STREAM API ASSIGNMENT

Name:Shruti Bhosale

1)Fruit class:

```java
public class fruit {
    String name;
    int calories;
    int price;
    String colour;
    public fruit(String name, int calories, int price, String colour) {
        super();
        this.name = name;
        this.calories = calories;
        this.price = price;
        this.colour = colour;}
    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 getColour() {
        return colour;}
    public void setColour(String colour) {
        this.colour = colour;}
    public String toString()
    {
        return ("name:"+ name +"calories:"+ calories + "price:"+ price + "colour:" + colour);

    }
}
```

```java
import java.util.ArrayList;
public class fruitsort{
    public static void main(String[] args) {

         List<fruit> Stream=new ArrayList<>();
         Stream.add(new fruit("Apple",100,50,"Red"));
         Stream.add(new fruit("Mango",200,100,"Yellow"));
         Stream.add(new fruit("Orange",50,600,"Orange"));
         Stream.add(new fruit("Guava",60,50,"Green"));
         Stream.add(new fruit("Cherry",70,90,"Red"));
         Stream.add(new fruit("Chichoo",50,50,"Brown"));
         Stream.add(new fruit("Guava",90,50,"Green"));


        //Que1
        System.out.println("Sorted By Calories less than 100 in Descending Order: \n");
        Stream<fruit> listSorted = Stream.stream().sorted(Comparator.comparingLong(fruit::getCalories).reversed()).filter(p -> p.getCalories()<100);

         listSorted.forEach(p -> System.out.println("FruitName: "+p.getName()+ "," + " Calories: "+p.getCalories()+ "," +
         " Price: "+p.getPrice()+ "," +" Colour: "+p.colour));
        //Que2
        System.out.println("\nColour wise fruit names:\n");
        Stream<fruit> listsorted2=Stream.stream().sorted(Comparator.comparing(fruit::getColour));
        listsorted2.forEach(p -> System.out.println("FruitName: "+p.getName()+ "," + " Calories: "+p.getCalories()+ "," +
                " Price: "+p.getPrice()+ "," +" Colour: "+p.colour));

         //Que3
         System.out.println("\nSorted Red Colour fruit as per the price\n");
         Stream<fruit> listSorted1 = Stream.stream().sorted(Comparator.comparingLong(fruit::getPrice)).filter(p -> p.getColour().matches("Red"));

         listSorted1.forEach(p -> System.out.println("FruitName: "+p.getName()+ "," + " Calories: "+p.getCalories()+ "," +
                 " Price: "+p.getPrice()+ "," +" Colour: "+p.colour));

    }
}
```

Output:

```
Problems  Javadoc  Declaration  Console  Error Log
<terminated> fruitsort [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe  (12-Aug-202
Sorted By Calories less than 100 in Descending Order:

FruitName: Guava, Calories: 90, Price: 50, Colour: Green
FruitName: Cherry, Calories: 70, Price: 90, Colour: Red
FruitName: Guava, Calories: 60, Price: 50, Colour: Green
FruitName: Orange, Calories: 50, Price: 600, Colour: Orange
FruitName: Chichoo, Calories: 50, Price: 50, Colour: Brown

Colour wise fruit names:

FruitName: Chichoo, Calories: 50, Price: 50, Colour: Brown
FruitName: Guava, Calories: 60, Price: 50, Colour: Green
FruitName: Guava, Calories: 90, Price: 50, Colour: Green
FruitName: Orange, Calories: 50, Price: 600, Colour: Orange
FruitName: Apple, Calories: 100, Price: 50, Colour: Red
FruitName: Cherry, Calories: 70, Price: 90, Colour: Red
FruitName: Mango, Calories: 200, Price: 100, Colour: Yellow

Sorted Red Colour fruit as per the price

FruitName: Apple, Calories: 100, Price: 50, Colour: Red
FruitName: Cherry, Calories: 70, Price: 90, Colour: Red
```

