**Adisri Sarode**

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:

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

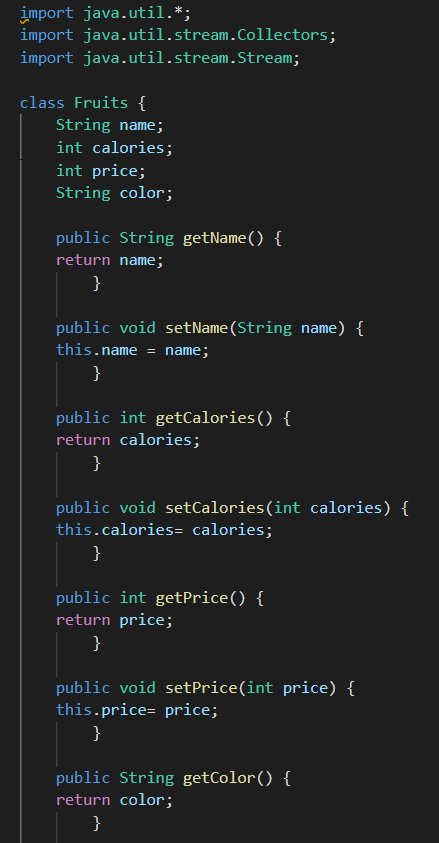
    public static List<String>reverseSort(ArrayList<Fruit> fruits) {}  
    public static ArrayList<Fruits> sort(ArrayList<Fruits> Fruits) {}

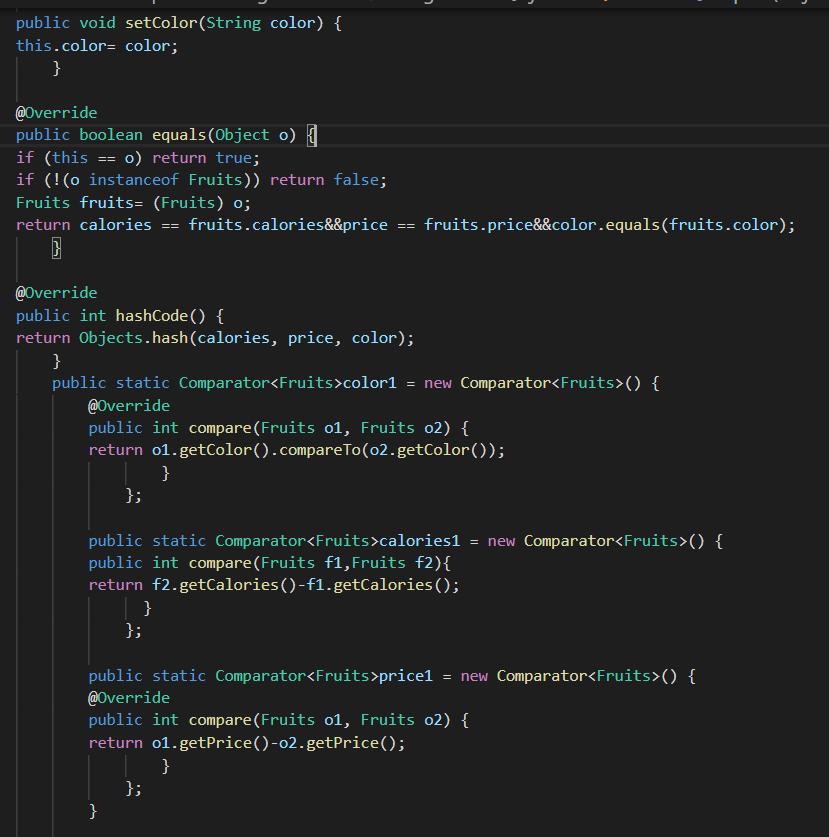
    public static ArrayList<Fruit>filterRedSortPrice(ArrayList<Fruit>

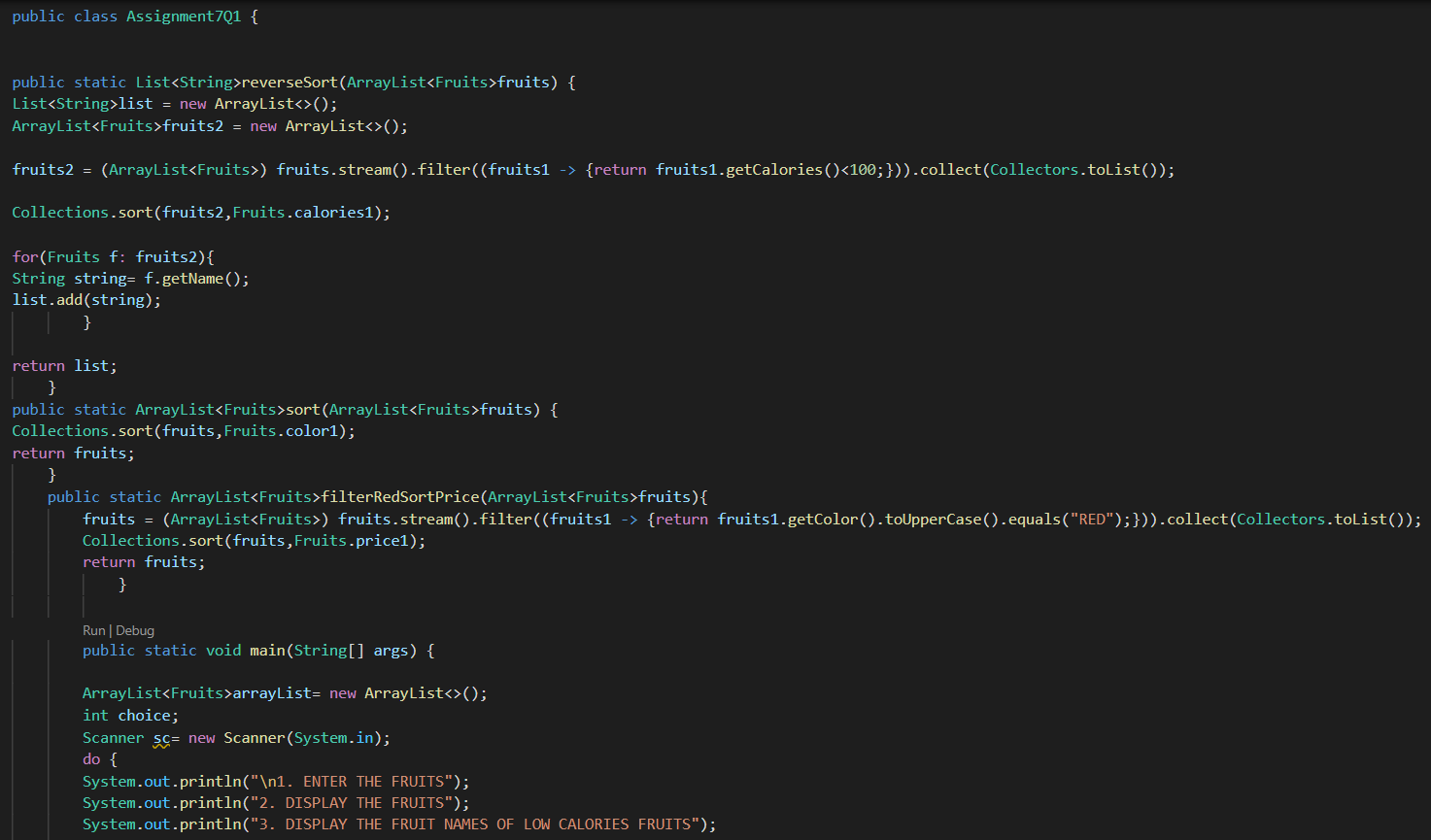
    public static void main(String[] args) {}

