```
/Fruit Class , News Class, Trader Class, Transaction Class


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;
    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 + "]";
        }

}
```

```
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;
```

```
/Main Class

public class Main1 {

    public static void main(String[] args) {
        List<Fruits> fruits = Arrays.asList(
                new Fruits("apple",200,100,"red"),
                new Fruits("Strawberry",300,150,"red"),
                new Fruits("Cherry",50,100,"red"),
                new Fruits("mango",200,100,"yellow"),
                new Fruits("pear",300,150,"green"),
                new Fruits("banana",50,100,"yellow")
```

```
        );

        List<News> news = Arrays.asList(
                new News(101,"Riddhi","The budget is increasing day by day","4"),
                new News(102,"Priya","The budget needs to fall down","7"),
                new News(103,"Sonu","Please adjust in given budget","5")
        );

        List<Trader> trade = new ArrayList<>();
        Trader t1 = new Trader("riddhi ","mumbai");
        Trader t2 =new Trader("siddhi ","delhi");
        Trader t3 = new Trader("raj ","nagpur");
        trade.add(t1);
        trade.add(t2);
        trade.add(t3);



        List<Transaction> transactions = Arrays.asList(
                new Transaction(t1,2011,300000),
                new Transaction(t2,2020,2000000),
                new Transaction(t3,2012,8526699)
        );

        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("-----------------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-------------------------------");
```

```java
        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:

```
Outline  Console
<terminated> Main1 [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe  (13-Aug-2021, 3:51:59 pm – 3:52:02 pm)
----------------1-------------------------------
Cherry
banana
------------------2-------------------------------
name= apple, Color= red
name= Strawberry, Color= red
name= Cherry, Color= red
name= mango, Color= yellow
name= pear, Color= green
name= banana, Color= yellow
------------------3-------------------------------
apple
Cherry
Strawberry
------------------4-------------------------------
newId is 102
------------------5-------------------------------
no of times budget appeared=  3
-----------------7-------------------------------
UserComments= The budget is increasing day by day, no of Comments= 4
UserComments= The budget needs to fall down, no of Comments= 7
UserComments= Please adjust in given budget, no of Comments= 5
-------------------8-------------------------------
Transaction [trader=Trader [ name= riddhi , city= mumbai, ], year=2011, value=300000]
-----------------9-------------------------------
Unique city [Trader [ name= riddhi , city= mumbai, ], Trader [ name= siddhi , city= delhi, ], Trader [ name= raj , city= nagpur, ]]
---------------------10-----------------------------
----------------------11-----------------------------
raj riddhi siddhi
----------------------12-----------------------------
----------------------13-----------------------------
Trader [ name= siddhi , city= delhi, ]
----------------------14-----------------------------
Max value: 8526699
---------------------15-----------------------------
Min value: 300000
```