**Stream Assignment**

Q1

Create the following classes:

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

**Display the following:**

**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.**

**Code Specifications:**

**c**lass Fruit {  
    private String name;  
    private int calories;  
    private int price;  
    private String color;  
}  
  
public class Assignment5Q1 {

    public static List<String> reverseSort(ArrayList<Fruit> fruits) {}  
    public static ArrayList<Fruits> sort(ArrayList<Fruits> Fruits) {}

    public static ArrayList<Fruit> filterRedSortPrice(ArrayList<Fruit>

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

}

**Code:**

package StreamAssignment;

import java.util.\*;

import java.util.stream.Collectors;

class Fruit {

private String name;

private int calories;

private int price;

private String color;

public Fruit(String name, int calories, int 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 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 getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

*@Override*

public String toString() {

return "Fruit{" +

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

", calories=" + calories +

", price=" + price +

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

'}';

}

}

public class Assignment5Q1 {

public static List<String> reverseSort(ArrayList<Fruit> fruits) {

//System.out.println(fruits);

ArrayList<Fruit> fruitName = fruits.stream()

.filter(p -> p.getCalories() < 100)

.sorted(Comparator.*comparingInt*(Fruit::getCalories).reversed())

.collect(Collectors.*toCollection*(ArrayList::new));

ArrayList<String> fruitName1 = new ArrayList<>();

for(Fruit fruit:fruitName){

fruitName1.add(fruit.getName());

}

System.***out***.println(fruitName1);

return fruitName1;

}

public static ArrayList<Fruit> sort(ArrayList<Fruit> Fruits) {

Fruits=Fruits.stream()

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

.collect(Collectors.*toCollection*(ArrayList::new));

return Fruits;

}

public static ArrayList<Fruit> filterRedSortPrice(ArrayList<Fruit> fruits){

fruits=fruits.stream()

.filter(p->p.getColor().equals("Red"))

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

.collect(Collectors.*toCollection*(ArrayList::new));

return fruits;

}

public static void main(String[] args) {

ArrayList<Fruit>fruits = new ArrayList<>();

fruits.add( new Fruit("mango",70,50,"Yellow"));

fruits.add(new Fruit("Apple",80,80,"Red"));

fruits.add(new Fruit("Banana",110,40,"Yellow"));

fruits.add( new Fruit("Pear",100,30,"Yellow"));

fruits.add(new Fruit("Grapes",60,50,"Green"));

*reverseSort*( fruits);

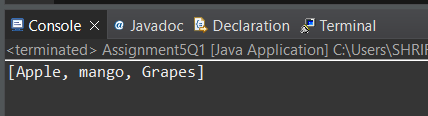
*sort*(fruits);

*filterRedSortPrice*(fruits);

}

}

**Output:**



Q2

Create the following classes:

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

**Find Out:**

**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.**

**3. Find out which user has posted maximum comments.**

**4. Display commentByUser wise number of comments.**

**Code Specifications:**

class News {  
    private int newsId;  
    private String postedByUser;  
    private String commentByUser;  
    private String comment;  
}  
  
public class Assignment5Q2 {  
    public static int maxComments(List<News> news) {}

    public static int budgetCount (List < News > news) {}  
    public static String maxCommentsByUser (List < News > news) {}

    public static Map<String, Integer> sortMaxCommentsByUser (List < News > news) {}

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

**Code:**

package StreamAssignment;

import java.util.\*;

import java.util.function.Function;

import java.util.stream.Collectors;

import java.util.stream.Stream;

class News {

private int newsId;

private String postedByUser;

private String commentByUser;

private String comment;

public News(int newsId, String postedByUser, String commentByUser, String comment) {

this.newsId = newsId;

this.postedByUser = postedByUser;

this.commentByUser = commentByUser;

this.comment = comment;

}

*@Override*

public String toString() {

return "News{" +

"newsId=" + newsId +

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

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

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

'}';

}

public int getNewsId() {

return newsId;

}

public String getPostedByUser() {

return postedByUser;

}

public String getCommentByUser() {

return commentByUser;

}

public String getComment() {

return comment;

}

}

public class Assignment5Q2 {

public static int maxComments(List<News> news) {

List<News>news1 = news;

List<News> list=new ArrayList<>();

int mx=0;

list= news.stream()

