1 . a . The median of a list is its middle value when the values are placed in order. For example, if a list contains 1, 4, 7, 8, and 9, then the median is 7. Write an application that allows you to enter nine double values and display them and their median. Save the file as Median.java.

b. Revise the Median class so that the user can enter any number of values up to nine. If the list has an even number of values, the median is the numeric average of the values in the two middle positions. Save the file as Median2.java.

A

import java.util.Arrays;

import java.util.Scanner;

public class Medianex {public static void main(String[] args)

{

Scanner input = new Scanner(System.in);

int i = 0; // iterator for the input loop

String userInput = "";

double[] numbers = new double[9];

System.out.println("Please enter nine numbers.");

do

{

System.out.print("Number " + (i + 1) + "/9: ");

userInput = input.nextLine();

try

{

numbers[i] = Double.parseDouble(userInput);

i++;

}

catch (Exception e)

{

System.out.println("Error: " + userInput + " is not a valid number");

}

} while (i < 9);

Arrays.sort(numbers);

System.out.print("\n\nYou entered: ");

for(Double num : numbers)

System.out.print(num + " ");

System.out.println("\nThe median value is " + numbers[4]);

}

}

B.

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

public class MedianRevise {

//Method for calculating Median value

public static double getMedian(double[] nums)

{

double median = 0.0;

int[] middle = new int[2];

if (nums.length % 2 == 0)

{

middle[0] = (nums.length / 2) - 1; // nums.length counts from "1" instead of "0"

middle[1] = middle[0] + 1; // get the other middle value in an even length array

median = nums[middle[0]] + nums[middle[1]];

median = median / 2.0;

}

else

{

middle[0] = nums.length / 2;

median = nums[middle[0]];

}

return median;

}

public static void main(String[] args)

{

Scanner input = new Scanner(System.in);

int LIMIT = 9;

int i=0;

String userInput = "";

ArrayList<String> numberList = new ArrayList<String>();

double[] numbers;

System.out.println("Please enter up to " + LIMIT + " numbers.\nTo stop just press enter.");

do

{

System.out.print("Number " + (i + 1) + "/" + LIMIT + ": ");

userInput = input.nextLine().toLowerCase();

if (!(userInput.isEmpty()))

{

try

{

Double.parseDouble(userInput);

numberList.add(userInput);

i++;

}

catch (Exception e)

{

System.out.println("Error: " + userInput + " is not a valid number");

if (userInput.indexOf(' ') > -1)

System.out.println("Please use a single line for each number.");

}

}

} while (!(userInput.isEmpty()) && i < LIMIT);

if (numberList.size() == 0)

{

System.out.println("\nNothing was entered exiting...");

return; // Kills program if not even one value was entered

}

else

{

numbers = new double[numberList.size()]; // Create an array the same size as the List

}

for(int c = 0; c < numberList.size(); c++)

numbers[c] = Double.parseDouble(numberList.get(c).toString()); // Transfer list values to the array

Arrays.sort(numbers);

System.out.print("\n\nYou entered: ");

for(Double num : numbers)

System.out.print(num + " ");

System.out.println("\nThe median value is " + getMedian(numbers));

input.close();

}

}

2. a.Write an application containing an array of 15 double values. Include a method to sort and display the values in ascending order. Compile, run, and check the results. Save the file as SortDoubles.java.

b. Modify the SortDoubles application to prompt the user whether to view the list in ascending or descending order. Save the file as SortDoublesWithOption.java.

public class Arraysort {

public static void main(String[] args)

{

Double[] ar = {4.0, 6.0, 1.0, 8.0, 3.0, 9.0, 7.0, 4.0, 20.0,45.0,78.0,12.0,14.0,52.0,160.9,32.0,24.0};

// To sort array in ascending order.

Arrays.sort(ar);

System.out.println("Sorted array : Ascending order" + Arrays.toString(ar));

//// To sort array in descending order.

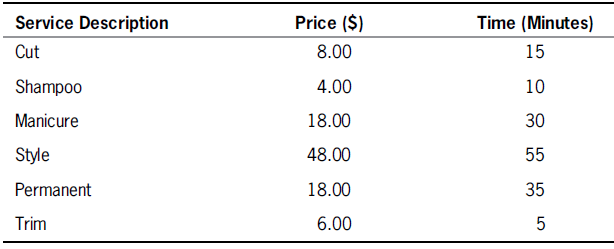
Arrays.sort(ar, Collections.reverseOrder());

System.out.println("Sorted array : descending order" + Arrays.toString(ar));

}

}

3. Table shows the various services offered by a hair salon, including its prices and times required:



Create a class that holds the service description, price, and the number of minutes it takes to perform the service. Include a constructor that requires arguments for all three data fields and three get methods that each return one of the data field’s values. Save the class as Service.java.

