Source code:

Order.java:

/\*\*

\* File Name: Order.java

\* Date: Sep 7, 2023

\* Author: Richard Fritsche

\* Project Name: portfolio

\* Project Purpose:

\* Algorithm Used:

\* Program Inputs:

\* Program Limitations:

\* Program Errors:

\*/

**package** portfolio;

**import** java.util.ArrayList;

**import** java.util.NoSuchElementException;

/\*\*

\* **@author** richard

\*

\*/

**public** **class** Order{

**private** ArrayList<OrderDetail> orders;

**private** **boolean** veritas;//boolian variable to confirm object is valid.

**public** Order() {

veritas = **false**;

orders = **new** ArrayList<>();

veritas = **true**;

}//end constructor

**public** **void** add(String name, **int** orderNum, **double** orderCost) {

add(**new** OrderDetail(name, orderNum, orderCost));

}//end add method

**public** **void** display() {

**if** (orders.size() <= 0) {//if there are no elements in the queue

System.***out***.println("There are no elements to display");

**return**;

}//end if

System.***out***.println("Displaying orders in the order they were entered: ");

**for**(OrderDetail order: orders) {//for each order in orders

order.display();

}//end for

}//end display method

/\*\*

\*

\*/

**public** **void** isValid()**throws** IllegalStateException

{

**if**(!veritas) {//if not valid

**throw** **new** IllegalStateException();

}//end if

}//end isValid

**public** **int** size()

{

**try** {

isValid();

**return** orders.size();

}//end try

**catch** (IllegalStateException e) {//if the Order Queue is not valid

**return** 0;

}//end catch

}

**public** **boolean** isEmpty()

{

**return** orders.isEmpty();

}

**public** OrderDetail[] toArray()

{

**return** (OrderDetail[]) orders.toArray();

}//end toArray method

**public** **void** clear()

{

orders.clear();

}

**public** **boolean** add(OrderDetail o)

{

**try** {

isValid();

**return** orders.add(o);

}//end try

**catch** (IllegalStateException e) {

**return** **false**;

}//end catch

}//end add method

**public** **boolean** offer(OrderDetail e)

{

**return** add(e);

}//end offer method

**public** OrderDetail remove()

{

**try** {

isValid();

**if** (orders.size() <= 0) {

**throw** **new** NoSuchElementException();

}

**else**{

**return** orders.remove(0);

}//end else

}//end try

**catch** (IllegalStateException e) {

**return** **null**;

}//end catch IllegalStateException

}//end remove

**public** OrderDetail poll()

{

**return** remove();

}

**public** OrderDetail peek()

{

**return** orders.get(0);

}//end peek method

}//end class

OrderDetail.java:

/\*\*

\* File Name: OrderDetail.java

\* Date: Aug 23, 2023

\* Author: Richard Fritsche

\* Project Name: portfolio

\* Project Purpose:

\* Algorithm Used:

\* Program Inputs:

\* Program Limitations:

\* Program Errors:

\*/

package portfolio;

/\*\*

\* @author richard

\*

\*/

public class OrderDetail {

/\*\*

\* Constructor

\*

\* @param

\*

\* @param lastName

\* @param orderNumber

\* @param orderCost

\*/

public OrderDetail(String lastName, int orderNumber, double orderCost)

{

veritas = false;

this.lastName = lastName;

this.orderNumber = orderNumber;

this.orderCost = orderCost;

veritas = true;

}//end constructor

public OrderDetail(String lastName, int orderNumber) {

this(lastName, orderNumber, 0.0);

}//end constructor

/\*\*

\* Constructor

\*

\* If this constructor is ever called, it will invalidate the object

\* @param

\*

\*/

private OrderDetail() {

veritas = false;

}//end private constructor

/\*\*

\* @return the orderCost

\*/

public double getOrderCost()

{

return orderCost;

}//end getOrderCost method

/\*\*

\* @param orderCost the orderCost to set

\*/

public void setOrderCost(double orderCost)

{

this.orderCost = orderCost;

}//end set OrderCost method

/\*\*

\* @return the lastName

\*/

public String getLastName()

{

return lastName;

}//end getLastName method

/\*\*

\* @return the orderNumber

\*/

public int getOrderNumber()