.filter(news2->news1.stream()

.anyMatch(one->one.getNewsId()==news2.getNewsId() && one!=news2)).collect(Collectors.*toCollection*(ArrayList::new));

Map<Integer,Integer>mp=new HashMap<>();

for(News news2:list){

if(mp.containsKey(news2.getNewsId())){

mp.put(news2.getNewsId(),mp.get(news2.getNewsId())+1);

}

else{

mp.put(news2.getNewsId(),1);

}

}

Set s1 = mp.entrySet();

Iterator itr = s1.iterator();

int id=0;

while(itr.hasNext()){

Map.Entry m1 = (Map.Entry)itr.next();

System.***out***.println(m1.getKey()+" "+m1.getValue());

if(mx<(int)m1.getValue()) {

mx = (int) m1.getValue();

id = (int)m1.getKey();

}

}

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

return id;

}

public static int budgetCount (List < News > news) {

String budget="budget";

long budgtC=news.stream()

.filter(p->p.getComment().contains(budget)).count();

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

return (int)budgtC;

}

public static String maxCommentsByUser (List < News > news) {

List<News>news1 = news;

List<News> list=new ArrayList<>();

list= news.stream()

.filter(news2->news1.stream()

.anyMatch(one->one.getCommentByUser()==news2.getCommentByUser() && one!=news2)).collect(Collectors.*toCollection*(ArrayList::new));

Map<String,Integer>mp=new HashMap<>();

for(News news2:list){

if(mp.containsKey(news2.getCommentByUser())){

mp.put(news2.getCommentByUser(),mp.get(news2.getCommentByUser())+1);

}

else{

mp.put(news2.getCommentByUser(),1);

}

}

Set s1 = mp.entrySet();

Iterator itr = s1.iterator();

String id="";

int mx=0;

while(itr.hasNext()){

Map.Entry m1 = (Map.Entry)itr.next();

System.***out***.println(m1.getKey()+" "+m1.getValue());

if(mx<(int)m1.getValue()) {

mx = (int) m1.getValue();

id = (String)m1.getKey();

}

}

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

return id;

}

public static Map<String, Integer> sortMaxCommentsByUser (List < News > news) {

Map<String,Long>mp1=new HashMap<>();

mp1=news.stream().collect(Collectors.*groupingBy*(News::getCommentByUser,Collectors.*counting*()));

List<Map.Entry<String, Long> > list =

new LinkedList<Map.Entry<String, Long> >(mp1.entrySet());

// Sort the list

Collections.*sort*(list, new Comparator<Map.Entry<String, Long> >() {

public int compare(Map.Entry<String, Long> o1,

Map.Entry<String, Long> o2)

{

return -(o1.getValue()).compareTo(o2.getValue());

}

});

HashMap<String, Integer> temp = new LinkedHashMap<String, Integer>();

for (Map.Entry<String, Long> aa : list) {

temp.put(aa.getKey(), (int) (long)aa.getValue());

}

return temp;

}

public static void main(String[] args) {

News news1 = new News(1, "Ajay", "Vijay", "comment");

News news2 =new News(2, "Ajay1", "Vijay1", "budget");

News news3 = new News(1, "Ajay2", "Vijay2", "comment");

News news4 = new News(3, "Ajay3", "Vijay", "budget");

News news5 = new News(1, "Ajay4", "Vijay4", "comment");

News news6 = new News(3, "Ajay5", "Vijay5", "budget");

News news7 = new News(1, "Ajay6", "Vijay6" ,"comment");

List<News> news = new ArrayList<>();

news.add(news1);

news.add(news2);

news.add(news3);

news.add(news4);

news.add(news5);

news.add(news6);

news.add(news7);

*maxComments*(news);

*budgetCount*(news);

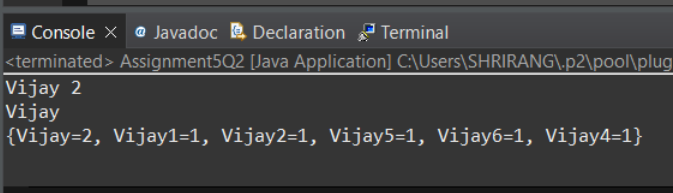
*maxCommentsByUser*(news);

