Name of the Course : Complete Java SE8 Developer Bootcamp

Level : Difficult

Tool Stack : Java8 and Junit5

Problem Statement : Provide a code solution to this requirement.

In this requirement, you need to sort the list of auditorium based on costPerDay, and capacity.

Description :  
  
a) Create a Class **Auditorium** with the following attributes: 



|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| name | String |
| costPerDay | Double |
| capacity | Integer |

Mark all the attributes as private, Create / Generate appropriate Getters & Setters, Add a parameterized constructor to take in all attributes in the given order: **Auditorium( String name, Double costPerDay, Integer capacity )**  
  
b) Create the following static methods in the **Auditorium** class, 



|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Auditorium createNew(String detail) | This method accepts a String. The auditorium detail separated by commas is passed as the argument. Split the details and create a hall object and returns it. |

The details should be given as a comma-separated value in the below order,  
**name, costPerDay, capacity**  
  
c) The class should implement the **Comparable** interface which sorts the Hall list based on costPerDay. While comparing, all the costPerDay attributes in the list are unique.  
  
d) Create a class **CapacityComparator** which implements Comparator interface and sort the list based on capacity. While comparing, all the capacity attributes in the list are unique.  
  
e) Creat a MainClass with main method for getting inputs from user Get the number of Auditorium and details and create a list. Sort the Auditorium according to the given option and display the list.  
  
When the object is printed, it should display the following details  
Print format:  
**System.out.format("%-15s %-15s %s\n","Name","Cost Per Day","Capacity");**  
Display one digit after decimal point for Double datatype.  
  
**Sample Input and Output 1:**  
Enter the number of the Auditorium:  
**3**

**Concert Hall,15000,550**

**Opera Hall,10000,400**

**Symphony Hall,20000,500**  
Enter a type to sort:  
1.Sort by Cost Per Day  
2.Sort by Capacity  
**1**

|  |  |  |
| --- | --- | --- |
| Name | Cost Per Day | Capacity |
| Opera Hall | 10000.0 | 400 |
| Concert Hall | 15000.0 | 550 |
| Symphony Hall | 20000.0 | 500 |

**Sample Input and Output 2:**   
  
Enter the number of the Auditorium:   
**3  
Concert Hall,15000,550  
Opera Hall,10000,400  
Symphony Hall,20000,500**   
Enter a type to sort:   
1.Sort by Cost Per Day   
2.Sort by Capacity   
**2**

|  |  |  |
| --- | --- | --- |
| Name | Cost Per Day | Capacity |
| Opera Hall | 10000.0 | 400 |
| Symphony Hall | 20000.0 | 500 |
| Concert Hall | 15000.0 | 550 |

Code:

import java.util.ArrayList;

import java.util.Collection;

import java.util.Collections;

import java.util.Comparator;

import java.util.Scanner;

public class Auditorium implements Comparable<Auditorium>{

private String name;

private Double costperday;

private Integer capacity;

public String getName() {

return name;

}

@Override

public String toString() {

return String.format("%-15s %-15s %-15s\n ", name , costperday ,capacity );

}

public void setName(String name) {

this.name = name;

}

public Double getCostperday() {

return costperday;

}

public void setCostperday(Double costperday) {

this.costperday = costperday;

}

public Integer getCapacity() {

return capacity;

}

public void setCapacity(Integer capacity) {

this.capacity = capacity;

}

public Auditorium(String name, Double costperday, Integer capacity) {

super();

this.name = name;

this.costperday = costperday;

this.capacity = capacity;

}

public static Auditorium createNew(String detail) {

//Concert Hall,15000,550

String[] details =detail.split(",");

Auditorium audi = new Auditorium(details[0],Double.parseDouble(details[1]),Integer.parseInt(details[2]));

return audi;

}

@Override

public int compareTo(Auditorium o) {

return Double.compare(this.costperday, o.costperday) ;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int nos=0;

ArrayList<Auditorium> audilist = new ArrayList<Auditorium> ();

System.out.println("Enter no of Auditorium");

nos = Integer.parseInt(sc.nextLine());

for (int i = 0; i < nos; i++) {

audilist.add(Auditorium.createNew(sc.nextLine()));

}

System.out.format("%-15s %-15s %s\n" ,"Name","Cost per Day","Capacity");

for (Auditorium auditorium : audilist) {

System.out.println(auditorium);

}

System.out.println("Enter a type to sort:");

System.out.println("1.Sort by Cost Per Day");

System.out.println("2.Sort by capacity");

System.out.println();

int sorttype= Integer.parseInt(sc.nextLine());

//int sorttype = 0;

switch (sorttype) {

case 1: Collections.sort(audilist);

break;

case 2: Collections.sort(audilist, new CapacityComparator());

break;

default:

break;

}

System.out.format("%-15s %-15s %s\n" ,"Name","Cost per Day","Capacity");

for (Auditorium auditorium : audilist) {

System.out.println(auditorium);

}

}

}

import java.util.Comparator;

public class CapacityComparator implements Comparator<Auditorium>{

@Override

public int compare(Auditorium o1, Auditorium o2) {

// TODO Auto-generated method stub

return Integer.compare(o1.getCapacity(), o2.getCapacity());

}

}

Test Data1

**Sample Input and Output 1:**  
Enter the number of the Auditorium:  
**3**

**Concert Hall,15000,550**

**Opera Hall,10000,400**

**Symphony Hall,20000,500**  
Enter a type to sort:  
1.Sort by Cost Per Day  
2.Sort by Capacity  
**1**

|  |  |  |
| --- | --- | --- |
| Name | Cost Per Day | Capacity |
| Opera Hall | 10000.0 | 400 |
| Concert Hall | 15000.0 | 550 |
| Symphony Hall | 20000.0 | 500 |

**Sample Input and Output 2:**   
  
Enter the number of the Auditorium:   
**3  
Concert Hall,15000,550  
Opera Hall,10000,400  
Symphony Hall,20000,500**   
Enter a type to sort:   
1.Sort by Cost Per Day   
2.Sort by Capacity   
**2**

|  |  |  |
| --- | --- | --- |
| Name | Cost Per Day | Capacity |
| Opera Hall | 10000.0 | 400 |
| Symphony Hall | 20000.0 | 500 |
| Concert Hall | 15000.0 | 550 |

Learning outcome: Participant could able to learn how to use the Collections API.