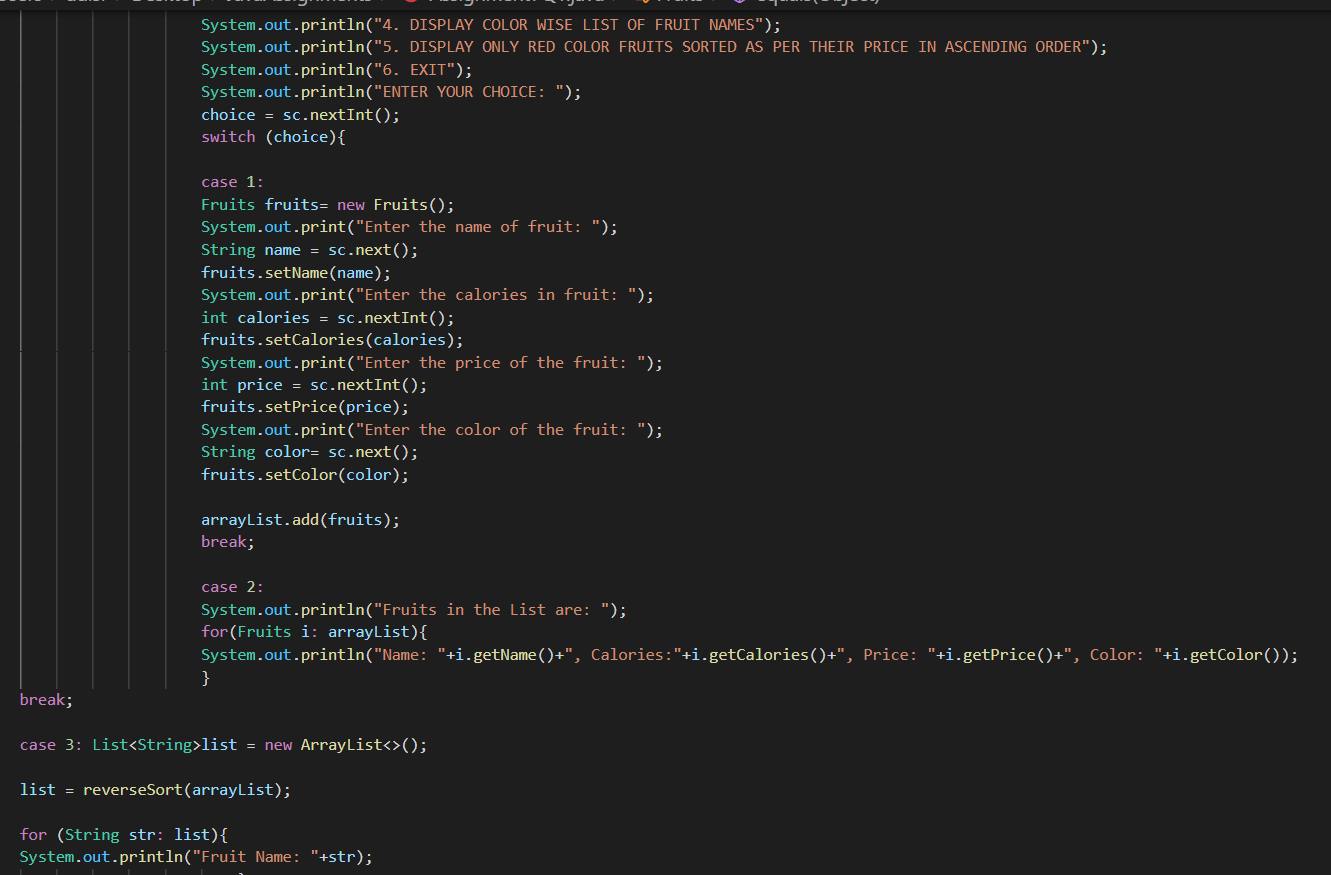
}

**Code**:



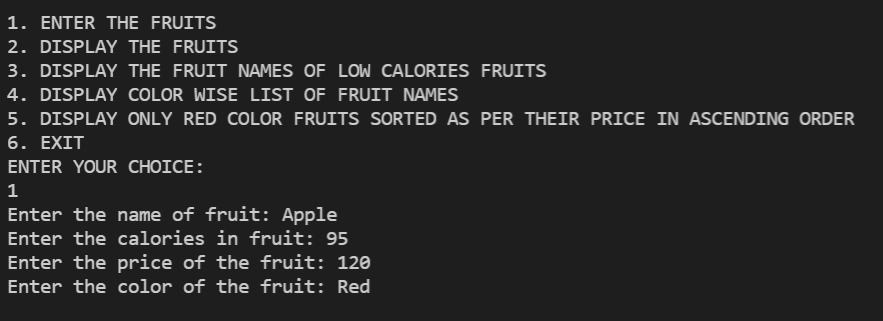


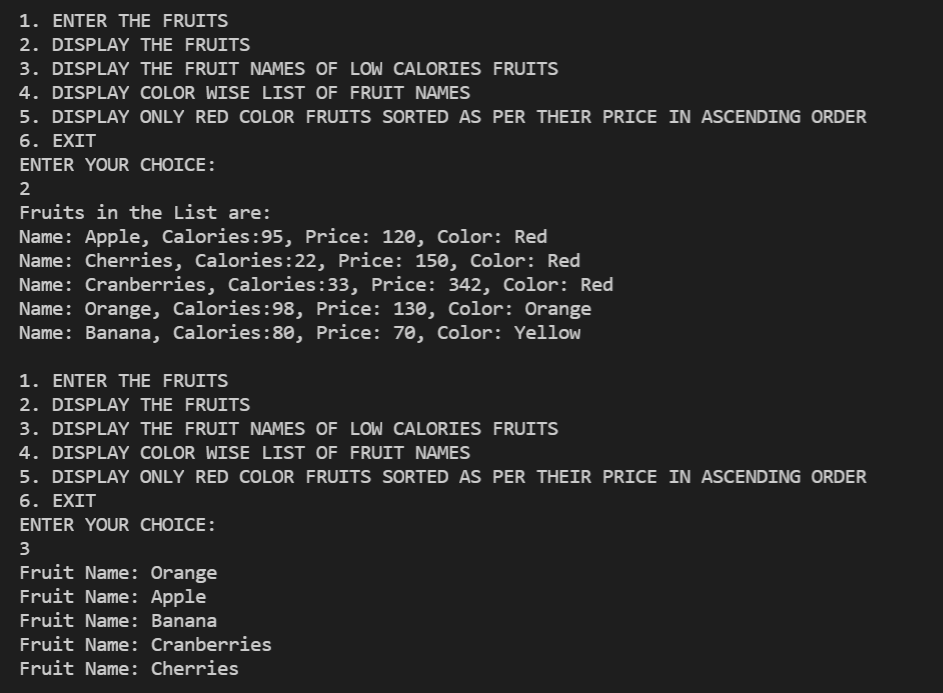


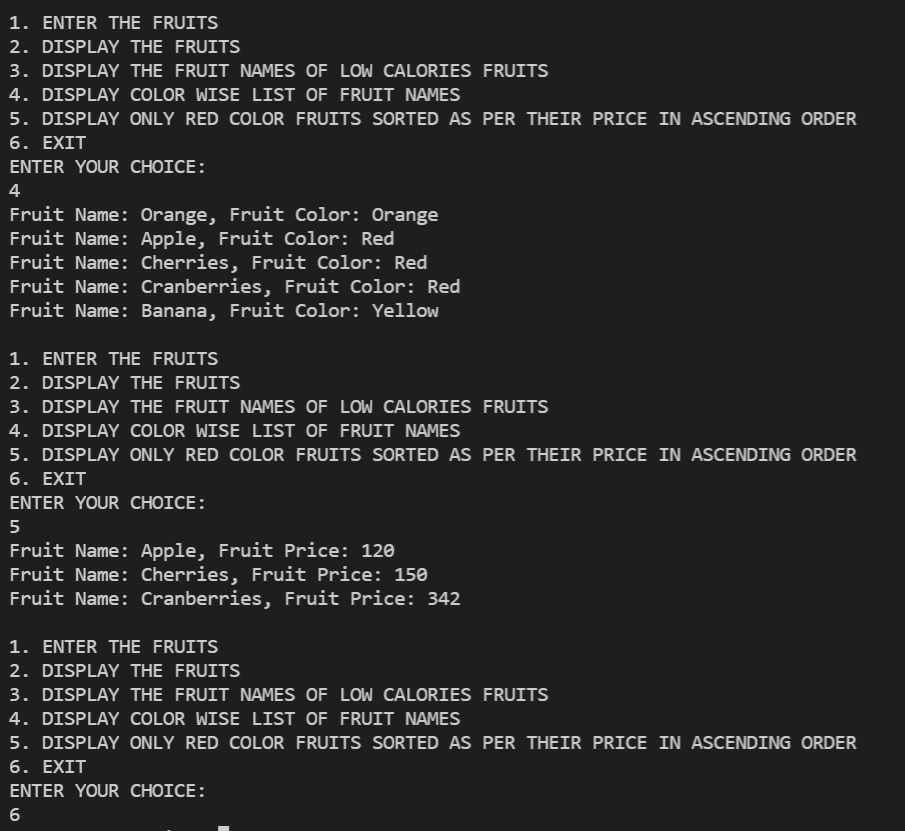




OUTPUT:







**Q2)Setup:**

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:

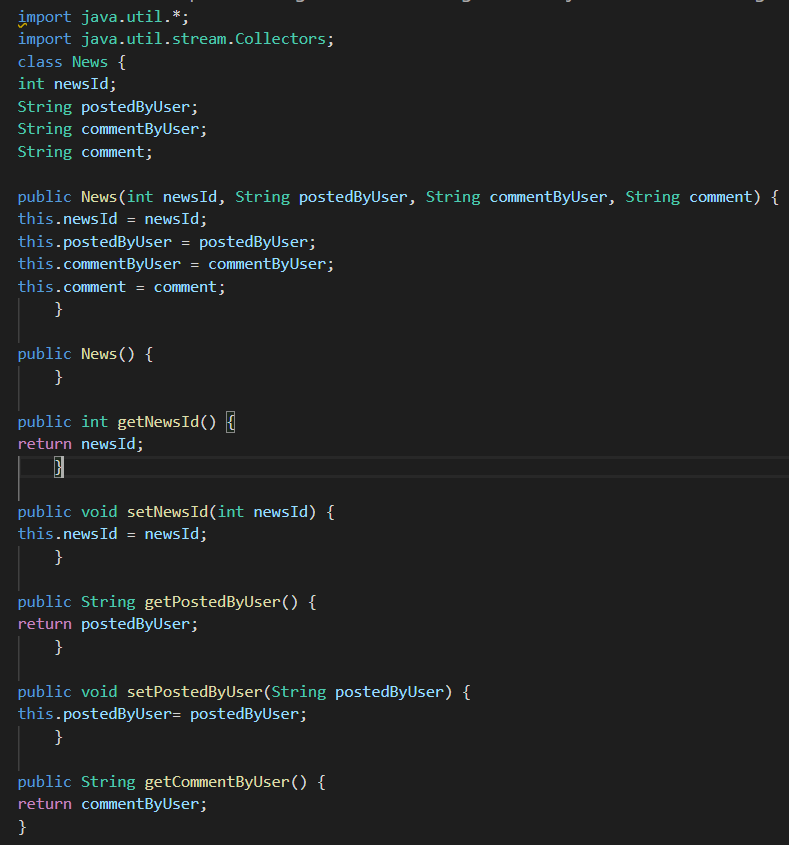
class News {  
    private int newsId;  
    private String postedByUser;  
    private String commentByUser;  
    private String comment;  
}  
  
public class Assignment5Q2 {  
    public static int maxComments(List<News> news) {}

