Streams

```
package streams;

import java.util.Comparator;

public class Fruits {
    String name;
    int calories;
    int price;
    String color;

    public Fruits(String name , int calories, int price, String color){
        super();
        this.name=name;
        this.calories=calories;
        this.price=price;
        this.color=color;
    }

    public String getName(){
        return name;
    }
    public int getCalories(){
        return calories;
    }
    public int getPrice(){
        return price;
    }
    public String getColor(){
        return color;
    }
    public String toString(){
        return getName();

    }

}
class News{
    public static int getNewsId=102;
      public Object System;
    int newsId;
    String postedByuser;
    String commentByuser;
    String comment;

    public News(int newsId, String postedByuser , String commentByuser, String
comment){
        super();
        this.comment=comment;
        this.commentByuser=commentByuser;
        this.postedByuser=postedByuser;
        this.newsId=newsId;
    }

        public int getNewsId(){
        return newsId;
        }
        public String getPostedByuser(){
```

```
        return postedByuser;
        }
        public String getCommentByuser(){
        return commentByuser;
        }
        public String getComment(){
        return comment;
        }
    @Override
    public String toString(){
        return "News [ NewsId="+ newsId+", postedby=" +postedByuser+ "+commentby+
,"+ commentByuser+ "+comments no "+comment+" ]";
    }

}
class Trader{
    String name;
    String city;

    public Trader(String name, String city){
        super();
        this.city=city;
        this.name=name;
    }


    public String getCity(){
        return city;
    }
    public String getName(){
        return name;
    }
    @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 int getYear(){
        return year;
    }
    public int getValue(){return value;
    }
    @Override
    public String toString(){
```

```
        return "Transaction [ trader="+trader+", year= "+year+", value= "+value+"
]";
    }
}
```

&

```
package streams;

import javax.swing.*;
import java.util.*;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.stream.Collectors;
import java.util.function.Function;
import java.util.stream.Collectors;
import java.util.Comparator;
import java.util.concurrent.ConcurrentHashMap;
import java.util.Map;
import java.util.ArrayList;

public class Main {


    public static void main(String[] args) {
        List<Fruits> fruits = Arrays.asList(
                new Fruits("banana",200,100,"yellow"),
                new Fruits("apple",300,250,"red"),
                new Fruits("orange",100,200,"orange")


        );

        List<News> news = Arrays.asList(
                new News(11,"rutuja","the performance is lowering its
standard","5"),
                new News(12,"shalva","keep practising more","4"),
                new News(13,"prajakta","be confident","2")
        );

        List<Trader> trade = new ArrayList<>();
        Trader t1 = new Trader("rutuja","mumbai");
        Trader t2 =new Trader("lalit","nashik");
        Trader t3 = new Trader("om","pune");
        trade.add(t1);
        trade.add(t2);
        trade.add(t3);



        List<Transaction> transactions = Arrays.asList(
                new Transaction(t1,2010,380000),
                new Transaction(t2,2020,208000),
                new Transaction(t3,2015,855677)
        );

        System.out.println("-----------------1------------------------------");
        fruits.stream()
                .filter(p->p.getCalories() < 100)
```

```
                .sorted(Comparator.comparingInt(Fruits::getCalories).reversed())
                .forEach(name-> System.out.println(name));

        System.out.println("------------------2------------------------------");
        fruits.forEach((Fruits)->{
            System.out.println("name= "+Fruits.getName()+","+" Color=
"+Fruits.getColor());
        });

        System.out.println("------------------3------------------------------");
        fruits.stream()
                .filter(f->f.getColor().matches("red"))
                .sorted(Comparator.comparing(Fruits::getPrice))
                .forEach(name-> System.out.println(name));

        System.out.println("------------------4------------------------------");
        news.stream()
                .max(Comparator.comparing(News::getComment));
        System.out.println("newId is "+ News.getNewsId);


        System.out.println("------------------5------------------------------");
        long count=news.stream()
                .filter(n->n.getCommentByuser().contains("budget"))
                .count();
        System.out.println("no of times budget appeared=  "+ count);

        /*System.out.println("------------------6------------------------------
--");
        news.stream()
                .max(Comparator.comparing(News::getComment));
         System.out.println("max comments by user" + News.getComment());


*/
        System.out.println("-----------------7------------------------------");
        news.forEach((News)->{
            System.out.println("UserComments= "+News.getCommentByuser()+","+" no
of Comments= "+News.getComment());
        });

        System.out.println("-------------------8------------------------------
");
        transactions.stream()
                .filter(t->t.getYear()==2011)
                .sorted(Comparator.comparing(Transaction::getValue))
                .forEach(System.out::println);

        System.out.println("------------------9------------------------------
");
        List<Trader> distinctElements = trade.stream().filter(distinctByKey(c-
>c.getCity()))
                .collect(Collectors.toList());
        System.out.println("Unique city "+distinctElements);

        System.out.println("---------------------10-----------------------------
");
        trade.stream()
                .filter(p->p.getCity().matches("pune"))
```

```
                .sorted(Comparator.comparing(Trader::getName))
                .forEach(System.out::println);

        System.out.println("----------------------11-----------------------------
");
        StringBuilder str = new StringBuilder();
        trade.stream()
                .sorted(Comparator.comparing(Trader::getName))
                .forEach((Trader)->{
                    str.append(Trader.getName());});
        System.out.println(str);


        System.out.println("----------------------12-----------------------------
");
        trade.stream()
                .filter(t->t.getCity().matches("indore"))
                .forEach(System.out::println);

        System.out.println("----------------------13-----------------------------
");
        trade.stream()
                .filter(d->d.getCity().matches("delhi"))
                .forEach(System.out::println);

        System.out.println("----------------------14-----------------------------
");
        Transaction maxi = transactions.stream()
                .max(Comparator.comparingInt(Transaction::getValue))
                .get();
        System.out.println("Max value: "+maxi.value);


        System.out.println("----------------------15-----------------------------
");
        Transaction mini = transactions.stream()
                .min(Comparator.comparingInt(Transaction::getValue))
                .get();
        System.out.println("Min value: "+ mini.value);

    }

    public static <T> Predicate<T> distinctByKey(Function<? super T, Object>
keyExtractor)
    {
     Map<Object, Boolean> seen = new ConcurrentHashMap<>();
     return t-> seen.putIfAbsent(keyExtractor.apply(t),Boolean.TRUE) == null;
    }

}
```

OUTPUT

```
-----------------1-------------------------------
-----------------2-------------------------------
name= banana, Color= yellow
name= apple, Color= red
name= orange, Color= orange
-----------------3-------------------------------
apple
-----------------4-------------------------------
newId is 102
-----------------5-------------------------------
no of times budget appeared=  0
-----------------7-------------------------------
UserComments= the performance is lowering its standard, no of Comments= 5
UserComments= keep practising more, no of Comments= 4
UserComments= be confident, no of Comments= 2
-----------------8-------------------------------
-----------------9-------------------------------
Unique city [Trader [ name= rutuja, city= mumbai, ], Trader [ name= lalit, city=
nashik, ], Trader [ name= om, city= pune, ]]
---------------------10------------------------------
Trader [ name= om, city= pune, ]
---------------------11------------------------------
lalitomrutuja
---------------------12------------------------------
---------------------13------------------------------
---------------------14------------------------------
Max value: 855677
---------------------15------------------------------
Min value: 208000
```