UOW-logo

Informatics Institute of Technology

Department of Computing

Software Development II Coursework Report

Module : 4COSC010C.3: Software Development II

Module Leader : Mr Deshan Sumanathilake

Date of submission : 08/08/2022

Student ID : <IIT No – 20210057> / <w1912796 >

Student First Name : Sandeep

Student Surname : Abeykoon

"I confirm that I understand what plagiarism / collusion / contract cheating is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged."

Name : Sandeep Chanura Abeykoon

Student ID : 20210057/w1912796

**Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Test Case** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | Fuel Queue Initialized Correctly  After program starts, 100 or VFQ | Displays ‘empty’ for all queues. | Displays ‘empty’ for all Queues. | Pass |
| 2 | Add passenger “Jane” to Queue 1  102 or ACQ  Enter Queue: 2  Enter Name: Jane | Display “The customer, Jane is successfully added to the pump 1 Queue....”” | Display “The customer, Jane is successfully added to the pump 1 Queue....”” | Pass |
| 3 | Remove passenger 3 from Queue 2  103 / RCQ  Enter Queue : 2 | Display “The Queue 2 have no customers to remove....” | Display “The Queue 2 have no customers to remove....’ | Pass |
| 4 | 100 / VFQ ----> View all Fuel Queues | Displays all the fuel queues with the data | Displays all the fuel queues with data | Pass |
| 5 | 101 / VEQ ----> View all Empty Queues | Displays all the empty queues with the data | Displays all the empty queues with the data | Pass |
| 6 | Adding passenger “Sandeep” to Queue 3  102 / ACQ ----> Add customer to a Queue | Enter the Queue number (1/2/3) : 3  Enter the Customer name : Sandeep  The customer, Sandeep is successfully added to the pump 3 Queue.... | Enter the Queue number (1/2/3) : 3  Enter the Customer name : Sandeep  The customer, Sandeep is successfully added to the pump 3 Queue.... | Pass |
| 7 | Removing the third passenger from the Queue 2  103 / RCQ ----> Remove a customer from a Queue | nter the Queue number (1/2/3) : 2  Enter the the location of the customer to be removed (1 - 6) : 3  The customer of the location 3 is successfully removed... | nter the Queue number (1/2/3) : 2  Enter the the location of the customer to be removed (1 - 6) : 3  The customer of the location 3 is successfully removed... | Pass |
| 8 | Removing the served customer of the Queue 1  104 / PCQ ----> Remove a served customer | Enter your option : 104  Enter the Queue number (1/2/3) : 1  The served customer of queue 1 is removed | Enter your option : 104  Enter the Queue number (1/2/3) : 1  The served customer of queue 1 is removed | Pass |
| 9 | 105 / VCS ----> View Customers Sorted in alphabetical order | Displaying the customers sorted in alphabetical order | Displays the customers sorted in alphabetical order | Pass |
| 10 | 106 / SPD ----> Store Program Data into file | The program data stores to the text file | Stores the data into the text file | Pass |
| 11 | 107 / LPD ----> Load Program Data from file | The data is loaded from the text file to the program | The data loads from the text file to the program | Pass |
| 12 | 108 / STK ----> View Remaining Fuel Stock | The current remaining fuel quantity ----> 6550 | The current remaining fuel quantity ----> 6550 | Pass |
| 13 | 109 / AFS ----> Add Fuel Stock | Enter your option : 109  The current remaining fuel quantity ----> 6550  Enter the fuel stock to be added : 10  The new fuel quantity has been successfully added.... | Enter your option : 109  The current remaining fuel quantity ----> 6550  Enter the fuel stock to be added : 10  The new fuel quantity has been successfully added.... | Pass |
| 14 | 999 / EXT ----> Exit the Program | Enter your option : 999  Good bye!!!...  Program terminated!!!. | Enter your option : 999  Good bye!!!...  Program terminated!!!. | Pass |
| 15 | 110 / IFQ ----> View Queue Income | The Incomes of all the Queeus should be displayed | The income of the Queue 1 is : 4300.0  The income of the Queue 2 is : 2150.0  The income of the Queue 3 is : 5160.0  The income of the Queue 4 is : 19350.0  The income of the Queue 5 is : 12900.0 | Pass |
| 16 | Entering an invalid input for menu option  Enter your option : dsd | The program identifies the invalid input | Enter your option : dsd  Please enter a valid option!!!.... | Pass |
| 17 | Entering a number for the passenger name  Enter the customer first name : 66456 | The program identifies the invalid input | Enter the customer first name : 66456  Invalid input (Input cannot be completely a number.... | Pass |
| 18 | Entering Strings for number of Liters | The pogram identifies the invalid input | Enter the number of liters of fuel required : gfgdg  Invalid input!!! Only numbers are allowed.... | Pass |
| 19 | Adding additional fuel exceeding the tank capacity | The program identifies the problem and informs the user | Enter your option : 109  The current remaining fuel quantity ----> 6488  Enter the fuel stock to be added : 1000 | Pass |
| 20 | Adding a customer when all the queues are full | The customer is added to the waiting queue | Enter the customer first name : Sandeep  Enter the customer last name : Abeykoon  Enter the customer vehicle number : CAL-4578  Enter the number of liters of fuel required : 10  All the queues are full. The passenger is added to the waiting queue... | Pass |

