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	: Iresh Bandara
Date of submission	: 26.07.2021
Student ID	: 20200696/ w1833520
Student First Name	: Kaliyugavarathan
Student Surname	: Sean Jesuharun

"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 : Kaliyugavarathan Sean Jesuharun

Student ID : 20200696

Test Cases

	Test Case [Task 1 & Task 2]	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	(View all empty Booths) After program starts, 101 or VEB	Display only the "empty" booths.	Display only the "empty" booths.	Pass
3	(Add patient to a Booth) Enter 102 or APB	Prompt for the Booth Number. Add patient to the relevant Booth Number	Prompt for the Booth Number. Add patient to the relevant Booth Number	pass
4	(Remove patient from a Booth) Enter 103 or RPB	Prompt for the booth Number from where to remove patient. Remove patient from the specified Booth number.	Prompt for the booth Number from where to remove patient. Remove patient from the specified Booth number.	Pass
5	(View Patient Names sorted in Alphabetical Order) Enter 104 or VPS	Showing Patients Names sorted in Alphabetical order.	Showing Patients Names sorted in Alphabetical order.	Pass
6	(Store program data into File) Enter 105 or SPD	[Task1] Storing the names of the Patient in every line of PatientFileTask1.txt [Task 2] Storing the names of the Patient in every line of PatientFileTask2.txt	[Task1] Storing the names of the Patient in every line of PatientFileTask1.txt [Task 2] Storing the names of the Patient in every line of PatientFileTask2.txt	Pass

7	(Load program data from a File) Enter 106 or LPD	[Task1] Load the data From PatientFileTask1.txt and arrange the booths according to the stored data. [Task2] Load the data From PatientFileTask2.txt and arrange the booths according to the stored data.	[Task1] Load the data From PatientFileTask1.txt and arrange the booths according to the stored data. [Task2] Load the data From PatientFileTask2.txt and arrange the booths according to the stored data.	Pass
8	(View Remaining Vaccinations) Enter 107 or VRV	Display the remaining vaccination according to the no of patients added. (1 Vaccine for 1 person).	Display the remaining vaccination according to the no of patients added. (1 Vaccine for 1 person).	Pass
9	(Add vaccinations to the stocks) Enter 108 or AVS	Prompt the no of vaccinations. Add vaccinations to the stocks and update the Stocks.	Prompt the no of vaccinations. Add vaccinations to the stocks and update the Stocks.	Pass
10	(Exit the whole Program) Enter 999 or EXT	Display Program is Existing. And Exit from the whole program.	Display Program is Existing. And Exit from the whole program.	Pass

Test Case [Task 3 (Arrays Version)]	Expected Result	Actual Result	Pass/Fail
(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'empty' for all booths	Displays 'empty' for all booths	Pass
(View all empty Booths) After program starts, 101 or VEB	Display only the "empty" booths.	Display only the "empty" booths.	Pass
(Add patient to a Booth) Enter 102 or APB	Prompt for the First Name. Prompt for Sur Name. Prompt for the Vaccination Type. Placing the patient in a booth according to his request of Vaccination Type.	Prompt for the First Name. Prompt for Sur Name. Prompt for the Vaccination Type Placing the patient in a booth according to his request of Vaccination Type.	pass
(Remove patient from a Booth) Enter 103 or RPB	Prompt for the booth Number from where to remove patient. Remove patient from the specified Booth number.	Prompt for the booth Number from where to remove patient. Remove patient from the specified Booth number.	Pass
(View Patient Names sorted in Alphabetical Order) Enter 104 or VPS	Showing Patients Names sorted in Alphabetical order.	Showing Patients Names sorted in Alphabetical order.	Pass
(Store program data into File) Enter 105 or SPD	Storing the First name, Sur Name and the Vaccination Type of a Patient in single line of patientFileTask1Extended.txt And storing the details of other patient in next Line	Storing the First name, Sur Name and the Vaccination Type of a Patient in single line of patientFileTask1Extended.txt And storing the details of other patient in next Line	Pass
(Load program data from a File) Enter 106 or LPD	Load the data from patientFileTask1Extended.txt and arrange the booths according to the stored data.	Load the data from patientFileTask1Extended.txt and arrange the booths according to the stored data.	Pass

(View Remaining Vaccinations) Enter 107 or VRV	Display the remaining vaccination according to the no of patients added. (1 Vaccine for 1 person).	Display the remaining vaccination according to the no of patients added. (1 Vaccine for 1 person).	Pass
(Add vaccinations to the stocks) Enter 108 or AVS	Prompt the no of vaccinations. Add vaccinations to the stocks and update the Stocks.	Prompt the no of vaccinations. Add vaccinations to the stocks and update the Stocks.	Pass
(Exit the whole Program) Enter 999 or EXT	Display Program is Existing. And Exit from the whole program.	Display Program is Existing. And Exit from the whole program.	Pass

Test Case [Task 3 (Classes Version Extended)]	**Expected Result**	**Actual Result**	**Pass/Fail**
(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'empty' for all booths	Displays 'empty' for all booths	Pass
(View all empty Booths) After program starts, 101 or VEB	Display only the "empty" booths.	Display only the "empty" booths.	Pass
(Add patient to a Booth) Enter 102 or APB	Prompt for the First Name. Prompt for Sur Name. Prompt for the Age. Prompt for the City. Prompt for the NIC or Passport number. Prompt for the Vaccination Type. Placing the patient in a booth according to his request of Vaccination Type.	Prompt for the First Name. Prompt for Sur Name. Prompt for the Age. Prompt for the City. Prompt for the NIC or Passport number. Prompt for the Vaccination Type. Placing the patient in a booth according to his request of Vaccination Type.	pass

(Remove patient from a Booth) Enter 103 or RPB	Prompt for the booth Number from where to remove patient. Remove patient from the specified Booth number.	Prompt for the booth Number from where to remove patient. Remove patient from the specified Booth number.	Pass
(View Patient Names sorted in Alphabetical Order) Enter 104 or VPS	Showing Patients Names sorted in Alphabetical order.	Showing Patients Names sorted in Alphabetical order.	Pass
(Store program data into File) Enter 105 or SPD	Storing the First name, Sur Name, Age, City, NIC or Passport Number and the Vaccination Type of a Patient in single line of patientFileTask2Extended.txt And storing the details of other patient in next Line	Storing the First name, Sur Name, Age, City, NIC or Passport Number and the Vaccination Type of a Patient in single line of patientFileTask2Extended.txt And storing the details of other patient in next Line	Pass
(Load program data from a File) Enter 106 or LPD	Load the data from patientFileTask2Extended.txt and arrange the booths according to the stored data.	Load the data from patientFileTask2Extended.txt and arrange the booths according to the stored data.	Pass
(View Remaining Vaccinations) Enter 107 or VRV	Display the remaining vaccination according to the no of patients added. (1 Vaccine for 1 person).	Display the remaining vaccination according to the no of patients added. (1 Vaccine for 1 person).	Pass
(Add vaccinations to the stocks) Enter 108 or AVS	Prompt the no of vaccinations. Add vaccinations to the stocks and update the Stocks.	Prompt the no of vaccinations. Add vaccinations to the stocks and update the Stocks.	Pass
(Exit the whole Program) Enter 999 or EXT	Display Program is Existing. And Exit from the whole program.	Display Program is Existing. And Exit from the whole program.	Pass

Test Case [Task 4 (Linked List Version)]	Expected Result	Actual Result	Pass/Fail
(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'empty' for all booths	Displays 'empty' for all booths	Pass
(View all empty Booths) After program starts, 101 or VEB	Display only the "empty" booths.	Display only the "empty" booths.	Pass
(Add patient to a Booth) Enter 102 or APB	Prompt for the First Name. Prompt for Sur Name. Prompt for the Age. Prompt for the City. Prompt for the NIC or Passport number. Prompt for the Vaccination Type. Placing the patient in a booth according to his request of Vaccination Type. If the Booths related to the vaccination type that the patient mentioned above is being filled that patient will added to the waiting list.	Prompt for the First Name. Prompt for Sur Name. Prompt for the Age. Prompt for the City. Prompt for the NIC or Passport number. Prompt for the Vaccination Type. Placing the patient in a booth according to his request of Vaccination Type. If the Booths related to the vaccination type that the patient mentioned above is being filled that patient will added to the waiting list.	pass
(Remove patient from a Booth) Enter 103 or RPB	Prompt for the booth Number from where to remove patient. Remove patient from the specified Booth number. And check whether there are patients available in the waiting list to get the vaccination that provided by the specified booth which removed the patient. And if	Prompt for the booth Number from where to remove patient. Remove patient from the specified Booth number. And check whether there are patients available in the waiting list to get the vaccination that provided by the specified booth which removed the patient. And if	Pass

	yes adding that patient automatically to that Booth.	yes adding that patient automatically to that Booth.	
(View Patient Names sorted in Alphabetical Order) Enter 104 or VPS	Showing Patients Names sorted in Alphabetical order.	Showing Patients Names sorted in Alphabetical order.	Pass
(Store program data into File) Enter 105 or SPD	Storing the First name, Sur Name, Age, City, NIC or Passport Number and the Vaccination Type of a Patient in single line of patientFileTask4.txt And storing the details of other patient in next Line	Storing the First name, Sur Name, Age, City, NIC or Passport Number and the Vaccination Type of a Patient in single line of patientFileTask4.txt And storing the details of other patient in next Line	Pass
(Load program data from a File) Enter 106 or LPD	Load the data from patientFileTask4.txt and arrange the booths according to the stored data.	Load the data from patientFileTask4.txt and arrange the booths according to the stored data.	Pass
(View Remaining Vaccinations) Enter 107 or VRV	Display the remaining vaccination according to the no of patients added. (1 Vaccine for 1 person).	Display the remaining vaccination according to the no of patients added. (1 Vaccine for 1 person).	Pass
(Add vaccinations to the stocks) Enter 108 or AVS	Prompt the no of vaccinations. Add vaccinations to the stocks and update the Stocks.	Prompt the no of vaccinations. Add vaccinations to the stocks and update the Stocks.	Pass
(Exit the whole Program) Enter 999 or EXT	Display Program is Existing. And Exit from the whole program.	Display Program is Existing. And Exit from the whole program.	Pass

Test Case [Task 5]	Expected Result	Actual Result	Pass/Fail
"Generate Receipt" Button Click Generate Receipt Button	Creating a new Stage "Receipts". Closing the previous stage.	Creating a new Stage "Receipts". Closing the previous stage.	Pass
Showing Details in Receipt	Showing Patient Name, Vaccination Type, Booth Number and Date and time Stamp.	Showing Patient Name, Vaccination Type, Booth Number and Date and time Stamp.	Pass

Discussion

Task 1

I have implemented test cases for all the operations(methods) so it ensures that all my test cases cover all the aspects of Task 1.