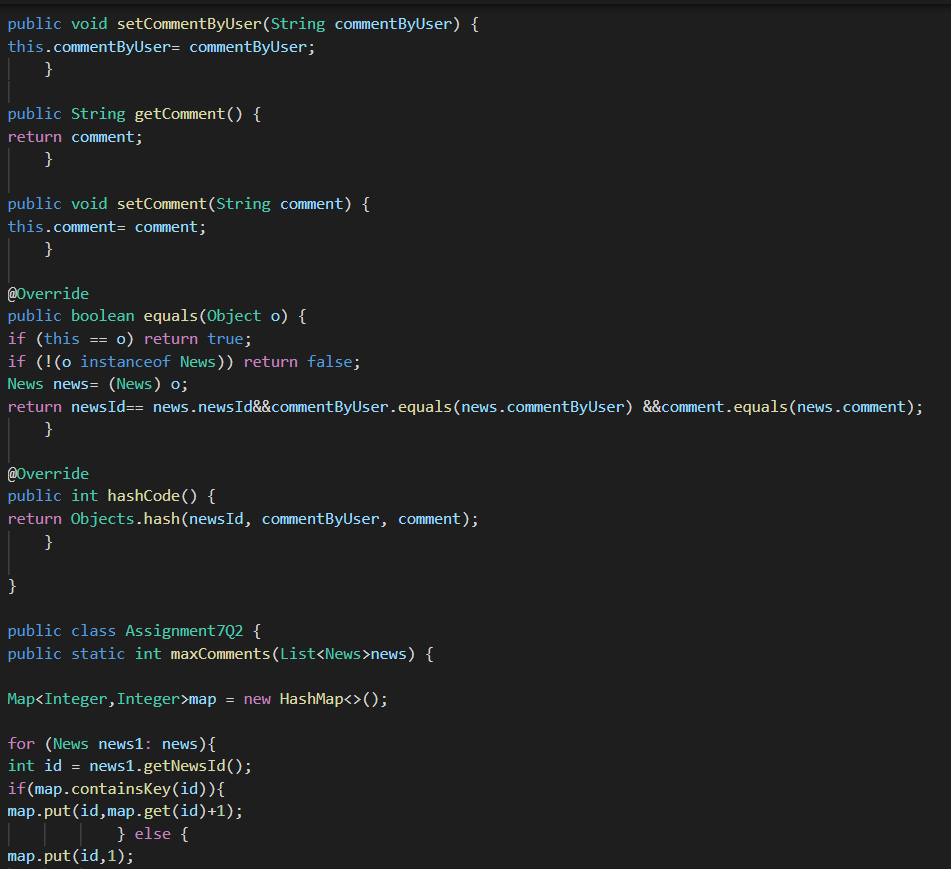
    public static int budgetCount (List < News > news) {}  
    public static String maxCommentsByUser (List < News > news) {}

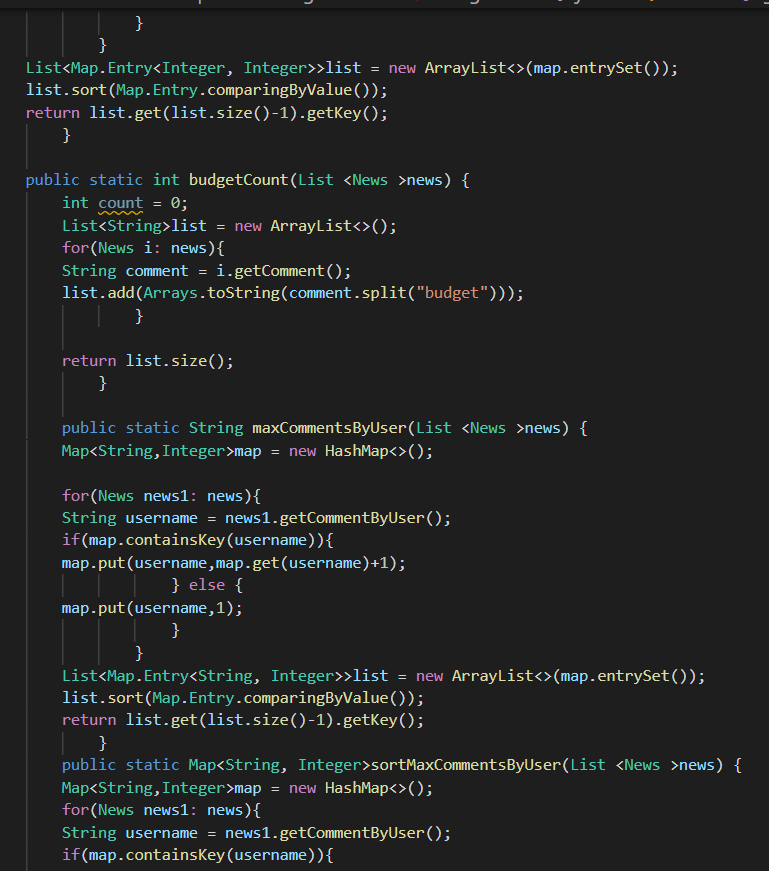
    public static Map<String, Integer>sortMaxCommentsByUser (List < News > news) {}

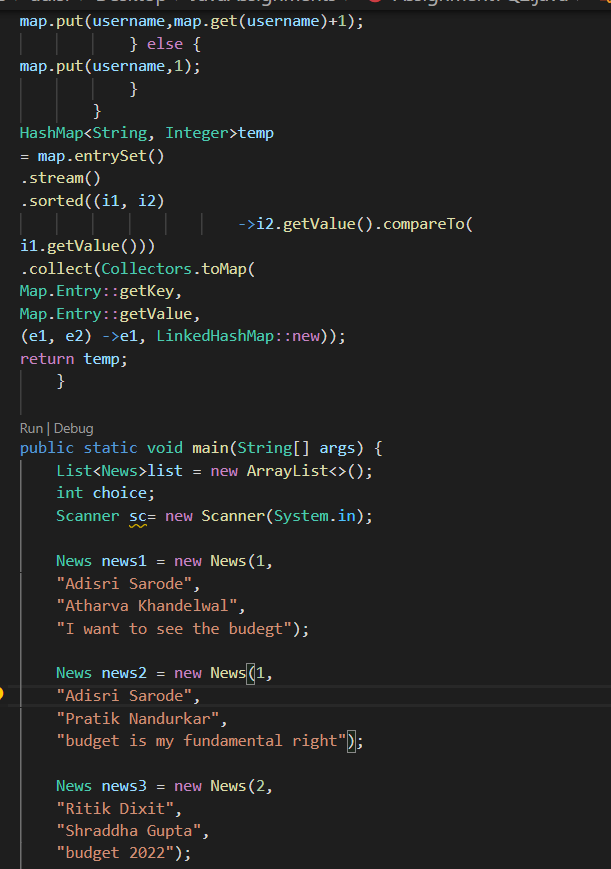
    public static void main(String[] args) {}  
}

