**Assignment – 9**

package newjava;

import java.util.\*;

import java.util.stream.Collectors;

import java.util.stream.Stream;

class fruits

{

    String name, color;

    int calories,price;

    public fruits(String name, String color, int calories, int price) {

        this.name = name;

        this.color = color;

        this.calories = calories;

        this.price = price;

    }

    //getters and setters for every field we have created.................

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getColor() {

        return color;

    }

    public void setColor(String color) {

        this.color = color;

    }

    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;

    }

    @Override

    public String toString() {

        return "fruits [name=" + name + ", color=" + color + ", calories=" + calories + ", price=" + price + "]";

    }

}

class news

{

    int newsid;

    String postedByuser,commentByuser,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,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 Transaction{

    Trader trader;

    int year,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 SteamAPIassign {

    public static void main(String[] args) {

        //ARRAY FOR for fruits

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

        //aRRAY FOR NEWS

        List<news> list2= new ArrayList<news>();

        //array for traders

        List<Trader> list3 = new ArrayList<Trader>();

        //array for transactions

        List<Transaction> list4 = new ArrayList<>();

// Fruits class data is starting from here.................................................................

        //GETTING vALUES FOR FRUITS...

        list1.add(new fruits("mango","yellow",2200,200));

        list1.add(new fruits("appple","red",1001,1300));

        list1.add(new fruits("banana","yellow",1121,220));

        list1.add(new fruits("blueberry","blue",31,120));

        list1.add(new fruits("orange","orange",11,100));

        list1.add(new fruits("cherry","red",120,100));

        //data for calories more than 100

        List<fruits> data=list1.stream().

                filter((fruits)-> fruits.getCalories()>100).

                collect(Collectors.toList());

        //data for all aspects

        List<fruits> data1=list1.stream().

                collect(Collectors.toList());

        //Red color fruit data

        List<fruits> datared=list1.stream().

                filter((fruits)-> fruits.getColor()=="red").

                collect(Collectors.toList());

        //data for red color fruits

        System.out.println(datared);

        //fetching name only..

        List<String> names =

                data.stream()

                          .map(fruits::getName)

                          .collect(Collectors.toList());

        System.out.println(names);

        //fetching colors only...

        List<String> colors =

                data1.stream()

                          .map(fruits::getColor)

                          .collect(Collectors.toList());

        System.out.println(colors);

//      list1.stream().filter((fruits)-> fruits.getPrice()>100).collect(Collectors.toList()).forEach(System.out::println);

//

//      list1.stream().filter((fruits)-> fruits.getPrice()>100).collect(Collectors.toList()).forEach(System.out::println);

//      getcolor().stream().collect(Collectors.toList()).forEach(System.out::println);

// Fruits class data is ending here.................................................................

//news class data is starting from here.....

        list2.add(new news(1001,"About Mentality of human being","please check it out","Amazing post man"));

        list2.add(new news(1002,"About Cruelity of human Nature","new post is here","Reality of todays world"));

        list2.add(new news(1003,"Current Affairs","Demonatizations","i have suffered alot form this"));

        List<news> dataid=list2.stream().

                filter((news)-> news.getComment().length()>26).

                collect(Collectors.toList());

        System.out.println(dataid);

//news class end here............................................................................

//traders class is going to start...............................................................

        list3.add(new Trader("jackkie","indore"));

        list3.add(new Trader("giant","luchnow"));

        list3.add(new Trader("sammie","indore"));

        list3.add(new Trader("watlon","luchnow"));

        list3.add(new Trader("susi","pune"));

        list3.add(new Trader("amy","delhi"));

        list3.add(new Trader("lakhan","pune"));

        list3.add(new Trader("jammer","delhi"));

        list3.add(new Trader("jacob","pune"));

        //COMMON USE DATA TO FETCH OTHERS DETAILS....

        List<Trader> datatrader=list3.stream().

                collect(Collectors.toList());

        //Traders name works on pune only

        System.out.println("\n=======Traders works on pune only=========");

        List<String> datatradername=datatrader.stream().

                filter((Trader)->Trader.getCity()=="pune").

                map(Trader::getName).sorted().

                collect(Collectors.toList());

        System.out.println(datatradername);

        System.out.println("\n=======Sorted name of all traders========");

        //Names of all traders sorted alphabetically

        List<String> data\_all\_name = datatrader.stream().

                map(Trader::getName).sorted().

                collect(Collectors.toList());

        System.out.println(data\_all\_name);

        //Traders name works on indore only

        System.out.println("\n=======Traders works on indore only=========");

        List<String> datatradername\_indore=datatrader.stream().

                filter((Trader)->Trader.getCity()=="indore").

                map(Trader::getName).sorted().

                collect(Collectors.toList());

        System.out.println(datatradername\_indore);

//Trader class ended here.......................................................................................

//Transaction started from here.................................................................................

        list4.add(new Transaction (list3.get(0),2013, 23441441));

        list4.add(new Transaction (list3.get(1),2015, 3111111));

        list4.add(new Transaction (list3.get(2),2012, 123455555));

        list4.add(new Transaction (list3.get(3),2015, 6662312));

        list4.add(new Transaction (list3.get(4),2016, 666234));

        list4.add(new Transaction (list3.get(5),2011, 23441441));

        list4.add(new Transaction (list3.get(6),2019, 97453636));

        list4.add(new Transaction (list3.get(7),2012, 523425252));

        list4.add(new Transaction (list3.get(8),2011, 663424234));

//      System.out.println(list4);

        //this code is common, use for all queries,\..........

        List<Transaction> datatransaction=list4.stream().

                collect(Collectors.toList());

        System.out.println(datatransaction);

        //find transaction 2011...........

        System.out.println("\n Transaction occurs in 2011 year..............");

        List<Transaction> data\_2012=datatransaction.stream().

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

                collect(Collectors.toList());

        System.out.println(data\_2012);

        System.out.println("\n Transaction on delhi only............");

        List<Transaction> Transaction\_delhi\_city = datatransaction.stream().

                filter((Trader)->Trader.getTrader().getCity()=="delhi").

                collect(Collectors.toList());

        System.out.println(Transaction\_delhi\_city);

//

//      System.out.println("\n Unique city............");

//      List<Transaction> unique\_city = datatransaction.stream().

//              map(Trader->Trader::getTrader()::getCity()).

//              collect(Collectors.toList());

//      System.out.println(Transaction\_delhi\_city);

//

        //find transaction 2011...........

//              System.out.println("\n Transaction occurs Delhi..............");

//              List<Transaction> data\_delhi=datatransaction.stream().

//                      filter((Trader)->Trader.getCity()=="delhi").

//                      collect(Collectors.toList());

//              System.out.println(data\_2012);

        //highest and lowest value of transaction..............

        System.out.println("\n Highest Transaction in Transaction class..............");

        Optional<Integer> data\_Highest=datatransaction.stream().map(Transaction::getValue).max((t1,t2)->t1.compareTo(t2));

        System.out.println(data\_Highest);

        System.out.println("\n Smallest Transaction in Transaction class..............");

        Optional<Integer> data\_lowest=datatransaction.stream().map(Transaction::getValue).min((t1,t2)->t1.compareTo(t2));

        System.out.println(data\_lowest);

    }

}

**--Thank you--**-