Task 2

I have implemented test cases for all the operations(methods) so it ensures that all my test cases cover all the aspects of Task 2.

Task 3 [Arrays Version]

I have implemented test cases for all the operations(methods). Especially when adding a patient if the relevant vaccination booth is filled at that time, then it will display a message like

“The Booths that allocated to provide AstraZeneca/ Sinopharm/ Pfizer vaccine are currently busy...”

so, it ensures that all my test cases cover all the aspects of Task 3[Arrays Version].

Task 3 [Classes Version]

I have implemented test cases for all the operations(methods). Especially when adding a patient if the relevant vaccination booth is filled at that time, then it will display a message like

"The Booths that allocated to provide AstraZeneca/ Sinopharm/ Pfizer vaccine are currently busy..."

"Do you prefer to go for other vaccination? if yes press " y " else press " n “: "

“if ‘y’ then it will change the Vaccination Request and place that patient in a booth once if it’s being empty.”

“if ‘n’ then it will Display a message like ‘Sorry...You have to wait until the booths are getting empty and register yourself again’.”

so, it ensures that all my test cases cover all the aspects of Task 3[Classes Version].

Task 4 [Linked List Version]

I have implemented test cases for all the operations(methods). Especially when adding a patient if the relevant vaccination booth is filled by that time. Then that patient will be added to waiting list. Once the booth becomes empty then the patient from the waiting list will automatically added to that booth according to his vaccination request. Like if we are removing patient from booth 0 then if there are patients willing to get “AstraZenaca” vaccine in the waiting list then only the booth 0 will be filled by that particular patient. so, it ensures that all my test cases cover all the aspects of Task 4[Linked List Version].

Task 5 [JavaFX]

I have implemented test cases for “Clicking Generate Button” and “Showing Details in Receipt” so it ensures that all my test cases cover all the aspects of Task 5.

Code :

Task 01[Arrays Version]

```
import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.FileNotFoundException;

import java.util.Scanner;


public class Arrays_Version {


    //Making the array Static so Each Static methods can access it.....

    static String[] serviceCenterBooths = new String[6];

    static int stocks = 150;

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


    public static void main(String[] args){


        // Initialising the ServiceCenterbooths as "e" This means that every booth is empty initially.....
```

```
initialise();


//Menu Options
System.out.println
    (
        "100 0r 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"
    );
while (true) {


   System.out.println();
   System.out.print("Select the task from the menu : ");
   String operation = myScanner.next();
```

```
//operations begins here
if (operation.equals("100") || operation.equals("VVB"))
{
   viewAllVaccinationBooths();
}
else if (operation.equals("101") || operation.equals("VEB"))
{
   viewAllEmptyBooths();
}
else if (operation.equals("102") || operation.equals("APB"))
{
   addPatientToABooth();
}
else if (operation.equals("103") || operation.equals("RPB"))
{
   removePatientFromABooth();
}
else if (operation.equals("104") || operation.equals("VPS"))
{
   viewPatientsSortedInAlphabeticalOrder();
```

```
}
else if (operation.equals("105") || operation.equals("SPD"))
{
    storeProgramDataIntoFile();
}
else if (operation.equals("106") || operation.equals("LPD"))
{
    loadProgramDataFromFile();
}
else if (operation.equals("107") || operation.equals("VRV"))
{
    viewRemainingVaccinations();
}
else if (operation.equals("108") || operation.equals("AVS"))
{
    addVaccinationsToTheStock();
}
else if (operation.equals("999") || operation.equals("EXT"))
{
    System.out.println("The program is Exiting.....");
    break;
```

```
        }
        else{
            System.out.println("Enter the correct value for the TASK...");
        }
    }
}

private static void initialise(){
    // Initialising the ServiceCenterbooths as "e" This means that every booth is empty initially.....
    for (int i = 0; i < serviceCenterBooths.length; i++)
    {
        serviceCenterBooths[i] = "e";
    }
}

private static void viewAllVaccinationBooths(){

    for (int i = 0; i < serviceCenterBooths.length; i++)
    {
        if (!serviceCenterBooths[i].equals("e"))
        {
```

```
            System.out.println("Booth " + i + " is occupied by " + serviceCenterBooths[i]);


        }
        else{
            System.out.println("Booth " + i + " is empty.");
        }
    }

}

private static void viewAllEmptyBooths(){
    //Making a boolean called "flag" to check whether atleast one booth is being empty...
    boolean flag = true;
    for (int i = 0; i < serviceCenterBooths.length; i++)
    {
        if (serviceCenterBooths[i].equals("e"))
        {
            System.out.println("Booth " + i + " is Empty.");
            flag = false;
        }
    }
```

```
    if (flag)
    {
        System.out.println("Currently All Booths are Occupied.");
    }
}

private static void addPatientToABooth(){

    while (true) {

        //Making a boolean called "flag" to check whether atleast one booth is being empty to add a patient...
        boolean flag = true;
        for (int i = 0; i < serviceCenterBooths.length; i++)
        {
            if (serviceCenterBooths[i].equals("e"))
            {
                flag = false;
            }
        }
        if (flag)
        {
```

```
            System.out.println("Can't add patients because all the 6 Booths have been occupied.");
            break;
        }

        //Giving operator an idea about where he can add patients by showing him the currently empty booths...
        System.out.println();
        System.out.println("Currently empty booth are: ");
        viewAllEmptyBooths();
        System.out.println();

        System.out.print("In which booth you want to add the patient [Enter a Booth Number from 0 to 5] : ");
        int boothnum = myScanner.nextInt();

        if (serviceCenterBooths[boothnum].equals("e")) {

            System.out.print("What is the name of the patient : ");
            String patientname = myScanner.next();

            serviceCenterBooths[boothnum] = patientname;
```

```
            System.out.println("\"" + patientname + "\"" + " has been successfully added to booth " + boothnum);

            stocks--;

            if (stocks == 20)

            {

               System.out.println();

               System.out.println("WARNING !!! ..... There are only " + stocks + " vaccinations remaining..");

            }

            break;

         } else {

            System.out.println("You can't add a patient here because it's already occupied by a patient name called " + "\"" + serviceCenterBooths[boothnum] + "\"");

         }

      }


   }


   private static void removePatientFromABooth(){


      while(true) {
```

```
//Making a boolean called "flag" to check whether atleast one booth is not being empty...
boolean flag = true;
for (int i = 0; i < serviceCenterBooths.length; i++)
{
   if (!serviceCenterBooths[i].equals("e"))
   {
      flag = false;
   }
}
if (flag)
{
   System.out.println("Can't remove patients because all the 6 Booths are currently being Empty.");
   break;
}

//Giving operator an idea about from where he can remove patients by showing him the currently occupied booths...
System.out.println();
System.out.println("Currently Occupied booths are: ");
for (int i = 0; i < serviceCenterBooths.length; i++) {
   if (!serviceCenterBooths[i].equals("e")) {
```

```
                System.out.println("Booth " + i + " is Occupied by " + "\"" + serviceCenterBooths[i] + "\"");
            }
        }

        System.out.println();
        System.out.print("From which Booth you want to remove the patient [Enter a Booth Number from 0 to 5] : ");
        int boothnum = myScanner.nextInt();

        if (!serviceCenterBooths[boothnum].equals("e")) {
            String patientName = serviceCenterBooths[boothnum];
            serviceCenterBooths[boothnum] = "e";
            System.out.println("\"" + patientName + "\"" + " has been successfully removed from booth " + boothnum);

            break;
        } else {
            System.out.println("The Booth is already being Empty...");
        }
    }
}
```

```
    private static void viewPatientsSortedInAlphabeticalOrder(){


        //Making a boolean called "flag" to check whether atleast one booth is not being empty...
        while (true) {
            boolean flag = true;
            for (int i = 0; i < serviceCenterBooths.length; i++) {
                if (!serviceCenterBooths[i].equals("e")) {
                    flag = false;
                }
            }
            if (flag) {
                System.out.println("Can't sorted patients names because all the 6 Booths are Empty.");
                break;
            }


            //Copying elements from serviceCenterBooths Array to sortedNames array because changes won't affect serviceCenterBooth array...
            String[] sortedNames = new String[6];
            for (int i = 0; i < sortedNames.length; i++)
            {
                sortedNames[i] = serviceCenterBooths[i];
```

```
}

//To hold Temporary Values....
String temporaryString;
for (int i = 0; i < sortedNames.length; i++) {
   for (int j = i + 1; j < sortedNames.length; j++)
   {
      if (!sortedNames[i].equals("e") && !sortedNames[j].equals("e"))
      {
         if (sortedNames[i].compareToIgnoreCase(sortedNames[j]) > 0) {
            temporaryString = sortedNames[i];
            sortedNames[i] = sortedNames[j];
            sortedNames[j] = temporaryString;
         }
      }
   }
}

System.out.println("And The Names Sorted in Alphabetical Order Are As Follows...");

for (int i = 0; i < sortedNames.length; i++)
```

```
        {
            if(!sortedNames[i].equals("e"))
            {
                System.out.println(sortedNames[i]);
            }
        }
        break;
    }
}

private static void storeProgramDataIntoFile(){

    try {
        File patientFileTask1 = new File("PatientFileTask1.txt");
        patientFileTask1.createNewFile();

        FileWriter myWriter = new FileWriter("PatientFileTask1.txt");
        for (int i = 0; i < serviceCenterBooths.length; i++)
        {
            myWriter.write(serviceCenterBooths[i]+ "\n");
        }
```

```
            myWriter.close();


            System.out.println("Patient names have been Successfully Stored in \" PatientFileTask1 \" TextFile..");


        }
        catch (IOException e){
            System.out.println("An Error has occurred");
        }


    }


    private static void loadProgramDataFromFile(){


        try{
            File PatientFile2 = new File("PatientFileTask1.txt");
            Scanner myReader = new Scanner(PatientFile2);


            while (myReader.hasNextLine())
            {
                for (int i = 0; i < serviceCenterBooths.length; i++)
```

```
            {
                String data = myReader.nextLine();
                serviceCenterBooths[i] = data;
            }
        }

        System.out.println("Patient names have been Successfully loaded from the \" PatientFileTask1 \" TextFile");
    }
    catch(FileNotFoundException e){
        System.out.println("File not found..");
    }
}

private static void viewRemainingVaccinations(){
    System.out.println("The remaining vaccinations are : " + stocks);
}

private static void addVaccinationsToTheStock(){

    System.out.print("How many vaccinations that you gonna add to the Stocks: ");
```

```
        int vaccinationnumber = myScanner.nextInt();


        stocks = stocks + vaccinationnumber;
        System.out.println(vaccinationnumber + " vaccinations has been successfully added to stocks");
    }


}
```

Task 02[Classes Version]