System.***out***.println( *sortMaxCommentsByUser*(news));

}

}

**Output:**



Q3

Create the following classes:

class Trader { String name; String city; }

**Find Out:**

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

**2. Find all traders from Pune and sort them by name.**

**3. Return a string of all traders’ names sorted alphabetically.**

**4. Are any traders based in Indore?**

**Code Specifications:**

class Trader {  
    private String name;  
    private String city;  
}  
  
public class Assignment5Q9 {  
    public static List<String> printUniqueCities (List <Trader> traders) {  }

    public static List<String> tradersFromPuneSortByName(List<Trader> traders) {}  
    public static String allTrader3Names(List<Trader> traders) {}  
    public static ArrayList<Trader> areAnyTradersFromIndore(ArrayList<Trader> traders) {}

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

**Code:**

package StreamAssignment;

import java.util.\*;

import java.util.stream.Collector;

import java.util.stream.Collectors;

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 + '\'' +

'}';

}

}

public class Assignment5Q3 {

public static List<String> printUniqueCities (List<Trader> traders) {

Set<String> list = new TreeSet<>();

List <Trader> uniqueList = new ArrayList<>();

uniqueList= traders.stream()

.filter(p-> list.add(p.getCity()))

.distinct().collect(Collectors.*toCollection*(ArrayList::new));

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

System.***out***.println("\_\_\_\_\_");

List<String> list1 = new ArrayList<>();

for(Trader trader:uniqueList){

list1.add(trader.getCity());

}

return list1;

}

public static List<String> trader2sFromPuneSortByName(List<Trader> traders) {

List<String> uniqueList = new ArrayList<>();

traders.stream()

.filter(p-> p.getCity().equals("Pune"))

.sorted(Comparator.*comparing*(Trader::getName))

.collect(Collectors.*toList*())

.forEach(p->uniqueList.add(p.getName()));

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

return uniqueList;

}

public static String allTrader3Names(List<Trader> traders) {

String allTrader="";

ArrayList<Trader> arrayList = new ArrayList<>();

arrayList=traders.stream()

.sorted(Comparator.*comparing*(Trader::getName))

.collect(Collectors.*toCollection*(ArrayList::new));

for(Trader trader:arrayList){

allTrader+=trader.getName();

}

return allTrader;

}

public static ArrayList<Trader> areAnyTrader4sFromIndore(ArrayList<Trader> traders) {

ArrayList<Trader>traders1 = traders;

ArrayList<Trader> traders2 = new ArrayList<>();

traders2=traders1.stream()

.filter(p->p.getCity().equals("Indore"))

.collect(Collectors.*toCollection*(ArrayList::new));

return traders2;

}

public static void main(String[] args) {

Trader t1 = new Trader("name1", "Bangalore");

Trader t2 = new Trader("name2", "Pune");

Trader t3 = new Trader("name3", "Bangalore");

Trader t4 = new Trader("name4", "Delhi");

Trader t5= new Trader("name5", "Chennai");

Trader t6 = new Trader("name6", "Pune");

Trader t7 = new Trader("name7", "Delhi");

Trader t8 = new Trader("name8", "Delhi");

Trader t9 = new Trader("name9", "Pune");

List<Trader> traders = new ArrayList<>();

traders.add(t1);

traders.add(t2);

traders.add(t3);

traders.add(t4);

traders.add(t5);

traders.add(t6);

traders.add(t7);

traders.add(t8);

traders.add(t9);

ArrayList<Trader> traders1 = new ArrayList<>();

traders1.add(t1);

traders1.add(t2);

traders1.add(t3);

traders1.add(t4);

traders1.add(t5);

traders1.add(t6);

traders1.add(t7);

traders1.add(t8);

traders1.add(t9);

*printUniqueCities*(traders);

*trader2sFromPuneSortByName*(traders);

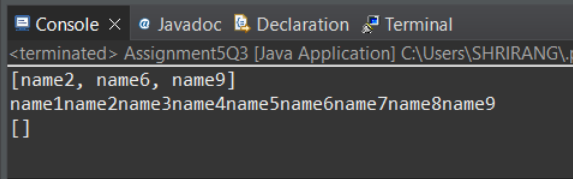
System.***out***.println(*allTrader3Names*(traders));