{

return orderNumber;

}//end getOrderNumber method

//class variables

private boolean veritas;//boolean variable to confirm object is valid.

private String lastName;

private int orderNumber;

private double orderCost;

private void isValid() {

if (!veritas) {

throw new IllegalStateException();

}//end if

}//end isValid method

public void display() {

System.out.println("Customer Last Name: " + getLastName());

System.out.println("Order Number: " + getOrderNumber());

System.out.printf("Order Total: $%,.2f\n\n", getOrderCost());

}

}//end class

OrderDisplay.java:

/\*\*

\* File Name: OrderDisplay.java

\* Date: Sep 7, 2023

\* Author: Richard Fritsche

\* Project Name: portfolio

\* Project Purpose:

\* Algorithm Used:

\* Program Inputs:

\* Program Limitations:

\* Program Errors:

\*/

package portfolio;

import java.util.ArrayList;

import java.util.NoSuchElementException;

/\*\*

\* @author richard

\*

\*/

public class OrderDisplay {

//class variables

private ArrayList<OrderDetail> nameOrderedList;//copy of the customer orders by last name

private OrderDetail[] nameOrdered;//copy of the customer orders by last name

private ArrayList<OrderDetail> numOrderedList;//copy of the custormer orders by order number

private OrderDetail[] numOrdered;

private int size;

private boolean veritas;//boolean variable to confirm object is valid.

private void isValid() {

if (!veritas) {

throw new IllegalStateException();

}//end if

}//end isValid method

public OrderDisplay() {

veritas = false;

nameOrderedList = new ArrayList<>();

nameOrdered = new OrderDetail[10];

numOrderedList = new ArrayList<>();

numOrdered = new OrderDetail[10];

size = 0;

veritas = true;

}//end constructor

public void add(OrderDetail o) {

try {

isValid();

if(size == nameOrdered.length) {//if the name Ordered array is full

resize();

}//end if

// nameOrderedList.add(o);

nameOrdered[size] = o;

// numOrderedList.add(o);

numOrdered[size] = o;

size++;

nameSort(0,size);

numSort(0,size);

displayByName();

displayByNum();

}//end try

catch (IllegalStateException e) {

return;

}//end catch

}//end add method

/\*\*

\* resize

\*

\*/

private void resize()

{

OrderDetail[]temp = new OrderDetail[size \* 2];

for(int i = 0; i<nameOrdered.length;i++ ) {//for each element currently in nameOrdered array

temp[i] = nameOrdered[i];//copy element over to temp array

}//end for

nameOrdered = temp;

for(int i = 0; i <numOrdered.length; i++) {//for each element currently in numOrdered array

temp[i] = numOrdered[i];

}//end for

numOrdered = temp;

}//end resize method

/\*\*

\* displayByNum

\*

\*/

public void displayByNum()

{

System.out.println("Printing List ordered by Order Number:");

for(int i = 0; i < size; i++) {//for each order in the name ordered list

//System.out.println("Index: " + i);

numOrdered[i].display();

}//for each loop

}//end displayByNum method

/\*\*

\* displayByName

\*

\*/

public void displayByName()

{

System.out.println("Printing List ordered by Customer Last Name:");

for(int i = 0; i < size; i++) {//for each order in the name ordered list

//System.out.println("Index: " + i);

nameOrdered[i].display();

}//for each loop

}//end displayByName method

/\*\*

\* numSort

\* @param end

\* @param start

\*

\*/

private void numSort(int start, int end)

{

int lower = start;

int upper = end-1;

int mid = (start + end)/2;

OrderDetail temp;

//System.out.println("Start numSort method with indexes " + start + " through " + end);

if(start >= upper) {//if the start and end are the same

//System.out.println("Set of 1, returning");

return;

}//end if

if((end - start) < 3) {//if there are two elements in the subarray

//System.out.println("Set of 2");

//System.out.println("Comparing " + nameOrdered[start].getLastName() + " to " + nameOrdered[end-1].getLastName() + " results as " + nameOrdered[start].getLastName().compareTo(nameOrdered[end-1].getLastName()));

if (numOrdered[start].getOrderNumber() < numOrdered[upper].getOrderNumber()) {//if the first element is lower value than the second value

