JAVA STREAM ASSIGNMENT 9

Class fruit

**package** nikita;

**public** **class** Fruit

{

String name;

**int** calories;

**int** price;

String color;

**public** Fruit(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;

}

@Override

**public** String toString()

{

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

}

}

Class news

**package** nikita;

**public** **class** News

{

**public** **static** **int** *getNewsid* = 5;

**public** **static** String *getCommentByUser* = "mahek";

**public** **static** **int** *getComment* = 5;

**int** newsid;

String postedbyUser;

String commentByUser;

String 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** String getPostedbyUser()

{

**return** postedbyUser;

}

**public** String getCommentByUser()

{

**return** commentByUser;

}

**public** String getComment()

{

**return** comment;

}

@Override

**public** String toString()

{

**return** "news[newsId = " +newsid+ ",PostedbyUser = " +postedbyUser+ ",CommentByUser = " +commentByUser+ ",Comment = " +comment+ "]";

}

}

Class traders

**package** nikita;

**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** String getCity()

{

**return** city;

}

@Override

**public** String toString()

{

**return** "traders[name = " +name+ ",city = " +city+ "]";

}

}

Class transaction

**package** nikita;

**public** **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+ "]";

}

}

1,2,3 ques

package nikita;

import java.util.Comparator;

import java.util.Arrays;

import java.util.List;

public class FruitMain

{

public static void main(String arg[])

{

List<Fruit> fruit = Arrays.asList(

new Fruit("banana",200,100,"yellow"),

new Fruit("litchi",300,150,"red"),

new Fruit("apple",150,190,"red"),

new Fruit("mango",350,200,"yellow"),

new Fruit("pineapple",50,180,"green"));

System.out.println("-----------------1-------------------------------");

fruit.stream()

.filter(p->p.getCalories() < 100)

.sorted(Comparator.comparing(Fruit::getCalories).reversed())

.forEach(name-> System.out.println(name));

System.out.println("------------------2-------------------------------");

fruit.stream()

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

System.out.println("------------------3-------------------------------");

fruit.stream()

.filter(f->f.getColor().matches("red"))

.sorted(Comparator.comparing(Fruit::getPrice))

.forEach(name-> System.out.println(name));

}

}

4,5,6,7 ques

package nikita;

import java.util.Comparator;

import java.util.Arrays;

import java.util.List;

public class NewsMain

{

public static void main(String[] args)

{

List<News> news = Arrays.asList(

new News(1,"nikita","learning java","1"),

new News(1,"pankaj","learning mongodb","2"),

new News(1,"kamal","learning git","3"),

new News(1,"mithi","learning uml","4"),

new News(1,"mahek","learning spring","5"));

System.out.println("------------------4--------------------------------");

news.stream()

.max(Comparator.comparing(News::getComment));

System.out.println("newsId 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("user that commented most is "+News.getCommentByUser+ " that is " +News.getComment+ " times");

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

news.forEach((News)-> System.out.println("UserComments= "+News.getCommentByUser()+","+" no of Comments= "+News.getComment()));

}

}

8,9,10,11,12,13,14,15 ques

**package** nikita;

**import** java.util.Comparator;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.function.Function;

**import** java.util.stream.Collectors;

**import** java.util.function.Predicate;

**import** java.util.concurrent.ConcurrentHashMap;

**import** java.util.Map;

**public** **class** TraderMain

{

**public** **static** **void** main(String arg[])

{

List<Trader> trade = **new** ArrayList<>();

Trader t1 = **new** Trader("nikita","mumbai");

Trader t2 =**new** Trader("ruchi","indore");

Trader t3 = **new** Trader("pankaj","delhi");

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

StringBuilder str1 = **new** StringBuilder();

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;

} \*/

}