UOW-logo

Informatics Institute of Technology

Department of Computing

Software Development II Coursework Report

Module : 4COSC010C: Software Development II

Module Leader : Iresh Bandara

Date of submission : 26/07/2021

Student ID : 20200868/ w1837850

Student First Name : Umesh

Student Surname : Dharmasena

"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 : Umesh Dharmasena

Student ID : 20200868

**Test Cases:**

**Task\_1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Test Case** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | (Booths Initialised correctly)  After program starts, 100 or VVB | Displays ‘empty’ for all booths | Displays ‘empty’ for all booths | Pass |
| 2 | After program starts, 102 or APB | Displays all ‘empty’ booths and asks for name and booth number | Displays all ‘empty’ booths and asks for name and booth number | Pass |
| 3 | After program starts, 101 or VEB | Displays all ‘empty’ booths | Displays all ‘empty’ booths | Pass |
| 4 | After program starts, 103 or RPB | Asks for booth number and change value of booth to ‘empty’ | Asks for booth number and change value of booth to ‘empty’ | Pass |
| 5 | After program starts, 104 or VPS | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Pass |
| 6 | After program starts, 105 or SPD | Writes all the elements in the booths array to a text file. | Writes all the elements in the booths array to a text file. | Pass |
| 7 | After program starts, 106 or LPD | Writes all the records in the text file as elements in the booths array | Writes all the records in the text file as elements in the booths array | Pass |
| 8 | After program starts, 107 or VRV | Displays how many vaccines are remaining | Displays how many vaccines are remaining | Pass |
| 9 | After program starts, 108 or AVS | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Pass |
| 10 | After program starts, 999 or EXT | Exits program | Exits program | Pass |

**Task\_2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test Case | Expected Result | Actual Result | Pass/Fail |
| 1 | After program starts, 100 or VVB | Displays ‘empty’ for all booths | Displays ‘empty’ for all booths | Pass |
| 2 | After program starts, 102 or APB | Displays all ‘empty’ booths and asks for name and booth number | Displays all ‘empty’ booths and asks for name and booth number | Pass |
| 3 | After program starts, 101 or VEB | Displays all ‘empty’ booths | Displays all ‘empty’ booths | Pass |
| 4 | After program starts, 103 or RPB | Asks for booth number and change value of booth to ‘empty’ | Asks for booth number and change value of booth to ‘empty’ | Pass |
| 5 | After program starts, 104 or VPS | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Pass |
| 6 | After program starts, 105 or SPD | Writes all the elements in the booths array to a text file. | Writes all the elements in the booths array to a text file. | Pass |
| 7 | After program starts, 106 or LPD | Writes all the records in the text file as elements in the booths array | Writes all the records in the text file as elements in the booths array | Pass |
| 8 | After program starts, 107 or VRV | Displays how many vaccines are remaining | Displays how many vaccines are remaining | Pass |
| 9 | After program starts, 108 or AVS | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Pass |
| 10 | After program starts, 999 or EXT | Exits program | Exits program | Pass |

**Task\_3\_1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test Case | Expected Result | Actual Result | Pass/Fail |
| 1 | After program starts, 100 or VVB | Displays ‘empty’ for all booths | Displays ‘empty’ for all booths | Pass |
| 2 | After program starts, 102 or APB | Displays all ‘empty’ booths which belong to the vaccine type they choose and asks for First name, Surname, which vaccination they would like and booth number | Displays all ‘empty’ booths which belong to the vaccine type they choose and asks for First name, Surname, which vaccination they would like and booth number | Pass |
| 3 | After program starts, 101 or VEB | Displays all ‘empty’ booths | Displays all ‘empty’ booths | Pass |
| 4 | After program starts, 103 or RPB | Asks for booth number and change value of booth to ‘empty’ | Asks for booth number and change value of booth to ‘empty’ | Pass |
| 5 | After program starts, 104 or VPS | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Pass |
| 6 | After program starts, 105 or SPD | Writes all the elements in the booths array to a text file. | Writes all the elements in the booths array to a text file. | Pass |
| 7 | After program starts, 106 or LPD | Writes all the records in the text file as elements in the booths array | Writes all the records in the text file as elements in the booths array | Pass |
| 8 | After program starts, 107 or VRV | Displays how many vaccines are remaining | Displays how many vaccines are remaining | Pass |
| 9 | After program starts, 108 or AVS | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Pass |
| 10 | After program starts, 999 or EXT | Exits program | Exits program | Pass |

**Task\_3\_2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test Case | Expected Result | Actual Result | Pass/Fail |
| 1 | After program starts, 100 or VVB | Displays ‘empty’ for all booths | Displays ‘empty’ for all booths | Pass |
| 2 | After program starts, 102 or APB | Displays all ‘empty’ booths which belong to the vaccine type they choose and asks for First name, Surname, Age, City, NIC or Passport Number, which vaccination they would like and booth number | Displays all ‘empty’ booths which belong to the vaccine type they choose and asks for First name, Surname, Age, City, NIC or Passport Number, which vaccination they would like and booth number | Pass |
| 3 | After program starts, 101 or VEB | Displays all ‘empty’ booths | Displays all ‘empty’ booths | Pass |
| 4 | After program starts, 103 or RPB | Asks for booth number and change value of booth to ‘empty’ | Asks for booth number and change value of booth to ‘empty’ | Pass |
| 5 | After program starts, 104 or VPS | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Pass |
| 6 | After program starts, 105 or SPD | Writes all the elements in the booths array to a text file. | Writes all the elements in the booths array to a text file. | Pass |
| 7 | After program starts, 106 or LPD | Writes all the records in the text file as elements in the booths array | Writes all the records in the text file as elements in the booths array | Pass |
| 8 | After program starts, 107 or VRV | Displays how many vaccines are remaining | Displays how many vaccines are remaining | Pass |
| 9 | After program starts, 108 or AVS | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Pass |
| 10 | After program starts, 999 or EXT | Exits program | Exits program | Pass |

**Task\_4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Test Case | Expected Result | Actual Result | Pass/Fail |
| 1 | After program starts, 100 or VVB | Displays ‘empty’ for all booths | Displays ‘empty’ for all booths | Pass |
| 2 | After program starts, 102 or APB | Displays all ‘empty’ booths which belong to the vaccine type they choose and asks for First name, Surname, Age, City, NIC or Passport Number, which vaccination they would like and booth number | Displays all ‘empty’ booths which belong to the vaccine type they choose and asks for First name, Surname, Age, City, NIC or Passport Number, which vaccination they would like and booth number | Pass |
| 3 | After program starts, 101 or VEB | Displays all ‘empty’ booths | Displays all ‘empty’ booths | Pass |
| 4 | After program starts, 103 or RPB | Asks for booth number and change value of booth to ‘empty’ | Asks for booth number and change value of booth to ‘empty’ | Pass |
| 5 | After program starts, 104 or VPS | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Temporarily Sorts all elements in booths array in alphabetical order but does not change the element locations | Pass |
| 6 | After program starts, 105 or SPD | Writes all the elements in the booths array to a text file. | Writes all the elements in the booths array to a text file. | Pass |
| 7 | After program starts, 106 or LPD | Writes all the records in the text file as elements in the booths array | Writes all the records in the text file as elements in the booths array | Pass |
| 8 | After program starts, 107 or VRV | Displays how many vaccines are remaining | Displays how many vaccines are remaining | Pass |
| 9 | After program starts, 108 or AVS | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Asks how many more vaccines are to be added and increments total number of vaccines remaining with user input | Pass |
| 10 | After program starts, 999 or EXT | Exits program | Exits program | Pass |

**Discussion**

Test cases were chosen to show that all options in Task\_1 works.