//System.out.println("Swapping the elements");

temp = numOrdered[start];

numOrdered[start] = numOrdered[upper];

numOrdered[upper] = temp;

}//end if

//System.out.println("Returning");

return;

}//end if

while ((lower < mid) ||(upper > mid)) {//while lower is less than mid or upper is less than mid

//System.out.println("Mid " + mid + " is between Lower " + lower + " and Upper " + upper);

while ((numOrdered[lower].getOrderNumber() > numOrdered[mid].getOrderNumber()) && (lower < mid)) {//while the elements set before the mid point are less than the mid point element and lower is less than mid

//System.out.println("increment lower");

lower++;

}//end while

while ((numOrdered[upper].getOrderNumber() < numOrdered[mid].getOrderNumber()) && (upper > mid)) {//while the elements set before the mid point are less than the mid point element and lower is less than mid

//System.out.println("decrement upper");

upper--;

}//end while

//System.out.println("Swap upper and lower");

swap(numOrdered, lower, upper);

lower = (lower < mid) ? lower + 1 : lower;

upper = (upper > mid) ? upper - 1 : upper;

}//end while

//ensure that the mid element is in the proper spot.

if(numOrdered[mid].getOrderNumber() > numOrdered[lower].getOrderNumber()) {//if mid should be sorted before lower

swap(numOrdered, mid, lower);

}//end if

if(numOrdered[mid].getOrderNumber() < numOrdered[upper].getOrderCost()) {//if the mid should be sorted after the upper

swap(numOrdered, mid, upper);

}//end if

//System.out.println("Run on lower half");

numSort(start, mid+1);

//System.out.println("Run on upper half");

numSort(mid, end);

}//end numSort method

/\*\*

\* swap

\*

\* @param numOrdered2

\* @param lower

\* @param upper

\*/

private void swap(OrderDetail[] orders, int lower, int upper)

{

OrderDetail temp = orders[lower];

orders[lower] = orders[upper];

orders[upper] = temp;

}

/\*\*

\* nameSort

\* @param start

\* @param end

\*

\*/

private void nameSort(int start, int end)

{

int lower = start;

int upper = end-1;

int mid = (start + end)/2;

OrderDetail temp;

//System.out.println("Start nameSort method with indexes " + start + " through " + end);

if(start >= upper) {//if the start and end are the same

//System.out.println("Set of 1, returning");

return;

}//end if

if((end - start) < 3) {//if there are two elements in the subarray

//System.out.println("Set of 2");

///System.out.println("Comparing " + nameOrdered[start].getLastName() + " to " + nameOrdered[end-1].getLastName() + " results as " + nameOrdered[start].getLastName().compareTo(nameOrdered[end-1].getLastName()));

if (nameOrdered[start].getLastName().compareToIgnoreCase(nameOrdered[upper].getLastName()) < 0) {//if the first element is lower value than the second value

//System.out.println("Swapping the elements");

temp = nameOrdered[start];

nameOrdered[start] = nameOrdered[upper];

nameOrdered[upper] = temp;

}//end if

//System.out.println("Returning");

return;

}//end if

while ((lower < mid) ||(upper > mid)) {//while lower is less than mid or upper is less than mid

//System.out.println("Mid " + mid + " is between Lower " + lower + " and Upper " + upper);

while ((nameOrdered[lower].getLastName().compareToIgnoreCase(nameOrdered[mid].getLastName()) > 0) && (lower < mid)) {//while the elements set before the mid point are less than the mid point element and lower is less than mid

//System.out.println("increment lower");

lower++;

}//end while

while ((nameOrdered[upper].getLastName().compareToIgnoreCase(nameOrdered[mid].getLastName()) < 0) && (upper > mid)) {//while the elements set before the mid point are less than the mid point element and lower is less than mid

//System.out.println("decrement upper");

upper--;

}//end while

//System.out.println("Swap upper and lower");

temp = nameOrdered[lower];

nameOrdered[lower] = nameOrdered[upper];

nameOrdered[upper] = temp;

lower = (lower < mid) ? lower + 1 : lower;

upper = (upper > mid) ? upper -1 : upper;

}//end while

//System.out.println("Run on lower half");

nameSort(start, mid+1);