4)News Class:

```
1  class News {
2
3      int newsId;
4      String postedByUser;
5      String commentByUser;
6      String comment;
7      public News(int newsId, String postedByUser, String commentByUser, String comment) {
8          super();
9          this.newsId = newsId;
10         this.postedByUser = postedByUser;
11         this.commentByUser = commentByUser;
12         this.comment = comment;
13     }
14     public int getNewsId() {
15         return newsId;
16     }
17     public void setNewsId(int newsId) {
18         this.newsId = newsId;
19     }
20     public String getPostedByUser() {
21         return postedByUser;
22     }
23     public void setPostedByUser(String postedByUser) {
24         this.postedByUser = postedByUser;
25     }
26     public String getCommentByUser() {
27         return commentByUser;
28     }
29     public void setCommentByUser(String commentByUser) {
30         this.commentByUser = commentByUser;
31     }
32     public String getComment() {
33         return comment;
34     }
35     public void setComment(String comment) {
36         this.comment = comment;
37     }
38
39     public String toString() {
40         return "News [newsId" + newsId + ", postedByUser=" + postedByUser + ", commentByUser=" + comment +", comment=" + comment + "]";
41     }
42     }
43
```

mockito-example1/pom.xml | *News.java | *SortNews.java

```java
import java.util.ArrayList;
public class SortNews {
    public static void main(String[] args) {
          List<News> list=new ArrayList<>();
          list.add(new News(1,"Shruti","Sakshi","Your post is so good"));
          list.add(new News(2,"Rohit","Amit","Your Budget is going too long"));
          list.add(new News(3,"Akash","Ritesh","What is the Budget of this project"));
          list.add(new News(1,"Shruti","Ritesh","I didnt like your post"));
          list.add(new News(1,"Shruti","Megha","I like your Idea But Budget is too high"));
          list.add(new News(1,"Shruti","Ritesh","I like your Idea But Budget is too high"));
           //Que4
          int I=0;
          for(News i:list) {
              if(i.getComment().length()>I) {
                  I=i.getComment().length();
              }
          }
          for(News i:list) {
              if(i.getComment().length()==I) {
                  System.out.println("NewsId which has received a maximum comments is "+i.getNewsId());
              }
          }
          System.out.println("==========================================================");
         //Que5
         Long count=list.stream().filter(p -> p.getComment().contains("Budget")).count();
         System.out.println("\nBudget word appeared in the comments "+count+" times");
         //Que6
         List<String> list1 =new ArrayList<String>();
         for(News i:list) {
             list1.add(i.getCommentByUser());
         }

}
}
```

Console | Problems | Debug Shell

```
<terminated> SortNews [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe  (13-Aug-2021, 10:12:54 pm – 10:12:57 pm)
NewsId which has received a maximum comments is 1
==========================================================

Budget word appeared in the comments 4 times
```

3)Trader Class:

```java

public class Trader {
     String name;
     String city;
    public Trader(String name, String city) {
        super();
        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;
    }

    public String toString(){
        return "Trader [name=" + name+ ", City=" + city +"]";

    }
}

```

```java
import java.util.ArrayList;

public class SortTrader {
	public static void main(String[] args) {
		List<Trader> trade=new ArrayList<>();
		trade.add(new Trader("Shruti","Mumbai"));
		trade.add(new Trader("Ritesh","Pune"));
		trade.add(new Trader("Swarda","Aurangabad"));
		trade.add(new Trader("Teju","Pune"));
		trade.add(new Trader("Piu","Indore"));
		trade.add(new Trader("Akash","Khed"));
		trade.add(new Trader("Yogesh","Chiplun"));
		trade.add(new Trader("Sanket","Delhi"));
		trade.add(new Trader("Bunty","Aurangabad"));
		trade.add(new Trader("Akshay","Delhi"));
		trade.add(new Trader("Aish","Aurangabad"));
		//Que10
		Stream<Trader> sorted = trade.stream().filter(p -> p.getCity().equals("Pune")).sorted(Comparator.comparing(Trader :: getName));
		sorted.forEach(p -> System.out.println("Name:" +p.getName()+ " city: "+p.getCity()));
		System.out.println("====================================");
		//Que12
		Stream<Trader> sorted1 = trade.stream().filter(p -> p.getCity().equals("Indore"));
		sorted1.forEach(p -> System.out.println("Name:" +p.getName()+ " city: "+p.getCity()));
		System.out.println("====================================");
		//Que11
		Stream<Trader> sorted2 = trade.stream().sorted(Comparator.comparing(Trader :: getName));
		sorted2.forEach(p -> System.out.println("Name: " +p.getName()+ " city: "+p.getCity()));
		System.out.println("====================================");
		//Que9
		Stream<Trader> sorted3 = trade.stream().filter(DistinctCustomPropertyExample.distinctByKey(p -> p.getCity()));
		sorted3.forEach(p -> System.out.println("Name: " +p.getName()+ " city: "+p.getCity()));
	}


}

```

```java
import java.util.Map;

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