Task\_2 program functions exactly like Task\_1, the only difference between them is that Task\_2 is made out of 2 classes Task\_2\_VacinationCenter and Task\_2\_Booth.

Task\_3\_1 works just like Task\_1 but with more additional features such as when adding a patient to booth, the program will prompt you for your First Name, Surname and which Vaccination you wish to take, then it will find the 2 booths which are designated to the vaccination of your preference and display the booth number of each booth is they are free.

Task\_3\_2 works just like Task\_2 but with more additional features such as when adding a patient to booth, the program will prompt you for your First Name, Surname, Age, City, NIC or Passport Number and which Vaccination you wish to take, then it will find the 2 booths which are designated to the vaccination of your preference and display the booth number of each booth is they are free. Task\_3\_2 is made out of 3 classes Task\_3\_2\_VacinationCenter, Task\_3\_2\_Patient and Task\_3\_2\_Booth.

Task\_4 works just like Task\_3\_2 but with more additional features such as when adding a patient to booth, if both booths which are designated for the preferred vaccine type are full then the patient gets added to a linked list to wait, once a patient leaves a booth then a patient who was in the head of the waiting list for that vaccine will get sent into the now free booth. Task\_4 is made out of 7 classes Task\_4\_VacinationCenter, Task\_4\_Patient, Task\_4\_Booth, Node, Task\_4\_LinkedList\_A, Task\_4\_LinkedList\_S, Task\_4\_LinkedList\_P.

**Code :**

**Task\_1:**

import java.util.Scanner;

import java.io.FileWriter;

import java.io.File;

import java.io.IOException;

// https://www.w3schools.com/java/java\_classes.asp

public class Task\_1{

    static String selection;

    static String patient;

    static Integer Vaccinations = 150 ;

    static String [] booths = {"empty","empty","empty","empty","empty","empty"};

    static Scanner sc = new Scanner(System.in);

    public static *void* main(String[] *args*){

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

        System.out.println(" 100 or VVB: View all Vaccination Booths \n 101 or VEB: View all Empty Booths \n 102 or APB: Add Patient to a Booth \n 103 or RPB: Remove Patient from a Booth \n 104 or VPS: View Patients Sorted in alphabetical order \n 105 or SPD: Store Program Data into file \n 106 or LPD: Load Program Data from file \n 107 or VRV: View Remaining Vaccinations \n 108 or AVS: Add Vaccinations to the Stock \n 999 or EXT: Exit the Program");

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

        while (true) {

            do{

                System.out.println("Type in your selection :");

                selection = sc.nextLine();

                System.out.println("                                                                                ");

            }

            while (!selection.equalsIgnoreCase("100") && !selection.equalsIgnoreCase("VVB") && !selection.equalsIgnoreCase("101") && !selection.equalsIgnoreCase("VEB") && !selection.equalsIgnoreCase("102") && !selection.equalsIgnoreCase("APB") && !selection.equalsIgnoreCase("103") && !selection.equalsIgnoreCase("RPB") && !selection.equalsIgnoreCase("104") && !selection.equalsIgnoreCase("VPS") && !selection.equalsIgnoreCase("105") && !selection.equalsIgnoreCase("SPD") && !selection.equalsIgnoreCase("106") && !selection.equalsIgnoreCase("LPD") && !selection.equalsIgnoreCase("107") && !selection.equalsIgnoreCase("VRV") && !selection.equalsIgnoreCase("108") && !selection.equalsIgnoreCase("AVS") && !selection.equalsIgnoreCase("999") && !selection.equalsIgnoreCase("EXT"));

            switch (selection) {

                case "100": case "VVB":

                    VVB();

                    break;

                case "101": case "VEB":

                    VEB();

                    break;

                case "102": case "APB":

                    APB();

                    break;

                case "103": case "RPB":

                    RPB();

                    break;

                case "104": case "VPS":

                    VPS();

                    break;

                case "105": case "SPD":

                    SPD();

                    break;

                case "106": case "LPD":

                    LPD();

                    break;

                case "107": case "VRV":

                    System.out.println("\nVaccinations remaining in stock:"+ Vaccinations);

                    break;

                case "108": case "AVS":

                    AVS();

                    break;

                case "999": case "EXT":

                    EXT();

            }

            if (Vaccinations == 20){

                System.out.println("Warning : only 20 Vaccinations remain!");

            }

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

        }

    }

// View all Vaccination Booths

    public static *void* VVB(){

        for (*int* i = 0; i < 6; i++) {

            if (booths[i] == "empty"){

                System.out.println("Booth " + (i+1) + " is : empty");

            } else {

                System.out.println("Booth " + (i+1) + " is occupied by : " + booths[i]);

            }

        }

    }

// View all Empty Booths

    public static *void* VEB(){

        for (*int* i = 0; i < 6; i++) {

            if (booths[i] == "empty"){

                System.out.println("Booth " + (i+1) + " is : empty");

            }

        }

    }

// Add Patient to a Booth

    public static *void* APB(){

        VEB();

        System.out.println("Select a booth from the above mentioned booths :");

        Integer number = sc.nextInt();

        System.out.println("                                                                                ");

        System.out.println("Enter patient First name :");

        sc.nextLine();

        patient = sc.nextLine();

        booths[number-1]=patient;

        System.out.println("Patient " + patient + " is assigned to booth number " + number) ;

        Vaccinations-=1 ;

    }

// Remove Patient from a booth

    public static *void* RPB(){

        System.out.println("Enter booth number 1 - 6 :");

        String Number = sc.nextLine();

        System.out.println("                                                                                ");

        Integer value = Integer.parseInt(Number);

        patient = booths[value - 1];

        booths[value-1] = "empty";

        System.out.println("Patient " + patient + " is has been removed from booth number " + Number);

    }

// View Patients Sorted in alphabetical order

    public static *void* VPS(){

                // https://www.javatpoint.com/bubble-sort-in-java

        String[] arr  = {booths[0],booths[1],booths[2],booths[3],booths[4],booths[5]};

        for (*int* j = 0; j < 6 - 1; j++){

            for (*int* i = j + 1; i < 6; i++) {

                if ((arr[j].toLowerCase()).compareTo((arr[i]).toLowerCase()) > 0){

                    String temp1 = arr[j];

                    arr[j] = arr[i];

                    arr[i] = temp1;

                }

            }

        }

        for (*int* i = 0; i < 6; i++){

            System.out.println("Patient " + (i + 1) + " : " + arr[i]);

        }

    }

// Store Program Data into file

    public static *void* SPD(){

            // https://www.w3schools.com/java/java\_files\_create.asp

        try {

            String str="";

            FileWriter writer = new FileWriter("Textfile1.txt");

            for (*int* i=0; i< 6 ; i++){

                str="Booth "+(i+1)+" :"+booths[i];

                writer.write(str + "\n");

            }

            writer.close();

            System.out.println("Successfully updated file.");

        }catch (IOException except){

            System.out.println("Error");

            except.printStackTrace();

        }

    }

// Load Program Data from file

    public static *void* LPD(){

            // https://www.w3schools.com/java/java\_files\_read.asp

            // https://beginnersbook.com/2013/12/java-string-substring-method-example/

        try{

            File line = new File("Textfile1.txt");

            Scanner reader = new Scanner(line);

            for (*int* i=0; i< 6 ; i++){

                String data = reader.nextLine();

                data = data.substring(9);

                if (data.equals("empty")){

                    booths[i] = "empty";

                }else{

                booths[i] = data;

                }

            }

            reader.close();

            System.out.println("Successfully updated Array.");

        }catch (IOException except){

            System.out.println("Error");

            except.printStackTrace();

        }

    }

// Add Vaccinations to the Stock

    public static *void* AVS(){

        System.out.println("Enter number of Vaccinations to be added to stock : ");