//System.out.println("Run on upper half");

nameSort(mid, end);

}//end nameSort method

public void remove(OrderDetail order) {

int i;

if (size <= 0) {//if there are no elements to remove

throw new NoSuchElementException();

}//end if

//System.out.println("Searching in nameOrdered");

i = findElement(nameOrdered, order);

nameOrdered[i] = null;

nameOrdered[i] = nameOrdered[size - 1];

//System.out.println("Searching in numOrdered");

i= findElement(numOrdered, order);

numOrdered[i] = null;

numOrdered[i] = numOrdered[size - 1];

size--;

nameSort(0, size);

numSort(0, size);

}//end remove method

/\*\*

\* findElement

\*

\* @param nameOrdered2

\* @param order

\*/

private int findElement(OrderDetail[] orders, OrderDetail order)

{

for(int i = 0; i < orders.length; i++) {//for each element in orders

if(orders[i] == order) {//if the order has been found

//System.out.println("Found");

return i;

}//end if

}//end for

//System.out.println("Not found");

return -1;

}

}//end class

Order Handler.java

/\*\*

\* File Name: OrderHandler.java

\* Date: Sep 7, 2023

\* Author: Richard Fritsche

\* Project Name: portfolio

\* Project Purpose:

\* Algorithm Used:

\* Program Inputs:

\* Program Limitations:

\* Program Errors:

\*/

package portfolio;

import java.util.InputMismatchException;

import java.util.NoSuchElementException;

import java.util.Scanner;

/\*\*

\* @author richard

\*

\*/

public class OrderHandler {

private static Scanner scnr;

private static Order list;

private static OrderDisplay display;

/\*\*

\* main

\*

\* @param args

\*/

public static void main(String[] args) {

scnr = new Scanner(System.in);

list = new Order();

display = new OrderDisplay();

OrderDetail order;

int option;

do {

option = menu();

switch (option) {//switch on result of the menu method

case 1:

scnr.nextLine();

add();

break;

case 2:

scnr.nextLine();

System.out.println("Removing: ");

order = remove();

if(order != null) {//if an element was returned

order.display();

}//end if

break;

case 3:

scnr.nextLine();

display();

break;

case 4:

scnr.close();

break;

default:

System.out.println("\n\nInvalid entry. Please try again.");

}//end switch

System.out.println("\n\n");

}while(option!=4);//end do while loop

}//end main method

private static int menu() {

try {

System.out.println("Welcome to the order processing system.");

System.out.println("Please select what you want to do.");

System.out.println("1: Add a new order to the system");

System.out.println("2: Remove the next order from the system");

System.out.println("3: Display the list of orders in the system");

System.out.println("4: Exit the system");

System.out.print("Which option do you want?");

return scnr.nextInt();

}//end try

catch (InputMismatchException e) {

scnr.nextLine();

return 0;//return an invalid value if the value cannot be parsed to a number

}//end catch

}

private static void add() {

try {

String name;

int orderNum;

double total;

OrderDetail temp;

System.out.print("\n\nGreat! What is the customer's last name? ");

name = scnr.nextLine();

System.out.print("What is the order number? ");

orderNum = scnr.nextInt();

scnr.nextLine();

System.out.print("And now was was the total to the order? $");

total = scnr.nextDouble();

scnr.nextLine();

temp = new OrderDetail(name, orderNum, total);

System.out.println("\n");

list.add(temp);

System.out.println("\n");

display.add(temp);

}//end try

catch (InputMismatchException e) {

System.out.println("\n\nThere seems to be a problem. Going back to the menu.");

scnr.nextLine();

}//end catch

}//end add method

private static OrderDetail remove() {

try {

OrderDetail temp = list.remove();

display.remove(temp);

// System.out.println("\n");

// display.displayByName();

// System.out.println("\n");

// display.displayByNum();

return temp;

}//end try

catch (NoSuchElementException e) {

System.out.println("There were no elements to remove.");

return null;

}//end catch

}//end remove method

private static void display() {

list.display();

// System.out.println("\n\n");

// display.displayByName();

// System.out.println("\n\n");

// display.displayByNum();

}//end display method

}//end class

Portfolio Project screenshots:

Start:





