Booth Class

```
public class Booth {


    private int boothNum;
    private String patientName;


    //Overloaded Constructor....
```

```
public Booth(int boothNum, String patientName){
    this.boothNum = boothNum;
    this.patientName = patientName;
}

public int getBoothNum() {
    return boothNum;
}

public void setBoothNum(int boothNum) {
    this.boothNum = boothNum;
}

public String getPatientName() {
    return patientName;
}

public void setPatientName(String patientName) {
    this.patientName = patientName;
}
```

```
    public void addPatientToABooth(String patientName){
        this.patientName = patientName;
        System.out.println("\"" + patientName + "\"" + " has been successfully added to booth " + this.boothNum);
    }

    public void removePatientFromABooth(String patientName){
        this.patientName = "e";
        System.out.println("\"" + patientName + "\"" + " has been successfully removed from booth " + this.boothNum);
    }
}
```

VaccinationCenter Class

```
import java.io.File;
import java.io.IOException;
import java.io.FileWriter;
import java.io.FileNotFoundException;
import java.util.Scanner;
```

```
public class VaccinationCenter {

    static Booth[] boothObjectsArray = new Booth[6];
    static int stocks = 150;
    static Scanner myScanner = new Scanner(System.in);

    public static void main(String[] args){

        //Initialized 6 booth objects in the boothObjectArray with boothNum and "e"....
        for (int i = 0; i < boothObjectsArray.length; i++)
        {
            boothObjectsArray[i] = new Booth(i,"e");
        }

        //Menu Options....
        System.out.println
                (
                        "100 0r 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"
        );

while (true){

    System.out.println();
    System.out.print("Select the task from the menu : ");
    String operation = myScanner.next();

    //operations begins here
    if (operation.equals("100") || operation.equals("VVB"))
    {
        viewAllVaccinationBooths();
    }
    else if (operation.equals("101") || operation.equals("VEB"))
```

```
{
    viewAllEmptyBooths();
}
else if (operation.equals("102") || operation.equals("APB"))
{
    addDetails();
}
else if (operation.equals("103") || operation.equals("RPB"))
{
    removeDetails();
}
else if (operation.equals("104") || operation.equals("VPS"))
{
    viewPatientsSortedInAlphabeticalOrder();
}
else if (operation.equals("105") || operation.equals("SPD"))
{
    storeProgramDataIntoFile();
}
else if (operation.equals("106") || operation.equals("LPD"))
{
```

```
            loadProgramDataFromFile();
        }
        else if (operation.equals("107") || operation.equals("VRV"))
        {
            viewRemainingVaccinations();
        }
        else if (operation.equals("108") || operation.equals("AVS"))
        {
            addVaccinationsToTheStock();
        }
        else if (operation.equals("999") || operation.equals("EXT"))
        {
            System.out.println("The program is Exiting.....");
            break;
        }
        else{
            System.out.println("Enter the correct value for the TASK...");
        }
    }
}
```

```
    private static void viewAllVaccinationBooths(){


        for (int i = 0; i < boothObjectsArray.length; i++)
        {
            if (boothObjectsArray[i].getPatientName().equals("e"))
            {
                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Empty");
            }
            else
            {
                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Occupied by " + "\""
+ boothObjectsArray[i].getPatientName() + "\"");
            }


        }
    }


    private static void viewAllEmptyBooths(){
        //Making a boolean called "flag" to check whether atleast one booth is being empty...
        boolean flag = true;
        for (int i = 0; i < boothObjectsArray.length; i++)
```

```
    {
        if (boothObjectsArray[i].getPatientName().equals("e"))
        {
            System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Empty");
            flag = false;
        }
    }
    if (flag)
    {
        System.out.println("Currently All Booths are Occupied.");
    }
}

private static void addDetails(){

    while (true) {
        //Making a boolean called "flag" to check whether atleast one booth is being empty to add a patient...
        boolean flag = true;
        for (int i = 0; i < boothObjectsArray.length; i++) {
            if (boothObjectsArray[i].getPatientName().equals("e")) {
                flag = false;
```

```
            }
        }
        if (flag) {
            System.out.println("Can't add patients because all the 6 Booths have been occupied.");
            break;
        }

        //Giving operator an idea about where he can add patients by showing him the currently empty booths...
        System.out.println();
        System.out.println("Currently empty booth are: ");
        viewAllEmptyBooths();
        System.out.println();

        System.out.print("In which booth you want to add the patient [Enter a Booth Number from 0 to 5] : ");
        int boothNumber = myScanner.nextInt();

        //BoothNum and index of Booth Objects in boothObjectArray are same.. so It can be done easily....
        if (boothObjectsArray[boothNumber].getPatientName().equals("e")) {
            System.out.print("What is the name of the patient : ");
            String name = myScanner.next();
```

```
            boothObjectsArray[boothNumber].addPatientToABooth(name);

            stocks--;



            if (stocks == 20) {

               System.out.println();

               System.out.println("WARNING !!! ..... There are only " + stocks + " vaccinations remaining....");

            }

            break;

         } else

         {

            System.out.println("This booth has already been Occupied by " + "\"" + boothObjectsArray[boothNumber].getPatientName() + "\"");

         }

      }



   }



   private static void removeDetails(){



      while (true) {
```

```
        //Making a boolean called "flag" to check whether atleast one booth is not being empty...
        boolean flag = true;
        for (int i = 0; i < boothObjectsArray.length; i++) {
            if (!boothObjectsArray[i].getPatientName().equals("e")) {
                flag = false;
            }
        }
        if (flag) {
            System.out.println("Can't remove patients because all the 6 Booths are currently being Empty.");
            break;
        }


        System.out.println();
        System.out.println("Currently Occupied booths are: ");
        for (int i = 0; i < boothObjectsArray.length; i++) {
            if (!boothObjectsArray[i].getPatientName().equals("e")) {
                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Occupied by " +
"\"" + boothObjectsArray[i].getPatientName() + "\"");
            }
        }
```

```
        //Giving operator an idea about from where he can remove patients by showing him the currently occupied booths...
        System.out.println();
        System.out.print("From which Booth you want to remove the patient [Enter a Booth Number from 0 to 5] : ");
        int boothNumber = myScanner.nextInt();

        //BoothNum and index of Booth Objects in boothObjectArray are same.. so It can be done easily....
        if (!boothObjectsArray[boothNumber].getPatientName().equals("e")) {
            String name = boothObjectsArray[boothNumber].getPatientName();
            boothObjectsArray[boothNumber].removePatientFromABooth(name);
            break;
        } else {
            System.out.println("This Booth is already being empty....");
        }
    }
}

private static void viewPatientsSortedInAlphabeticalOrder(){

    //Making a boolean called "flag" to check whether atleast one booth is not being empty...
    while (true) {
```

```
boolean flag = true;
for (int i = 0; i < boothObjectsArray.length; i++) {
    if (!boothObjectsArray[i].getPatientName().equals("e")) {
        flag = false;
    }
}
if (flag) {
    System.out.println("Can't sorted patients names because all the 6 Booths are Empty.");
    break;
}

//Making an array called "sortedNames" which contains boothObjectsArray patients names...
String[] sortedNames = new String[6];
for (int i = 0; i < sortedNames.length; i++)
{
    sortedNames[i] = boothObjectsArray[i].getPatientName();
}

//To hold Temporary Values....
String temporaryString;
for (int i = 0; i < sortedNames.length; i++) {
```

```
    for (int j = i + 1; j < sortedNames.length; j++)
    {
        if (!sortedNames[i].equals("e") && !sortedNames[j].equals("e"))
        {
            if (sortedNames[i].compareToIgnoreCase(sortedNames[j]) > 0) {
                temporaryString = sortedNames[i];
                sortedNames[i] = sortedNames[j];
                sortedNames[j] = temporaryString;
            }
        }
    }
}

System.out.println("And The Names Sorted in Alphabetical Order Are As Follows...");

for (int i = 0; i < sortedNames.length; i++)
{
    if(!sortedNames[i].equals("e"))
    {
        System.out.println(sortedNames[i]);
    }
```

```
        }
        break;
    }
}

private static void storeProgramDataIntoFile(){

    try {
        File patientFileTask2 = new File("PatientFileTask2.txt");
        patientFileTask2.createNewFile();

        FileWriter myWriter = new FileWriter("PatientFileTask2.txt");
        for (int i = 0; i < boothObjectsArray.length; i++)
        {
            myWriter.write(boothObjectsArray[i].getPatientName()+ "\n");
        }
        myWriter.close();

        System.out.println("Patient names have been Successfully Stored in \" PatientFileTask2 \" TextFile..");
```

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

private static void loadProgramDataFromFile(){

    try{
        File PatientFile2 = new File("PatientFileTask2.txt");
        Scanner myReader = new Scanner(PatientFile2);

        while (myReader.hasNextLine())
        {
            for (int i = 0; i < boothObjectsArray.length; i++)
            {
                String data = myReader.nextLine();
                boothObjectsArray[i].setPatientName(data);

            }
        }
```

```
            System.out.println("Patient name have been Successfully loaded from the \" PatientFileTask2 \" TextFile");
        }
        catch(FileNotFoundException e){
            System.out.println("File not found..");
        }

    }

    private static void viewRemainingVaccinations() {
        System.out.println("The remaining vaccinations are : " + stocks);
    }

    private static void addVaccinationsToTheStock(){

        System.out.print("How many vaccinations that you gonna add to the Stocks: ");
        int vaccinationnumber = myScanner.nextInt();

        stocks = stocks + vaccinationnumber;
        System.out.println(vaccinationnumber + " vaccinations has been successfully added to stocks");
```