**Discussion**

<<Discussion of how you chose your test cases to ensure that your tests cover all aspects of your program >>

The program starts to work

**Code :**

**Array Version**

import java.util.Arrays;  
import java.util.Scanner;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.File; //File class  
import java.io.FileNotFoundException; //Handles file not found errors  
  
public class FuelCenter {  
 */\*\*  
 \* This is the main method  
 \** ***@param*** *args is a String array which is used to take the system arguments  
 \*/* public static void main(String[] args) {  
 //Declaring and Initializing the local variables  
 Scanner input = new Scanner(System.*in*);  
 String[][] pumpQueues = new String[3][6]; // 2D array initialized.  
 int fuelStock = 6600;  
  
 for (String[] rows : pumpQueues) { //Changing the default array values to "Empty"  
 Arrays.*fill*(rows, "empty");  
 }  
 mainLoop:  
 while (true) {  
 //User Menu  
 System.*out*.println("""  
 \n  
 --------------------MENU-----------------------  
 \n\t100 / VFQ ----> View all Fuel Queues  
 \t101 / VEQ ----> View all Empty Queues  
 \t102 / ACQ ----> Add customer to a Queue  
 \t103 / RCQ ----> Remove a customer from a Queue  
 \t104 / PCQ ----> Remove a served customer  
 \t105 / VCS ----> View Customers Sorted in alphabetical order  
 \t106 / SPD ----> Store Program Data into file  
 \t107 / LPD ----> Load Program Data from file  
 \t108 / STK ----> View Remaining Fuel Stock  
 \t109 / AFS ----> Add Fuel Stock  
 \t999 / EXT ----> Exit the Program  
   
 ----------------------------------------------  
 """);  
  
 System.*out*.println(fuelStock <= 500 ? "WARNING !!!... THE FUEL STOCK IS VERY LOW ( " + fuelStock + " Liters )\n" : "");  
 // Getting user option  
 System.*out*.print("Enter your option : ");  
 String option = input.next();  
  
 switch (option.toUpperCase()) {  
 case "100", "VFQ" -> *viewQueues*(pumpQueues, false, input);  
 case "101", "VEQ" -> *viewQueues*(pumpQueues, true, input);  
 case "102", "ACQ" -> fuelStock = *addCustomer*(pumpQueues, input, fuelStock);  
 case "103", "RCQ" -> fuelStock = *removeCustomer*(pumpQueues, input, fuelStock);  
 case "104", "PCQ" -> *remServedCustomer*(pumpQueues, input);  
 case "105", "VCS" -> *sorting*(pumpQueues, input);  
 case "106", "SPD" -> *storeData*(pumpQueues, fuelStock, input);  
 case "107", "LPD" -> fuelStock = *loadData*(pumpQueues, fuelStock, input);  
 case "108", "STK" -> *fuelManage*(fuelStock, false, input);  
 case "109", "AFS" -> fuelStock = *fuelManage*(fuelStock, true, input);  
 case "999", "EXT" -> {  
 System.*out*.println("\nGood bye!!!...\nProgram terminated!!!.");  
 break mainLoop;  
 }  
 default -> System.*out*.println("\nPlease enter a valid option!!!....");  
 }  
 }  
 }  
  