```

Output:

```
Console
<terminated> SortTrader [Java Application] C:\Program Files\Java\jdk-13.0.
Name:Ritesh city: Pune
Name:Teju city: Pune
====================================
Name:Piu city: Indore
====================================
Name: Aish city: Aurangabad
Name: Akash city: Khed
Name: Akshay city: Delhi
Name: Bunty city: Aurangabad
Name: Piu city: Indore
Name: Ritesh city: Pune
Name: Sanket city: Delhi
Name: Shruti city: Mumbai
Name: Swarda city: Aurangabad
Name: Teju city: Pune
Name: Yogesh city: Chiplun
====================================
Name: Shruti city: Mumbai
Name: Ritesh city: Pune
Name: Swarda city: Aurangabad
Name: Piu city: Indore
Name: Akash city: Khed
Name: Yogesh city: Chiplun
Name: Sanket city: Delhi
```

4)Transaction:

```
public class Transaction{

    final Trader trader;
    final int year;
    private final int value;

    public Transaction(Trader trader, int year, int value){
        this.trader = trader;
        this.year = year;
        this.value = value;
    }

    public Trader getTrader(){
        return this.trader;
    }

    public int getYear(){
        return this.year;
    }

    public int getValue(){
        return this.value;
    }

    public String toString(){
        return "{" + this.trader + ", " +
               "year: "+this.year+", " +
               "value:" + this.value +"}";
    }
}
```

```
import java.util.ArrayList;

public class Transort {

    public static void main(String[] args) {
    Trader t1 = new Trader("Shruti", "Mumbai");
    Trader t2 = new Trader("Ritesh","Pune");
    Trader t3 = new Trader("Akash","Delhi");
    Trader t4 = new Trader("Swarda","Delhi");
    Trader t5 = new Trader("Shital","Roha");
    Trader t6 = new Trader("Anu","Mumbai");
    Trader t7 = new Trader("Piu","Delhi");

    Transaction T1=new Transaction(t1,2011,4000);
    Transaction T2=new Transaction(t2,2010,4000);
    Transaction T3=new Transaction(t3,2014,4000);
    Transaction T4=new Transaction(t4,2011,4000);
    Transaction T5=new Transaction(t5,2015,4000);
    Transaction T6=new Transaction(t6,2016,4000);
    Transaction T7=new Transaction(t7,2011,4000);

    List<Transaction> trans=new ArrayList<>();
    trans.add(T1);
    trans.add(T2);
    trans.add(T3);
    trans.add(T4);
    trans.add(T5);
    trans.add(T6);
    trans.add(T7);

            //Que 8
            System.out.println("--------TRANSACTION IN YEAR 2011-------");
            List<Transaction> Year2011=trans.stream().filter(P -> P.year==2011).sorted(Comparator.comparing(Transaction::getValue))
                    .collect(Collectors.toList());
            Year2011.forEach(System.out::println);
```

```java
            //Que14
            System.out.println("\n"+"--------Highest TRANSACTION-------");
            Transaction Max=trans.stream().max(Comparator.comparing(Transaction::getValue)).get();
            System.out.println(Max);

            //Que15
            System.out.println("\n"+"--------Lowest TRANSACTION-------");
            Transaction Min=trans.stream().min(Comparator.comparing(Transaction::getValue)).get();
            System.out.println(Min);

            //Que13
            System.out.println("\n"+"--------TRANSACTION OF TRADER FROM DELHI-------");
            List<Transaction> FromDelhi=trans.stream().filter(P -> P.trader.getCity().equals("Delhi")).collect(Collectors.toList());
            FromDelhi.forEach(System.out::println);



    }
}

```

Console Problems Debug Shell

<terminated> Transort [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (13-Aug-2021, 10:32:56 pm – 10:32:58 pm)

```
--------TRANSACTION IN YEAR 2011-------
{Trader [name=Shruti, City=Mumbai], year: 2011, value:4000}
{Trader [name=Swarda, City=Delhi], year: 2011, value:4000}
{Trader [name=Piu, City=Delhi], year: 2011, value:4000}

--------Highest TRANSACTION-------
{Trader [name=Shruti, City=Mumbai], year: 2011, value:4000}

--------Lowest TRANSACTION-------
{Trader [name=Shruti, City=Mumbai], year: 2011, value:4000}

--------TRANSACTION OF TRADER FROM DELHI-------
{Trader [name=Akash, City=Delhi], year: 2014, value:4000}
{Trader [name=Swarda, City=Delhi], year: 2011, value:4000}
{Trader [name=Piu, City=Delhi], year: 2011, value:4000}
```