System.***out***.println(*areAnyTrader4sFromIndore*(traders1));

}

}

**Output:**



Q4

Create the following classes:

class Trader { String name; String city; }

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

**1. Find all transactions in the year 2011 and sort them by value (small to high).**

**2. Print all transactions’ values from the traders living in Delhi.**

**3. What’s the highest value of all the transactions?**

**4. Find the transaction with the smallest value.**

**Code Specifications:**

class Trader {  
    private String name;  
    private String city;  
}  
  
class Transaction {  
    private Trader trader;  
    private int year;  
    private int value;  
}  
public class Assignment5Q4 {  
    public static List<Transaction> sortTransactions(List <Transaction> transactions) {}

    public static List<Integer> transactionsValuesDelhi(List<Transaction> transactions) {}  
    public static int highestTransaction(List<Transaction> transactions){}

    public static int smallestTransaction(List<Transaction> transactions){}

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

**Code:**

package StreamAssignment;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import java.util.stream.Collectors;

class Transaction {

private Trader trader;

private int year;

private int value;

public Transaction(Trader trader, int year, int value) {

this.trader = trader;

this.year = year;

this.value = value;

}

public Trader getTrader() {

return trader;

}

public int getYear() {

return year;

}

public int getValue() {

return value;

}

*@Override*

public String toString() {

return "Transaction{" +

"trader=" + trader +

", year=" + year +

", value=" + value +

'}';

}

}

public class Assignment5Q4 {

public static List<Transaction> sortTransactions(List <Transaction> transactions) {

List<Transaction> sortedTrans= new ArrayList<>();

sortedTrans=transactions.stream()

.filter(p->p.getYear()==2011)

.sorted(Comparator.*comparingInt*(Transaction::getValue))

.collect(Collectors.*toCollection*(ArrayList::new));

return sortedTrans;

}

public static List<Integer> transaction3sValuesDelhi(List<Transaction> transactions) {

List<Integer> transactioValueDelhi = new ArrayList<>();

transactions.stream()

.filter(p->p.getTrader().getCity().equals("Delhi"))

.forEach(p->transactioValueDelhi.add(p.getValue()));

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

return transactioValueDelhi;

}

public static int highestTransaction2(List<Transaction> transactions){

int mx=Integer.***MIN\_VALUE***;

List<Integer>list = new ArrayList<>();

transactions.stream()

.forEach(p->list.add(p.getValue()));

mx =Collections.*max*(list);

return mx;

}

public static int smallestTransaction1(List<Transaction> transactions){

int mn=Integer.***MAX\_VALUE***;

List<Integer>list = new ArrayList<>();

transactions.stream()

.forEach(p->list.add(p.getValue()));

mn =Collections.*min*(list);

return mn;

}

public static void main(String[] args) {

Trader t1 = new Trader("name1", "Bangalore");

Trader t2 = new Trader("name2", "Pune");

Trader t3 = new Trader("name3", "Bangalore");

Trader t4 = new Trader("name4", "Delhi");

Trader t5= new Trader("name5", "Chennai");

Trader t6 = new Trader("name6", "Pune");

Trader t7 = new Trader("name7", "Delhi");

Trader t8 = new Trader("name8", "Delhi");

Trader t9 = new Trader("name9", "Pune");

Transaction ts1 = new Transaction(t1,2011,300);

Transaction ts2 = new Transaction(t2,2011,200);

Transaction ts3 = new Transaction(t3,2012,100);

Transaction ts4 = new Transaction(t4,2011,400);

Transaction ts5 = new Transaction(t5,2011,400);

Transaction ts6 = new Transaction(t6,2010,300);

Transaction ts7 = new Transaction(t7,2013,200);

Transaction ts8 = new Transaction(t8,2012,150);

Transaction ts9 = new Transaction(t9,2009,400);

ArrayList<Transaction> transactions = new ArrayList<>();

transactions.add(ts1);

transactions.add(ts2);

transactions.add(ts3);

transactions.add(ts4);

transactions.add(ts5);

transactions.add(ts6);

transactions.add(ts7);

transactions.add(ts8);

transactions.add(ts9);

*sortTransactions*(transactions);

*transaction3sValuesDelhi*(transactions);

System.***out***.println(*highestTransaction2*(transactions));

}

}

**Output:**