b. Write an application named SalonReport that contains an array to hold six Service objects, and fill it with the data from Table 9-6. Include methods to sort the array in ascending order by each of the data fields. Prompt the user for the preferred sort order, and display the list of services in the requested order. Save the program as SalonReport.java.

public class SalonService {//parameters declartions

String servDescript;

double price;

int time;

// constructor for Service

public SalonService(String s, double p, int m) {

servDescript = s;

price = p;

time = m;

}

//get Methods to return service description, price, and average

// minutes to perform services

public String getServiceType() {

return servDescript;

}

public double getPrice() {

return price;

}

public int getMinutes() {

return time;

}

@Override

public String toString() {

return "servDescript = " + servDescript +

", price = " + price +

", time = " + time + System.getProperty("line.separator");

}

}

b.

import java.util.Arrays;

import java.util.Scanner;

public class SalonREport {

public static void main(String[] args) {

SalonService[] myService= new SalonService[6];

myService[0] = new SalonService("Cut",8.00,15);

myService[1] = new SalonService("Shampoo",4.00,10);

myService[2] = new SalonService("Manicure",18.00,30);

myService[3] = new SalonService("Style",48.00,55);

myService[4] = new SalonService("Permanent",18.00,35);

myService[5] = new SalonService("Trim",6.00,5);

Scanner input = new Scanner(System.in);

System.out.println("enter a number to sort the data\n"+ "1. according service description\n"+

"2. according to price\n"+"3. According to time");

int choice = input.nextInt();

System.out.println("enter the order\t 1 for ascending\t 2 for descending");

int order = input.nextInt();

switch(choice){

case 1: sortDescrp(myService,order);

break;

case 2: sortPrice(myService,order);

break;

case 3: sortTime(myService,order);

break;

default: System.out.println("Invalid input");

}

}

//(a,b) -> is lambda expression

public static void sortDescrp(SalonService[] myService,int order){

if(order==1){

Arrays.sort(myService, (a,b)-> a.servDescript.compareTo(b.servDescript));

System.out.println((Arrays.asList(myService)));

}

else

{

Arrays.sort(myService, (a,b)-> b.servDescript.compareTo(a.servDescript));

System.out.println((Arrays.asList(myService)));

}

}

public static void sortPrice(SalonService[] myService,int order){

if(order==1){

Arrays.sort(myService,(a,b)->Double.compare(a.price,b.price));

System.out.println((Arrays.asList(myService)));

}

else

{

Arrays.sort(myService,(a,b)->Double.compare(b.price,a.price));

System.out.println((Arrays.asList(myService)));

}

}

public static void sortTime(SalonService[] myService, int order){

if(order==1){

Arrays.sort(myService, (a,b)->Integer.compare(a.time, b.time));

System.out.println((Arrays.asList(myService)));

}

else{

Arrays.sort(myService, (a,b)->Integer.compare(b.time, a.time));

System.out.println((Arrays.asList(myService)));

}

}

}

**The following defines a \_\_\_\_\_\_\_\_\_\_\_\_\_ array:**

**int[][] nums = {{1, 2}, {3, 4}, {5, 6}};**

a. one-dimensional

**b. two-dimensional**

c. three-dimensional

d. six-dimensional

**How many rows are contained in the following array?**

**double[][] prices = {{2.56, 3.57, 4.58, 5.59},**

**{12.35, 13.35, 14.35, 15.00}};**

a. 1

**b. 2**

c. 4

d. 8

**In the following array, what is the value of code[2][1]?**

**char[][] code = {{ 'A ', 'D ', 'M '},**

**{ 'P ', 'R ', 'S '};**

**{ 'U ', 'V ', 'Z '}};**

a. 'P'

b. 'R'

c. 'U'

**d. 'V'**

**In the following array, what is the value of address[1][1]?**

**String address = {{ "123 Oak ", "345 Elm "}; { "87 Maple ", "901 Linden "}};**

a. "123 Oak "

b. "345 Elm "

c. "87 Maple "

**d. "901 Linden "**

**In the following array, what is the value of fees.length?**

**double[][] fees = {{3.00, 3.50, 4.00, 5.00}, {6.35, 7.35, 8.35, 9.00}};**

**a. 2**

b. 4

c. 8

d. none of the above

**In the following array, what is the value of fees[1].length?**

**double[][] fees = {{3.00, 3.50, 4.00, 5.00}, {6.35, 7.35, 8.35, 9.00}};**

a. 2

**b. 4**

c. 8

d. none of the above

**The chief advantage to using the ArrayList class instead of the Arrays class is that**

an ArrayList \_\_\_\_\_.

a. can be much larger

b. is easier to search

**c. is dynamically resizable**

d. can be used as an argument to a static method