 */\*\*  
 \* This method takes an integer user input within a specific range and returns the int value  
 \** ***@param*** *prompt displays when getting the user input  
 \** ***@param*** *input is the Scanner object  
 \** ***@param*** *endRange is the input validation upper range  
 \** ***@return*** *returns the validated integer output  
 \*/* private static int inputValidation(String prompt, Scanner input, int endRange) {  
 int intValue;  
  
 while (true) {  
 System.*out*.print("\n" + prompt);  
 if (input.hasNextInt()) {  
 intValue = input.nextInt();  
 input.nextLine();  
  
 if (intValue >= 1 & intValue <= endRange) {  
 break;  
 }else if (1 != endRange) {  
 System.*out*.println("Out of range!!!. Only values from " + 1 + " to " + endRange + " is allowed....");  
 }else {  
 System.*out*.println("Invalid Input!!!. Only " + 1 + " is allowed");  
 }  
 } else{  
 System.*out*.print("Invalid input!!! Only numbers are allowed....\n");  
 input.next();  
 }  
 }  
 return intValue;  
 }  
  
 */\*\*  
 \* This method prints all the fuel queues or "Empty" queues  
 \** ***@param*** *array A 2D String array which contains the queue elements  
 \** ***@param*** *filter A boolean parameter to check whether to filter the "Empty" queue slots or not  
 \** ***@param*** *input is a Scanner object  
 \*/* private static void viewQueues(String[][] array, Boolean filter, Scanner input) {  
 int NumberOfElements = ((array.length) \* (array[0].length)); //Each sub array contains equal number of elements  
  
 System.*out*.println(filter?"\n\t----------All the Empty Queues available----------\n":"\n\t----------All the Fuel Queues available----------\n");  
  
 for (int i = 1; i <= array.length; i++) { //This for loop prints the headings  
 System.*out*.print("QUEUE " + i + "\t\t\t\t\t");  
 }  
 System.*out*.println("\n");  
  
 for (int i = 0, maxIndex = 0, minIndex = 0; (i < NumberOfElements); i++, maxIndex++) {  
 if (maxIndex == array.length) {  
 maxIndex = 0;  
 minIndex++;  
 System.*out*.println();  
 }  
 String element = array[maxIndex][minIndex];  
 System.*out*.printf("%-24s", ((minIndex + 1) + ") " + (filter ? (element.equals("empty") ? "empty" : "") : element)));  
 }  
 System.*out*.print("\n");  
 *inputValidation*("Enter 1 to continue to the main menu : ", input, 1);  
 }  
  
 */\*\*  
 \* This method is to add a new customer to a fuel queue  
 \** ***@param*** *array A 2D String array which contains the queue elements  
 \** ***@param*** *input is a Scanner object  
 \** ***@param*** *fuelStock is an integer value which holds the current fuel stock  
 \** ***@return*** *returns the updated fuel stock  
 \*/* private static int addCustomer(String[][] array, Scanner input, int fuelStock) {  
 System.*out*.println("\n-------- ADD A CUSTOMER --------");  
 int queueNum = *inputValidation*("Enter the Queue number (1/2/3) : ", input, 3);  
 int lastIndex = (array[queueNum - 1].length) - 1;  
 String cusName = "";  
  
 // Checks the last index of the array is "Empty" or not to confirm the queue is full or not  
 if (!(array[queueNum - 1][lastIndex].equals("empty"))) {  
 System.*out*.println("The Queue " + queueNum + " is not empty at the moment, please try again later...");  
 } else {  
 while (true) { // Assuming a customer name can be partially numerical for identification purposes  
 try {  
 System.*out*.print("Enter the Customer name : "); //But cannot be completely numerical  
 cusName = input.nextLine();  
 Integer.*parseInt*(cusName);  
 System.*out*.println("Customer Name cannot be completely a number !!!.... Please re-enter the name....\n");  
 } catch (Exception e) {  
 break;  
 }  
 }  
 for (int i = 0; i < (array[queueNum - 1].length); i++) {  
 if (array[queueNum - 1][i].equals("empty")) {  
 array[queueNum - 1][i] = cusName.toUpperCase();  
 // Customer name is updated to the array in uppercase for sorting purpose.  
 System.*out*.println("The customer, " + cusName + " is successfully added to the pump " + queueNum + " Queue....");  
 fuelStock -= 10;  
 break;  
 }  
 }  
 }  
 if(*inputValidation*("1 ----> Go to main menu\n2 ----> Add another customer\n\tOption : ", input, 2) == 2){  
 fuelStock = *addCustomer*(array, input, fuelStock);  
 }  
 return fuelStock;  
 }  
  