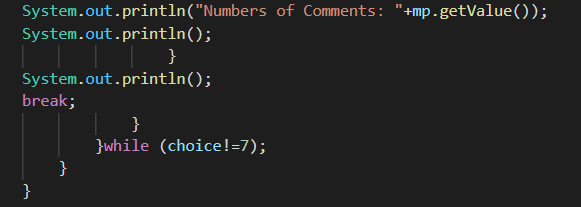
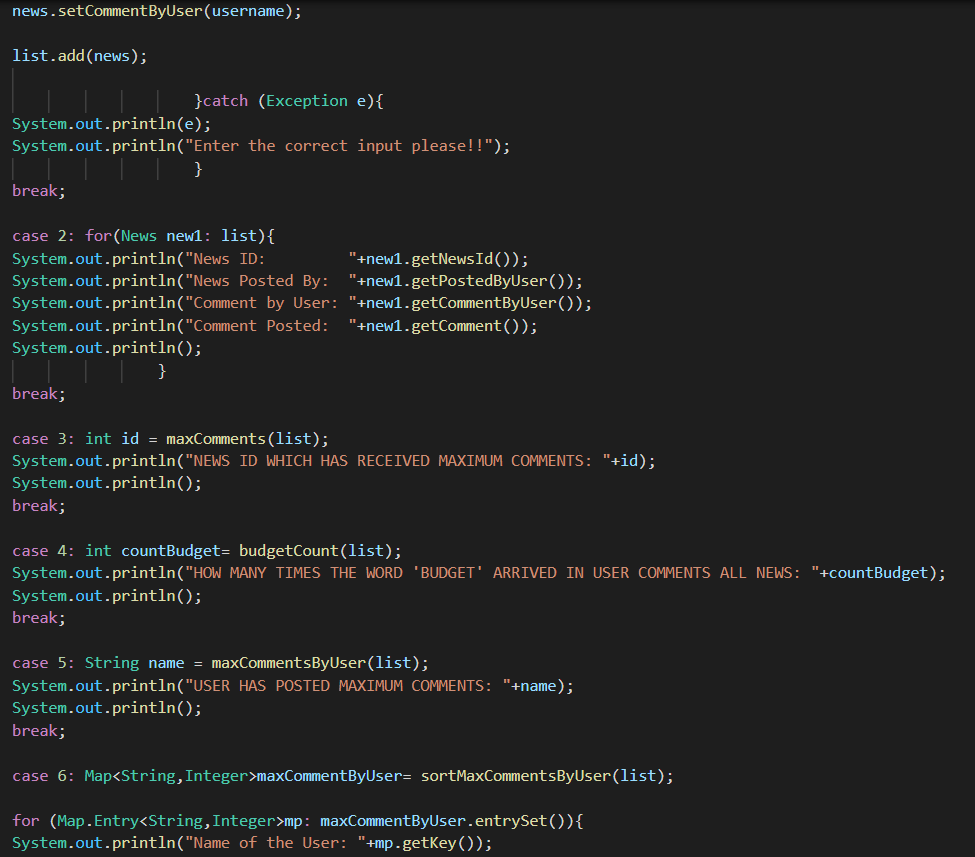
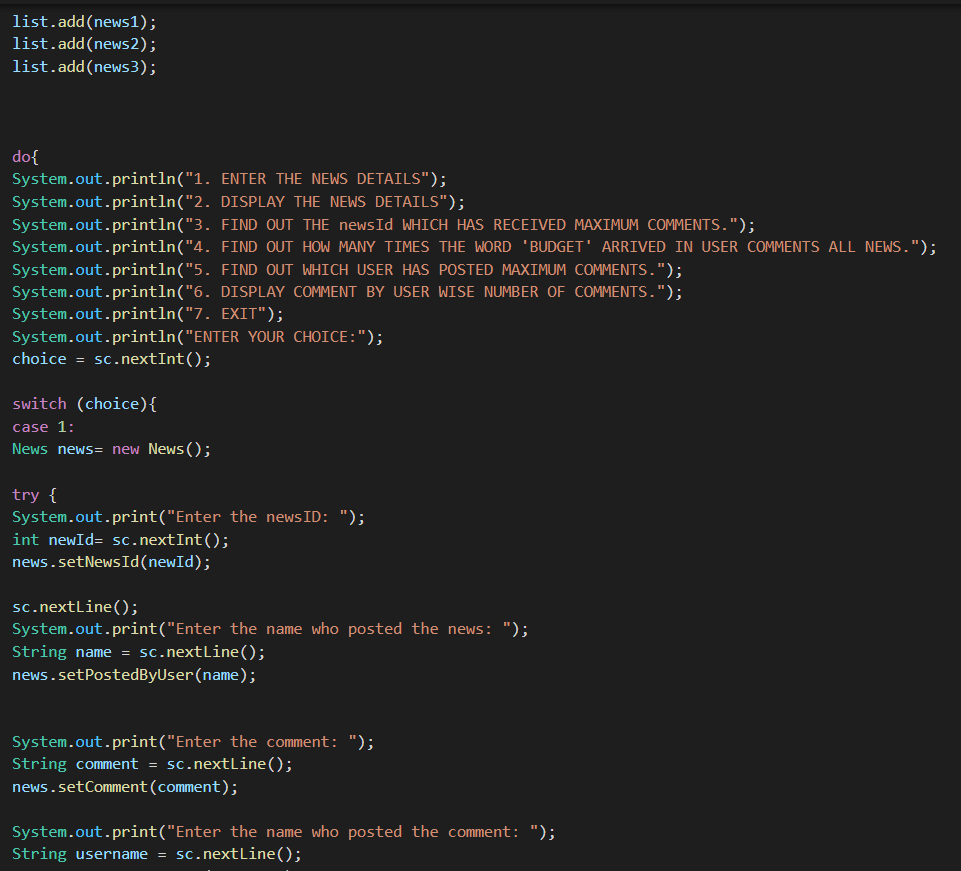
Code:



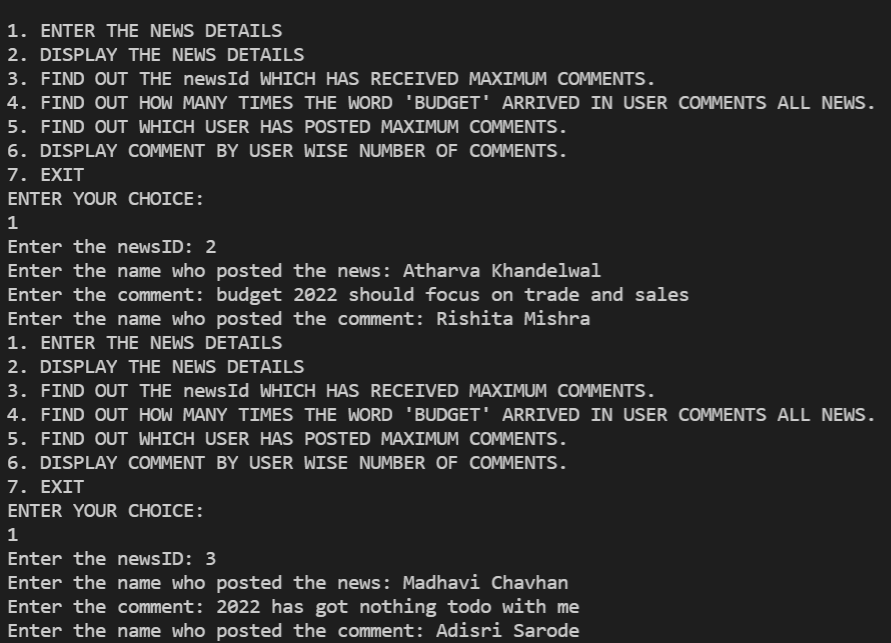


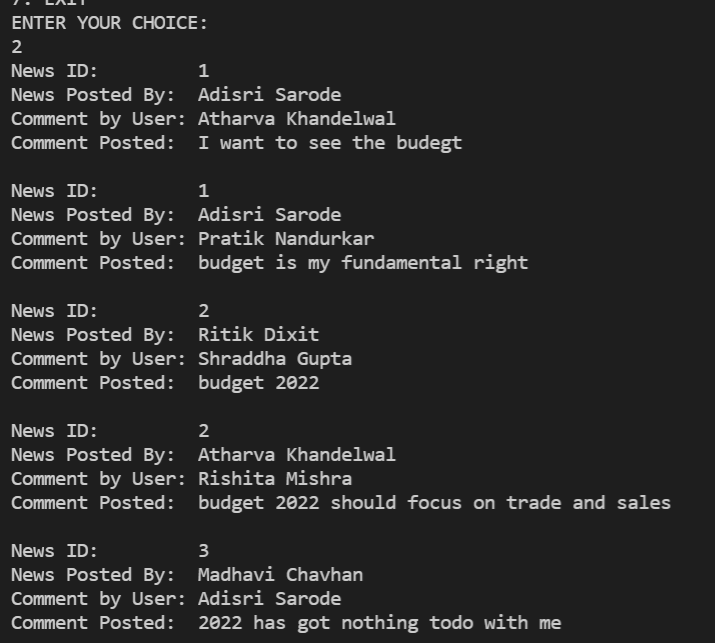


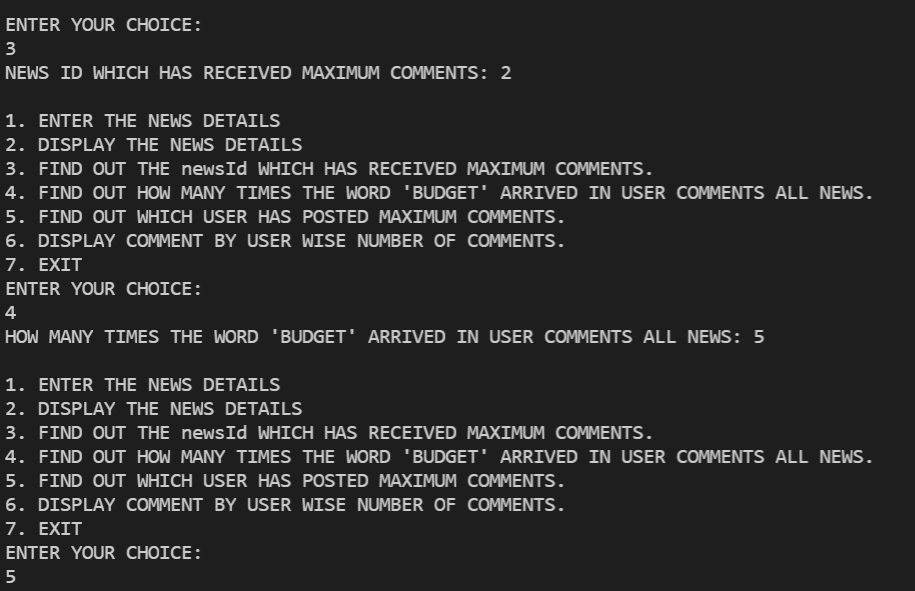


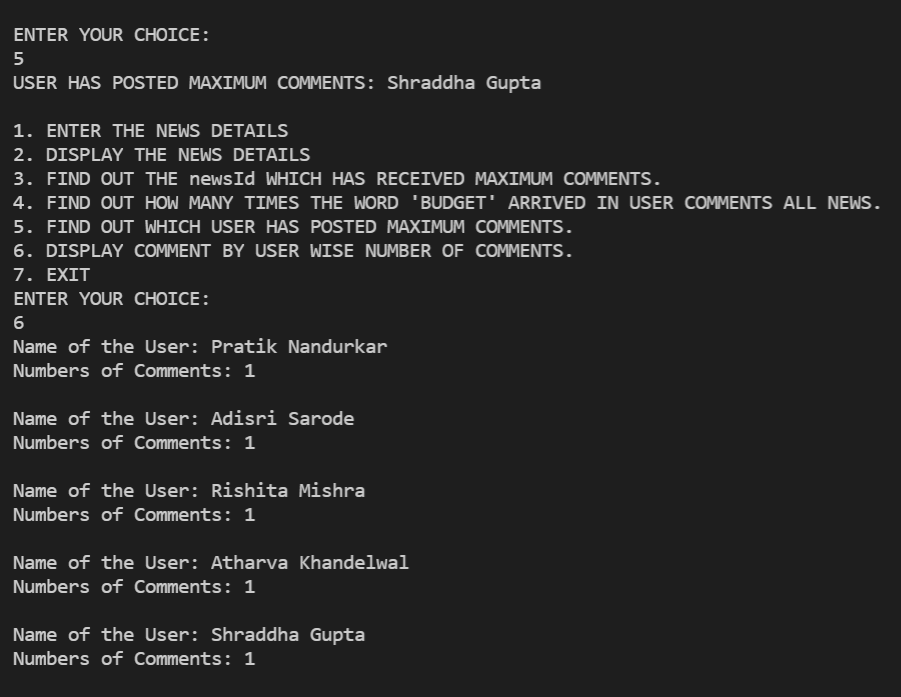


OUTPUT:









**Q3)Setup:**

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:

import java.util.\*;

class Trader {

String name;

String city;

public Trader(String name, String city) {

this.name = name;

this.city = city;

    }

public Trader(){

    }

    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 boolean equals(Object o) {

        if (this == o) return true;

        if (!(o instanceof Trader)) return false;

        Trader trader= (Trader) o;

        return name.equals(trader.name) &&city.equals(trader.city);

            }

        @Override

        public int hashCode() {

        return Objects.hash(name, city);

            }

        public static Comparator<Trader>comparator = new Comparator<Trader>() {

        @Override

        public int compare(Trader o1, Trader o2) {

        return o1.getName().compareTo(o2.getName());

                }

            };

        }

        public class Assignment7Q3 {

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

            Map<String,String>map = new HashMap<>();

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

            for (Trader trader: traders){

            map.putIfAbsent(trader.getCity(),trader.getName());

                    }

            for (Map.Entry<String,String>mp: map.entrySet()){

            cities.add(mp.getKey());

                    }

            return cities;

                }

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

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

            for (Trader trader: traders){

            if(trader.getCity().toUpperCase().equals("PUNE")){

            tradersFromPune.add(trader.getName());

                        }

                    }

            Collections.sort(tradersFromPune);

            return tradersFromPune;

                }

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

            traders.sort(Trader.comparator);

            return traders;

                }

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

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

            for(Trader trader: traders){

            if(trader.getCity().toUpperCase().equals("INDORE")){

            traderList.add(trader);

                        }

                    }

            return traderList;

                }

                public static void main(String[] args) {

                    Scanner sc= new Scanner(System.in);

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

                    Trader trader1 = new Trader("Ayush Agrawal","Pune");

                    Trader trader2 = new Trader("RitikDixit","Bangalore");

                    Trader trader3 = new Trader("Atharva Upadhye","Pune");

                    Trader trader4 = new Trader("Pratik Nandurkar","Pune");

                    Trader trader5 = new Trader("Shraddha Gupta","Indore");

                    Trader trader6 = new Trader("Akanksha Ramgirwar","Indore");

                    arrayList.add(trader1);

                    arrayList.add(trader2);

                    arrayList.add(trader3);

                    arrayList.add(trader4);

                    arrayList.add(trader5);

                    arrayList.add(trader6);

                    int choice;

                    do{

                    System.out.println("\n1. ENTER THE TRADERS DETAILS");

                    System.out.println("2. DISPLAY THE TRADERS DETAILS");

                    System.out.println("3. DISPLAY ALL UNIQUE CITIES WHERE TRADERS WORK");

                    System.out.println("4. DISPLAY ALL TRADERS FROM PUNE SORTED BY NAME");

                    System.out.println("5. DISPLAY ALL TRADERS NAMES SORTED ALPHABETICALLY");

                    System.out.println("6. DISPLAY ALL TRADERS FROM INDORE");

                    System.out.println("7. EXIT");

                    System.out.println("ENTER YOUR CHOICE");

                    choice = sc.nextInt();

                    switch (choice){

                    case 1:

                    Trader trader= new Trader();

                    sc.nextLine();

                    System.out.println("Enter the Trader Name: ");

                    String name = sc.nextLine();

                    trader.setName(name);

                    System.out.println("Enter the City Name: ");

                    String city = sc.nextLine();

                    trader.setCity(city);

                    arrayList.add(trader);

                    break;

                    case 2:

                    System.out.println("List of traders: ");

                    for (Trader trade: arrayList){

                    System.out.println("Trader Name: "+trade.getName());

                    System.out.println("Trader City: "+trade.getCity());

                                        }

                    System.out.println();

                    break;

                    case 3:

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

                    list = printUniqueCities(arrayList);

                    System.out.println("All uniques cities where traders work: ");

                    for (String str: list){

                    System.out.println(str);

                                        }

                    System.out.println();

                    break;

                    case 4:

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

                    list1 = tradersFromPuneSortByName(arrayList);

                    System.out.println("All traders from Pune: ");

                    for (String str: list1){

                    System.out.println(str);

                                        }

                    System.out.println();

                    break;

                    case 5: List<Trader>arraylist2 = new ArrayList<>();

                    arraylist2 = allTrader3Names(arrayList);

                    System.out.println("All Traders Name:");

                    for(Trader trader7: arraylist2){

                    System.out.println("Name: "+trader7.getName());

                    System.out.println("City: "+trader7.getCity());

                    System.out.println();

                                        }

                    System.out.println();

                    break;

                    case 6: List<Trader>traderList= new ArrayList<>();

                    traderList= areAnyTradersFromIndore(arrayList);

                    System.out.println("All traders from Indore: ");

                    for(Trader trader7: traderList){

                    System.out.println("Name: "+trader7.getName());

                                        }

                    System.out.println();

                    break;

                    case 7: System.exit(0);

                    default:

                    System.out.println("PLEASE ENTER THE RIGHT CHOICE!");

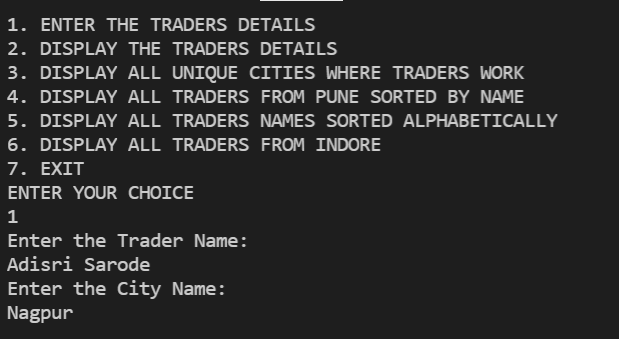
                                }

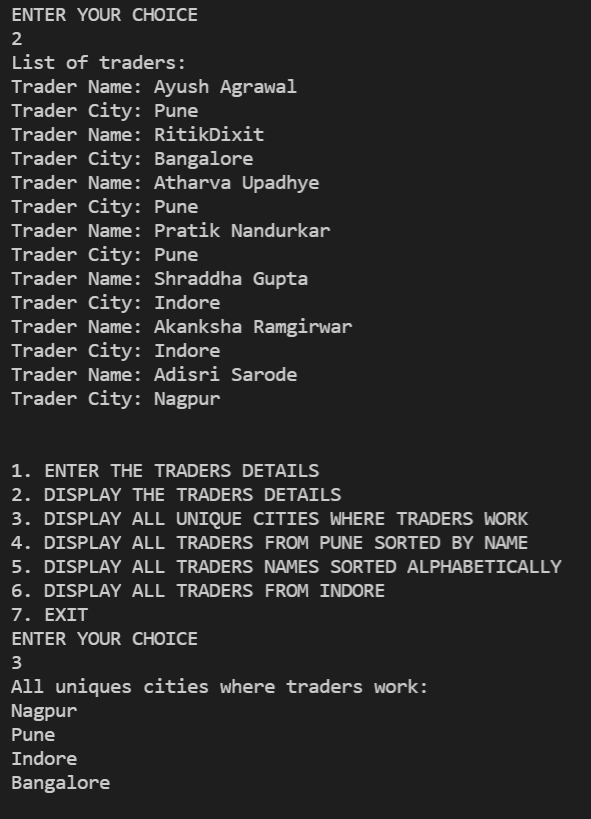
                            }while (choice!=7);