```
    }
}
```

Task 03[Arrays Version]

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.Scanner;
```

```
public class Arrays_Version_Extended {


    //Making the array Static so Each Static methods can access it.....
    static String[] serviceCenterBooths = new String[6];


    //Parallel Arrays to hold relationship between indexes of each arrays....
    static String[] firstNameArray = new String[6];
    static String[] surNameArray = new String[6];
    static String[] vaccinationRequestArray = new String[6];


    static int stocks = 150;
    static Scanner myScanner = new Scanner(System.in);


    public static void main(String[] args){
        // Initialising the ServiceCenterbooths as "e" This means that every booth is empty initially.....
        initialise();


        //Menu Options
        System.out.println
                (
```

```
            "100 0r 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"
        );
while (true) {

    System.out.println();
    System.out.print("Select the task from the menu : ");
    String operation = myScanner.next();

    //operations begins here
    if (operation.equals("100") || operation.equals("VVB"))
    {
        viewAllVaccinationBooths();
```

```
}
else if (operation.equals("101") || operation.equals("VEB"))
{
   viewAllEmptyBooths();
}
else if (operation.equals("102") || operation.equals("APB"))
{
   addPatientToABooth();
}
else if (operation.equals("103") || operation.equals("RPB"))
{
   removePatientFromABooth();
}
else if (operation.equals("104") || operation.equals("VPS"))
{
   viewPatientsSortedInAlphabeticalOrder();
}
else if (operation.equals("105") || operation.equals("SPD"))
{
   storeProgramDataIntoFile();
}
```

```
    else if (operation.equals("106") || operation.equals("LPD"))
    {
        loadProgramDataFromFile();
    }
    else if (operation.equals("107") || operation.equals("VRV"))
    {
        viewRemainingVaccinations();
    }
    else if (operation.equals("108") || operation.equals("AVS"))
    {
        addVaccinationsToTheStock();
    }
    else if (operation.equals("999") || operation.equals("EXT"))
    {
        System.out.println("The program is Exiting.....");
        break;
    }
    else{
        System.out.println("Enter the correct value for the TASK...");
    }
}
```

```
}

private static void initialise(){
    // Initialising the ServiceCenterbooths as "e" This means that every booth is empty initially.....
    for (int i = 0; i < serviceCenterBooths.length; i++)
    {
        serviceCenterBooths[i] = "e";
    }
}

private static void viewAllVaccinationBooths(){

    for (int i = 0; i < serviceCenterBooths.length; i++)
    {
        if (!serviceCenterBooths[i].equals("e"))
        {
            System.out.println("Booth " + i + " is occupied by " + serviceCenterBooths[i]);

        }
        else{
            System.out.println("Booth " + i + " is empty.");
```

```
        }
    }

}

private static void viewAllEmptyBooths(){
    //Making a boolean called "flag" to check whether atleast one booth is being empty...
    boolean flag = true;
    for (int i = 0; i < serviceCenterBooths.length; i++)
    {
        if (serviceCenterBooths[i].equals("e"))
        {
            System.out.println("Booth " + i + " is Empty.");
            flag = false;
        }
    }
    if (flag)
    {
        System.out.println("Currently All Booths are Occupied.");
    }
}
```

```
private static void addPatientToABooth(){


    while (true) {


        //Making a boolean called "flag" to check whether atleast one booth is being empty to add a patient...
        boolean flag = true;
        for (int i = 0; i < serviceCenterBooths.length; i++)
        {
            if (serviceCenterBooths[i].equals("e"))
            {
                flag = false;
            }
        }
        if (flag)
        {
            System.out.println("Can't add patients because all the 6 Booths have been occupied.");
            break;
        }


        System.out.println();
```

```
System.out.println("Booth Allocation Details...");
System.out.println
    (
        "Booth 0 and Booth 1 FOR : \" AstraZeneca \" \n" +
            "Booth 2 and Booth 3 FOR : \" Sinopharm \" \n" +
            "Booth 4 and Booth 5 FOR : \" Pfizer \" "
    );

System.out.println();
System.out.print("Patient Details....");
System.out.println();

System.out.print("First Name          : ");
String FirstName = myScanner.next();

System.out.print("Sur Name           : ");
String SurName = myScanner.next();

System.out.println();
System.out.println("Vaccine List is follows..");
System.out.println();
```

```
System.out.println
    (
        "For AstraZeneca Press 1 \n" +
            "For Sinopharm Press 2 \n" +
            "For Pfizer Press 3"
    );
System.out.println();
System.out.print("Which Vaccine you prefer to get : ");
int VaccinationRequest = myScanner.nextInt();


if (VaccinationRequest == 1){


    //Making a boolean called "flag1" to check whether atleast one booth that Allocated to Provide "AstraZeneca" vaccine is being empty...
    boolean flag1 = true;
    for (int i = 0; i < 2; i++)
    {
        if (serviceCenterBooths[i].equals("e"))
        {
            flag1 = false;
```

```
                firstNameArray[i] = FirstName;

                surNameArray[i] = SurName;

                vaccinationRequestArray[i] = "AstraZeneca";

                serviceCenterBooths[i] = FirstName;

                System.out.println("\"" + serviceCenterBooths[i] + "\"" + " has been successfully added to booth " + i);

                stocks--;

                break;

            }

        }

        if (flag1) {

            System.out.println("The Booths that allocated to provide \" AstraZeneca \" vaccine [Booth 0 and Booth 1] are currently busy...");

        }

    }

    else if (VaccinationRequest == 2){



        //Making a boolean called "flag2" to check whether atleast one booth that Allocated to Provide "Sinopharm" vaccine is being empty...

        boolean flag2 = true;

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

        {
```

```
            if (serviceCenterBooths[i].equals("e"))
            {
                flag2 = false;
                firstNameArray[i] = FirstName;
                surNameArray[i] = SurName;
                vaccinationRequestArray[i] = "Sinopharm";
                serviceCenterBooths[i] = FirstName;
                System.out.println("\"" + serviceCenterBooths[i] + "\"" + " has been successfully added to booth " + i);
                stocks--;
                break;
            }
        }
        if (flag2) {
            System.out.println("The Booths that allocated to provide \" Sinopharm \" vaccine [Booth 2 and Booth 3] are currently busy...");
        }
    }
    else if (VaccinationRequest == 3){

        //Making a boolean called "flag3" to check whether atleast one booth that Allocated to Provide "Pfizer" vaccine is being empty...
```

```
            boolean flag3 = true;
            for (int i = 4; i < 6; i++)
            {
               if (serviceCenterBooths[i].equals("e"))
               {
                  flag3 = false;
                  firstNameArray[i] = FirstName;
                  surNameArray[i] = SurName;
                  vaccinationRequestArray[i] = "Pfizer";
                  serviceCenterBooths[i] = FirstName;
                  System.out.println("\"" + serviceCenterBooths[i] + "\"" + " has been successfully added to booth " + i);
                  stocks--;
                  break;
               }
            }
            if (flag3) {
               System.out.println("The Booths that allocated to provide \" Pfizer \" vaccine [Booth 2 and Booth 3] are currently busy...");
            }
         }
         if (stocks == 20) {
```

```
            System.out.println();
            System.out.println("WARNING !!! ..... There are only " + stocks + " vaccinations remaining....");
        }
        break;
      }
   }

   private static void removePatientFromABooth(){

      while(true) {

         //Making a boolean called "flag" to check whether atleast one booth is not being empty...
         boolean flag = true;
         for (int i = 0; i < serviceCenterBooths.length; i++)
         {
            if (!serviceCenterBooths[i].equals("e"))
            {
               flag = false;
            }
         }
```

```
        if (flag)
        {
            System.out.println("Can't remove patients because all the 6 Booths are currently being Empty.");
            break;
        }

        //Giving operator an idea about from where he can remove patients by showing him the currently occupied booths...
        System.out.println();
        System.out.println("Currently Occupied booths are: ");
        for (int i = 0; i < serviceCenterBooths.length; i++) {
            if (!serviceCenterBooths[i].equals("e")) {
                System.out.println("Booth " + i + " is Occupied by " + "\"" + serviceCenterBooths[i] + "\"");
            }
        }

        System.out.println();
        System.out.print("From which Booth you want to remove the patient [Enter a Booth Number from 0 to 5] : ");
        int boothnum = myScanner.nextInt();

        if (!serviceCenterBooths[boothnum].equals("e")) {
```

```
            String patientName = serviceCenterBooths[boothnum];
            serviceCenterBooths[boothnum] = "e";
            System.out.println("\"" + patientName + "\"" + " has been successfully removed from booth " + boothnum);

            firstNameArray[boothnum] = null;
            surNameArray[boothnum] = null;
            vaccinationRequestArray[boothnum] = null;
            break;
        } else {
            System.out.println("The Booth is already being Empty...");
        }
    }
}

private static void viewPatientsSortedInAlphabeticalOrder(){

    //Making a boolean called "flag" to check whether atleast one booth is not being empty...
    while (true) {
        boolean flag = true;
        for (int i = 0; i < serviceCenterBooths.length; i++) {
```

```
            if (!serviceCenterBooths[i].equals("e")) {
                flag = false;
            }
        }
        if (flag) {
            System.out.println("Can't sorted patients names because all the 6 Booths are Empty.");
            break;
        }

        //Copying elements from serviceCenterBooths Array to sortedNames array because changes won't affect serviceCenterBooth array...
        String[] sortedNames = new String[6];
        for (int i = 0; i < sortedNames.length; i++)
        {
            sortedNames[i] = serviceCenterBooths[i];
        }

        //To hold Temporary Values....
        String temporaryString;
        for (int i = 0; i < sortedNames.length; i++) {
            for (int j = i + 1; j < sortedNames.length; j++)
```

```
    {
        if (!sortedNames[i].equals("e") && !sortedNames[j].equals("e"))
        {
            if (sortedNames[i].compareToIgnoreCase(sortedNames[j]) > 0) {
                temporaryString = sortedNames[i];
                sortedNames[i] = sortedNames[j];
                sortedNames[j] = temporaryString;
            }
        }
    }
}

System.out.println("And The Names Sorted in Alphabetical Order Are As Follows...");

for (int i = 0; i < sortedNames.length; i++)
{
    if(!sortedNames[i].equals("e"))
    {
        System.out.println(sortedNames[i]);
    }
}
```

```
        break;
    }
}

private static void storeProgramDataIntoFile(){

    try {
        File patientFileTask1Extended = new File("patientFileTask1Extended.txt");
        patientFileTask1Extended.createNewFile();

        FileWriter myWriter = new FileWriter("patientFileTask1Extended.txt");
        for (int i = 0; i < serviceCenterBooths.length; i++)
        {
            if (!serviceCenterBooths[i].equals("e")) {
                myWriter.write(firstNameArray[i] + " ");
                myWriter.write(surNameArray[i] + " ");
                myWriter.write(vaccinationRequestArray[i] + "\n");
            }
            else{
                myWriter.write("empty" + "\n");
            }
```

```
            }

            myWriter.close();


            System.out.println("Patient details have been Successfully Stored in \" patientFileTask1Extended \" TextFile..");


        }

        catch (IOException e){

            System.out.println("An Error has occurred");

        }


    }


    private static void loadProgramDataFromFile(){


        try{

            File PatientFile2 = new File("patientFileTask1Extended.txt");

            Scanner myReader = new Scanner(PatientFile2);


            while (myReader.hasNextLine())

            {
```

```
    for (int i = 0; i < serviceCenterBooths.length; i++)
    {
        String data = myReader.nextLine();
        if (data.equals("empty")){
            firstNameArray[i] = null;
            surNameArray[i] = null;
            vaccinationRequestArray[i] = null;
            serviceCenterBooths[i] = "e";
            continue;
        }

        //Creating an array called "dataArray" with all the Patient details as its elements...
        String[] dataArray = data.split(" ");
        firstNameArray[i] = dataArray[0];
        surNameArray[i] = dataArray [1];
        vaccinationRequestArray[i] = dataArray[2];
        serviceCenterBooths[i] = dataArray[0];
    }
}
```

```
            System.out.println("Patient details have been Successfully loaded from the \" patientFileTask1Extended \" TextFile..");
        }
        catch(FileNotFoundException e){
            System.out.println("File not found..");
        }
    }

    private static void viewRemainingVaccinations(){
        System.out.println("The remaining vaccinations are : " + stocks);
    }

    private static void addVaccinationsToTheStock(){

        System.out.print("How many vaccinations that you gonna add to the Stocks: ");
        int vaccinationnumber = myScanner.nextInt();

        stocks = stocks + vaccinationnumber;
        System.out.println(vaccinationnumber + " vaccinations has been successfully added to stocks");
    }
```

```
}
```

Task 03[Classes Version]

Booth Class

```
public class Booth {

    private int boothNum;

    private Patient patientObject;


    //Overloaded Constructor....

    public Booth(int boothNum){

        this.boothNum = boothNum;
```

```
}

public int getBoothNum() {
    return boothNum;
}

public void setBoothNum(int boothNum) {
    this.boothNum = boothNum;
}

public Patient getPatientObject() {
    return patientObject;
}

public void setPatientObject(Patient patientObject) {
    this.patientObject = patientObject;
}

public void addPatientToABooth(Patient patient){
    this.patientObject = patient;
```

```
        System.out.println("\"" + this.patientObject.getFirstName() + "\"" + " has been successfully added to booth " + this.boothNum);
    }

    public void removePatientFromABooth(String patientName){
        this.patientObject = null;
        System.out.println("\"" + patientName + "\"" + " has been successfully removed from booth " + this.boothNum);
    }
}
```

Patient Class

```
public class Patient {

    private String firstName;
    private String surName;
    private int age;
    private String city;
    private String NICorPassport;
```

```
    private int vaccinationRequest;


    //Overloaded Constructor....
    public Patient (String firstName, String surName, int age, String city, String NICorPassport, int vaccinationRequest){
        this.firstName = firstName;
        this.surName = surName;
        this.age = age;
        this.city = city;
        this.NICorPassport = NICorPassport;
        this.vaccinationRequest = vaccinationRequest;
    }


    public String getFirstName() {
        return firstName;
    }


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

```
public String getSurName() {
    return surName;
}

public void setSurName(String surName) {
    this.surName = surName;
}

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 String getNICorPassport() {
        return NICorPassport;
    }

    public void setNICorPassport(String NICorPassport) {
        this.NICorPassport = NICorPassport;
    }

    public int getVaccinationRequest() {
        return vaccinationRequest;
    }

    public void setVaccinationRequest(int vaccinationRequest) {
        this.vaccinationRequest = vaccinationRequest;
    }
}
```

VaccinationCenter Class

```
import java.io.File;

import java.io.IOException;

import java.io.FileWriter;

import java.io.FileNotFoundException;

import java.util.Scanner;


public class VaccinationCenter {
```

```
static Booth[] boothObjectsArray = new Booth[6];

static int stocks = 150;

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


public static void main(String[] args){


   //Initialized 6 booth objects in the boothObjectArray with boothNum and "e"....

   for (int i = 0; i < boothObjectsArray.length; i++)

   {

      boothObjectsArray[i] = new Booth(i);

   }


   //Menu Options....

   System.out.println

         (

               "100 0r 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"
        );

while (true){

    System.out.println();
    System.out.print("Select the task from the menu : ");
    String operation = myScanner.next();

    //operations begins here
    if (operation.equals("100") || operation.equals("VVB"))
    {
        viewAllVaccinationBooths();
    }
    else if (operation.equals("101") || operation.equals("VEB"))
    {
        viewAllEmptyBooths();
    }
```

```
else if (operation.equals("102") || operation.equals("APB"))
{
   addDetails();
}
else if (operation.equals("103") || operation.equals("RPB"))
{
   removeDetails();
}
else if (operation.equals("104") || operation.equals("VPS"))
{
   viewPatientsSortedInAlphabeticalOrder();
}
else if (operation.equals("105") || operation.equals("SPD"))
{
   storeProgramDataIntoFile();
}
else if (operation.equals("106") || operation.equals("LPD"))
{
   loadProgramDataFromFile();
}
else if (operation.equals("107") || operation.equals("VRV"))
```

```
        {
            viewRemainingVaccinations();
        }
        else if (operation.equals("108") || operation.equals("AVS"))
        {
            addVaccinationsToTheStock();
        }
        else if (operation.equals("999") || operation.equals("EXT"))
        {
            System.out.println("The program is Exiting.....");
            break;
        }
        else{
            System.out.println("Enter the correct value for the TASK...");
        }
    }
}

private static void viewAllVaccinationBooths(){

    for (int i = 0; i < boothObjectsArray.length; i++)
```

```
        {
            if (boothObjectsArray[i].getPatientObject() == null)
            {
                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Empty");
            }
            else
            {
                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Occupied by " + "\""
+ boothObjectsArray[i].getPatientObject().getFirstName() + "\"");
            }

        }
    }

    private static void viewAllEmptyBooths(){
        //Making a boolean called "flag" to check whether atleast one booth is being empty...
        boolean flag = true;
        for (int i = 0; i < boothObjectsArray.length; i++)
        {
            if (boothObjectsArray[i].getPatientObject() == null)
            {
```

```
            System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Empty");
            flag = false;
        }
    }
    if (flag)
    {
        System.out.println("Currently All Booths are Occupied.");
    }
}

private static void addDetails(){

    //Making a boolean called "flag" to check whether atleast one booth is being empty to add a patient...
    while (true) {
        boolean flag = true;
        for (int i = 0; i < boothObjectsArray.length; i++) {
            if (boothObjectsArray[i].getPatientObject() == null) {
                flag = false;
            }
        }
        if (flag) {
```

```
    System.out.println("Can't add patients because all the 6 Booths have been occupied.");
    break;
}

System.out.println();
System.out.println("Booth Allocation Details...");
System.out.println
      (
            "Booth 0 and Booth 1 FOR : \" AstraZeneca \" \n" +
                  "Booth 2 and Booth 3 FOR : \" Sinopharm \" \n" +
                  "Booth 4 and Booth 5 FOR : \" Pfizer \" "
      );

System.out.println();
System.out.print("Patient Details....");
System.out.println();

System.out.print("First Name          : ");
String FirstName = myScanner.next();

System.out.print("Sur Name            : ");
```

```
String SurName = myScanner.next();


System.out.print("Age                 : ");
int Age = myScanner.nextInt();


System.out.print("City                : ");
String City = myScanner.next();


System.out.print("NIC/Passport Number : ");
String NICorPassport = myScanner.next();


System.out.println();
System.out.println("Vaccine List is follows..");
System.out.println();


System.out.println
    (
        "For AstraZeneca Press 1 \n" +
            "For Sinopharm Press 2 \n" +
            "For Pfizer Press 3"
    );
```

```
        System.out.println();
        System.out.print("Which Vaccine you prefer to get : ");
        int VaccinationRequest = myScanner.nextInt();

        while (true) {
            if (VaccinationRequest == 1) {

                //Making a boolean called "flag1" to check whether atleast one booth that Allocated to Provide "AstraZeneca" vaccine is being empty...
                boolean flag1 = true;
                for (int i = 0; i < boothObjectsArray.length; i++) {
                    if ((boothObjectsArray[i].getBoothNum() == 0) || (boothObjectsArray[i].getBoothNum() == 1)) {
                        if (boothObjectsArray[i].getPatientObject() == null) {

                            flag1 = false;

                            boothObjectsArray[i].addPatientToABooth(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));
                            stocks--;
                            break;
                        }
```

```
            }
        }

        if (flag1) {
            System.out.println("The Booths that allocated to provide \" AstraZeneca \" vaccine [Booth 0 and Booth 1] are currently busy...");
            System.out.println();
            System.out.print("Do you prefer to go for other vaccination? if yes press \" y \" else press \" n \" : ");
            String checker = myScanner.next();

            if (checker.equals("y")) {
                System.out.println();
                System.out.println
                        (
                                "For Sinopharm Press 2 \n" +
                                        "For Pfizer Press 3"
                        );
                System.out.println();
                System.out.print("Which Vaccine you prefer to get : ");
                VaccinationRequest = myScanner.nextInt();
```

```
                continue;

            } else if (checker.equals("n")) {

                System.out.println("Sorry...You have to wait until the booths are getting empty and Register yourself again");

            }

        }

        break;



    }

    else if (VaccinationRequest == 2) {



        //Making a boolean called "flag2" to check whether atleast one booth that Allocated to Provide "Sinopharm" vaccine is being empty...

        boolean flag2 = true;

        for (int i = 0; i < boothObjectsArray.length; i++) {

            if ((boothObjectsArray[i].getBoothNum() == 2) || (boothObjectsArray[i].getBoothNum() == 3)) {

                if (boothObjectsArray[i].getPatientObject() == null) {

                    flag2 = false;



                    boothObjectsArray[i].addPatientToABooth(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));

                    stocks--;
```

```
                    break;
                }


            }
        }
        if (flag2) {
            System.out.println("The Booths that allocated to provide \" Sinopharm \" vaccine [Booth 2 and Booth 3] are currently busy...");
            System.out.println();
            System.out.print("Do you prefer to go for other vaccination? if yes press \" y \" else press \" n \" : ");
            String checker = myScanner.next();

            if (checker.equals("y"))
            {
                System.out.println();
                System.out.println
                        (
                                "For AstraZeneca Press 1 \n" +
                                        "For Pfizer Press 3"
                        );
                System.out.println();
```

```
                System.out.print("Which Vaccine you prefer to get : ");
                VaccinationRequest = myScanner.nextInt();
                continue;
            }
            else if (checker.equals("n"))
            {
                System.out.println("Sorry...You have to wait until the booths are getting empty and Register yourself again");
            }
        }
        break;
    }
    else if (VaccinationRequest == 3) {

        //Making a boolean called "flag3" to check whether atleast one booth that Allocated to Provide "Pfizer" vaccine is being empty...
        boolean flag3 = true;
        for (int i = 0; i < boothObjectsArray.length; i++) {
            if ((boothObjectsArray[i].getBoothNum() == 4) || (boothObjectsArray[i].getBoothNum() == 5)) {
                if (boothObjectsArray[i].getPatientObject() == null) {
                    flag3 = false;
```

```
                    boothObjectsArray[i].addPatientToABooth(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));
                    stocks--;
                    break;
                }

            }
        }
        if (flag3) {
            System.out.println("The Booths that allocated to provide \" Pfizer \" vaccine [Booth 4 and Booth 5] are currently busy...");
            System.out.println();
            System.out.print("Do you prefer to go for other vaccination? if yes press \" y \" else press \" n \" : ");
            String checker = myScanner.next();

            if (checker.equals("y"))
            {
                System.out.println();
                System.out.println
                        (
```