 */\*\*  
 \* This method is to remove a customer from any location of any fuel queue  
 \** ***@param*** *array A 2D String array which contains the queue elements  
 \** ***@param*** *input is a Scanner object  
 \** ***@param*** *fuelStock is an integer value which holds the current fuel stock  
 \** ***@return*** *returns the updated fuel stock  
 \*/* private static int removeCustomer(String[][] array, Scanner input, int fuelStock) {  
 int location;  
 System.*out*.println("\n-------- REMOVE A CUSTOMER --------");  
  
 int queueNum = *inputValidation*("Enter the Queue number (1/2/3) : ", input, 3);  
  
 if (array[queueNum - 1][0].equals("empty")) {  
 System.*out*.println("The Queue " + queueNum + " have no customers to remove....");  
 } else {  
 location = *inputValidation*("Enter the the location of the customer to be removed (1 - 6) : ", input, 6);  
  
 if (array[queueNum - 1][location - 1].equals("empty")) {  
 System.*out*.println("The location " + location + " was already empty....");  
 } else {  
 // When a customer is removed all the customers after the removed customer will be stepped down  
 for (int i = location - 1; i < (array[queueNum - 1].length - 1); i++) {  
 array[queueNum - 1][i] = array[queueNum - 1][i + 1];  
 }  
 array[queueNum - 1][array[queueNum - 1].length - 1] = "empty";  
 System.*out*.println("The customer of the location " + location + " is successfully removed...");  
 fuelStock = ((fuelStock + 10) > 6600? fuelStock : fuelStock + 10);  
 }  
 }  
 if(*inputValidation*("1 ----> Main Menu\n2 ----> Remove another customer\n\tOption : ", input, 2) == 2 ){  
 *removeCustomer*(array, input, fuelStock);  
 }  
 return fuelStock;  
 }  
  
 */\*\*  
 \* This method checks and removes the very first customer of a queue  
 \** ***@param*** *array A 2D String array which contains the queue elements  
 \** ***@param*** *input is a Scanner object  
 \*/* private static void remServedCustomer(String[][] array, Scanner input) {  
 int queueNum = *inputValidation*("Enter the Queue number (1/2/3) : ", input, 3);  
  
 //Assumes that the first customer of a queue is the served customer.  
 if (array[queueNum - 1][0].equals("empty")) {  
 System.*out*.println("No served customers can be found in queue " + queueNum);  
 } else {  
 for (int i = 0; i < (array[queueNum - 1].length - 1); i++) {  
 array[queueNum - 1][i] = array[queueNum - 1][i + 1];  
 }  
 array[queueNum - 1][array[queueNum - 1].length - 1] = "empty";  
 System.*out*.println("The served customer of queue " + queueNum + " is removed");  
 }  
 if(*inputValidation*("1 ----> Main Menu\n2 ----> Remove another served customer\n\tOption : ", input, 2) == 2 ) {  
 *remServedCustomer*(array, input);  
 }  
 }  
  
 */\*\*  
 \* This method is to sort the available customers of all the queues separately and display the sorted data  
 \** ***@param*** *array A 2D String array which contains the queue elements  
 \*/* private static void sorting(String[][] array, Scanner input) {  
 // Checking whether the arrays have customers before sorting (increases optimization)  
 if (array[0][0].equals("empty") & array[1][0].equals("empty") && array[2][0].equals("empty")) {  
 System.*out*.println("There are no any customers in the Queues to be sorted... ");  
 *inputValidation*("Enter 1 to continue to the main menu : ", input, 1);  
 } else {  
 String[][] sortingArray = new String[3][6]; // new array is created to sort data  
  
 for (int i = 0; i < array.length; i++) { // Copying the array into created array  
 System.*arraycopy*(array[i], 0, sortingArray[i], 0, array[i].length);  
 }  
 for (String[] subArray : sortingArray) { // Sorting the array  
 for (int i = 0; i < (subArray.length - 1); i++) {  
 for (int j = (i + 1); j < (subArray.length); j++) {  
 if ((subArray[i].compareTo(subArray[j])) > 0) {  
 String temp = subArray[i];  
 subArray[i] = subArray[j];  
 subArray[j] = temp;  
 }  
 }  
 }  
 }  
 *viewQueues*(sortingArray, false, input); // Displaying the sorted array  
 }  
 }  
  