        Scanner vacc = new Scanner(System.in);

        Integer add = vacc.nextInt();

        Vaccinations = Vaccinations + add;

    }

// Exit the Program

    public static *void* EXT(){

        System.exit(0);

    }

}

**Task\_2\_CacinationCenter:**

// https://www.w3schools.com/java/java\_arraylist.asp

// https://www.w3schools.com/java/java\_classes.asp

// https://www.geeksforgeeks.org/inheritance-in-java/

import java.util.Scanner;

import java.io.FileWriter;

import java.io.File;

import java.io.IOException;

public class Task\_2\_VacinationCenter {

    static String selection;

    static String patient;

    static Integer Vaccinations = 150;

    static Scanner sc = new Scanner(System.in);

    // https://www.youtube.com/watch?v=cCNpZZVslik

    static Task\_2\_Booth[] booths = new Task\_2\_Booth[6];

    public static *void* main(String[] *args*) {

        for (*int* i = 0; i < 6; i++) {

            Task\_2\_Booth h1 = new Task\_2\_Booth("empty");

            booths[i] = h1;

        }

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

        System.out.println(

                " 100 or VVB: View all Vaccination Booths \n 101 or VEB: View all Empty Booths \n 102 or APB: Add Patient to a Booth \n 103 or RPB: Remove Patient from a Booth \n 104 or VPS: View Patients Sorted in alphabetical order \n 105 or SPD: Store Program Data into file \n 106 or LPD: Load Program Data from file \n 107 or VRV: View Remaining Vaccinations \n 108 or AVS: Add Vaccinations to the Stock \n 999 or EXT: Exit the Program");

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

        while (true) {

            do {

                System.out.println("Type in your selection :");

                selection = sc.nextLine();

                System.out.println("                                                                                ");

            } while (!selection.equalsIgnoreCase("100") && !selection.equalsIgnoreCase("VVB")

                    && !selection.equalsIgnoreCase("101") && !selection.equalsIgnoreCase("VEB")

                    && !selection.equalsIgnoreCase("102") && !selection.equalsIgnoreCase("APB")

                    && !selection.equalsIgnoreCase("103") && !selection.equalsIgnoreCase("RPB")

                    && !selection.equalsIgnoreCase("104") && !selection.equalsIgnoreCase("VPS")

                    && !selection.equalsIgnoreCase("105") && !selection.equalsIgnoreCase("SPD")

                    && !selection.equalsIgnoreCase("106") && !selection.equalsIgnoreCase("LPD")

                    && !selection.equalsIgnoreCase("107") && !selection.equalsIgnoreCase("VRV")

                    && !selection.equalsIgnoreCase("108") && !selection.equalsIgnoreCase("AVS")

                    && !selection.equalsIgnoreCase("999") && !selection.equalsIgnoreCase("EXT"));

            switch (selection) {

                case "100":

                case "VVB":

                    VVB();

                    break;

                case "101":

                case "VEB":

                    VEB();

                    break;

                case "102":

                case "APB":

                    APB();

                    break;

                case "103":

                case "RPB":

                    RPB();

                    break;

                case "104":

                case "VPS":

                    VPS();

                    break;

                case "105":

                case "SPD":

                    SPD();

                    break;

                case "106":

                case "LPD":

                    LPD();

                    break;

                case "107":

                case "VRV":

                    System.out.println("\nVaccinations remaining in stock:" + Vaccinations);

                    break;

                case "108":

                case "AVS":

                    AVS();

                    break;

                case "999":

                case "EXT":

                    EXT();

            }

            if (Vaccinations == 20) {

                System.out.println("Warning : only 20 Vaccinations remain!");

            }

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

        }

    }

    // View all Vaccination Booths

    public static *void* VVB() {

        for (*int* i = 0; i < 6; i++) {

            if (booths[i].name == "empty") {

                System.out.println("Booth " + (i + 1) + " is : empty");

            } else {

                System.out.println("Booth " + (i + 1) + " is occupied by : " + booths[i].name);

            }

        }

    }

    // View all Empty Booths

    public static *void* VEB() {

        for (*int* i = 0; i < 6; i++) {

            if (booths[i].name == "empty") {

                System.out.println("Booth " + (i + 1) + " is occupied by : empty");

            }

        }

    }

    // Add Patient to a Booth

    public static *void* APB() {

        VEB();

        System.out.println("Select a booth from the above mentioned booths :");

        Integer number = sc.nextInt();

        System.out.println("                                                                                ");

        System.out.println("Enter patients First name :");

        sc.nextLine();

        patient = sc.nextLine();

        booths[number - 1].name = patient;

        System.out.println("                                                                                ");

        System.out.println("Patient " + patient + " is assigned to booth number " + number);

        Vaccinations -= 1;

    }

    // Remove Patient from a booth

    public static *void* RPB() {

        System.out.println("Enter booth number 1 - 6 :");

        Integer Number = sc.nextInt();

        System.out.println("                                                                                ");

        patient = booths[Number - 1].name;

        booths[Number - 1].name = "empty";

        System.out.println("Patient " + patient + " is has been removed from booth number " + Number);

    }

    // View Patients Sorted in alphabetical order

    public static *void* VPS() {

        // https://www.javatpoint.com/bubble-sort-in-java

        String[] arr = { "empty", "empty", "empty", "empty", "empty", "empty" };

        for (*int* x = 0; x < 6; x++) {

            arr[x] = booths[x].name;

        }

        for (*int* j = 0; j < 6 - 1; j++) {

            for (*int* i = j + 1; i < 6; i++) {

                if ((arr[j].toLowerCase()).compareTo((arr[i]).toLowerCase()) > 0) {

                    String temp1 = arr[j];

                    arr[j] = arr[i];

                    arr[i] = temp1;

                }

            }

        }

        for (*int* i = 0; i < 6; i++) {

            System.out.println("Patient " + (i + 1) + " : " + arr[i]);

        }

    }

    // Store Program Data into file

    public static *void* SPD() {

        // https://www.w3schools.com/java/java\_files\_create.asp

        try {

            String str = "";

            FileWriter writer = new FileWriter("Textfile2.txt");

            for (*int* i = 0; i < 6; i++) {

                str = "Booth " + i + " :" + booths[i].name;

                writer.write(str + "\n");

            }

            writer.close();

            System.out.println("Successfully updated file.");

        } catch (IOException except) {

            System.out.println("Error");

            except.printStackTrace();

        }

    }

    // Load Program Data from file

    public static *void* LPD() {

        // https://www.w3schools.com/java/java\_files\_read.asp

        // https://beginnersbook.com/2013/12/java-string-substring-method-example/

        try {

            File line = new File("Textfile2.txt");

            Scanner reader = new Scanner(line);

            for (*int* i = 0; i < 6; i++) {

                String data = reader.nextLine();

                data = data.substring(9);

                if (data.equals("empty")) {

                    booths[i].name = "empty";

                } else {

                    booths[i].name = data;

                }

            }

            reader.close();

            System.out.println("Successfully updated Array.");

        } catch (IOException except) {

            System.out.println("Error");

            except.printStackTrace();

        }

    }

    // Add Vaccinations to the Stock

    public static *void* AVS() {

        System.out.println("Enter number of Vaccinations to be added to stock : ");

        Scanner vacc = new Scanner(System.in);

        Integer add = vacc.nextInt();

        Vaccinations = Vaccinations + add;

    }

    // Exit the Program

    public static *void* EXT() {

        System.exit(0);

    }

}

**Task\_2\_Booth:**

// https://www.w3schools.com/java/java\_arraylist.asp

// https://www.w3schools.com/java/java\_classes.asp

// https://www.youtube.com/watch?v=cCNpZZVslik

// https://www.geeksforgeeks.org/inheritance-in-java/

public class Task\_2\_Booth {

    String name;

    Task\_2\_Booth(String *name*) {

        this.name = *name*;

    }

}

**Task\_3\_1:**

import java.util.Scanner;

import java.io.FileWriter;

import java.io.File;

import java.io.IOException;

// https://www.w3schools.com/java/java\_classes.asp

public class Task\_3\_1 {

    static String selection;

    static String patient;

    static Integer Vaccinations = 150;

    static String[] Abooths = { "empty", "empty" };

    static String[] Sbooths = { "empty", "empty" };

    static String[] Pbooths = { "empty", "empty" };

    static Scanner sc = new Scanner(System.in);

    public static *void* main(String[] *args*) {

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

        System.out.println(

                " 100 or VVB: View all Vaccination Booths \n 101 or VEB: View all Empty Booths \n 102 or APB: Add Patient to a Booth \n 103 or RPB: Remove Patient from a Booth \n 104 or VPS: View Patients Sorted in alphabetical order \n 105 or SPD: Store Program Data into file \n 106 or LPD: Load Program Data from file \n 107 or VRV: View Remaining Vaccinations \n 108 or AVS: Add Vaccinations to the Stock \n 999 or EXT: Exit the Program");

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

        while (true) {

            do {

                System.out.println("Type in your selection :");

                selection = sc.nextLine();

                System.out.println("                                                                                ");

            } while (!selection.equalsIgnoreCase("100") && !selection.equalsIgnoreCase("VVB")

                    && !selection.equalsIgnoreCase("101") && !selection.equalsIgnoreCase("VEB")

                    && !selection.equalsIgnoreCase("102") && !selection.equalsIgnoreCase("APB")

                    && !selection.equalsIgnoreCase("103") && !selection.equalsIgnoreCase("RPB")

                    && !selection.equalsIgnoreCase("104") && !selection.equalsIgnoreCase("VPS")

                    && !selection.equalsIgnoreCase("105") && !selection.equalsIgnoreCase("SPD")

                    && !selection.equalsIgnoreCase("106") && !selection.equalsIgnoreCase("LPD")

                    && !selection.equalsIgnoreCase("107") && !selection.equalsIgnoreCase("VRV")

                    && !selection.equalsIgnoreCase("108") && !selection.equalsIgnoreCase("AVS")

                    && !selection.equalsIgnoreCase("999") && !selection.equalsIgnoreCase("EXT"));

            switch (selection) {

                case "100":

                case "VVB":

                    VVB();

                    break;

                case "101":

                case "VEB":

                    VEB();

                    break;

                case "102":

                case "APB":

                    APB();

                    break;

                case "103":

                case "RPB":

                    RPB();

                    break;

                case "104":

                case "VPS":

                    VPS();

                    break;

                case "105":

                case "SPD":

                    SPD();

                    break;

                case "106":

                case "LPD":

                    LPD();

                    break;

                case "107":

                case "VRV":

                    System.out.println("\nVaccinations remaining in stock:" + Vaccinations);

                    break;

                case "108":

                case "AVS":

                    AVS();

                    break;

                case "999":

                case "EXT":

                    EXT();

            }

            if (Vaccinations == 20) {

                System.out.println("Warning : only 20 Vaccinations remain!");

            }

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

        }

    }

    // View all Vaccination Booths

    public static *void* VVB() {

        for (*int* i = 0; i < 2; i++) {

            if (Abooths[i] == "empty") {

                System.out.println("Booth " + (i + 1) + " is : empty");

            } else {

                System.out.println("Booth " + (i + 1) + " is occupied by : " + Abooths[i]);

            }

        }

        for (*int* i = 0; i < 2; i++) {

            if (Sbooths[i] == "empty") {

                System.out.println("Booth " + (i + 3) + " is : empty");

            } else {

                System.out.println("Booth " + (i + 3) + " is occupied by : " + Sbooths[i]);

            }

        }

        for (*int* i = 0; i < 2; i++) {

            if (Pbooths[i] == "empty") {

                System.out.println("Booth " + (i + 5) + " is : empty");

            } else {

                System.out.println("Booth " + (i + 5) + " is occupied by : " + Pbooths[i]);

            }

        }

    }

    // View all Empty Booths

    public static *void* VEB() {

        for (*int* i = 0; i < 2; i++) {

            if (Abooths[i] == "empty") {

                System.out.println("Booth " + (i + 1) + " is : empty");

            }

        }

        for (*int* i = 0; i < 2; i++) {

            if (Sbooths[i] == "empty") {

                System.out.println("Booth " + (i + 3) + " is : empty");

            }

        }

        for (*int* i = 0; i < 2; i++) {

            if (Pbooths[i] == "empty") {

                System.out.println("Booth " + (i + 5) + " is : empty");

            }

        }

    }

    // Add Patient to a Booth

    public static *void* APB() {

        System.out.println("We have the following Vaccinations : \nAstraZeneca \nSinopharm \nPfizer");

        System.out.println("Which Vaccination do you want?");

        String choice = sc.nextLine();

        switch (choice) {

            case "AstraZeneca":

                for (*int* i = 0; i < 2; i++) {

                    if (Abooths[i] == "empty") {

                        System.out.println("Booth " + (i + 1) + " is : empty");

                    }

                }

                System.out.println("Select a booth from the above mentioned booths :");

                Integer number = sc.nextInt();

                System.out.println("                                                                                ");

                System.out.println("Enter patient First name :");

                sc.nextLine();

                String firstname = sc.nextLine();

                System.out.println("Enter patient Surname :");

                String surname = sc.nextLine();

                System.out.println("                                                                                ");

                patient = (firstname + "#" + surname + "#" + choice);

                Abooths[number - 1] = patient;

                System.out.println("Patient " + firstname + " " + surname + " " + "Vaccinated with " + choice

                        + " is assigned to booth number " + number);

                Vaccinations -= 1;

                break;

            case "Sinopharm":

                for (*int* i = 0; i < 2; i++) {

                    if (Sbooths[i] == "empty") {

                        System.out.println("Booth " + (i + 3) + " is : empty");

                    }

                }

                System.out.println("Select a booth from the above mentioned booths :");

                number = sc.nextInt();

                System.out.println("                                                                                ");

                System.out.println("Enter patient First name :");

                sc.nextLine();

                firstname = sc.nextLine();

                System.out.println("Enter patient Surname :");

                surname = sc.nextLine();

                patient = (firstname + "#" + surname + "#" + choice);

                System.out.println("                                                                                ");

                Sbooths[number - 3] = patient;

                System.out.println("Patient " + firstname + " " + surname + " " + "Vaccinated with " + choice

                        + " is assigned to booth number " + number);

                Vaccinations -= 1;

                break;

            case "Pfizer":

                for (*int* i = 0; i < 2; i++) {

                    if (Pbooths[i] == "empty") {

                        System.out.println("Booth " + (i + 5) + " is : empty");

                    }

                }

                System.out.println("Select a booth from the above mentioned booths :");

                number = sc.nextInt();

                System.out.println("                                                                                ");

                System.out.println("Enter patient First name :");

                sc.nextLine();

                firstname = sc.nextLine();

                System.out.println("Enter patient Surname :");

                surname = sc.nextLine();

                patient = (firstname + "#" + surname + "#" + choice);

                System.out.println("                                                                                ");

                Pbooths[number - 5] = patient;

                System.out.println("Patient " + firstname + " " + surname + " " + "Vaccinated with " + choice

                        + " is assigned to booth number " + number);

                Vaccinations -= 1;

                break;

        }

    }

    // Remove Patient from a booth

    public static *void* RPB() {

        System.out.println("Enter booth number 1 - 6 :");

        String Number = sc.nextLine();

        System.out.println("                                                                                ");

        switch (Number) {

            case "1":

            case "2":

                Integer value = Integer.parseInt(Number);

                patient = Abooths[value - 1];

                Abooths[value - 1] = "empty";

                System.out.println("Patient " + patient + " is has been removed from booth number " + Number);

                break;

            case "3":

            case "4":

                value = Integer.parseInt(Number);

                patient = Sbooths[value - 3];

                Sbooths[value - 3] = "empty";

                System.out.println("Patient " + patient + " is has been removed from booth number " + Number);

                break;

            case "5":

            case "6":

                value = Integer.parseInt(Number);

                patient = Pbooths[value - 5];

                Pbooths[value - 5] = "empty";

                System.out.println("Patient " + patient + " is has been removed from booth number " + Number);

                break;

        }

    }

    // View Patients Sorted in alphabetical order

    public static *void* VPS() {

        // https://www.javatpoint.com/bubble-sort-in-java

        String[] arr = { Abooths[0], Abooths[1], Sbooths[0], Sbooths[1], Pbooths[0], Pbooths[1] };

        for (*int* j = 0; j < 6 - 1; j++) {

            for (*int* i = j + 1; i < 6; i++) {

                if ((arr[j].toLowerCase()).compareTo((arr[i]).toLowerCase()) > 0) {

                    String temp1 = arr[j];

                    arr[j] = arr[i];

                    arr[i] = temp1;

                }

            }

        }

        for (*int* i = 0; i < 6; i++) {

            System.out.println("Patient " + (i + 1) + " : " + arr[i]);

        }

    }

    // Store Program Data into file

    public static *void* SPD() {

        // https://www.w3schools.com/java/java\_files\_create.asp

        try {

            String str = "";

            FileWriter writer = new FileWriter("Textfile3\_1.txt");

            for (*int* i = 0; i < 2; i++) {

                str = "Booth " + (i + 1) + " :" + Abooths[i];

                writer.write(str + "\n");

            }

            for (*int* i = 0; i < 2; i++) {

                str = "Booth " + (i + 3) + " :" + Sbooths[i];

                writer.write(str + "\n");

            }

            for (*int* i = 0; i < 2; i++) {

                str = "Booth " + (i + 5) + " :" + Pbooths[i];

                writer.write(str + "\n");

            }

            writer.close();

            System.out.println("Successfully updated file.");

        } catch (IOException except) {

            System.out.println("Error");

            except.printStackTrace();

        }

    }

    // Load Program Data from file

    public static *void* LPD() {

        // https://www.w3schools.com/java/java\_files\_read.asp

        // https://beginnersbook.com/2013/12/java-string-substring-method-example/

        try {

            File line = new File("Textfile3\_1.txt");

            Scanner reader = new Scanner(line);

            for (*int* i = 0; i < 2; i++) {

                String data = reader.nextLine();

                data = data.substring(9);

                if (data.equals("empty")) {

                    Abooths[i] = "empty";

                } else {

                    Abooths[i] = data;

                }

            }

            for (*int* i = 0; i < 2; i++) {

                String data = reader.nextLine();

                data = data.substring(9);

                if (data.equals("empty")) {

                    Sbooths[i] = "empty";

                } else {

                    Sbooths[i] = data;

                }

            }

            for (*int* i = 0; i < 2; i++) {

                String data = reader.nextLine();

                data = data.substring(9);

                if (data.equals("empty")) {

                    Pbooths[i] = "empty";

                } else {

                    Pbooths[i] = data;

                }

            }

            reader.close();

            System.out.println("Successfully updated Array.");

        } catch (IOException except) {

            System.out.println("Error");

            except.printStackTrace();

        }

    }

    // Add Vaccinations to the Stock

    public static *void* AVS() {

        System.out.println("Enter number of Vaccinations to be added to stock : ");

        Scanner vacc = new Scanner(System.in);

        Integer add = vacc.nextInt();

        Vaccinations = Vaccinations + add;

    }

    // Exit the Program

    public static *void* EXT() {

        System.exit(0);

    }

}

**Task\_3\_2\_VacinationCenter:**

// https://www.w3schools.com/java/java\_arraylist.asp

// https://www.w3schools.com/java/java\_classes.asp

// https://www.geeksforgeeks.org/inheritance-in-java/

import java.util.Scanner;

import java.io.FileWriter;

import java.io.File;

import java.io.IOException;

public class Task\_3\_2\_VacinationCenter {

    static String selection;

    static String patient;

    static Integer Vaccinations = 150;

    static Scanner sc = new Scanner(System.in);

    // https://www.youtube.com/watch?v=cCNpZZVslik

    static Task\_3\_2\_Patient[] booths = new Task\_3\_2\_Patient[6];

    public static *void* main(String[] *args*) {

        for (*int* i = 0; i < 6; i++) {

            Task\_3\_2\_Patient h1 = new Task\_3\_2\_Patient("empty", "empty", "empty", "empty", "empty", "empty");

            booths[i] = h1;

        }

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

        System.out.println(

                " 100 or VVB: View all Vaccination Booths \n 101 or VEB: View all Empty Booths \n 102 or APB: Add Patient to a Booth \n 103 or RPB: Remove Patient from a Booth \n 104 or VPS: View Patients Sorted in alphabetical order \n 105 or SPD: Store Program Data into file \n 106 or LPD: Load Program Data from file \n 107 or VRV: View Remaining Vaccinations \n 108 or AVS: Add Vaccinations to the Stock \n 999 or EXT: Exit the Program");

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

        while (true) {

            do {

                System.out.println("Type in your selection :");

                selection = sc.nextLine();

                System.out.println("                                                                                ");

            } while (!selection.equalsIgnoreCase("100") && !selection.equalsIgnoreCase("VVB")

                    && !selection.equalsIgnoreCase("101") && !selection.equalsIgnoreCase("VEB")

                    && !selection.equalsIgnoreCase("102") && !selection.equalsIgnoreCase("APB")

                    && !selection.equalsIgnoreCase("103") && !selection.equalsIgnoreCase("RPB")

                    && !selection.equalsIgnoreCase("104") && !selection.equalsIgnoreCase("VPS")

                    && !selection.equalsIgnoreCase("105") && !selection.equalsIgnoreCase("SPD")

                    && !selection.equalsIgnoreCase("106") && !selection.equalsIgnoreCase("LPD")

                    && !selection.equalsIgnoreCase("107") && !selection.equalsIgnoreCase("VRV")

                    && !selection.equalsIgnoreCase("108") && !selection.equalsIgnoreCase("AVS")

                    && !selection.equalsIgnoreCase("999") && !selection.equalsIgnoreCase("EXT"));

            switch (selection) {

                case "100":

                case "VVB":

                    VVB();

                    break;

                case "101":

                case "VEB":

                    VEB();

                    break;

                case "102":

                case "APB":

                    APB();

                    break;

                case "103":

                case "RPB":

                    RPB();

                    break;

                case "104":

                case "VPS":

                    VPS();

                    break;

                case "105":

                case "SPD":

                    SPD();

                    break;

                case "106":

                case "LPD":

                    LPD();

                    break;

                case "107":

                case "VRV":

                    System.out.println("\nVaccinations remaining in stock:" + Vaccinations);

                    break;

                case "108":

                case "AVS":

                    AVS();

                    break;

                case "999":

                case "EXT":

                    EXT();

            }

            if (Vaccinations == 20) {

                System.out.println("Warning : only 20 Vaccinations remain!");

            }

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

        }

    }

    // View all Vaccination Booths

    public static *void* VVB() {

        for (*int* i = 0; i < 6; i++) {

            if (booths[i].Firstname == "empty") {

                System.out.println("Booth " + (i + 1) + " is : empty");

            } else {

                System.out.println("Booth " + (i + 1) + " is occupied by : " + booths[i].Firstname);

            }

        }

    }

    // View all Empty Booths

    public static *void* VEB() {

        for (*int* i = 0; i < 6; i++) {

            if (booths[i].Firstname == "empty") {

                System.out.println("Booth " + (i + 1) + " is occupied by : empty");

            }

        }

    }

    // Add Patient to a Booth

    public static *void* APB() {

        System.out.println("Enter patients First name :");

        patient = sc.nextLine();

        System.out.println("Enter patients Surname :");

        String surname = sc.nextLine();

        System.out.println("Enter patients Age :");

        String age = sc.nextLine();

        System.out.println("Enter patients City :");

        String city = sc.nextLine();

        System.out.println("Enter patients NIC or Passport Number :");

        String id = sc.nextLine();

        System.out.println("We have the following Vaccinations : \nAstraZeneca \nSinopharm \nPfizer");

        System.out.println("Which Vaccination do you want?");

        String choice = sc.nextLine();

        switch (choice) {

            case "AstraZeneca":

                for (*int* i = 0; i < 2; i++) {

                    if (booths[i].Firstname == "empty") {

                        System.out.println("Booth " + (i + 1) + " is occupied by : empty");

                    }

                }

                System.out.println("Select a booth from the above mentioned booths :");

                Integer number = sc.nextInt();

                System.out.println("                                                                                ");

                booths[number - 1].Firstname = patient;

                booths[number - 1].Surname = surname;

                booths[number - 1].Age = age;

                booths[number - 1].City = city;

                booths[number - 1].Id = id;

                booths[number - 1].Vaccination = choice;

                System.out.println("                                                                                ");

                System.out.println("Patient " + patient + " is assigned to booth number " + number);

                Vaccinations -= 1;

                break;

            case "Sinopharm":

                for (*int* i = 2; i < 4; i++) {

                    if (booths[i].Firstname == "empty") {

                        System.out.println("Booth " + (i + 1) + " is occupied by : empty");

                    }

                }

                System.out.println("Select a booth from the above mentioned booths :");

                number = sc.nextInt();

                System.out.println("                                                                                ");

                booths[number - 1].Firstname = patient;

                booths[number - 1].Surname = surname;

                booths[number - 1].Age = age;

                booths[number - 1].City = city;

                booths[number - 1].Id = id;

                booths[number - 1].Vaccination = choice;

                System.out.println("                                                                                ");

                System.out.println("Patient " + patient + " is assigned to booth number " + number);

                Vaccinations -= 1;

                break;

            case "Pfizer":

                for (*int* i = 4; i < 6; i++) {

                    if (booths[i].Firstname == "empty") {

                        System.out.println("Booth " + (i + 1) + " is occupied by : empty");

                    }

                }

                System.out.println("Select a booth from the above mentioned booths :");

                number = sc.nextInt();

                System.out.println("                                                                                ");

                booths[number - 1].Firstname = patient;

                booths[number - 1].Surname = surname;

                booths[number - 1].Age = age;

                booths[number - 1].City = city;

                booths[number - 1].Id = id;

                booths[number - 1].Vaccination = choice;

                System.out.println("                                                                                ");

                System.out.println("Patient " + patient + " is assigned to booth number " + number);

                Vaccinations -= 1;

                break;

        }

    }

    // Remove Patient from a booth

    public static *void* RPB() {

        System.out.println("Enter booth number 1 - 6 :");

        Integer Number = sc.nextInt();

        System.out.println("                                                                                ");

        patient = booths[Number - 1].Firstname;

        booths[Number - 1].Firstname = "empty";

        System.out.println("Patient " + patient + " is has been removed from booth number " + Number);

    }

    // View Patients Sorted in alphabetical order

    public static *void* VPS() {

        // https://www.javatpoint.com/bubble-sort-in-java

        String[] arr = { "empty", "empty", "empty", "empty", "empty", "empty" };

        for (*int* x = 0; x < 6; x++) {

            arr[x] = booths[x].Firstname;

        }

        for (*int* j = 0; j < 6 - 1; j++) {

            for (*int* i = j + 1; i < 6; i++) {

                if ((arr[j].toLowerCase()).compareTo((arr[i]).toLowerCase()) > 0) {

                    String temp1 = arr[j];

                    arr[j] = arr[i];

                    arr[i] = temp1;

                }

            }

        }

        for (*int* i = 0; i < 6; i++) {

            System.out.println("Patient " + (i + 1) + " : " + arr[i]);

        }

    }

    // Store Program Data into file

    public static *void* SPD() {

        // https://www.w3schools.com/java/java\_files\_create.asp

        try {

            String str = "";

            FileWriter writer = new FileWriter("Textfile3\_2.txt");

            for (*int* i = 0; i < 6; i++) {

                str = "Booth " + i + " :" + booths[i].Firstname + "#" + booths[i].Surname + "#" + booths[i].Age + "#"

                        + booths[i].City + "#" + booths[i].Id + "#" + booths[i].Vaccination;

                writer.write(str + "\n");

            }

            writer.close();

            System.out.println("Successfully updated file.");

        } catch (IOException except) {

            System.out.println("Error");

            except.printStackTrace();

        }

    }

    // Load Program Data from file

    public static *void* LPD() {

        // https://www.w3schools.com/java/java\_files\_read.asp

        // https://beginnersbook.com/2013/12/java-string-substring-method-example/

        try {

            File line = new File("Textfile3\_2.txt");

            Scanner reader = new Scanner(line);

            for (*int* i = 0; i < 6; i++) {

                String data = reader.nextLine();

                data = data.substring(9, 14);

                if (data.equals("empty")) {

                    booths[i].Firstname = "empty";

                } else {

                    booths[i].Firstname = data;

                }

            }

            reader.close();

            System.out.println("Successfully updated Array.");

        } catch (IOException except) {

            System.out.println("Error");

            except.printStackTrace();

        }

    }

    // Add Vaccinations to the Stock

    public static *void* AVS() {

        System.out.println("Enter number of Vaccinations to be added to stock : ");

        Scanner vacc = new Scanner(System.in);

        Integer add = vacc.nextInt();

        Vaccinations = Vaccinations + add;

    }

    // Exit the Program

    public static *void* EXT() {

        System.exit(0);

    }

}

**Task\_3\_2\_Booth:**

// https://www.w3schools.com/java/java\_arraylist.asp

// https://www.w3schools.com/java/java\_classes.asp

// https://www.youtube.com/watch?v=cCNpZZVslik

// https://www.geeksforgeeks.org/inheritance-in-java/

public class Task\_3\_2\_Booth {

    String Firstname;

    Task\_3\_2\_Booth(String *Firstname*) {

        this.Firstname = *Firstname*;

    }

}

**Task\_3\_2\_Patient:**

// https://www.geeksforgeeks.org/inheritance-in-java/

public class Task\_3\_2\_Patient extends Task\_3\_2\_Booth {

    String Surname;

    String Age;

    String City;

    String Id;

    String Vaccination;

    public Task\_3\_2\_Patient(String *Firstname*, String *Surname*, String *Age*, String *City*, String *Id*, String *Vaccination*) {

        super(*Firstname*);

        this.Surname = *Surname*;

        this.Age = *Age*;

        this.City = *City*;

        this.Id = *Id*;

        this.Vaccination = *Vaccination*;

    }

}

**Task\_4\_VacinationCenter**

// https://www.w3schools.com/java/java\_arraylist.asp

// https://www.w3schools.com/java/java\_classes.asp

// https://www.geeksforgeeks.org/inheritance-in-java/

// https://youtu.be/SMIq13-FZSE

// https://www.youtube.com/playlist?list=PLsyeobzWxl7oRKwDi7wjrANsbhTX0IK0J

import java.util.Scanner;

import java.io.FileWriter;

import java.io.File;

import java.io.IOException;

import java.util.LinkedList;

import java.util.Objects;

public class Task\_4\_VacinationCenter {

    static String selection;

    static String patient;

    static Integer Vaccinations = 150;

    static Scanner sc = new Scanner(System.in);

    // https://www.youtube.com/watch?v=cCNpZZVslik

    static Task\_4\_Patient[] booths = new Task\_4\_Patient[6];

    static Task\_4\_LinkedList\_A listA = new Task\_4\_LinkedList\_A();

    static Task\_4\_LinkedList\_S listS = new Task\_4\_LinkedList\_S();

    static Task\_4\_LinkedList\_P listP = new Task\_4\_LinkedList\_P();

    public static *void* main(String[] *args*) throws IOException {

        for (*int* i = 0; i < 6; i++) {

            Task\_4\_Patient h1 = new Task\_4\_Patient("empty", "empty", "empty", "empty", "empty", "empty");

            booths[i] = h1;

        }

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

        System.out.println(

                " 100 or VVB: View all Vaccination Booths \n 101 or VEB: View all Empty Booths \n 102 or APB: Add Patient to a Booth \n 103 or RPB: Remove Patient from a Booth \n 104 or VPS: View Patients Sorted in alphabetical order \n 105 or SPD: Store Program Data into file \n 106 or LPD: Load Program Data from file \n 107 or VRV: View Remaining Vaccinations \n 108 or AVS: Add Vaccinations to the Stock \n 999 or EXT: Exit the Program");

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

        while (true) {

            do {

                System.out.println("Type in your selection :");

                selection = sc.nextLine();

                System.out.println("                                                                                ");

            } while (!selection.equalsIgnoreCase("100") && !selection.equalsIgnoreCase("VVB")

                    && !selection.equalsIgnoreCase("101") && !selection.equalsIgnoreCase("VEB")

                    && !selection.equalsIgnoreCase("102") && !selection.equalsIgnoreCase("APB")

                    && !selection.equalsIgnoreCase("103") && !selection.equalsIgnoreCase("RPB")

                    && !selection.equalsIgnoreCase("104") && !selection.equalsIgnoreCase("VPS")

                    && !selection.equalsIgnoreCase("105") && !selection.equalsIgnoreCase("SPD")

                    && !selection.equalsIgnoreCase("106") && !selection.equalsIgnoreCase("LPD")

                    && !selection.equalsIgnoreCase("107") && !selection.equalsIgnoreCase("VRV")

                    && !selection.equalsIgnoreCase("108") && !selection.equalsIgnoreCase("AVS")

                    && !selection.equalsIgnoreCase("999") && !selection.equalsIgnoreCase("EXT"));

            switch (selection) {

                case "100":

                case "VVB":

                    VVB();

                    break;

                case "101":

                case "VEB":

                    VEB();

                    break;

                case "102":

                case "APB":

                    APB();

                    break;

                case "103":

                case "RPB":

                    RPB();

                    break;

                case "104":

                case "VPS":

                    VPS();

                    break;

                case "105":

                case "SPD":

                    SPD();

                    break;

                case "106":

                case "LPD":

                    LPD();

                    break;

                case "107":

                case "VRV":

                    System.out.println("\nVaccinations remaining in stock:" + Vaccinations);

                    break;

                case "108":

                case "AVS":

                    AVS();

                    break;

                case "999":

                case "EXT":

                    EXT();

            }

            if (Vaccinations == 20) {

                System.out.println("Warning : only 20 Vaccinations remain!");

            }

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

        }

    }

    // View all Vaccination Booths

    public static *void* VVB() {

        for (*int* i = 0; i < 6; i++) {

            if (booths[i].Firstname == "empty") {

                System.out.println("Booth " + (i + 1) + " is : empty");

            } else {

                System.out.println("Booth " + (i + 1) + " is occupied by : " + booths[i].Firstname);

            }

        }

    }

    // View all Empty Booths

    public static *void* VEB() {

        for (*int* i = 0; i < 6; i++) {

            if (booths[i].Firstname == "empty") {

                System.out.println("Booth " + (i + 1) + " is occupied by : empty");

            }

        }

    }

    // Add Patient to a Booth

    public static *void* APB() {

        Boolean found = false;

        System.out.println("Enter patients First name :");

        patient = sc.nextLine();

        System.out.println("Enter patients Surname :");

        String surname = sc.nextLine();

        System.out.println("Enter patients Age :");

        String age = sc.nextLine();

        System.out.println("Enter patients City :");

        String city = sc.nextLine();

        System.out.println("Enter patients NIC or Passport Number :");

        String id = sc.nextLine();

        System.out.println("We have the following Vaccinations : \nAstraZeneca \nSinopharm \nPfizer");

        System.out.println("Which Vaccination do you want?");

        String choice = sc.nextLine();

        switch (choice) {

            case "AstraZeneca":

                for (*int* i = 0; i < 2; i++) {

                    if (booths[i].Firstname == "empty") {

                        System.out.println("Booth " + (i + 1) + " is occupied by : empty");

                        found = true;

                    }

                }

                if (found.equals(true)) {

                    System.out.println("Select a booth from the above mentioned booths :");

                    Integer number = sc.nextInt();

                    System.out.println(

                            "                                                                                ");

                    booths[number - 1].Firstname = patient;

                    booths[number - 1].Surname = surname;

                    booths[number - 1].Age = age;

                    booths[number - 1].City = city;

                    booths[number - 1].Id = id;

                    booths[number - 1].Vaccination = choice;

                    System.out.println(

                            "                                                                                ");

                    System.out.println("Patient " + patient + " is assigned to booth number " + number);

                    Vaccinations -= 1;

                } else {

                    String data = patient + "/" + surname + "/" + age + "/" + city + "/" + id + "/" + choice;

                    listA.insert(data);

                    System.out.println(listA.head);

                }

                break;

            case "Sinopharm":

                for (*int* i = 2; i < 4; i++) {

                    if (booths[i].Firstname == "empty") {

                        System.out.println("Booth " + (i + 1) + " is occupied by : empty");

                        found = true;

                    }

                }

                if (found.equals(true)) {

                    System.out.println("Select a booth from the above mentioned booths :");

                    Integer number = sc.nextInt();

                    System.out.println(

                            "                                                                                ");

                    booths[number - 1].Firstname = patient;

                    booths[number - 1].Surname = surname;

                    booths[number - 1].Age = age;

                    booths[number - 1].City = city;

                    booths[number - 1].Id = id;

                    booths[number - 1].Vaccination = choice;

                    System.out.println(

                            "                                                                                ");

                    System.out.println("Patient " + patient + " is assigned to booth number " + number);

                    Vaccinations -= 1;

                } else {

                    String data = patient + "/" + surname + "/" + age + "/" + city + "/" + id + "/" + choice;

                    listA.insert(data);

                    System.out.println("pls wait");

                }

                break;

            case "Pfizer":

                for (*int* i = 4; i < 6; i++) {

                    if (booths[i].Firstname == "empty") {

                        System.out.println("Booth " + (i + 1) + " is occupied by : empty");

                        found = true;

                    }

                }

                if (found.equals(true)) {

                    System.out.println("Select a booth from the above mentioned booths :");

                    Integer number = sc.nextInt();

                    System.out.println(

                            "                                                                                ");

                    booths[number - 1].Firstname = patient;

                    booths[number - 1].Surname = surname;

                    booths[number - 1].Age = age;

                    booths[number - 1].City = city;

                    booths[number - 1].Id = id;

                    booths[number - 1].Vaccination = choice;

                    System.out.println(

                            "                                                                                ");

                    System.out.println("Patient " + patient + " is assigned to booth number " + number);

                    Vaccinations -= 1;

                } else {

                    String data = patient + "/" + surname + "/" + age + "/" + city + "/" + id + "/" + choice;

                    listA.insert(data);

                    System.out.println("pls wait");

                }

                break;

        }

    }

    // Remove Patient from a booth

    public static *void* RPB() {

        System.out.println("Enter booth number 1 - 6 :");

        Integer Number = sc.nextInt();

        System.out.println("                                                                                ");

        patient = booths[Number - 1].Firstname;

        booths[Number - 1].Firstname = "empty";

        System.out.println("Patient " + patient + " is has been removed from booth number " + Number);

        // https://www.geeksforgeeks.org/split-string-java-examples/

        switch (Number) {

            case 1:

            case 2:

                String str;

                if (!Objects.isNull(listA.head)) {

                    str = listA.head.data;

                    String[] parts = str.split("/");

                    String part1 = parts[0];

                    String part2 = parts[1];

                    String part3 = parts[2];

                    String part4 = parts[3];

                    String part5 = parts[4];

                    String part6 = parts[5];

                    booths[Number - 1].Firstname = part1;

                    booths[Number - 1].Surname = part2;

                    booths[Number - 1].Age = part3;

                    booths[Number - 1].City = part4;

                    booths[Number - 1].Id = part5;

                    booths[Number - 1].Vaccination = part6;

                    String name = part1;

                    listA.delete();

                    System.out.println(

                            "                                                                                ");

                    System.out.println("Patient " + part1 + " is assigned to booth number " + Number);

                    Vaccinations -= 1;

                    break;

                }

            case 3:

            case 4:

                if (!Objects.isNull(listS.head)) {

                    str = listS.head.data;

                    String[] parts = str.split("/");

                    String part1 = parts[0];

                    String part2 = parts[1];

                    String part3 = parts[2];

                    String part4 = parts[3];

                    String part5 = parts[4];

                    String part6 = parts[5];

                    booths[Number - 1].Firstname = part1;

                    booths[Number - 1].Surname = part2;

                    booths[Number - 1].Age = part3;

                    booths[Number - 1].City = part4;

                    booths[Number - 1].Id = part5;

                    booths[Number - 1].Vaccination = part6;

                    listS.delete();

                    System.out.println(

                            "                                                                                ");

                    System.out.println("Patient " + part1 + " is assigned to booth number " + Number);

                    Vaccinations -= 1;

                    break;

                }

            case 5:

            case 6:

                if (!Objects.isNull(listA.head)) {

                    str = listP.head.data;

                    String[] parts = str.split("/");

                    String part1 = parts[0];

                    String part2 = parts[1];

                    String part3 = parts[2];

                    String part4 = parts[3];

                    String part5 = parts[4];

                    String part6 = parts[5];

                    booths[Number - 1].Firstname = part1;

                    booths[Number - 1].Surname = part2;

                    booths[Number - 1].Age = part3;

                    booths[Number - 1].City = part4;

                    booths[Number - 1].Id = part5;

                    booths[Number - 1].Vaccination = part6;

                    listP.delete();

                    System.out.println(

                            "                                                                                ");

                    System.out.println("Patient " + part1 + " is assigned to booth number " + Number);

                    Vaccinations -= 1;

                    break;

                }

        }

    }

    // View Patients Sorted in alphabetical order

    public static *void* VPS() {

        // https://www.javatpoint.com/bubble-sort-in-java

        String[] arr = { "empty", "empty", "empty", "empty", "empty", "empty" };

        for (*int* x = 0; x < 6; x++) {

            arr[x] = booths[x].Firstname;

        }

        for (*int* j = 0; j < 6 - 1; j++) {

            for (*int* i = j + 1; i < 6; i++) {

                if ((arr[j].toLowerCase()).compareTo((arr[i]).toLowerCase()) > 0) {

                    String temp1 = arr[j];

                    arr[j] = arr[i];

                    arr[i] = temp1;

                }

            }

        }

        for (*int* i = 0; i < 6; i++) {

            System.out.println("Patient " + (i + 1) + " : " + arr[i]);

        }

    }

    // Store Program Data into file

    public static *void* SPD() {

        // https://www.w3schools.com/java/java\_files\_create.asp

        try {

            String str = "";

            FileWriter writer = new FileWriter("Textfile4.txt");

            for (*int* i = 0; i < 6; i++) {

                str = "Booth " + i + " :" + booths[i].Firstname + "#" + booths[i].Surname + "#" + booths[i].Age + "#"

                        + booths[i].City + "#" + booths[i].Id + "#" + booths[i].Vaccination;

                writer.write(str + "\n");

            }

            writer.close();

            System.out.println("Successfully updated file.");

        } catch (IOException except) {

            System.out.println("Error");

            except.printStackTrace();

        }

    }

    // Load Program Data from file

    public static *void* LPD() {

        // https://www.w3schools.com/java/java\_files\_read.asp

        // https://beginnersbook.com/2013/12/java-string-substring-method-example/

        try {

            File line = new File("Textfile4.txt");

            Scanner reader = new Scanner(line);

            for (*int* i = 0; i < 6; i++) {

                String data = reader.nextLine();

                data = data.substring(9, 14);

                if (data.equals("empty")) {

                    booths[i].Firstname = "empty";

                } else {

                    booths[i].Firstname = data;

                }

            }

            reader.close();

            System.out.println("Successfully updated Array.");

        } catch (IOException except) {

            System.out.println("Error");

            except.printStackTrace();

        }

    }

    // Add Vaccinations to the Stock

    public static *void* AVS() {

        System.out.println("Enter number of Vaccinations to be added to stock : ");

        Scanner vacc = new Scanner(System.in);

        Integer add = vacc.nextInt();

        Vaccinations = Vaccinations + add;

    }

    // Exit the Program

    public static *void* EXT() {

        System.exit(0);

    }

}

**Task\_4\_LinkedList\_A**

public class Task\_4\_LinkedList\_A {

    Node head;

    public String data;

    public *void* insert(String *data*) {

        Node node = new Node();

        node.data = *data*;

        node.next = null;

        if (head == null) {

            head = node;

        } else {

            Node n = head;

            while (n.next != null) {

                n = n.next;

            }

            n.next = node;

        }

    }

    public *void* delete() {

        head = head.next;

    }

}

**Task\_4\_LinkedList\_S**

public class Task\_4\_LinkedList\_S {

    Node head;

    public String data;

    public *void* insert(String *data*) {

        Node node = new Node();

        node.data = *data*;

        if (head == null) {

            head = node;

        } else {

            Node n = head;

            while (n.next != null) {

                n = n.next;

            }

            n.next = node;

        }

    }

    public *void* delete() {

        if (head.next != null) {

            head = head.next;

        }

    }

    public static String isEmpty() {

        String check = Task\_4\_LinkedList\_S.isEmpty();

        return check;

    }

}

**Task\_4\_LinkedList\_P**

public class Task\_4\_LinkedList\_P {

    Node head;

    public String data;

    public *void* insert(String *data*) {

        Node node = new Node();

        node.data = *data*;

        if (head == null) {

            head = node;

        } else {

            Node n = head;

            while (n.next != null) {

                n = n.next;

            }

            n.next = node;

        }

    }

    public *void* delete() {

        if (head.next != null) {

            head = head.next;

        }

    }

}

**Node**

public class Node {

    String data;

    Node next;

}

<<END>>