```
                    "For AstraZeneca Press 1 \n" +
                            "For Sinopharm Press 2"
                    );
                System.out.println();
                System.out.print("Which Vaccine you prefer to get : ");
                VaccinationRequest = myScanner.nextInt();
                continue;
            }
            else if (checker.equals("n"))
            {
                System.out.println("Sorry...You have to wait until the booths are getting empty and Register yourself again");
            }
        }
        break;
    }
}
if (stocks == 20) {
    System.out.println();
    System.out.println("WARNING !!! ..... There are only " + stocks + " vaccinations remaining....");
}
```

```
        break;
    }
}

private static void removeDetails(){

    while (true) {
        //Making a boolean called "flag" to check whether atleast one booth is not being empty...
        boolean flag = true;
        for (int i = 0; i < boothObjectsArray.length; i++) {
            if (!(boothObjectsArray[i].getPatientObject() == null)) {
                flag = false;
            }
        }
        if (flag) {
            System.out.println("Can't remove patients because all the 6 Booths are currently being Empty.");
            break;
        }

        System.out.println();
        System.out.println("Currently Occupied booths are: ");
```

```
        for (int i = 0; i < boothObjectsArray.length; i++) {
            if (!(boothObjectsArray[i].getPatientObject() == null)) {
                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Occupied by " + "\"" + boothObjectsArray[i].getPatientObject().getFirstName() + "\"");
            }
        }

        //Giving operator an idea about from where he can remove patients by showing him the currently occupied booths...
        System.out.println();
        System.out.print("From which Booth you want to remove the patient [Enter a Booth Number from 0 to 5] : ");
        int boothNumber = myScanner.nextInt();

        //BoothNum and index of Booth Objects in boothObjectArray are same.. so It can be done easily....
        if (!(boothObjectsArray[boothNumber].getPatientObject() == null)) {
            String name = boothObjectsArray[boothNumber].getPatientObject().getFirstName();
            boothObjectsArray[boothNumber].removePatientFromABooth(name);
            break;
        } else {
            System.out.println("This Booth is already being empty....");
        }
```

```
    }
}

private static void viewPatientsSortedInAlphabeticalOrder(){

    while (true) {
        //Making a boolean called "flag" to check whether atleast one booth is not being empty...
        boolean flag = true;
        for (int i = 0; i < boothObjectsArray.length; i++) {
            if (!(boothObjectsArray[i].getPatientObject() == null)) {
                flag = false;
            }
        }
        if (flag) {
            System.out.println("Can't sorted patients names because all the 6 Booths are Empty.");
            break;
        }

        //Making an array called "sortedNames" which contains boothObjectsArray patients names...
        String[] sortedNames = new String[6];
        for (int i = 0; i < sortedNames.length; i++)
```

```
{
    if (!(boothObjectsArray[i].getPatientObject() == null)) {
        sortedNames[i] = boothObjectsArray[i].getPatientObject().getFirstName();
    }
    else{
        sortedNames[i] = "e";
    }
}

//To hold Temporary Values....
String temporaryString;
for (int i = 0; i < sortedNames.length; i++) {
    for (int j = i + 1; j < sortedNames.length; j++)
    {
        if (sortedNames[i].compareToIgnoreCase(sortedNames[j]) > 0) {
            temporaryString = sortedNames[i];
            sortedNames[i] = sortedNames[j];
            sortedNames[j] = temporaryString;
        }
    }
}
```

```
        System.out.println("And The Names Sorted in Alphabetical Order Are As Follows...");


        for (int i = 0; i < sortedNames.length; i++)
        {
           if (!sortedNames[i].equals("e")) {
              System.out.println(sortedNames[i]);
           }
        }
        break;
    }
}


private static void storeProgramDataIntoFile(){


    try {
        File patientFileTask2Extended = new File("patientFileTask2Extended.txt");
        patientFileTask2Extended.createNewFile();


        FileWriter myWriter = new FileWriter("patientFileTask2Extended.txt");
        for (int i = 0; i < boothObjectsArray.length; i++)
```

```
        {
            if (!(boothObjectsArray[i].getPatientObject() == null)) {
                myWriter.write(boothObjectsArray[i].getPatientObject().getFirstName() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getSurName() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getAge() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getCity() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getNICorPassport() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getVaccinationRequest() + "\n");
            }
            else{
                myWriter.write("empty" + "\n");
            }
        }
        myWriter.close();

        System.out.println("Patient Details have been Successfully Stored in \" patientFileTask2Extended \" TextFile...");

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

```
    }
}

private static void loadProgramDataFromFile(){

    try{
        File PatientFile2 = new File("patientFileTask2Extended.txt");
        Scanner myReader = new Scanner(PatientFile2);

        while (myReader.hasNextLine())
        {
            for (int i = 0; i < boothObjectsArray.length; i++)
            {
                String data = myReader.nextLine();
                if (data.equals("empty")){
                    boothObjectsArray[i].setPatientObject(null);
                    continue;
                }

                //Creating an array with all the Patient attribute...
                String[] dataArray = data.split(" ");
```

```
                //setting up the attributes in the same place....

                boothObjectsArray[i].setPatientObject(new
Patient(dataArray[0],dataArray[1],Integer.parseInt(dataArray[2]),dataArray[3],dataArray[4],Integer.parse
Int(dataArray[5])));



            }

        }


        System.out.println("Patient Details have been Successfully loaded from the \"
patientFileTask2Extended \" TextFile...");

    }

    catch(FileNotFoundException e){

        System.out.println("File not found..");

    }


}


  private static void viewRemainingVaccinations() {

    System.out.println("The remaining vaccinations are : " + stocks);

  }
```

```
    private static void addVaccinationsToTheStock(){


        System.out.print("How many vaccinations that you gonna add to the Stocks: ");
        int vaccinationnumber = myScanner.nextInt();


        stocks = stocks + vaccinationnumber;
        System.out.println(vaccinationnumber + " vaccinations has been successfully added to stocks");
    }
}
```

Task04 [Linked List Version]

Booth Class

```
public class Booth {


    private int boothNum;
    private Patient patientObject;


    //Overloaded Constructor....
    public Booth(int boothNum){
        this.boothNum = boothNum;
```

```
}

public int getBoothNum() {
    return boothNum;
}

public void setBoothNum(int boothNum) {
    this.boothNum = boothNum;
}

public Patient getPatientObject() {
    return patientObject;
}

public void setPatientObject(Patient patientObject) {
    this.patientObject = patientObject;
}

public void addPatientToABooth(Patient patient){
    this.patientObject = patient;
```

```
        System.out.println("\"" + this.patientObject.getFirstName() + "\"" + " has been successfully added to booth " + this.boothNum);

    }


    public void removePatientFromABooth(String patientName){

        this.patientObject = null;

        System.out.println("\"" + patientName + "\"" + " has been successfully removed from booth " + this.boothNum);

    }

}
```

Patient Class

```
public class Patient {


    private String firstName;

    private String surName;

    private int age;

    private String city;

    private String NICorPassport;
```

```
    private int vaccinationRequest;


    //Overloaded Constructor....
    public Patient (String firstName, String surName, int age, String city, String NICorPassport, int vaccinationRequest){
        this.firstName = firstName;
        this.surName = surName;
        this.age = age;
        this.city = city;
        this.NICorPassport = NICorPassport;
        this.vaccinationRequest = vaccinationRequest;
    }



    public String getFirstName() {
        return firstName;
    }


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

```
public String getSurName() {
    return surName;
}

public void setSurName(String surName) {
    this.surName = surName;
}

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 String getNICorPassport() {
        return NICorPassport;
    }

    public void setNICorPassport(String NICorPassport) {
        this.NICorPassport = NICorPassport;
    }

    public int getVaccinationRequest() {
        return vaccinationRequest;
    }

    public void setVaccinationRequest(int vaccinationRequest) {
        this.vaccinationRequest = vaccinationRequest;
    }
}
```

VaccinationCenter Class

```
import java.io.File;

import java.io.IOException;

import java.io.FileWriter;

import java.io.FileNotFoundException;

import java.util.LinkedList;

import java.util.Scanner;
```

```
public class VaccinationCenter {


    static Booth[] boothObjectsArray = new Booth[6];
    static int stocks = 150;
    static Scanner myScanner = new Scanner(System.in);


    //waiting List to hold waiting patients...
    static LinkedList<Patient> waitingList = new LinkedList<Patient>();


    public static void main(String[] args){


        //Initialized 6 booth objects in the boothObjectArray with boothNum and "e"....
        for (int i = 0; i < boothObjectsArray.length; i++)
        {
            boothObjectsArray[i] = new Booth(i);
        }


        //Menu Options....
        System.out.println
              (
                    "100 0r 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"
        );

while (true){

    System.out.println();
    System.out.print("Select the task from the menu : ");
    String operation = myScanner.next();

    //operations begins here
    if (operation.equals("100") || operation.equals("VVB"))
    {
        viewAllVaccinationBooths();
```

```
}
else if (operation.equals("101") || operation.equals("VEB"))
{
   viewAllEmptyBooths();
}
else if (operation.equals("102") || operation.equals("APB"))
{
   addDetails();
}
else if (operation.equals("103") || operation.equals("RPB"))
{
   removeDetails();
}
else if (operation.equals("104") || operation.equals("VPS"))
{
   viewPatientsSortedInAlphabeticalOrder();
}
else if (operation.equals("105") || operation.equals("SPD"))
{
   storeProgramDataIntoFile();
}
```

```
    else if (operation.equals("106") || operation.equals("LPD"))
    {
        loadProgramDataFromFile();
    }
    else if (operation.equals("107") || operation.equals("VRV"))
    {
        viewRemainingVaccinations();
    }
    else if (operation.equals("108") || operation.equals("AVS"))
    {
        addVaccinationsToTheStock();
    }
    else if (operation.equals("999") || operation.equals("EXT"))
    {
        System.out.println("The program is Exiting.....");
        break;
    }
    else{
        System.out.println("Enter the correct value for the TASK...");
    }
}
```

```
    }

    private static void viewAllVaccinationBooths(){

        for (int i = 0; i < boothObjectsArray.length; i++)
        {
            if (boothObjectsArray[i].getPatientObject() == null)
            {
                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Empty");
            }
            else
            {
                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Occupied by " + "\""
+ boothObjectsArray[i].getPatientObject().getFirstName() + "\"");
            }

        }
    }

    private static void viewAllEmptyBooths(){
        //Making a boolean called "flag" to check whether atleast one booth is being empty...
```

```
    boolean flag = true;
    for (int i = 0; i < boothObjectsArray.length; i++)
    {
        if (boothObjectsArray[i].getPatientObject() == null)
        {
            System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Empty");
            flag = false;
        }
    }
    if (flag)
    {
        System.out.println("Currently All Booths are Occupied.");
    }
}

private static void addDetails(){

    while (true) {

        System.out.println();
        System.out.println("Booth Allocation Details...");
```

```
System.out.println
        (
                "Booth 0 and Booth 1 FOR : \" AstraZeneca \" \n" +
                        "Booth 2 and Booth 3 FOR : \" Sinopharm \" \n" +
                        "Booth 4 and Booth 5 FOR : \" Pfizer \" "
        );


System.out.println();
System.out.print("Patient Details....");
System.out.println();


System.out.print("First Name          : ");
String FirstName = myScanner.next();


System.out.print("Sur Name           : ");
String SurName = myScanner.next();


System.out.print("Age                : ");
int Age = myScanner.nextInt();


System.out.print("City               : ");
```

```
String City = myScanner.next();


System.out.print("NIC/Passport Number : ");
String NICorPassport = myScanner.next();


System.out.println();
System.out.println("Vaccine List is follows..");
System.out.println();


System.out.println
    (
        "For AstraZeneca Press 1 \n" +
            "For Sinopharm Press 2 \n" +
            "For Pfizer Press 3"
    );
System.out.println();
System.out.print("Which Vaccine you prefer to get : ");
int VaccinationRequest = myScanner.nextInt();



if (VaccinationRequest == 1) {
```

```
        //Making a boolean called "flag1" to check whether atleast one booth that Allocated to Provide "AstraZeneca" vaccine is being empty...

        boolean flag1 = true;

        for (int i = 0; i < boothObjectsArray.length; i++) {

            if ((boothObjectsArray[i].getBoothNum() == 0) || (boothObjectsArray[i].getBoothNum() == 1)) {

                if (boothObjectsArray[i].getPatientObject() == null) {



                    flag1 = false;



                    boothObjectsArray[i].addPatientToABooth(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));

                    stocks--;

                    break;

                }

            }

        }



        if (flag1) {

            System.out.println("The Booths that allocated to provide \" AstraZeneca \" vaccine [Booth 0 and Booth 1] are currently busy...");
```

```
            waitingList.add(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));

            System.out.println("\"" + FirstName + "\"" + " has been successfully added to waiting list...");

        }



    }

    else if (VaccinationRequest == 2) {


        //Making a boolean called "flag2" to check whether atleast one booth that Allocated to Provide "Sinopharm" vaccine is being empty...

        boolean flag2 = true;

        for (int i = 0; i < boothObjectsArray.length; i++) {

            if ((boothObjectsArray[i].getBoothNum() == 2) || (boothObjectsArray[i].getBoothNum() == 3)) {

                if (boothObjectsArray[i].getPatientObject() == null) {

                    flag2 = false;


                    boothObjectsArray[i].addPatientToABooth(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));

                    stocks--;

                    break;

                }
```

```
            }

        }

        if (flag2) {

            System.out.println("The Booths that allocated to provide \" Sinopharm \" vaccine [Booth 2 and Booth 3] are currently busy...");

            waitingList.add(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));

            System.out.println("\"" + FirstName + "\"" +" has been successfully added to waiting list...");



        }


    }

    else if (VaccinationRequest == 3) {



        //Making a boolean called "flag3" to check whether atleast one booth that Allocated to Provide "Pfizer" vaccine is being empty...

        boolean flag3 = true;

        for (int i = 0; i < boothObjectsArray.length; i++) {

            if ((boothObjectsArray[i].getBoothNum() == 4) || (boothObjectsArray[i].getBoothNum() == 5)) {

                if (boothObjectsArray[i].getPatientObject() == null) {

                    flag3 = false;
```

```
                boothObjectsArray[i].addPatientToABooth(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));

                stocks--;

                break;

            }


        }

    }

    if (flag3) {

        System.out.println("The Booths that allocated to provide \" Pfizer \" vaccine [Booth 4 and Booth 5] are currently busy...");

        waitingList.add(new Patient(FirstName, SurName, Age, City, NICorPassport, VaccinationRequest));

        System.out.println("\"" + FirstName + "\"" +" has been successfully added to waiting list...");


    }


}


if (stocks == 20) {

    System.out.println();
```

```
            System.out.println("WARNING !!! ..... There are only " + stocks + " vaccinations remaining....");
        }
        break;
      }
   }

   private static void removeDetails(){

      while (true) {
         //Making a boolean called "flag" to check whether atleast one booth is not being empty...
         boolean flag = true;
         for (int i = 0; i < boothObjectsArray.length; i++) {
            if (!(boothObjectsArray[i].getPatientObject() == null)) {
               flag = false;
            }
         }
         if (flag) {
            System.out.println("Can't remove patients because all the 6 Booths are currently being Empty.");
            break;
         }
```

```
        System.out.println();

        System.out.println("Currently Occupied booths are: ");

        for (int i = 0; i < boothObjectsArray.length; i++) {

            if (!(boothObjectsArray[i].getPatientObject() == null)) {

                System.out.println("Booth " + boothObjectsArray[i].getBoothNum() + " is Occupied by " + "\"" + boothObjectsArray[i].getPatientObject().getFirstName() + "\"");

            }

        }



        //Giving operator an idea about from where he can remove patients by showing him the currently occupied booths...

        System.out.println();

        System.out.print("From which Booth you want to remove the patient [Enter a Booth Number from 0 to 5] : ");

        int boothNumber = myScanner.nextInt();



        //BoothNum and index of Booth Objects in boothObjectArray are same.. so It can be done easily....

        if (!(boothObjectsArray[boothNumber].getPatientObject() == null)) {

            String name = boothObjectsArray[boothNumber].getPatientObject().getFirstName();

            boothObjectsArray[boothNumber].removePatientFromABooth(name);
```

```
            //After removing the patient from a particular Booth checking whether there are patients waiting to get vaccine that provided by this particular booth...

            checkWaitingList(boothNumber);



            break;

        } else {

            System.out.println("This Booth is already being empty....");

        }

    }

}



private static void checkWaitingList(int boothNumber){

    //Checking waiting list whether there are patients waiting to "AstraZeneca" Vaccine...

    if (boothNumber == 0 || boothNumber == 1)

    {

        for (int i = 0; i < waitingList.size(); i++)

        {

            if (waitingList.get(i).getVaccinationRequest() == 1)

            {

                System.out.println();

                System.out.println("From waiting list....");
```

```
            boothObjectsArray[boothNumber].addPatientToABooth(waitingList.get(i));
            stocks--;
            waitingList.remove(waitingList.get(i));
            break;
        }
    }
}
//Checking waiting list whether there are patients waiting to "Sinopharm" Vaccine...
else if (boothNumber == 2 || boothNumber == 3)
{
    for (int i = 0; i < waitingList.size(); i++)
    {
        if (waitingList.get(i).getVaccinationRequest() == 2)
        {
            System.out.println();
            System.out.println("From waiting list....");
            boothObjectsArray[boothNumber].addPatientToABooth(waitingList.get(i));
            stocks--;
            waitingList.remove(waitingList.get(i));
            break;
        }
```

```
        }
    }
    //Checking waiting list whether there are patients waiting to "Pfizer" Vaccine...
    else if (boothNumber == 4 || boothNumber == 5)
    {
        for (int i = 0; i < waitingList.size(); i++)
        {
            if (waitingList.get(i).getVaccinationRequest() == 3)
            {
                System.out.println();
                System.out.println("From waiting list....");
                boothObjectsArray[boothNumber].addPatientToABooth(waitingList.get(i));
                stocks--;
                waitingList.remove(waitingList.get(i));
                break;
            }
        }
    }
}

private static void viewPatientsSortedInAlphabeticalOrder(){
```

```
while (true) {
    //Making a boolean called "flag" to check whether atleast one booth is not being empty...
    boolean flag = true;
    for (int i = 0; i < boothObjectsArray.length; i++) {
        if (!(boothObjectsArray[i].getPatientObject() == null)) {
            flag = false;
        }
    }
    if (flag) {
        System.out.println("Can't sorted patients names because all the 6 Booths are Empty.");
        break;
    }

    //Making an array called "sortedNames" which contains boothObjectsArray patients names...
    String[] sortedNames = new String[6];
    for (int i = 0; i < sortedNames.length; i++)
    {
        if (!(boothObjectsArray[i].getPatientObject() == null)) {
            sortedNames[i] = boothObjectsArray[i].getPatientObject().getFirstName();
        }
```

```
    else{
        sortedNames[i] = "e";
    }
}

//To hold Temporary Values....
String temporaryString;
for (int i = 0; i < sortedNames.length; i++) {
    for (int j = i + 1; j < sortedNames.length; j++)
    {
        if (sortedNames[i].compareToIgnoreCase(sortedNames[j]) > 0) {
            temporaryString = sortedNames[i];
            sortedNames[i] = sortedNames[j];
            sortedNames[j] = temporaryString;
        }
    }
}

System.out.println("And The Names Sorted in Alphabetical Order Are As Follows...");

for (int i = 0; i < sortedNames.length; i++)
```

```
        {
            if (!sortedNames[i].equals("e")) {
                System.out.println(sortedNames[i]);
            }
        }
        break;
    }
}

private static void storeProgramDataIntoFile(){

    try {
        File patientFileTask4 = new File("patientFileTask4.txt");
        patientFileTask4.createNewFile();

        FileWriter myWriter = new FileWriter("patientFileTask4.txt");
        for (int i = 0; i < boothObjectsArray.length; i++)
        {
            if (!(boothObjectsArray[i].getPatientObject() == null)) {
                myWriter.write(boothObjectsArray[i].getPatientObject().getFirstName() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getSurName() + " ");
```

```
                myWriter.write(boothObjectsArray[i].getPatientObject().getAge() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getCity() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getNICorPassport() + " ");
                myWriter.write(boothObjectsArray[i].getPatientObject().getVaccinationRequest() + "\n");
            }
            else{
                myWriter.write("empty" + "\n");
            }
        }
        myWriter.close();

        System.out.println("Patient Details have been Successfully Stored in \" patientFileTask4 \" TextFile...");

    }
    catch (IOException e){
        System.out.println("An Error has occurred");
    }
}

private static void loadProgramDataFromFile(){
```

```
try{
    File PatientFile2 = new File("patientFileTask4.txt");
    Scanner myReader = new Scanner(PatientFile2);

    while (myReader.hasNextLine())
    {
       for (int i = 0; i < boothObjectsArray.length; i++)
       {
          String data = myReader.nextLine();
          if (data.equals("empty")){
             boothObjectsArray[i].setPatientObject(null);
             continue;
          }

          //Creating an array with all the Patient attribute...
          String[] dataArray = data.split(" ");

          //setting up the attributes in the same place....
          boothObjectsArray[i].setPatientObject(new
Patient(dataArray[0],dataArray[1],Integer.parseInt(dataArray[2]),dataArray[3],dataArray[4],Integer.parse
Int(dataArray[5])));
```

```
            }
        }

        System.out.println("Patient Details have been Successfully loaded from the \" patientFileTask4 \" TextFile...");
    }
    catch(FileNotFoundException e){
        System.out.println("File not found..");
    }

}

private static void viewRemainingVaccinations() {
    System.out.println("The remaining vaccinations are : " + stocks);
}

private static void addVaccinationsToTheStock(){

    System.out.print("How many vaccinations that you gonna add to the Stocks: ");
    int vaccinationnumber = myScanner.nextInt();
```

```
        stocks = stocks + vaccinationnumber;
        System.out.println(vaccinationnumber + " vaccinations has been successfully added to stocks");
    }
}
```

Task05 [JavaFX]

PatientDetails.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 PatientDetails extends Application {
```

```
    @Override
    public void start(Stage primaryStage) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("PatientDetails.fxml"));
        primaryStage.setTitle("Patient Details");
        primaryStage.setScene(new Scene(root, 750, 520));
        primaryStage.show();
    }



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

PatientDetailsController.Java

```
package sample;


import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.TextField;

import javafx.stage.Stage;


import java.util.Date;



public class PatientDetailsController {


    @FXML

    private Button btnGenerateReceipt;
```

```
@FXML
private TextField txtFirstName;
@FXML
private TextField txtSurName;
@FXML
private TextField txtVaccinationType;
@FXML
private TextField txtBoothNumber;



private Parent root;


@FXML
public void receiptWindowAppear(ActionEvent actionEvent) throws Exception{

    String FirstName = txtFirstName.getText();
    String SurName = txtSurName.getText();
    String VaccinationType = txtVaccinationType.getText();
    String BoothNumber = txtBoothNumber.getText();
    String dateString = getDateAndTime().toString();
```

```
        FXMLLoader loader = new FXMLLoader(getClass().getResource("Receipt.fxml"));
        root = loader.load();

        ReceiptController receiptControllerObject = loader.getController();
receiptControllerObject.printDetails(FirstName,SurName,VaccinationType,BoothNumber,dateString);

        //Creating new Stage
        Stage receiptStage = new Stage();
        receiptStage.setTitle("Receipt");

        receiptStage.setScene(new Scene(root,650,550));
        receiptStage.show();



        //Identifying and closing the previous Stage
        Stage previousStage = (Stage) ((Node)actionEvent.getSource()).getScene().getWindow();
        previousStage.close();
```

```
    }


   //To generate Current Time...
    public Date getDateAndTime(){
       Date dateAndTime = new Date();
       return dateAndTime;
    }


}
```

PatientDetails.fxml

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


<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>


<AnchorPane prefHeight="576.0" prefWidth="735.0" xmlns="http://javafx.com/javafx/16"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.PatientDetailsController">

  <children>

    <Label alignment="CENTER" layoutX="239.0" layoutY="14.0" prefHeight="60.0"
prefWidth="254.0" text="Patient Details">

      <font>

        <Font name="SF Atarian System Extended" size="32.0" />

      </font>

    </Label>

    <Label alignment="CENTER_RIGHT" layoutX="209.0" layoutY="102.0" prefHeight="35.0"
prefWidth="123.0" text="First Name :">

      <font>

        <Font name="SF Atarian System Extended" size="18.0" />
```

```
      </font>
    </Label>
    <TextField fx:id="txtFirstName" layoutX="346.0" layoutY="107.0" prefHeight="25.0"
prefWidth="181.0" promptText="Enter First Name" />
    <Label alignment="CENTER_RIGHT" layoutX="209.0" layoutY="149.0" prefHeight="35.0"
prefWidth="123.0" text="Sur Name :">
      <font>
        <Font name="SF Atarian System Extended" size="18.0" />
      </font>
    </Label>
    <TextField fx:id="txtSurName" layoutX="347.0" layoutY="154.0" prefHeight="25.0"
prefWidth="181.0" promptText="Enter Sur Name" />
    <Label alignment="CENTER_RIGHT" layoutX="209.0" layoutY="191.0" prefHeight="35.0"
prefWidth="123.0" text="Age :">
      <font>
        <Font name="SF Atarian System Extended" size="18.0" />
      </font>
    </Label>
    <TextField layoutX="347.0" layoutY="198.0" prefHeight="25.0" prefWidth="181.0"
promptText="Enter the Age" />
    <Label alignment="CENTER_RIGHT" layoutX="210.0" layoutY="238.0" prefHeight="35.0"
prefWidth="123.0" text="City :">
      <font>
```

```
            <Font name="SF Atarian System Extended" size="18.0" />
        </font>
    </Label>
    <TextField layoutX="347.0" layoutY="245.0" prefHeight="25.0" prefWidth="181.0" promptText="Enter the City" />
    <Label alignment="CENTER_RIGHT" layoutX="144.0" layoutY="287.0" prefHeight="35.0" prefWidth="189.0" text="NIC / Passport NO :">
        <font>
            <Font name="SF Atarian System Extended" size="18.0" />
        </font>
    </Label>
    <TextField layoutX="346.0" layoutY="292.0" prefHeight="25.0" prefWidth="181.0" promptText="Enter NIC / Passport NO" />
    <Label alignment="CENTER_RIGHT" layoutX="145.0" layoutY="334.0" prefHeight="35.0" prefWidth="189.0" text="Vaccination Type :">
        <font>
            <Font name="SF Atarian System Extended" size="18.0" />
        </font>
    </Label>
    <TextField fx:id="txtVaccinationType" layoutX="346.0" layoutY="340.0" prefHeight="25.0" prefWidth="181.0" promptText="Enter Vaccination Type" />
    <Label alignment="CENTER_RIGHT" layoutX="144.0" layoutY="381.0" prefHeight="35.0" prefWidth="189.0" text="Booth Number :">
```

```
        <font>
          <Font name="SF Atarian System Extended" size="18.0" />
        </font>
      </Label>
      <TextField fx:id="txtBoothNumber" layoutX="345.0" layoutY="386.0" prefHeight="25.0" prefWidth="181.0" promptText="Enter Booth NO" />
      <Button fx:id="btnGenerateReceipt" alignment="CENTER" layoutX="280.0" layoutY="442.0" mnemonicParsing="false" onAction="#receiptWindowAppear" prefHeight="49.0" prefWidth="172.0" text="Generate Receipt" textFill="#e11111">
        <font>
          <Font name="SF Atarian System Extended" size="16.0" />
        </font>
      </Button>
    </children>
</AnchorPane>
```

ReceiptController.java

```
package sample;


import javafx.fxml.FXML;

import javafx.scene.control.Label;


public class ReceiptController {


    @FXML

    private Label lblName;

    @FXML

    private Label lblVaccinationType;

    @FXML

    private Label lblBoothNumber;

    @FXML

    private Label lblDate;


    @FXML

    public void printDetails(String FirstName,String Surname, String VaccinationType, String
BoothNumber, String dateString){

        lblName.setText(FirstName + " " + Surname);
```

```
        lblVaccinationType.setText(VaccinationType);
        lblBoothNumber.setText(BoothNumber);
        lblDate.setText(dateString);

    }

}
```

Receipt.fxml

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

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

```
<AnchorPane prefHeight="544.0" prefWidth="635.0" xmlns="http://javafx.com/javafx/16"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.ReceiptController">
  <children>
    <Label alignment="CENTER" layoutX="178.0" layoutY="-1.0" prefHeight="66.0"
prefWidth="291.0" text="RECEIPT">
      <font>
        <Font name="SF Atarian System Extended" size="44.0" />
      </font>
    </Label>
    <Label alignment="CENTER_RIGHT" layoutX="134.0" layoutY="108.0" prefHeight="39.0"
prefWidth="137.0" text="Name :">
      <font>
        <Font name="SF Atarian System Extended" size="28.0" />
      </font>
    </Label>
    <Label fx:id="lblName" layoutX="277.0" layoutY="109.0" prefHeight="39.0" prefWidth="306.0"
textFill="#2894ae">
      <font>
        <Font name="SF Atarian System Extended" size="22.0" />
      </font>
    </Label>
```

```
    <Label alignment="CENTER_RIGHT" layoutX="29.0" layoutY="167.0" prefHeight="39.0"
prefWidth="243.0" text="Vaccination Type :">
      <font>
        <Font name="SF Atarian System Extended" size="28.0" />
      </font>
    </Label>
    <Label fx:id="lblVaccinationType" layoutX="282.0" layoutY="167.0" prefHeight="39.0"
prefWidth="306.0" textFill="#277fae">
      <font>
        <Font name="SF Atarian System Extended" size="22.0" />
      </font>
    </Label>
    <Label alignment="CENTER_RIGHT" layoutX="30.0" layoutY="226.0" prefHeight="39.0"
prefWidth="243.0" text="Booth NO :">
      <font>
        <Font name="SF Atarian System Extended" size="28.0" />
      </font>
    </Label>
    <Label fx:id="lblBoothNumber" layoutX="281.0" layoutY="224.0" prefHeight="39.0"
prefWidth="306.0" textFill="#2f8d94">
      <font>
        <Font name="SF Atarian System Extended" size="22.0" />
      </font>
```

```
    </Label>

    <Label fx:id="lblDate" alignment="CENTER" layoutX="88.0" layoutY="431.0" prefHeight="57.0"
prefWidth="476.0" textFill="#27837b">

      <font>

        <Font name="SF Atarian System Extended" size="25.0" />

      </font>

    </Label>

    <Label alignment="CENTER_RIGHT" layoutX="29.0" layoutY="277.0" prefHeight="39.0"
prefWidth="243.0" text="Address :">

      <font>

        <Font name="SF Atarian System Extended" size="28.0" />

      </font>

    </Label>

    <Label fx:id="lblBoothNumber1" layoutX="283.0" layoutY="277.0" prefHeight="39.0"
prefWidth="325.0" text="No.54 Pesalai Road Mannar">

      <font>

        <Font name="SF Atarian System Extended" size="22.0" />

      </font>

    </Label>

    <Label alignment="CENTER_RIGHT" layoutX="30.0" layoutY="326.0" prefHeight="39.0"
prefWidth="243.0" text="Tele NO :">

      <font>

        <Font name="SF Atarian System Extended" size="28.0" />
```

```
        </font>
      </Label>
      <Label fx:id="lblBoothNumber11" alignment="CENTER" layoutX="277.0" layoutY="327.0"
prefHeight="39.0" prefWidth="256.0" text="+94773446687">
        <font>
          <Font name="SF Atarian System Extended" size="22.0" />
        </font>
      </Label>
    </children>
</AnchorPane>
```

<<END>>