 */\*\*  
 \* This method extracts the data from the arrays and writes them to a text file  
 \* Writes the current fuel quantity to the text file  
 \** ***@param*** *array A 2D String array which contains the queue elements  
 \** ***@param*** *fuelStock is an integer value which holds the current fuel stock  
 \** ***@param*** *input is a Scanner object  
 \*/* private static void storeData(String[][] array, int fuelStock, Scanner input) {  
 try {  
 FileWriter writeData = new FileWriter("fuelCenter.txt"); //creating a file object  
 // Writing the data in the arrays to the file  
 for (String[] subArray : array) {  
 for (String data : subArray) {  
 writeData.write(data + "\n");  
 }  
 }  
 //Writing the current fuel stock  
 writeData.write(fuelStock + "\n");  
 writeData.close(); // closing the file writer  
 System.*out*.println("The data has been successfully written to the file....");  
  
 } catch (IOException e) {  
 System.*out*.println("Ooops!!!... an error occurred while writing data to the file");  
 }  
 *inputValidation*("Enter 1 to continue to the main menu : ", input, 1);  
 }  
  
 */\*\*  
 \* This method reads the data from a text file and updates the arrays  
 \* updates the fuel stock to the fuelStock variable  
 \** ***@param*** *array A 2D String array which contains the queue elements  
 \** ***@param*** *fuelStock is an integer value which holds the current fuel stock  
 \** ***@param*** *input is a Scanner object  
 \** ***@return*** *returns the new updated fuel stock to the program  
 \*/* private static int loadData(String[][] array, int fuelStock, Scanner input) {  
 try {  
 File fileObject = new File("fuelCenter.txt");  
 Scanner fileRead = new Scanner(fileObject);  
  
 // Updating the arrays from the file in the data  
 while (fileRead.hasNextLine()) {  
 for (int i = 0; i < array.length; i++) {  
 for (int j = 0; j < array[i].length; j++) {  
 array[i][j] = fileRead.nextLine();  
 }  
 }  
 fuelStock = Integer.*parseInt*(fileRead.nextLine()); // Updating the fuel stock from the file  
 }  
 fileRead.close();  
 System.*out*.println("The data has been successfully loaded from the file....");  
  
 } catch (FileNotFoundException e) {  
 System.*out*.println("An error occurred while reading the file ");  
 }  
 *inputValidation*("Enter 1 to continue to the main menu : ", input, 1);  
 return fuelStock;  
 }  
  
 */\*\*  
 \* This method is to view the current fuel quantity or add the fuel quantity  
 \** ***@param*** *fuelStock is an integer value which holds the current fuel stock  
 \** ***@param*** *add\_fuel is a boolean parameter which selects whether to add fuel or not  
 \** ***@param*** *input is a Scanner object  
 \** ***@return*** *returns the updated fuel stock to the program  
 \*/* private static int fuelManage(int fuelStock, boolean add\_fuel, Scanner input) {  
  
 System.*out*.println("\nThe current remaining fuel quantity ----> " + fuelStock); //Current fuel stock  
 // if the fuel level is less than or equal 500 Liters a warning message will be displayed  
 System.*out*.println(fuelStock <= 500 ? "WARNING !!!... THE FUEL STOCK IS VERY LOW ( " + fuelStock + " Liters )" : "");  
  
 if (add\_fuel) { // If fuel adding needs to be done  
 int newFuel = *inputValidation*("Enter the fuel stock to be added : ", input, 6600);  
 if ((fuelStock + newFuel) < 6600) {  
 fuelStock += newFuel;  
 System.*out*.println("The new fuel quantity has been successfully added....");  
 }else{  
 System.*out*.println("The fuel quantity exeeds the tank capacity !!!. Do you want to try again?");  
 if(*inputValidation*("1 ----> Main Menu\n2 ----> Add Fuel\n\tOption : ", input, 2) == 2) {  
 *fuelManage*(fuelStock, true, input);  
 }  
 }  
 }  
 *inputValidation*("Enter 1 to continue to the main menu : ", input, 1);  
 return fuelStock;  
 }  
}

**Class Version**

**FuelCenter.java**

<<END>>