output:User that posted max contents:");
69          newsList.stream().collect(Collectors.groupingBy(l->l.commentByUser, Collectors.counting()))
70            .entrySet()
71            .stream()
72            .max(Map.Entry.comparingByValue())
73            .ifPresent(l-> System.out.println("User Id : "+ l.getKey() + " has did the maximum comment i.e. :" + l.getValue()));
74
75          //7th Question
76          System.out.println("\n"+"Stream 7th Question output:CommentByUser wise number of comments:");
77          newsList.stream().collect(Collectors.groupingBy(l->l.commentByUser, Collectors.counting()))
78          .entrySet().stream()
79          .forEach(l -> System.out.println(l));
80
81          // 8th Question
82          System.out.println("\n"+"Stream 8th Question output Transactions in year 2011:");
83          transactionList.stream().filter(l -> l.year == 2011).sorted(Comparator.comparingInt(l-> l.value))
84          .forEach(l -> System.out.println(l));
85
86          //9th
87          System.out.println("\n"+"Stream 9th Question output-unique cities:");
88          traderList.stream().map(l-> l.city.toLowerCase()).distinct().forEach(l -> System.out.println(l));
89
90          // 10th Question
91          System.out.println("\n"+"Stream 10th Question output-traders from Pune:");
92          traderList.stream().filter(l -> l.city.equalsIgnoreCase("Pune")).sorted(Comparator.comparing(l -> l.name))
93          .forEach(l -> System.out.println(l));
94
95          //11th Question
96          System.out.println("\n"+"Stream 11th Question output-Returns string of  traders name:");
97          traderList.stream().sorted(Comparator.comparing(l -> l.name)).map(l -> l.name).forEach(System.out::println);
```

*StreamDemo.java

```
98
99          //12th Question
100         System.out.println("\n"+"Stream 12th Question output-Traders from Indore:");
101         traderList.stream().filter(Trader -> Trader.city == "Indore").forEach(Trader -> System.out.println(Trader.name));
102
103         //13th Question
104         System.out.println("\n"+"Stream 13th Question output-transaction values from traders in Delhi:");
105         transactionList.stream().filter(l-> l.trader.city.equalsIgnoreCase("Delhi")).forEach(System.out::println);
106
107         //14th question
108         System.out.println("\n"+"Stream 14th Question output-highest value of all the transactions:");
109         transactionList.stream().max(Comparator.comparingInt(l-> l.value)).ifPresent(System.out::println);
110
111         //15 Question
112         System.out.println("\n"+"Stream 15th Question output-smallest value of all the transactions:");
113         transactionList.stream().min(Comparator.comparingInt(l-> l.value)).ifPresent(System.out::println);
114 }
115 }
116 class Fruit
117 {
118     String name;
119     int calories;
120     int price;
121     String color;
122     public Fruit(String name, int calories, int price, String color)
123     {
124         super();
125         this.name = name;
126         this.calories = calories;
127         this.price = price;
128         this.color = color;
129     }
```

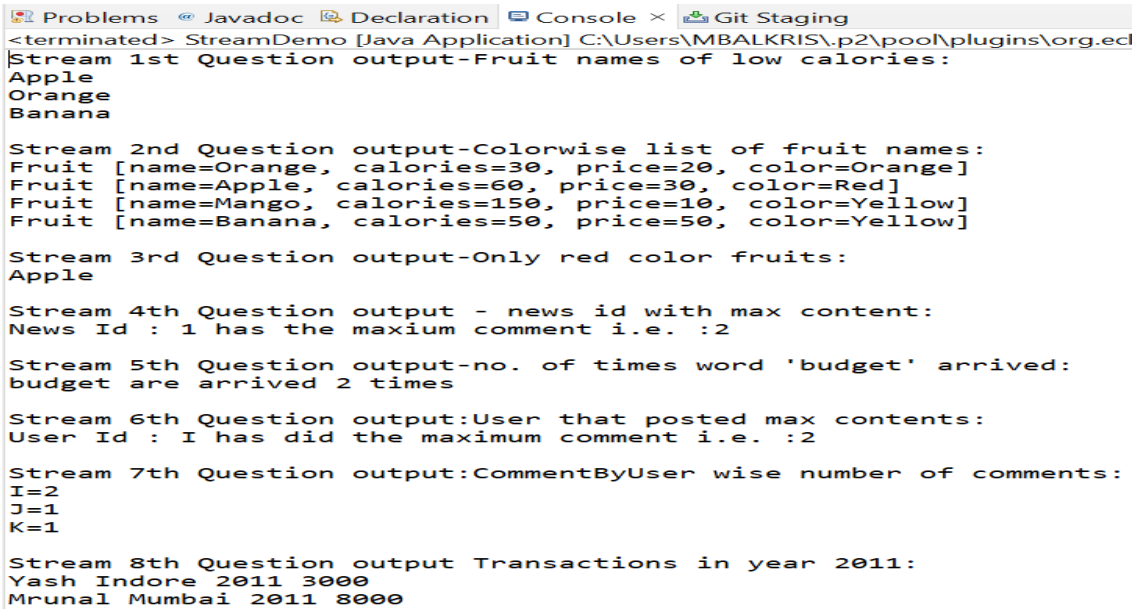
*StreamDemo.java

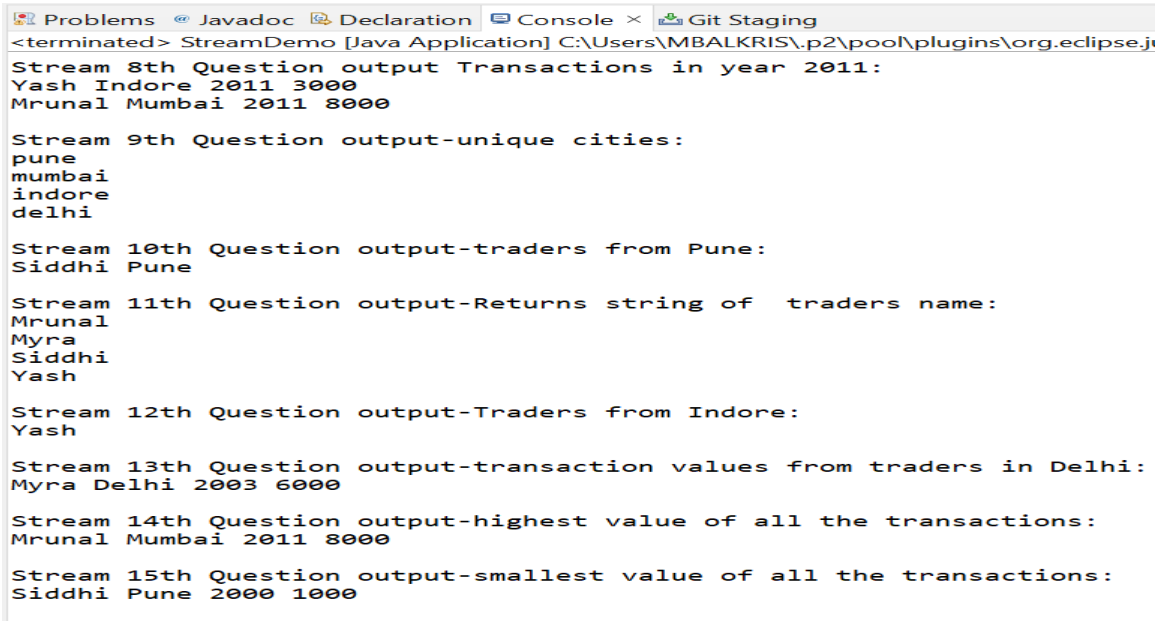
```
130     @Override
131     public String toString()
132     {