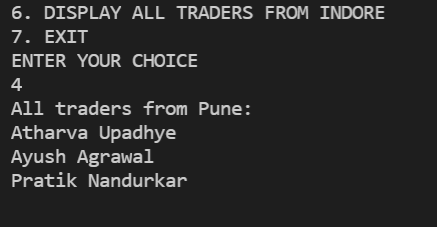
                        }

                    }

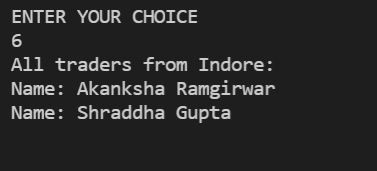
Output:











**Q4)Setup:**

Create the following classes:

class Trader { String name; String city; }

class Transaction { Tradertrader; 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:

import java.util.\*;

import java.util.stream.Collectors;

class Trader1 {

String name;

String city;

public Trader1(String name, String city) {

    this.name = name;

    this.city = city;

        }

    public Trader1(){

        }

    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 boolean equals(Object o) {

    if (this == o) return true;

    if (!(o instanceof Trader1)) return false;

    Trader1 trader1 = (Trader1) o;

    return name.equals(trader1.name) &&city.equals(trader1.city);

        }

    @Override

    public int hashCode() {

    return Objects.hash(name, city);

        }

    }

    class Transaction {

    Trader1 trader1;

    int year;

    int value;

    public Transaction(){}

    public Transaction(Trader1 trader1, int year, int value) {

    this.trader1 = trader1;

    this.year = year;

    this.value = value;

        }

    public Trader1 getTrader1() {

    return trader1;

        }

    public void setTrader1(Trader1 trader1) {

        this.trader1 = trader1;

    }

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 boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Transaction)) return false;

Transaction that = (Transaction) o;

return year == that.year &&value == that.value &&trader1.equals(that.trader1);

    }

@Override

public int hashCode() {

return Objects.hash(trader1, year, value);

    }

public static Comparator<Transaction>MaximumValue = new Comparator<Transaction>() {

@Override

public int compare(Transaction o1, Transaction o2) {

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

        }

    };

}

public class Assignment7Q4 {

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

transactions = transactions.stream().filter((t1)->t1.getYear()==2011).collect(Collectors.toList());

transactions.sort(Transaction.MaximumValue);

return transactions;

    }

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

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

transactions = transactions.stream().filter((t1)->t1.getTrader1().getCity().toUpperCase().equals("DELHI")).collect(Collectors.toList());

for(Transaction t1: transactions){

transactionValue.add(t1.getValue());

        }

return transactionValue;

    }

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

int MaxTransaction;

MaxTransaction = transactions.stream().mapToInt(Transaction::getValue).max().orElseThrow(NoSuchElementException::new);

return MaxTransaction;

    }

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

return transactions.stream().mapToInt(Transaction::getValue).min().orElseThrow(NoSuchElementException::new);

    }

public static void main(String[] args) {

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

Scanner sc= new Scanner(System.in);

int choice;

do{

System.out.println("1. ENTER THE DETAILS OF TRADERS & THEIR TRANSACTIONS");

System.out.println("2. DISPLAY THE DETAILS OF TRADERS & THEIR TRANSACTION");

System.out.println("3. DISPLAY ALL TRANSACTIONS IN THE YEAR 2011 IN SORTED ORDER");

System.out.println("4. DISPLAY ALL TRANSACTIONS FROM TRADERS WHO LIVES IN DELHI");

System.out.println("5. DISPLAY THE HIGHEST VALUE OF ALL THE TRANSACTIONS");

System.out.println("6. DISPLAY THE SMALLEST VALUE OF ALL THE TRANSACTIONS");

System.out.println("7. EXIT");

System.out.println("ENTER YOUR CHOICE");

choice = sc.nextInt();

switch (choice){

case 1:

Transaction transaction = new Transaction();

Trader1 trader1= new Trader1();

sc.nextLine();

System.out.print("ENTER THE NAME OF THE TRADER: ");

String name = sc.nextLine();

trader1.setName(name);

System.out.print("ENTER THE CITY NAME OF THE TRADER: ");

String city = sc.nextLine();

trader1.setCity(city);

transaction.setTrader1(trader1);

System.out.print("ENTER THE YEAR OF TRANSACTION: ");

int year = sc.nextInt();

transaction.setYear(year);

System.out.print("ENTER THE AMOUNT OF TRANSACTION: ");

int amount = sc.nextInt();

transaction.setValue(amount);

arraylist.add(transaction);

break;

case 2:

System.out.println("List of all Traders and their Transactions");

for (Transaction t: arraylist){

System.out.println("Trader Name: "+t.getTrader1().getName());

System.out.println("Trader City: "+t.getTrader1().getCity());

System.out.println("Year:        "+t.getYear());

System.out.println("Value:       "+t.getValue());

System.out.println();

                    }

break;

case 3:

System.out.println("All the transactions in the year 2011 in sorted order: ");

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

transactionList = sortTransactions(arraylist);

for (Transaction t1: transactionList){

System.out.println("Trader Name: "+t1.getTrader1().getName());

System.out.println("Trader City: "+t1.getTrader1().getCity());

System.out.println("Value: "+t1.getValue());

System.out.println("Year: "+t1.getYear());

System.out.println();

                    }

break;

case 4:

System.out.println("All transactions from the traders lives in Delhi: ");

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

list = transactionsValuesDelhi(arraylist);

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

break;

case 5: int Maxvalue = highestTransaction(arraylist);

System.out.println("Highest Transaction value: "+Maxvalue);

break;

case 6: int Minvalue = smallestTransaction(arraylist);

System.out.println("Lowest Transaction value: "+Minvalue);

break;

case 7: System.exit(0);

default:

System.out.println("PLEASE ENTER THE CORRECT CHOICE: ");

            }

        }while (choice!=7);

    }

}

OUTPUT:

