UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

Department of Computing

Software Development II Coursework Report

Module : 4COSC010C: Software Development II

Module Leader : Mr. Iresh Bandara

Date of submission : 26/07/2021

Student ID : 20200504/ w1830238

Student First Name : Rukshan

Student Surname : Liyanage

"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 : Rukshan Liyanage

Student ID : 20200504/ w1830238

Test Cases

Test Cases for Task 1 & 2

	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	(Add Patient "Scott" to Booth 5) 102 or APB enter "Scott"	100 or VVB Displays "Scott" for booth 5	100 or VVB Displays "Scott" for booth 5	Pass
3	(View empty Booths) 101 or VEB	Except for booth 5, view all other Booths as Empty	Except for booth 5, view all other Booths as Empty	Pass
4	(Add Patient "John" to Booth 0 Add Patient "Mark" to Booth 2) 102 or APB enter "John" 102 or APB enter "Mark"	100 or VVB Displays "John", "Mark" for booth 0 and 2	100 or VVB Displays "John", "Mark" for booth 0 and 2	Pass
5	(Sort patients names) Enter 104 or VPS to view alphabetically sorted Name list.	Display the Patient names in following order: John Mark Scott	Display the Patient names in following order: John Mark Scott	Pass
6	(Remove Patient from Booth 2) Enter option 103 or RPB and enter "2" After removing the Patient enter 100 or VVB	Display "empty" for Booth number 2	Display "empty" for Booth number 2	Pass
7	(Store Program Data) Enter option 105 or SPD	Display "Successfully wrote to the file"	Display "Successfully wrote to the file"	Pass
8	(Load Program Data) Enter Option 106 or LPD	All the stored data must display	All the stored data must display	Pass

9	(View Remaining Vaccinations) Enter option 107 or VRV	Display "Remaining vaccine amount is: 147"	Display "Remaining vaccine amount is: 147"	Pass
10	(Add new 30 vaccinations to the Stock) Enter option 108 or AVS and Enter 30	107 or VRV Remaining vaccination amount must be 177	107 or VRV Remaining vaccination amount must be 177	Pass
11	(Entering an invalid option) Enter option 200 or ABC	Display "Please enter a valid option"	Display "Please enter a valid option"	Pass
12	(Exiting the Program) Enter option 999 or EXT	Program Quits	Program Quits	Pass

Test Cases for Task 3 – Array Version

	Test Case	Expected Result	Actual Result	Pass/Fail
1	(Add a Patient with following Details and select the Option 100 or VVB) Enter option 102 or APB Enter Following Details: First Name - Rick Surname - Morty Vaccination (Sinopharm) – 2 Enter option 100 or VVB	100 or VVB Patient must be directed to booth no. 2 and display the additional information about the patient.	100 or VVB Patient must be directed to booth no. 2 and display the additional information about the patient.	Pass
2	(Add a Patient with following Details and select the Option 100 or VVB) Enter option 102 or APB Enter Following Details: First Name - Stan Surname - Lee Vaccination (AstraZeneca) – 1 Enter option 100 or VVB	100 or VVB Patient must be directed to booth no. 0 and display the additional information about the patient.	100 or VVB Patient must be directed to booth no. 0 and display the additional information about the patient.	Pass
3	(Add a Patient with following Details and select the Option 100 or VVB) Enter option 102 or APB Enter Following Details: First Name - Peter Surname - bush Vaccination (AstraZeneca) – 1 Enter option 100 or VVB	100 or VVB Patient must be directed to booth no. 1 and display the additional information about the patient.	100 or VVB Patient must be directed to booth no. 1 and display the additional information about the patient.	Pass
4	(Add patient with the requested vaccination AstraZeneca)	Display “Requested Vaccination type Booths are Occupied”	Display “Requested Vaccination type Booths are Occupied	Pass

	Enter option 102 or APB Enter Following Details: First Name - Saman Surname - Perera Vaccination (AstraZeneca) – 1			

Test Cases for Task 3 – Class Version

	Test Case	**Expected Result**	**Actual Result**	**Pass/Fail**
1	(Add a Patient with following Details and select the Option 100 or VVB) Enter option 102 or APB Enter Following Details: First Name - Rick Surname - Morty Age - 18 City - New York Nic - 12345678 Vaccination (Sinopharm) – 2 Enter option 100 or VVB	100 or VVB Patient must be directed to booth no. 2 and display the additional information about the patient.	100 or VVB Patient must be directed to booth no. 2 and display the additional information about the patient.	Pass
2	(Add a Patient with following Details and select the Option 100 or VVB) Enter option 102 or APB Enter Following Details: First Name - Stan Surname - Lee Age - 32 City - Colombo Nic - 20001234 Vaccination (AstraZeneca) – 1 Enter option 100 or VVB	100 or VVB Patient must be directed to booth no. 0 and display the additional information about the patient.	100 or VVB Patient must be directed to booth no. 0 and display the additional information about the patient.	Pass

3	(Add a Patient with following Details and select the Option 100 or VVB) Enter option 102 or APB Enter Following Details: First Name - Peter Surname - bush Age - 21 City - Kandy Nic - 654321 Vaccination (AstraZeneca) – 1 Enter option 100 or VVB	100 or VVB Patient must be directed to booth no. 1 and display the additional information about the patient.	100 or VVB Patient must be directed to booth no. 1 and display the additional information about the patient	Pass
4	(Add patient with the requested vaccination AstraZeneca) Enter option 102 or APB Enter Following Details: First Name - Saman Surname - Perera Age - 18 City - Galle Nic - 808080 Vaccination (AstraZeneca) – 1	Display "Requested Vaccination type Booths are Occupied"	Display "Requested Vaccination type Booths are Occupied	Pass

Test Cases for Task 4

	Test Case	**Expected Result**	**Actual Result**	**Pass/Fail**
1	(Add a Patient with following Details and select the Option 100 or VVB) Enter option 102 or APB Enter Following Details: First Name - Stan Surname - Lee Age - 32 City - Colombo Nic - 20001234	100 or VVB Patient must be directed to booth no. 0 and display the additional information about the patient.	100 or VVB Patient must be directed to booth no. 0 and display the additional information about the patient.	Pass

	Vaccination (AstraZeneca) – 1 Enter option 100 or VVB			
2	(Add a Patient with following Details and select the Option 100 or VVB) Enter option 102 or APB Enter Following Details: First Name - Peter Surname - bush Age - 21 City - Kandy Nic - 654321 Vaccination (AstraZeneca) – 1 Enter option 100 or VVB	100 or VVB Patient must be directed to booth no. 1 and display the additional information about the patient.	100 or VVB Patient must be directed to booth no. 1 and display the additional information about the patient.	Pass
3	(Add a Patient with following Details) Enter option 102 or APB Enter Following Details: First Name - Jack Surname - Sue Age - 20 City - Jaffna Nic - 101010 Vaccination (AstraZeneca) – 1	Display "Requested Vaccination type Booths are Occupied Patient will be added to a waiting list" Patient must be added to the Waiting List	Display "Requested Vaccination type Booths are Occupied Patient will be added to a waiting list" Patient must be added to the Waiting List	Pass
4	(View Patients in the waiting List) Enter the option 109 or VPW	Details about the Patient that waiting to get vaccinated in the waiting list must be displayed	Details about the Patient that waiting to get vaccinated in the waiting list must be displayed	Pass
5	(Remove Patient from Booth 0) Enter option 103 or RPB and enter Booth number 0. Enter option 100 or VVB	Patient in the Waiting List must be added to the booth number 0 if the vaccination request is AstraZeneca	Patient in the Waiting List must be added to the booth number 0 if the vaccination request is AstraZeneca	Pass

Discussion

Test cases for Task 1 & Task 2

Test cases for Tasks 1 & 2 are to make sure that all the options that you can enter on the Menu is functional. There are several test cases to make sure that Add Patient and Remove patient options are running smoothly. There's a test case to check what will happens if the user input an invalid option.

Test cases for Task 3

Test cases for Task3 is to make sure that Program get the additional details from the user When a new patient is adding and display them when needed. When the Patient is added to a booth it must be according to the Vaccination type. If the Requested vaccination type booths are Full a message will Display.

Test cases for Task 4

Test cases for Task 4 is to when user tries to add three patients that requested same vaccination type, first two will be added to the correct booths and the 3rd person will be added to a waiting list. If one of the patients removes from the booth, patient in the waiting list will be added to the booth according to the vaccination type.

Code:

Task 1 – Array Version

- VacinationCenter.java

```
import java.io.*;
import java.util.Scanner;
public class VacinationCenter {


    static Scanner input = new Scanner(System.in);
    static Scanner input2 = new Scanner(System.in);
    public static void main(String[] args) {


        int stock = 150;

        int boothNum = 0;
        String[] booths = new String[6];
        //Initialising
        initialise(booths);

        boolean loop = false;
        while(!loop) {
            input = new Scanner(System.in);
            input2 = new Scanner(System.in);
            try {
                String option;
                //Displaying option menu
                System.out.println("Enter 100 or VVB: View all Vaccination Booths");
                System.out.println("Enter 101 or VEB: View all Empty Booths");
                System.out.println("Enter 102 or APB: Add Patient to a Booth");
                System.out.println("Enter 103 or RPB: Remove Patient from a Booth");
                System.out.println("Enter 104 or VPS: View Patients Sorted in
alphabetical order");
                System.out.println("Enter 105 or SPD: Store Program Data into file");
                System.out.println("Enter 106 or LPD: Load Program Data from file");
                System.out.println("Enter 107 or VRV: View Remaining Vaccinations");
                System.out.println("Enter 108 or AVS: Add Vaccinations to the Stock");
                System.out.println("Enter 999 or EXT: Exit the Program");

                System.out.println(" ");
                //Getting the option from user
                System.out.println("Enter your Option: ");
                option = input2.nextLine().toLowerCase();
                //Option Menu
                switch (option) {

                    case "vvb":
                    case "100":
                        viewBooths(booths);
                        System.out.println(" ");
                        break;
                    case "veb":
                    case "101":
```

```
                    emptyBooths(booths);
                    System.out.println(" ");
                    break;
                case "apb":
                case "102":
                    addPatient(booths);
                    stock--;
                    System.out.println(" ");
                    break;
                case "rpb":
                case "103":
                    removePatient(booths);
                    System.out.println(" ");
                    break;
                case "vps":
                case "104":
                    sortPatient(booths);
                    System.out.println(" ");
                    break;
                case "spd":
                case "105":

                    StoreData(booths,stock);
                    break;

                case "lpd":
                case "106":

                    LoadData();
                    break;

                case "vrv":
                case "107":
                    remainingVacc(stock);
                    System.out.println(" ");
                    break;

                case "avs":
                case "108":
                    stock = addVaccs(stock);
                    System.out.println(" ");
                    break;
                case "ext":
                case "999":
                    loop = true;
                    break;
                default:
                    System.out.println("Please enter a valid option.");
                    System.out.println(" ");
                    break;



            }
            //Displaying a warning message when the stock reaches 20
            if (stock == 20) {
                System.out.println("Warning only " + stock + " are Remaining.");
                System.out.println(" ");
            }
        }catch (Exception e){
            System.out.println("Please enter a valid input.... ");
```

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

            }

        }


    }

    private static void initialise(String[] array) {
        for (int x = 0; x < 6; x++) array[x] = "Empty";
        System.out.println("initilise ");
    }
    //method to View all Vaccination Booths
    private static void viewBooths(String[] array){
        for (int x = 0; x < 6; x++){
            if (array[x].equals("Empty")){
                System.out.println("Booth " + x + " is Empty");

            }
            else {
                System.out.println("Booth " + x +" is Occupied by " + array[x] );

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

    }
    //View all Empty Booths method
    private static void emptyBooths(String[] array) {
        for (int x = 0; x < 6; x++) {
            if (array[x].equals("Empty")) {
                System.out.println("Booth " + x + " is Empty");
                System.out.println(" ");
            }
        }

    }
    //method to Add a patient to the booth
    private static void addPatient(String[] array){
        String name;
        int boothNum ;

        System.out.println("Enter a booth number to Add a Patient (0-5)");
        boothNum = input.nextInt();
        //if someone is already in the booth a message will display
        if (boothNum > 5){
            System.out.println("Please select a booth number between 0 and 5");
        }
        else if(!array[boothNum].equals("Empty")){
            System.out.println("This both is Occupied");
        }
        else {
            do {
                System.out.println("Enter the Patient name for the Booth " + boothNum +
": ");
                name = input2.nextLine();
            } while(name.isEmpty());
            array[boothNum] = name;
        }
```

```
    }
    //method to Remove a patient from the booth
    private static void removePatient(String[] array){
        int boothNum ;

        System.out.println("Enter a booth number to Remove a Patient (0-5)");
        boothNum = input.nextInt();
        if (boothNum > 5){
            System.out.println("Please select a booth number between 0 and 5");
        }
        else if (array[boothNum].equals("Empty")) {
            System.out.println("This Booth is already Empty");
        }
        else {
            array[boothNum] = "Empty";
        }

    }
    //method to Sort Patients Names
    private static void sortPatient(String[] array){
        //Counting the number of Patient that currently getting vaccinated
        int nameCount = 0;
        for(int n=0; n < array.length; n++){
            if(!array[n].equals("Empty")){
                nameCount +=1;
            }
        }
        //Creating an new list for the patient that currently getting vaccinated
        int q=0;
        String[] sortNames = new String[nameCount];
        for(int x=0; x< array.length; x++){

            if(!array[x].equals("Empty")){
                sortNames[q] = array[x];
                q ++;
            }
        }
        //Sorting list of names
        for(int n=0; n< sortNames.length; n++){
            for(int x = n + 1; x< sortNames.length; x++){
                String temp = sortNames[x];

                int num = sortNames[n].compareToIgnoreCase(sortNames[x]);
                if (num > 0){
                    sortNames[x] = sortNames[n];
                    sortNames[n] = temp ;
                }
            }
        }

        //Displaying the Sorted names list
        for(int n=0; n< sortNames.length; n++){
            System.out.println(sortNames[n]);
        }


    }

    private static void remainingVacc(int x){
        System.out.println("Remaining vaccine amount is: " + x);
    }
```

```
private static int addVaccs(int stock){
    System.out.println("Enter the amount of vaccinations adding to the Stock: ");
    int amount = input.nextInt();
    stock += amount;
    return stock;
}
//Saving Program Data to a File
private static void StoreData(String[] firstNameArr,int stock){
    try{
        FileWriter data = new FileWriter("Task1_Array.txt");


        for (int i=0; i < firstNameArr.length; i++){
            if (!firstNameArr[i].equals("Empty")) {
                data.write("--------Booth Number " + i + "--------" +"\n");
                data.write("First name : " + firstNameArr[i] + "\n");

                data.write("\n ");
            }
            else{
                data.write("--------Booth Number " + i + "--------" +"\n");
                data.write("This Booth is empty" + "\n");
                data.write("\n ");
            }
        }
        data.write("Remaining vaccine amount is: " + stock + "\n\n");
        data.close();
        System.out.println("Successfully wrote to the file.");
        System.out.println(" ");
    } catch (IOException e){
        System.out.println("An error occurred.");
        System.out.println(" ");
        e.printStackTrace();
    }
}

//Retrieving Program data from the File
private static void LoadData(){

    try {
        File load = new File("Task1_Array.txt");
        if (load.exists()) {
            BufferedReader br = new BufferedReader(new FileReader(load));

            String st;
            while ((st = br.readLine()) != null) {
                System.out.println(st);
                System.out.print("");
            }

        } else {
            System.out.println("The file does not exist.");
        }

    } catch (IOException e){
        System.out.println("An error occurred.");
        System.out.println(" ");
        e.printStackTrace();
    }
}
```

```
}
```

Test 2 – Class Version

- Booth.java Class

```
public class Booth {

    private  int stock = 150;
    private String name = "Empty";


    public Booth(){

    }



    //This is a method to view the situation of the booth
    public  void viewBooth(int x){
        if (name.equals("Empty")){
            System.out.println("Booth number " + x + " is Empty.");
        }
        else {
            System.out.println("Booth number " + x +" is Occupied by " + name);
        }
    }

    //This method will identify empty booth
    public  void emptyBooth(int x){
        if (name.equals("Empty")){
            System.out.println("Booth number " + x + " is Empty.");
        }
    }



    //This method will return the Name
    public  String getName(){
        return name;
    }
    //This method will set the New Name
    public  void setName(String n){
        name = n;
    }

    //Reducing the vaccine Stock
    public  void setStock(){
        stock --;
    }
    //Adding new Vaccines to the Stock
    public  void setStock(int num){
        stock += num;
    }
    //Returning the remaining Vaccine Stock
```

```
    public int getStock(){
        return stock;
    }

}
```

- VacinationCenter.java Class (Main Class)

```
import java.io.*;
import java.util.Scanner;

public class VacinationCenter {


    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Scanner input2 = new Scanner(System.in);
        int boothNum;
        String[] sortName;
        int sortNameCount = 0;

        String firstName;


        Booth stock = new Booth();

        Booth[] centerArr = new Booth[6];
        centerArr[0] = new Booth();
        centerArr[1] = new Booth();
        centerArr[2] = new Booth();
        centerArr[3] = new Booth();
        centerArr[4] = new Booth();
        centerArr[5] = new Booth();



        boolean loop = false;
        while (!loop) {
            input = new Scanner(System.in);
            input2 = new Scanner(System.in);
            try{
                //Displaying option menu
                System.out.println("Enter 100 or VVB: View all Vaccination Booths");
                System.out.println("Enter 101 or VEB: View all Empty Booths");
                System.out.println("Enter 102 or APB: Add Patient to a Booth");
                System.out.println("Enter 103 or RPB: Remove Patient from a Booth");
                System.out.println("Enter 104 or VPS: View Patients Sorted in
alphabetical order");
                System.out.println("Enter 105 or SPD: Store Program Data into file");
                System.out.println("Enter 106 or LPD: Load Program Data from file");
                System.out.println("Enter 107 or VRV: View Remaining Vaccinations");
                System.out.println("Enter 108 or AVS: Add Vaccinations to the Stock");
                System.out.println("Enter 999 or EXT: Exit the Program");
                System.out.println(" ");

                //Getting the option from user
```

```
                System.out.println("Enter your Option: ");
                String option = input2.nextLine().toLowerCase();

                //Option Menu
                switch (option) {
                    case "vvb":
                    case "100":
                        for (int x = 0; x < centerArr.length; x++) {
                            centerArr[x].viewBooth(x);
                            System.out.println(" ");
                        }

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

                    case "veb":
                    case "101":
                        for (int x = 0; x < centerArr.length; x++) {
                            centerArr[x].emptyBooth(x);

                        }
                        System.out.println(" ");
                        break;

                    case "apb":
                    case "102":

                        System.out.println("Enter a booth number to Add a Patient (0-5):
");
                        boothNum = input.nextInt();



                        if (boothNum > 5){
                            System.out.println("Please select a booth number between 0
and 5");
                        }
                        else if(!centerArr[boothNum ].getName().equals("Empty")){
                            System.out.println("This both is Occupied");
                        }
                        else {
                            do {
                                System.out.println("Enter the Patient name for the Booth
" + boothNum + ": ");
                                firstName = input2.nextLine();
                            } while (firstName.isEmpty());
                            centerArr[boothNum].setName(firstName);
                            stock.setStock();
                            sortNameCount ++;

                        }
                        System.out.println(" ");
                        break;

                    case "rpb":
                    case "103":
                        System.out.println("Enter a booth number to Remove a Patient (0-
5)");
                        boothNum = input.nextInt();

                        if (centerArr[boothNum].getName().equals("Empty")) {
```

```
                            System.out.println("This Booth is already Empty");
                        } else {
                            centerArr[boothNum].setName("Empty");
                            System.out.println("Patient will be removed from the booth");
                            sortNameCount--;
                        }
                        System.out.println(" ");
                        break;

                    case "vps":
                    case "104":
                        sortName = new String[sortNameCount];
                        int y = 0;
                        //Creating an new list for the patient that currently getting
vaccinated
                        for (int n = 0; n < centerArr.length; n++) {
                            if (!centerArr[n].getName().equals("Empty")) {
                                sortName[y] = centerArr[n].getName();
                                y++;
                            }
                        }
                        //Sorting list of names
                        for (int n = 0; n < sortName.length; n++) {
                            for (int x = n + 1; x < sortName.length; x++) {
                                String temp = sortName[x];

                                int num = sortName[n].compareToIgnoreCase(sortName[x]);
                                if (num > 0) {
                                    sortName[x] = sortName[n];
                                    sortName[n] = temp;
                                }
                            }
                        }
                        //Displaying the Sorted names list
                        for (int n = 0; n < sortName.length; n++) {
                            System.out.println(sortName[n]);
                        }
                        System.out.println(" ");
                        break;

                    case "spd":
                    case "105":

                        //Saving Program Data to a File
                        try{
                            FileWriter data = new FileWriter("Task2_Class.txt");


                            for (int i=0; i < centerArr.length; i++){
                                if (!centerArr[i].getName().equals("Empty")) {
                                    data.write("----------------Booth Number " + i + "---
------------" +"\n");
                                    data.write("Name      : " + centerArr[i].getName() +
"\n");

                                    data.write("\n ");
                                }
                                else{
                                    data.write("--------Booth Number " + i + "--------"
+"\n");
                                    data.write("This Booth is empty" + "\n");
```

```
                                data.write("\n ");
                            }
                        }
                        data.write("Remaining vaccine amount is: " + stock.getStock()
+ "\n\n");
                        data.close();
                        System.out.println("Successfully wrote to the file.");
                        System.out.println(" ");
                    } catch (IOException e){
                        System.out.println("An error occurred.");
                        System.out.println(" ");
                        e.printStackTrace();
                    }

                    break;
                case "lpd":
                case "106":

                    //Retrieving Program data from the File
                    try {
                        File load = new File("Task2_Class.txt");
                        if (load.exists()) {
                            BufferedReader br = new BufferedReader(new
FileReader(load));


                            String st;
                            while ((st = br.readLine()) != null) {
                                System.out.println(st);
                                System.out.print("");
                            }

                        } else {
                            System.out.println("The file does not exist.");
                        }

                    } catch (IOException e){
                        System.out.println("An error occurred.");
                        System.out.println(" ");
                        e.printStackTrace();
                    }


                    break;

                case "vrv":
                case "107":
                    int left = stock.getStock();
                    System.out.println("Remaining vaccine amount is: " + left);
                    System.out.println(" ");
                    break;


                case "avs":
                case "108":
                    System.out.println("Enter the amount of vaccinations adding to
the Stock: ");
                    int amount = input.nextInt();
                    stock.setStock(amount);
                    break;
```

```
                    case "ext":
                    case "999":
                        loop = true;
                        break;
                    default:
                        System.out.println("Please enter a valid option.");
                        System.out.println(" ");


                }
                //Displaying a warning message when the stock reaches 20
                if (stock.getStock() == 20) {
                    System.out.println("Warning only 20 vaccines are Remaining.");
                    System.out.println(" ");
                }
            }catch (Exception e) {
                System.out.println("Please enter a valid input.... ");
            }
        }


    }



}
```

Task 3 – Array Version

- VacinationCenter.java

```
import java.io.*;
import java.util.Scanner;
public class VacinationCenter{


    static Scanner input = new Scanner(System.in);
    static Scanner input2 = new Scanner(System.in);
    public static void main(String[] args) {


        int stock = 150;
        String firstName;
        String lastName;
        int boothNum = 0;
        String[] firstNameArr = new String[6];
        String[] lastNameArr = new String[6];

        //Initialising
        initialise(firstNameArr);

        boolean loop = false;
        while (!loop) {
            try {
                //Displaying option menu
                System.out.println("Enter 100 or VVB: View all Vaccination Booths");
                System.out.println("Enter 101 or VEB: View all Empty Booths");
                System.out.println("Enter 102 or APB:  Add Patient to a Booth");
                System.out.println("Enter 103 or RPB: Remove Patient from a Booth");
                System.out.println("Enter 104 or VPS: View Patients Sorted in
alphabetical order");
                System.out.println("Enter 105 or SPD: Store Program Data into file");
                System.out.println("Enter 106 or LPD: Load Program Data from file");
                System.out.println("Enter 107 or VRV: View Remaining Vaccinations");
                System.out.println("Enter 108 or AVS: Add Vaccinations to the Stock");
                System.out.println("Enter 999 or EXT: Exit the Program");

                System.out.println(" ");
                //Getting the option from user
                System.out.println("Enter your Option: ");
                String option = input2.nextLine().toLowerCase();

                //Option Menu
                switch (option) {
                    case "vvb":
                    case "100":
                        getPatientInfo(firstNameArr,lastNameArr);
                        System.out.println(" ");
                        break;
                    case "veb":
                    case "101":
```

```
                        emptyBooths(firstNameArr);
                        System.out.println(" ");
                        break;
                    case "apb":
                    case "102":
                        do {
                            System.out.println("Enter the First Name of the Patient  for
the Booth :");
                            firstName = input2.nextLine();
                        } while (firstName.isEmpty());

                        do {
                            System.out.println("Enter the Surname Name of the Patient:
");
                            lastName = input2.nextLine();
                        } while (lastName.isEmpty());

                        System.out.println("Enter the number of Requested vaccination "
);
                        System.out.println("AstraZeneca - 1" );
                        System.out.println("Sinopharm   - 2" );
                        System.out.println("Pfizer      - 3" );
                        int req = input.nextInt();

                        addPatient(firstNameArr,req,firstName);
                        setPatientInfo(firstNameArr,lastNameArr,firstName,lastName);
                        stock--;
                        System.out.println(" ");
                        break;

                    case "rpb":
                    case "103":
                        removePatient(firstNameArr);
                        System.out.println(" ");
                        break;
                    case "vps":
                    case "104":
                        sortPatient(firstNameArr);
                        System.out.println(" ");
                        break;
                    case "spd":
                    case "105":
                        StoreData(firstNameArr,lastNameArr,stock);
                        break;
                    case "lpd":
                    case "106":
                        LoadData();
                        break;

                    case "vrv":
                    case "107":
                        remainingVacc(stock);
                        System.out.println(" ");
                        break;
                    case "avs":
                    case "108":
                        stock = addVaccs(stock);
                        System.out.println(" ");
                        break;

                    case "ext":
```

```
                    case "999":
                        loop = true;
                    default:
                        System.out.println("Please enter a valid option.");
                        System.out.println(" ");

                }
                //Displaying a warning message when the stock reaches 20
                if (stock == 20) {
                    System.out.println("Warning only " + stock + " are Remaining.");
                    System.out.println(" ");
                }
            }catch (Exception e) {
                System.out.println("Please enter a valid input.... ");
            }
        }



    }

    private static void initialise(String[] array) {
        for (int x = 0; x < 6; x++) array[x] = "Empty";
        System.out.println("initilise ");
    }
    //method to View all Vaccination Booths
    private static void viewBooths(String[] array){
        for (int x = 0; x < 6; x++){
            if (array[x].equals("Empty")){
                System.out.println("Booth " + x + " is Empty");
            }
            else {
                System.out.println("Booth " + x +" is Occupied by " + array[x] );
            }
        }

    }
    //View all Empty Booths method
    private static void emptyBooths(String[] array) {
        for (int x = 0; x < 6; x++) {
            if (array[x].equals("Empty")) {
                System.out.println("Booth " + x + " is Empty");
            }
        }

    }
    //method to Add a patient to the booth
    private static void addPatient(String[] array, int requested, String name){
        int[] vaccs = {1,1,2,2,3,3};
        for (int x=0; x < vaccs.length; x++){
            if(requested ==vaccs[x]) {
                if (array[x].equals("Empty")) {
                    array[x] = name;
                    System.out.println("Patient has been directed to the proper booth");
                    break;
                } else if (array[x + 1].equals("Empty")) {
                    array[x + 1] = name;
                    System.out.println("Patient will be directed to the proper booth");
                    break;
                } else {
                    System.out.println("Requested Vaccination type Booths are Occupied");
```

```
                    break;
                }
            }

        }



    }
    //method to Remove a patient from the booth
    private static void removePatient(String[] array){
        int boothNum ;

        System.out.println("Enter a booth number to Remove a Patient (0-5)");
        boothNum = input.nextInt();

        if (boothNum > 5){
            System.out.println("Please select a booth number between 0 and 5");
        }
        else if (array[boothNum].equals("Empty")) {
            System.out.println("This Booth is already Empty");
        }
        else {
            array[boothNum] = "Empty";
            System.out.println("Patient will be removed from the booth");
        }

    }
    //method
    private static void sortPatient(String[] array){
        //Counting the number of Patient that currently getting vaccinated
        int nameCount = 0;
        for(int n=0; n < array.length; n++){
            if(!array[n].equals("Empty")){
                nameCount +=1;
            }
        }
        //Creating an new list for the patient that currently getting vaccinated
        int q=0;
        String[] sortNames = new String[nameCount];
        for(int x=0; x< array.length; x++){

            if(!array[x].equals("Empty")){
                sortNames[q] = array[x];
                q ++;
            }
        }
        //Sorting list of names
        for(int n=0; n< sortNames.length; n++){
            for(int x = n + 1; x< sortNames.length; x++){
                String temp = sortNames[x];

                int num = sortNames[n].compareToIgnoreCase(sortNames[x]);
                if (num > 0){
                    sortNames[x] = sortNames[n];
                    sortNames[n] = temp ;
                }
            }
        }

        //Displaying the Sorted names list
```

```
        for(int n=0; n< sortNames.length; n++){
            System.out.println(sortNames[n]);
        }


    }

    private static void remainingVacc(int x){
        System.out.println("Remaining vaccine amount is: " + x);
    }

    private static int addVaccs(int stock){
        System.out.println("Enter the amount of vaccinations adding to the Stock: ");
        int amount = input.nextInt();

        stock += amount;
        return stock;
    }

    private static void setPatientInfo(String[] array, String[] lastName,String name,
String LName) {


        for (int x = 0; x < array.length; x++) {
            if (array[x].equals(name)) {
                lastName[x] = LName;
                break;
            }
        }
    }

    private static void getPatientInfo(String[] array, String[] lastName ){
        String[] vaccType = {"AstraZeneca", "AstraZeneca", "Sinopharm", "Sinopharm",
"Pfizer", "Pfizer"};
        for (int n=0; n < array.length; n++){
            if (!array[n].equals("Empty")){
                System.out.println("Booth number  - " + n);
                System.out.println("First Name is " + array[n]);
                System.out.println("Last Name is " + lastName[n]);
                System.out.println("Requested Vacation is " + vaccType[n]);
                System.out.println(" ");
            }
            else {
                System.out.println("Booth number  - " + n);
                System.out.println("This Booth is empty");
                System.out.println(" ");
            }
        }


    }
    //Saving Program Data to a File
    private static void StoreData(String[] firstNameArr, String[] lastNameArr, int
stock){
        try{
            FileWriter data = new FileWriter("Task3_Array.txt");

            String[] vaccType = {"AstraZeneca", "AstraZeneca", "Sinopharm", "Sinopharm",
"Pfizer", "Pfizer"};
            for (int i=0; i < firstNameArr.length; i++){
                if (!firstNameArr[i].equals("Empty")) {
```

```
                    data.write("--------Booth Number " + i + "--------" +"\n");
                    data.write("First name : " + firstNameArr[i] + "\n");
                    data.write("Surname : " + lastNameArr[i] + "\n");
                    data.write("Vaccination Type: " + vaccType[i] + "\n");
                    data.write("\n ");
                }
                else{
                    data.write("--------Booth Number " + i + "--------" +"\n");
                    data.write("This Booth is empty" + "\n");
                    data.write("\n ");
                }
            }
            data.write("Remaining vaccine amount is: " + stock + "\n\n");
            data.close();
            System.out.println("Successfully wrote to the file.");
            System.out.println(" ");
        } catch (IOException e){
            System.out.println("An error occurred.");
            System.out.println(" ");
            e.printStackTrace();
        }
    }

    //Retrieving Program data from the File
    private static void LoadData(){

        try {
            File load = new File("Task3_Array.txt");
            if (load.exists()) {
                BufferedReader br = new BufferedReader(new FileReader(load));

                String st;
                while ((st = br.readLine()) != null) {
                    System.out.println(st);
                    System.out.print("");
                }

            } else {
                System.out.println("The file does not exist.");
            }

        } catch (IOException e){
            System.out.println("An error occurred.");
            System.out.println(" ");
            e.printStackTrace();
        }
    }

}
```

Task 3 – Class Version

- Booth.java Class

```
public class Booth {

    private int stock = 150;
    private  String name = "Empty";
    private int vaccsType;

    public Booth(){

    }
    //This Constructor will get the assigned value for the Vaccination type( 1-
AstraZeneca, 2-Sinopharm, 3-Pfizer
    public Booth(int type ){
        this.vaccsType = type;
    }

    //This is a method to view the situation of the booth
    public void viewBooth(int x){
        if (name.equals("Empty")){
            System.out.println("Booth number " + x + " is Empty.");
        }
        else {
            System.out.println("Booth number " + x +" is Occupied by " + name);
        }
    }

    //This method will identify empty booth
    public void emptyBooth(int x){
        if (name.equals("Empty")){
            System.out.println("Booth number " + x + " is Empty.");
        }
    }

    //This method will return the value that belongs to the assigned Vaccination type
    public int getVaccsType(){
        return vaccsType;
    }

    //This method will return the Name
    public String getName(){
        return name;
    }
    //This method will set the New Name
    public void setName(String name){
        this.name = name;
    }

    //Reducing the vaccine Stock
    public void setStock(){
        this.stock --;
    }
```

```
    //Adding new Vaccines to the Stock
    public void setStock(int num){
        this.stock += num;
    }
    //Returning the remaining Vaccine Stock
    public int getStock(){
        return stock;
    }

}
```

- Patient.java Class

```
public class Patient {

    private String firstName = "Empty";
    private String lasttName;
    private int age;
    private String city;
    private long id;
    private String vaccsType;
    public Patient(){

    }

    public void setValues (String firstName, String lasttName, int age, String city ,long
id , int type){
        this.firstName = firstName;
        this.lasttName = lasttName;
        this.age = age;
        this.city = city ;
        this.id = id;

        if (type == 1){
            this.vaccsType = "AstraZeneca";
        }
        else if (type == 2){
            this.vaccsType = "Sinopharm";
        }
        else{
            this.vaccsType = "Pfizer";
        }
    }

    public void getValues(int boothNum){
        if (firstName.equals("Empty")){
            System.out.println("---------Booth number " + boothNum + " is Empty---------
");
        }
        else {
            System.out.println("---------Booth number " + boothNum + "---------");
            System.out.println("First Name: " + firstName);
            System.out.println("Surname   : " + lasttName);
            System.out.println("Age       : " + age);
            System.out.println("City      : " + city);
            System.out.println("NIC or Passport Number: " + id);
```

```
            System.out.println("Vaccination Requested: " + vaccsType);
        }
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLasttName() {
        return lasttName;
    }

    public int getAge() {
        return age;
    }

    public String getCity() {
        return city;
    }

    public long getId() {
        return id;
    }

    public String getVaccsType() {
        return vaccsType;
    }
}
```

- VacinationCenter.java Class (Main Class)

```
import java.io.*;
import java.util.Scanner;

public class VacinationCenter {


    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Scanner input2 = new Scanner(System.in);
        int boothNum;
        String[] sortName;
        int sortNameCount = 0;

        String firstName;
        String lastName;
        int age;
        String city;
        long id;
```

```
        int vaccsReq;
        Booth stock = new Booth();

        Booth[] centerArr = new Booth[6];
        centerArr[0] = new Booth(1);
        centerArr[1] = new Booth(1);
        centerArr[2] = new Booth(2);
        centerArr[3] = new Booth(2);
        centerArr[4] = new Booth(3);
        centerArr[5] = new Booth(3);

        Patient[] patientsArr = new Patient[6];
        patientsArr[0] = new Patient();
        patientsArr[1] = new Patient();
        patientsArr[2] = new Patient();
        patientsArr[3] = new Patient();
        patientsArr[4] = new Patient();
        patientsArr[5] = new Patient();

        boolean loop = false;
        while (!loop) {
            input = new Scanner(System.in);
            input2 = new Scanner(System.in);
            try{
                //Displaying option menu
                System.out.println("Enter 100 or VVB: View all Vaccination Booths");
                System.out.println("Enter 101 or VEB: View all Empty Booths");
                System.out.println("Enter 102 or APB:  Add Patient to a Booth");
                System.out.println("Enter 103 or RPB: Remove Patient from a Booth");
                System.out.println("Enter 104 or VPS: View Patients Sorted in
alphabetical order");
                System.out.println("Enter 105 or SPD: Store Program Data into file");
                System.out.println("Enter 106 or LPD: Load Program Data from file");
                System.out.println("Enter 107 or VRV: View Remaining Vaccinations");
                System.out.println("Enter 108 or AVS: Add Vaccinations to the Stock");
                System.out.println("Enter 999 or EXT: Exit the Program");
                System.out.println(" ");

                //Getting the option from user
                System.out.println("Enter your Option: ");
                String option = input2.nextLine().toLowerCase();

                //Option Menu
                switch (option) {
                    case "vvb":
                    case "100":
                        for (int x = 0; x < patientsArr.length; x++) {
                            patientsArr[x].getValues(x);
                            System.out.println(" ");
                        }

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

                    case "veb":
                    case "101":
                        for (int x = 0; x < centerArr.length; x++) {
                            centerArr[x].emptyBooth(x);
                        }
                        System.out.println(" ");
                        break;
```

```
                    case "apb":
                    case "102":

                        do {
                            System.out.println("Enter the First Name of the Patient  for
the Booth: ");
                            firstName = input2.nextLine();
                        } while (firstName.isEmpty());

                        do {
                            System.out.println("Enter the Surname Name of the Patient:
");
                            lastName = input2.nextLine();
                        } while (lastName.isEmpty());

                        System.out.println("Enter the age of the Patient: ");
                        age = input.nextInt();

                        do {
                            System.out.println("Enter the City of the Patient: ");
                            city = input2.nextLine();
                        } while (city.isEmpty());

                        System.out.println("Enter the NIC or Passport Number of the
Patient: ");
                        id = input.nextLong();
                        System.out.println("Enter the number of Requested vaccination ");
                        System.out.println("AstraZeneca - 1");
                        System.out.println("Sinopharm   - 2");
                        System.out.println("Pfizer      - 3");
                        vaccsReq = input.nextInt();

                        for (int x = 0; x < centerArr.length; x++) {

                            if (centerArr[x].getVaccsType() == vaccsReq) {
                                if (centerArr[x].getName().equals("Empty")) {
                                    centerArr[x].setName(firstName);
                                    System.out.println("Patient has been directed to the
proper booth");
                                    patientsArr[x].setValues(firstName, lastName, age,
city, id, vaccsReq);
                                    stock.setStock();
                                    sortNameCount++;
                                    break;
                                } else if (centerArr[x + 1].getName().equals("Empty")) {
                                    centerArr[x + 1].setName(firstName);
                                    System.out.println("Patient will be directed to the
proper booth");
                                    patientsArr[x + 1].setValues(firstName, lastName,
age, city, id, vaccsReq);
                                    stock.setStock();
                                    sortNameCount++;
                                    break;
                                } else {
                                    System.out.println("Requested Vaccination type Booths
are Occupied");
                                    break;
                                }

                            }
```

```
                        }
                        System.out.println(" ");

                        break;

                    case "rpb":
                    case "103":
                        System.out.println("Enter a booth number to Remove a Patient (0-
5)");
                        boothNum = input.nextInt();
                        if (centerArr[boothNum].getName().equals("Empty")) {
                            System.out.println("This Booth is already Empty");
                        } else {
                            centerArr[boothNum].setName("Empty");
                            patientsArr[boothNum].setFirstName("Empty");
                            System.out.println("Patient will be removed from the booth");
                            sortNameCount--;
                        }
                        System.out.println(" ");
                        break;

                    case "vps":
                    case "104":
                        sortName = new String[sortNameCount];
                        int y = 0;
                        //Creating an new list for the patient that currently getting
vaccinated
                        for (int n = 0; n < centerArr.length; n++) {
                            if (!centerArr[n].getName().equals("Empty")) {
                                sortName[y] = centerArr[n].getName();
                                y++;
                            }
                        }
                        //Sorting list of names
                        for (int n = 0; n < sortName.length; n++) {
                            for (int x = n + 1; x < sortName.length; x++) {
                                String temp = sortName[x];

                                int num = sortName[n].compareToIgnoreCase(sortName[x]);
                                if (num > 0) {
                                    sortName[x] = sortName[n];
                                    sortName[n] = temp;
                                }
                            }
                        }
                        //Displaying the Sorted names list
                        for (int n = 0; n < sortName.length; n++) {
                            System.out.println(sortName[n]);
                        }
                        System.out.println(" ");
                        break;

                    case "spd":
                    case "105":

                        //Saving Program Data to a File
                        try{
                            FileWriter data = new FileWriter("Task3_Class.txt");
```

```
                            for (int i=0; i < centerArr.length; i++){
                                if (!centerArr[i].getName().equals("Empty")) {
                                    data.write("----------------Booth Number " + i + "---
-------------" +"\n");
                                    data.write("First name      : " +
patientsArr[i].getFirstName() + "\n");
                                    data.write("Surname         : " +
patientsArr[i].getLasttName() + "\n");
                                    data.write("Age             : " +
patientsArr[i].getAge() + "\n");
                                    data.write("City            : " +
patientsArr[i].getCity() + "\n");
                                    data.write("Nic or Passport : " +
patientsArr[i].getId() + "\n");
                                    data.write(  "Vaccination Type : " +
patientsArr[i].getVaccsType()  + "\n");
                                    data.write("\n ");
                                }
                                else{
                                    data.write("--------Booth Number " + i + "--------"
+"\n");
                                    data.write("This Booth is empty" + "\n");
                                    data.write("\n ");
                                }
                            }
                            data.write("Remaining vaccine amount is: " + stock.getStock()
+ "\n\n");
                            data.close();
                            System.out.println("Successfully wrote to the file.");
                            System.out.println(" ");
                        } catch (IOException e){
                            System.out.println("An error occurred.");
                            System.out.println(" ");
                            e.printStackTrace();
                        }

                        break;
                    case "lpd":
                    case "106":

                        //Retrieving Program data from the File
                        try {
                            File load = new File("Task3_Class.txt");
                            if (load.exists()) {
                                BufferedReader br = new BufferedReader(new
FileReader(load));

                                String st;
                                while ((st = br.readLine()) != null) {
                                    System.out.println(st);
                                    System.out.print("");
                                }

                            } else {
                                System.out.println("The file does not exist.");
                            }

                        } catch (IOException e){
                            System.out.println("An error occurred.");
                            System.out.println(" ");
                            e.printStackTrace();
```

```
                        }


                        break;

                    case "vrv":
                    case "107":
                        int left = stock.getStock();
                        System.out.println("Remaining vaccine amount is: " + left);
                        System.out.println(" ");
                        break;


                    case "avs":
                    case "108":
                        System.out.println("Enter the amount of vaccinations adding to
the Stock: ");
                        int amount = input.nextInt();
                        stock.setStock(amount);
                        break;


                    case "ext":
                    case "999":
                        loop = true;
                        break;
                    default:
                        System.out.println("Please enter a valid option.");
                        System.out.println(" ");


                }
                //Displaying a warning message when the stock reaches 20
                if (stock.getStock() == 20) {
                    System.out.println("Warning only 20 vaccines are Remaining.");
                    System.out.println(" ");
                }
            }catch (Exception e) {
                System.out.println("Please enter a valid input.... ");
            }
        }


    }



}
```

Task 4 – Class Version

- Booth.java Class

```
public class Booth {

    private int stock = 150;
    private  String name = "Empty";
    private int vaccsType;

    public Booth(){

    }
    public Booth(String name, int type){
        this.name = name;
        this.vaccsType = type;
    }
    //This Constructor will get the assigned value for the Vaccination type( 1-
AstraZeneca, 2-Sinopharm, 3-Pfizer
    public Booth(int type ){
        this.vaccsType = type;
    }

    //This is a method to view the situation of the booth
    public void viewBooth(int x){
        if (name.equals("Empty")){
            System.out.println("Booth number " + x + " is Empty.");
        }
        else {
            System.out.println("Booth number " + x +" is Occupied by " + name);
        }
    }

    //This method will identify empty booth
    public void emptyBooth(int x){
        if (name.equals("Empty")){
            System.out.println("Booth number " + x + " is Empty.");
        }
    }

    //This method will return the value that belongs to the assigned Vaccination type
    public int getVaccsType(){
        return vaccsType;
    }

    //This method will return the Name
    public String getName(){
        return name;
    }
    //This method will set the New Name
    public void setName(String name){
        this.name = name;
    }
```

```
    //Reducing the vaccine Stock
    public void setStock(){
        this.stock --;
    }
    //Adding new Vaccines to the Stock
    public void setStock(int num){
        this.stock += num;
    }
    //Returning the remaining Vaccine Stock
    public int getStock(){
        return stock;
    }

}
```

- Patient.java Class

```
public class Patient {

    private String firstName = "Empty";
    private String lasttName;
    private int age;
    private String city;
    private long id;
    private String vaccsType;
    public Patient(){

    }
    public  Patient(String firstName, String lasttName, int age, String city ,long
id , int type){
        this.setFirstName(firstName);
        this.setLasttName(lasttName);
        this.setAge(age);
        this.setCity(city);
        this.setId(id);

        if (type == 1){
            this.setVaccsType("AstraZeneca");
        }
        else if (type == 2){
            this.setVaccsType("Sinopharm");
        }
        else{
            this.setVaccsType("Pfizer");
        }
    }

    public void setValues (String firstName, String lasttName, int age, String city
,long id , int type){
        this.setFirstName(firstName);
        this.setLasttName(lasttName);
        this.setAge(age);
        this.setCity(city);
        this.setId(id);

        if (type == 1){
```

```
            this.setVaccsType("AstraZeneca");
        }
        else if (type == 2){
            this.setVaccsType("Sinopharm");
        }
        else{
            this.setVaccsType("Pfizer");
        }
    }

    public void getValues(int boothNum){
        if (getFirstName().equals("Empty")){
            System.out.println("----------Booth number " + boothNum + " is Empty---
-------");
        }
        else {
            System.out.println("----------Booth number - " + boothNum + "----------
");
            System.out.println("First Name: " + getFirstName());
            System.out.println("Surname   : " + getLasttName());
            System.out.println("Age       : " + getAge());
            System.out.println("City      : " + getCity());
            System.out.println("NIC or Passport Number: " + getId());
            System.out.println("Vaccination Requested: " + getVaccsType());
        }
    }


    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLasttName() {
        return lasttName;
    }

    public void setLasttName(String lasttName) {
        this.lasttName = lasttName;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public long getId() {
```

```
            return id;
        }

        public void setId(long id) {
            this.id = id;
        }

        public String getVaccsType() {
            return vaccsType;
        }

        public void setVaccsType(String vaccsType) {
            this.vaccsType = vaccsType;
        }
    }
```

- VacinationCenter.java Class (Main class)

```
import java.io.*;
import java.util.LinkedList;
import java.util.Scanner;

public class VacinationCenter {


    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        Scanner input2 = new Scanner(System.in);
        int boothNum;
        //sortName Array is to store the Alphabetically sorted name list
        String[] sortName;
        //"sortNameCount" variable is to get the count of patients that currently getting
vaccinated
        int sortNameCount = 0;
        //Following Variables will be used to store the User inputs
        String firstName;
        String lastName;
        int age;
        String city;
        long id;
        int vaccsReq;
        //This will help to keep the track of vaccination stock and add new stock
        Booth stock = new Booth();

        Booth[] centerArr = new Booth[6];
        centerArr[0] = new Booth(1);
        centerArr[1] = new Booth(1);
        centerArr[2] = new Booth(2);
        centerArr[3] = new Booth(2);
        centerArr[4] = new Booth(3);
        centerArr[5] = new Booth(3);

        Patient[] patientsArr = new Patient[6];
        patientsArr[0] = new Patient();
```

```
        patientsArr[1] = new Patient();
        patientsArr[2] = new Patient();
        patientsArr[3] = new Patient();
        patientsArr[4] = new Patient();
        patientsArr[5] = new Patient();

        LinkedList<Booth> waitingListArr = new LinkedList<>();
        LinkedList<Patient> waitingPatientList = new LinkedList<>();


        boolean loop = false;
        while (!loop) {
            try {
                input = new Scanner(System.in);
                input2 = new Scanner(System.in);
                //Displaying option menu
                System.out.println("Enter 100 or VVB: View all Vaccination Booths");
                System.out.println("Enter 101 or VEB: View all Empty Booths");
                System.out.println("Enter 102 or APB: Add Patient to a Booth");
                System.out.println("Enter 103 or RPB: Remove Patient from a Booth");
                System.out.println("Enter 104 or VPS: View Patients Sorted in
alphabetical order");
                System.out.println("Enter 105 or SPD: Store Program Data into file");
                System.out.println("Enter 106 or LPD: Load Program Data from file");
                System.out.println("Enter 107 or VRV: View Remaining Vaccinations");
                System.out.println("Enter 108 or AVS: Add Vaccinations to the Stock");
                System.out.println("Enter 999 or EXT: Exit the Program");
                System.out.println(" ");

                //Getting the option from user
                System.out.println("Enter your Option: ");
                String option = input2.nextLine().toLowerCase();

                //Option Menu
                switch (option) {


                    case "vvb":
                    case "100":
                        for (int x = 0; x < patientsArr.length; x++) {
                            patientsArr[x].getValues(x);
                            System.out.println(" ");
                        }

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

                    case "veb":
                    case "101":
                        for (int x = 0; x < centerArr.length; x++) {
                            centerArr[x].emptyBooth(x);
                        }
                        System.out.println(" ");
                        break;

                    case "apb":
                    case "102":

                        do {
                            System.out.println("Enter the First Name of the Patient  for
the Booth: ");
```

```
                            firstName = input2.nextLine();
                        } while (firstName.isEmpty());

                        do {
                            System.out.println("Enter the Surname Name of the Patient:
");
                            lastName = input2.nextLine();
                        } while (lastName.isEmpty());

                        System.out.println("Enter the age of the Patient: ");
                        age = input.nextInt();

                        do {
                            System.out.println("Enter the City of the Patient: ");
                            city = input2.nextLine();
                        } while (city.isEmpty());

                        System.out.println("Enter the NIC or Passport Number of the
Patient: ");
                        id = input.nextLong();
                        System.out.println("Enter the number of Requested vaccination ");
                        System.out.println("AstraZeneca - 1");
                        System.out.println("Sinopharm   - 2");
                        System.out.println("Pfizer      - 3");
                        vaccsReq = input.nextInt();
                        //Following Process will Add the patient to a booth right away or
wil be added to a waiting List.
                        for (int x = 0; x < centerArr.length; x++) {

                            if (centerArr[x].getVaccsType() == vaccsReq) {
                                if (centerArr[x].getName().equals("Empty")) {
                                    centerArr[x].setName(firstName);
                                    System.out.println("Patient has been directed to the
proper booth");

                                    patientsArr[x].setValues(firstName, lastName, age,
city, id, vaccsReq);

                                    stock.setStock();
                                    sortNameCount++;
                                    break;
                                } else if (centerArr[x + 1].getName().equals("Empty") ) {
                                    centerArr[x + 1].setName(firstName);
                                    System.out.println("Patient will be directed to the
proper booth");

                                    patientsArr[x + 1].setValues(firstName, lastName,
age, city, id, vaccsReq);

                                    stock.setStock();
                                    sortNameCount++;
                                    break;
                                } else {
                                    System.out.println("Requested Vaccination type Booths
are Occupied Patient will be added to a waiting list");
                                    //Adding the Patient to a waiting List Till an
opening comes in the requested vaccination type booths
                                    waitingListArr.add(new Booth(firstName,vaccsReq));
                                    //Adding the Addition information about the Patient
to a temporarily list till he finds a booth
                                    //After the Patient finds  proper booth this
Information will be available in the System.
                                    waitingPatientList.add(new
Patient(firstName,lastName,age,city,id,vaccsReq));
                                    break;
```

```
                                }

                            }

                        }
                        System.out.println(" ");

                        break;

                    case "rpb":
                    case "103":
                        System.out.println("Enter a booth number to Remove a Patient (0-
5)");
                        boothNum = input.nextInt();
                        if (centerArr[boothNum].getName().equals("Empty")) {
                            System.out.println("This Booth is already Empty");
                        } else {
                            //This process will remove a patient from the chosen booth
and if the empty booth vaccination
                            // brand is matching to a patient that waits in the waiting
list he will be added to the booth.
                            centerArr[boothNum].setName("Empty");
                            patientsArr[boothNum].setFirstName("Empty");
                            sortNameCount--;
                            System.out.println("Patient will be removed from the booth");
                            for (int x = 0; x < waitingListArr.size(); x++){
                                //Patients in the waiting will be only add to the booth
if the requested vaccinations booths are empty
                                if (centerArr[boothNum].getVaccsType() ==
waitingListArr.get(x).getVaccsType()){
                                    sortNameCount++;
                                    stock.setStock();
                                    System.out.println("New Patient will be add to the
booth");

centerArr[boothNum].setName(waitingListArr.get(x).getName());
                                    //Setting Additional information about the New
patient to the System

patientsArr[boothNum].setFirstName(waitingPatientList.get(x).getFirstName());

patientsArr[boothNum].setLasttName(waitingPatientList.get(x).getLasttName());

patientsArr[boothNum].setAge(waitingPatientList.get(x).getAge());

patientsArr[boothNum].setCity(waitingPatientList.get(x).getCity());

patientsArr[boothNum].setId(waitingPatientList.get(x).getId());

patientsArr[boothNum].setVaccsType(waitingPatientList.get(x).getVaccsType());
                                    //Removing the Patient From the waiting List
                                    waitingListArr.remove(x);
                                    waitingPatientList.remove(x);
                                    break;

                                }

                            }


                        }
```

```
                        System.out.println(" ");
                        break;

                    case "vps":
                    case "104":
                        sortName = new String[sortNameCount];
                        int y = 0;
                        //Creating an new list for the patient that currently getting
vaccinated
                        for (Booth arr : centerArr) {
                            if (!arr.getName().equals("Empty")) {
                                sortName[y] = arr.getName();
                                y++;
                            }
                        }
                        //Sorting list of names
                        for (int n = 0; n < sortName.length; n++) {
                            for (int x = n + 1; x < sortName.length; x++) {
                                String temp = sortName[x];

                                int num = sortName[n].compareToIgnoreCase(sortName[x]);
                                if (num > 0) {
                                    sortName[x] = sortName[n];
                                    sortName[n] = temp;
                                }
                            }
                        }
                        //Displaying the Sorted names list
                        for (String sort : sortName) {
                            System.out.println(sort);
                        }
                        System.out.println(" ");
                        break;

                    case "spd":
                    case "105":


                        try{
                            FileWriter data = new FileWriter("Task4.txt");


                            for (int i=0; i < centerArr.length; i++){
                                if (!centerArr[i].getName().equals("Empty")) {
                                    data.write("----------------Booth Number " + i + "---
------------" +"\n");
                                    data.write("First name      : " +
patientsArr[i].getFirstName() + "\n");
                                    data.write("Surname         : " +
patientsArr[i].getLasttName() + "\n");
                                    data.write("Age             : " +
patientsArr[i].getAge() + "\n");
                                    data.write("City            : " +
patientsArr[i].getCity() + "\n");
                                    data.write("Nic or Passport : " +
patientsArr[i].getId() + "\n");
                                    data.write(  "Vaccination Type : " +
patientsArr[i].getVaccsType()  + "\n");
                                    data.write("\n ");
                                }
                                else{
```

```
                                    data.write("--------Booth Number " + i + "--------"
+"\n");
                                    data.write("This Booth is empty" + "\n");
                                    data.write("\n ");
                                }
                            }
                            data.write("Remaining vaccine amount is: " + stock.getStock()
+ "\n\n");
                            data.close();
                            System.out.println("Successfully wrote to the file.");
                            System.out.println(" ");
                        } catch (IOException e){
                            System.out.println("An error occurred.");
                            System.out.println(" ");
                            e.printStackTrace();
                        }

                        break;
                    case "lpd":
                    case "106":


                        try {
                            File load = new File("Task4.txt");
                            if (load.exists()) {
                                BufferedReader br = new BufferedReader(new
FileReader(load));

                                String st;
                                while ((st = br.readLine()) != null) {
                                    System.out.println(st);
                                    System.out.print("");
                                }

                            } else {
                                System.out.println("The file does not exist.");
                            }

                        } catch (IOException e){
                            System.out.println("An error occurred.");
                            System.out.println(" ");
                            e.printStackTrace();
                        }


                        break;

                    case "vrv":
                    case "107":
                        //Assigning the remaining stock to the Variable "left"
                        int left = stock.getStock();
                        System.out.println("Remaining vaccine amount is: " + left);
                        System.out.println(" ");
                        break;


                    case "avs":
                    case "108":
                        //Getting Amount from the User to Add the Vaccination stock
                        System.out.println("Enter the amount of vaccinations adding to
the Stock: ");
```

```
                        int amount = input.nextInt();
                        stock.setStock(amount);
                        break;


                    case "ext":
                    case "999":
                        //Ending the program by Breaking the Loop
                        loop = true;
                        break;

                    default:
                        System.out.println("Please enter a valid option.");
                        System.out.println(" ");


                }
                //Displaying a warning message when the stock reaches 20
                if (stock.getStock() == 20) {
                    System.out.println("Warning only 20 vaccines are Remaining.");
                    System.out.println(" ");
                }
            }catch (Exception e) {
                System.out.println("Please enter a valid input.... ");
            }
        }



    }
}
```

Task 5 – JavaFx

- Main.java

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("ScreenA.fxml"));
        primaryStage.setTitle("GUI");
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
    }


    public static void main(String[] args) {
        launch(args);
    }
}
```

- ControllerA.java

```
package sample;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.stage.Stage;


public class ControllerA {

    @FXML
    private TextField txtFirst, txtLast, txtAge, txtCity, txtNic;

    @FXML
    private RadioButton r1, r2, r3;

    @FXML
    private RadioButton s0, s1, s2, s3, s4, s5;
```

```
    @FXML
    private Label lblVali;

    @FXML
    private CheckBox b0, b1, b2, b3, b4, b5;

    //This Method will give the selected Vaccination type
    public String radio(){
        String r ="";
        if (r1.isSelected()){
            r += r1.getText();
        }
        else if (r2.isSelected()){
            r += r2.getText();
        }
        else if (r3.isSelected()){
            r += r3.getText();
        }
        else {
            r = "Vaccination has not been Selected";
        }
        return r;

    }
    //This method will give the Selected Booth Number
    public String SelectBooth(){
        String select = "";
        if (s0.isSelected()){
            select += s0.getText();
        }
        else if (s1.isSelected()){
            select += s1.getText();
        }
        else if (s2.isSelected()){
            select += s2.getText();
        }
        else if (s3.isSelected()){
            select += s3.getText();
        }
        else if (s4.isSelected()){
            select += s4.getText();
        }
        else if (s5.isSelected()){
            select += s5.getText();
        }
        else {
            select = "Booth has not been Selected";
        }
        return select;

    }




    @FXML
    void change(ActionEvent event) {
        //Validating the Text Fields and Radio buttons
        if (txtFirst.getLength() == 0){
            lblVali.setText("Please Enter your Name ");
        }
```

```
        else if (txtLast.getLength() == 0){
            lblVali.setText("Please Enter your Surname ");
        }
        else if (txtAge.getLength() == 0){
            lblVali.setText("Please Enter your Age ");
        }
        else if (txtCity.getLength() == 0){
            lblVali.setText("Please Enter your City ");
        }
        else if (txtNic.getLength() == 0){
            lblVali.setText("Please Enter your NIC or Passport ");
        }
        else if (radio().equals("Vaccination has not been Selected")){
            lblVali.setText("Vaccination type has not been Selected");
        }
        else if (SelectBooth().equals("Booth has not been Selected")){
            lblVali.setText("Booth number has not been Selected");
        }
        else {


            try {
                FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("ScreenB.fxml"));
                Parent root1 = (Parent) fxmlLoader.load();

                ControllerB scene = fxmlLoader.getController();
                scene.name(txtFirst.getText(), txtLast.getText(), txtAge.getText(),
txtCity.getText(), txtNic.getText(), radio(), SelectBooth());
                lblVali.setText("");

                Stage stage = new Stage();

                stage.setTitle("Receipt");
                stage.setScene(new Scene(root1));
                stage.show();

            } catch (Exception e) {
                e.printStackTrace();
            }

        }
    }
}
```

- ControllerB.java

```
package sample;

import javafx.fxml.FXML;
import javafx.scene.control.Label;

import java.text.SimpleDateFormat;
import java.util.Date;

public class ControllerB {

    @FXML
    private Label label, lblAge, lblCity, lblNic, lblVaccs, lblBooth, lblDate;
    //Constructor to change the Label Text
    public void name (String a, String b, String c, String d, String e, String f, String
g){
        label.setText(a +" " + b);
        lblAge.setText(c);
        lblCity.setText(d);
        lblNic.setText(e);
        lblVaccs.setText(f);
        lblBooth.setText(g);

        Date date = new Date();
        SimpleDateFormat formatter = new SimpleDateFormat("dd-MM-yyyy     HH:mm:ss");
        lblDate.setText(formatter.format(date));
    }



}
```

- ScreenA.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.text.Font?>

<GridPane alignment="center" hgap="10" vgap="10" xmlns="http://javafx.com/javafx/16"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.ControllerA">
```

```
    <columnConstraints>
        <ColumnConstraints />
    </columnConstraints>
    <rowConstraints>
        <RowConstraints />
    </rowConstraints>
    <children>
        <AnchorPane prefHeight="379.0" prefWidth="682.0">
            <children>
                <TextField fx:id="txtFirst" layoutX="277.0" layoutY="64.0" />
                <Button fx:id="button" layoutX="287.0" layoutY="339.0"
mnemonicParsing="false" onAction="#change" text="Generate Receipt" />
                <TextField fx:id="txtAge" layoutX="277.0" layoutY="116.0" />
                <TextField fx:id="txtLast" layoutX="277.0" layoutY="90.0" />
                <TextField fx:id="txtCity" layoutX="277.0" layoutY="142.0" />
                <TextField fx:id="txtNic" layoutX="277.0" layoutY="167.0" />
                <Label layoutX="31.0" layoutY="68.0" prefHeight="18.0" prefWidth="138.0"
text="First Name                        " />
                <Label layoutX="31.0" layoutY="94.0" prefHeight="18.0" prefWidth="138.0"
text="Last Name                        " />
                <Label layoutX="31.0" layoutY="120.0" prefHeight="18.0" prefWidth="138.0"
text="Age                                " />
                <Label layoutX="31.0" layoutY="146.0" prefHeight="18.0" prefWidth="137.0"
text="City" />
                <Label layoutX="31.0" layoutY="171.0" prefHeight="18.0" prefWidth="150.0"
text="NIC or Passport Number" />
                <Label layoutX="31.0" layoutY="207.0" text="Select the Vaccination Type"
/>
                <Label layoutX="31.0" layoutY="255.0" text="Select the Booth" />
                <RadioButton fx:id="r1" layoutX="196.0" layoutY="207.0"
mnemonicParsing="false" text="AstraZeneca">
                    <toggleGroup>
                        <ToggleGroup fx:id="v" />
                    </toggleGroup>
                </RadioButton>
                <RadioButton fx:id="r2" layoutX="324.0" layoutY="207.0"
mnemonicParsing="false" text="Sinopharm" toggleGroup="$v" />
                <RadioButton fx:id="r3" layoutX="455.0" layoutY="207.0"
mnemonicParsing="false" text="Pfizer" toggleGroup="$v" />
                <RadioButton fx:id="s0" layoutX="46.0" layoutY="282.0"
mnemonicParsing="false" text="0">
                    <toggleGroup>
                        <ToggleGroup fx:id="booth" />
                    </toggleGroup>
                </RadioButton>
                <RadioButton fx:id="s1" layoutX="167.0" layoutY="282.0"
mnemonicParsing="false" text="1" toggleGroup="$booth" />
                <RadioButton fx:id="s2" layoutX="262.0" layoutY="282.0"
mnemonicParsing="false" text="2" toggleGroup="$booth" />
                <RadioButton fx:id="s3" layoutX="364.0" layoutY="282.0"
mnemonicParsing="false" text="3" toggleGroup="$booth" />
                <RadioButton fx:id="s4" layoutX="466.0" layoutY="282.0"
mnemonicParsing="false" text="4" toggleGroup="$booth" />
                <RadioButton fx:id="s5" layoutX="568.0" layoutY="282.0"
mnemonicParsing="false" text="5" toggleGroup="$booth" />
                <Label layoutX="192.0" text="Enter Patient Details">
                    <font>
                        <Font size="36.0" />
                    </font>
                </Label>
            <Label fx:id="lblVali" alignment="CENTER" layoutX="211.0" layoutY="312.0"
```

```
prefHeight="19.0" prefWidth="253.0" textFill="RED">
               <font>
                  <Font size="13.0" />
               </font>
            </Label>
            </children>
        </AnchorPane>
    </children>
</GridPane>
```

- ScreenB.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.text.Font?>

<Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="539.0" prefWidth="457.0" xmlns="http://javafx.com/javafx/16"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.ControllerB">

    <Label fx:id="label" layoutX="215.0" layoutY="104.0" text="Label" />
    <Label fx:id="lblAge" layoutX="215.0" layoutY="172.0" text="Label" />
    <Label layoutX="64.0" layoutY="104.0" text="Name          " />
    <Label layoutX="65.0" layoutY="172.0" text="Age             " />
    <Label layoutX="64.0" layoutY="235.0" text="City                  " />
    <Label fx:id="lblCity" layoutX="215.0" layoutY="235.0" text="Label" />
    <Label layoutX="63.0" layoutY="296.0" text="NIC or Passport Number " />
    <Label fx:id="lblNic" layoutX="215.0" layoutY="296.0" text="Label" />
    <Label layoutX="67.0" layoutY="365.0" text="Vaccinatioin Type " />
    <Label fx:id="lblVaccs" layoutX="215.0" layoutY="365.0" text="Label" />
    <Label fx:id="lblBooth" layoutX="215.0" layoutY="430.0" text="Label" />
    <Label layoutX="67.0" layoutY="430.0" text="Booth Number" />
    <Label fx:id="lblDate" layoutX="317.0" layoutY="468.0" text="Label" />
    <Label layoutX="169.0" layoutY="20.0" prefHeight="47.0" prefWidth="135.0"
text="Receipt">
        <font>
            <Font size="32.0" />
        </font>
    </Label>
    <Label layoutX="188.0" layoutY="104.0" text="    :" />
    <Label layoutX="188.0" layoutY="172.0" text="    :" />
    <Label layoutX="188.0" layoutY="235.0" text="    :" />
    <Label layoutX="188.0" layoutY="296.0" text="    :" />
    <Label layoutX="188.0" layoutY="365.0" text="    :" />
    <Label layoutX="188.0" layoutY="430.0" text="    :" />

</Pane>
```