Q1,2,3,4

package cg;

public class Fruits {

private String name;

private int calories;

private int price;

private String color;

public Fruits(String name,Integer calories,Integer price,String color) {

this.name=name;

this.calories=calories;

this.price=price;

this.color=color;

}

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 String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

@Override

public String toString() {

return this.name+ " " + this.calories+ " " + this.price+" " + this.color + " ";

}

}

+++++++++++++++++++++++++++++++++++++++++++++++++

package cg;

import java.util.Arrays;

import java.util.Comparator;

import java.util.List;

import java.util.stream.Collectors;

public class Color {

public static void main(String[] args) {

List<Fruits> fruitslist=Arrays.asList(new Fruits("Mango",99, 90,"Yellow"),

new Fruits("Apple", 95, 30, "Red"),

new Fruits("Strawberry", 25, 10,"Red"),

new Fruits("Cherry", 10, 10, "Red"),

new Fruits("Grapes", 15, 20, "Green"));

//fruits.forEach(System.out::println);

List<Fruits> colorList=fruitslist.stream().sorted(Comparator.comparing(Fruits::getColor)).collect(Collectors.toList());

System.out.println("Colorwise sorting of fruits is:");

colorList.forEach(System.out::println);

}

}

+++++++++++++++++++++++++++++++

package cg;

import java.util.Arrays;

import java.util.Comparator;

import java.util.List;

import java.util.stream.Collectors;

public class Calories {

public static void main(String[] args) {

List<Fruits> fruitslist=Arrays.asList(

new Fruits("Mango",95, 80,"Yellow"),

new Fruits("Apple", 95, 30, "Red"),

new Fruits("Strawberry", 25, 10,"Red"),

new Fruits("Cherry", 10, 10, "Red"),

new Fruits("Grapes", 15, 20, "Green"));

//fruits.forEach(System.out::println);

List<Fruits>sortedList=fruitslist.stream().sorted(Comparator.comparingInt(Fruits::getCalories).reversed()).collect(Collectors.toList());

System.out.println("Fruits with calories<100 are:");

sortedList.forEach(System.out::println);

}

}

===============================================================

Q 5,6,7

package cg;

import java.util.Arrays;

import java.util.Comparator;

import java.util.List;

import java.util.Optional;

import java.util.stream.Collector;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class News {

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 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;

}

public static void main(String[] args ) {

List<News> list=Arrays.asList(

new News(123, "Iron Man", "Thanos", "Finance Minister Nirmala Sitharaman on Saturday said that the government, undeterred by the COVID-19 pandemic, has been pursuing reforms for achieving sustained long-term growth in a bid to make India one of the top economies of the world in the coming decades."),

new News(234, "Thor", "Loki", "Replying to debate on Budget 2021-22 in Lok Sabha, she said, the Prime Minister did not lose any opportunity to continue with the reforms and this Budget has set the pace for India to become self-reliant or Aatmanirbhar."),

new News(345, "Hulk", "She Hulk", "She Budget for 2021-22, has provided the highest capex growth of 34.4 per cent by providing more money to railways, roads and defence."),

new News(456, "Hulk", "Loki", "She said her Budget for 2021-22, has provided the highest capex growth of 34.4 per cent by providing more money to railways, roads and defence.")

);

//Question 5

int count=(int) list.stream()

.filter(x-> x.comment.contains("Budget"))

.count();

System.out.println(count);

System.out.println("=====================================================");

//Question6

Optional<String> max = list.stream().map(x->x.commentByUser)

.max((i,j)->i.compareTo(j));

System.out.println(max.get());

Optional<News> max1=list.stream()

.collect(Collectors.maxBy(Comparator.comparing(News::getCommentByUser)));

System.out.println(max1.get());

list.stream().map(x->x.commentByUser)

.sorted()

.forEach(System.out::println);

//Question 7

}

@Override

public String toString() {

return "News [newsId=" + newsId + ", postedByUser=" + postedByUser + ", commentByUser=" + commentByUser

+ ", comment=" + comment + "]";

}

}

Q 9,10

package cg;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Stream;

public class Trader {

String name;

String city;

public Trader(String name, String city) {

super();

this.name = name;

this.city = city;

}

public static void main(String[] args) {

List<Trader> list=Arrays.asList(

new Trader("Iron Man", "Pune"),

new Trader("Thor", "Bangalore"),

new Trader("Hulk", "Pune"),

new Trader("Captain", "Delhi")

);

//Question 9

list.stream().map(x-> x.city)

.distinct()

.forEach(System.out::println);

System.out.println("==========================================");

//Question 10

list.stream().filter(x->x.city.contains("Pune"))

.map(x->x.name)

.sorted()

.forEach(System.out::println);

}

@Override

public String toString() {

return "Trader [name=" + name + ", city=" + city + "]";

}

}

====================================================================

Q 8,11,12,13,14,15

package cg;

import java.util.Arrays;

import java.util.List;

import java.util.Optional;

import java.util.stream.Stream;

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 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 static void main(String[] args) {

List<Transaction> list=Arrays.asList(

new Transaction(new Trader("Iron Man", "Pune"), 2011, 5000),

new Transaction(new Trader("Thor", "Bangalore"), 2012, 4000),

new Transaction(new Trader("Hulk", "Pune"), 2011, 3000),

new Transaction(new Trader("Captain", "Delhi"), 2012, 2000)

);

//Question 8

list.stream().filter(x-> x.year==2011)

.map(x-> x.value)

.sorted()

.forEach(System.out::println);

System.out.println("==========================================");

//Question 11

list.stream().map(x->x.trader.name)

.sorted()

.forEach(System.out::println);

System.out.println("==========================================");

//Question 12

list.stream().filter(x->x.trader.city.contains("Indore"))

.map(x->x.trader)

.forEach(System.out::println);

System.out.println("==========================================");

//Question 13

list.stream().filter(x->x.trader.city.contains("Delhi"))

.map(x->x.value)

.forEach(System.out::println);

System.out.println("==========================================");

//Question 14

Optional<Integer> max = list.stream().map(x->x.value)

.max((i,j)->i.compareTo(j));

System.out.println(max);

System.out.println("==========================================");

//question 15

Optional<Integer> min = list.stream().map(x->x.value)

.min((i,j)->i.compareTo(j));

System.out.println(min);

}

}