133         return "Fruit [name=" + name + ", calories=" + calories + ", price=" + price + ", color=" + color + "]";
134     }
135 }
136
137 class News
138 {
139     int newsId;
140     String postedByUser;
141     String commentByUser;
142     String comment;
143     public News(int newsId, String postedByUser, String commentByUser, String comment)
144     {
145         super();
146         this.newsId = newsId;
147         this.postedByUser = postedByUser;
148         this.commentByUser = commentByUser;
149         this.comment = comment;
150     }
151     @Override
152     public String toString()
153     {
154         return "News[newsId="+newsId+",postedByUser="+postedByUser+", commentByUser="+commentByUser+",comment="+comment+"]";
155     }
156 }
157
158 class Trader
159 {
160     String name;
161     String city;
```

*StreamDemo.java

```
162     public Trader(String name, String city)
163     {
164         super();
165         this.name = name;
166         this.city = city;
167     }
168     @Override
169     public String toString()
170     {
171
172         return name+" "+ city;
173     }
174 }
175
176 class Transaction
177 {
178     Trader trader;
179     int year;
180     int value;
181     public Transaction(Trader trader, int year, int value)
182     {
183         super();
184         this.trader = trader;
185         this.year = year;
186         this.value = value;
187     }
188     @Override
189     public String toString()
190     {
191         return trader +" "+year+ " " +value ;
192     }
193 }
```

Output: -


Problems
Javadoc
Declaration
Console
Git Staging
<terminated> StreamDemo [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.ecl
Stream 1st Question output-Fruit names of low calories:
Apple
Orange
Banana
Stream 2nd Question output-Colorwise list of fruit names:
Fruit [name=Orange, calories=30, price=20, color=Orange]
Fruit [name=Apple, calories=60, price=30, color=Red]
Fruit [name=Mango, calories=150, price=10, color=Yellow]
Fruit [name=Banana, calories=50, price=50, color=Yellow]
Stream 3rd Question output-Only red color fruits:
Apple
Stream 4th Question output - news id with max content:
News Id : 1 has the maxium comment i.e. :2
Stream 5th Question output-no. of times word 'budget' arrived:
budget are arrived 2 times
Stream 6th Question output:User that posted max contents:
User Id : I has did the maximum comment i.e. :2
Stream 7th Question output:CommentByUser wise number of comments:
I=2
J=1
K=1
Stream 8th Question output Transactions in year 2011:
Yash Indore 2011 3000
Mrunal Mumbai 2011 8000


Problems
Javadoc
Declaration
Console
Git Staging
<terminated> StreamDemo [Java Application] C:\Users\MBALKRIS\.p2\pool\plugins\org.eclipse.j
Stream 8th Question output Transactions in year 2011:
Yash Indore 2011 3000
Mrunal Mumbai 2011 8000
Stream 9th Question output-unique cities:
pune
mumbai
indore
delhi
Stream 10th Question output-traders from Pune:
Siddhi Pune
Stream 11th Question output-Returns string of traders name:
Mrunal
Myra
Siddhi
Yash
Stream 12th Question output-Traders from Indore:
Yash
Stream 13th Question output-transaction values from traders in Delhi:
Myra Delhi 2003 6000
Stream 14th Question output-highest value of all the transactions:
Mrunal Mumbai 2011 8000
Stream 15th Question output-smallest value of all the transactions:
Siddhi Pune 2000 1000