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	: 24th July 2021
Student ID	: 20200616 / w1833660
Student First Name	: Saadat
Student Surname	: Hamid Mansoor

"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 : Saadat Hamid Mansoor

Student ID : 20200616 / w1833660

Test Cases

Task 1

No	Test Case	Expected Result	Actual Result	Pass/Fail
1	(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'is Free' for all booths	Displays 'is Free' for all booths	Pass
2	(Add Patient "Speedy" to Booth 5) 102 or APB and enter "Speedy" and then choose Booth 5	Displays "Speedy" for booth 5 (when 100 or VVB)	Displays "Speedy" for booth 5 (when 100 or VVB)	Pass
3	(View All empty Booths) 101 or VEB	Displays ONLY empty booths	Displays ONLY empty booths	Pass
4	(Add Patient 7 to any booth after adding 6 patients) 102 or APB	Displays : warning unable to add more patients!	Displays : warning unable to add more patients!	Pass
5	(Stock count while adding patients) When the stock reaches a count value that is a multiple of 20 102 or APB	Displays: your vaccine stock is at : <value of CVStock>	Displays: your vaccine stock is at : <value of CVStock>	Pass
6	(Stock count while adding patients) When the stock reaches 20 102 or APB	Displays : Warning Covid Vaccine stock Critically Low!!	Displays : Warning Covid Vaccine stock Critically Low!!	Pass
7	(Add patient to already occupied booth) 102 or APB	Displays : this booth is Full try again!	Displays : this booth is Full try again!	Pass
8	(Remove patient from booth) correctly input a booth number of a occupied booth 103 or RPB	Displays : Patient <Name> Has been removed from Booth	Displays : Patient <Name> Has been removed from Booth	Pass
9	(Patient List Alphabetical Sorter) 104 or VPS	Displays Names of patients in alphabetical order	Displays Names of patients in alphabetical order	Pass

10	(Saving data into a file) 105 or SPD	Creates a Corona.txt file and stores vaccine count in first line and all patient names in following lines	Creates a Corona.txt file and stores vaccine count in first line and all patient names in following lines	Pass
11	(Loading the Data back into program) 106 or LPD	Reads the Corona.txt file created in 10 and assigns the values line by line to appropriate places	Reads the Corona.txt file created in 10 and assigns the values line by line to appropriate places	Pass
12	(Loading the Data back into program) Entering 100 or VVB after loading file	Displays the Data loaded	Displays the Data loaded	Pass
13	(View all Vaccines) 107 or VRV	Displays the Vaccine count	Displays the Vaccine count	Pass
14	(Add Vaccine) 107 or VRV after adding vaccine using 108or AVS	Displays Updated Vaccine Count	Displays Updated Vaccine Count	Pass

Task 2

No	Test Case	Expected Result	Actual Result	Pass/Fail
1	(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'is Free' for all booths	Displays 'is Free' for all booths	Pass
2	(Add Patient "Speedy" to Booth 5) 102 or APB and enter "Speedy" and then choose Booth 5	Displays "Speedy" for booth 5 (when 100 or VVB)	Displays "Speedy" for booth 5 (when 100 or VVB)	Pass
3	(View All empty Booths) 101 or VEB	Displays ONLY empty booths	Displays ONLY empty booths	Pass
4	(Add Patient 7 to any booth after adding 6 patients) 102 or APB	Displays : warning unable to add more patients!	Displays : warning unable to add more patients!	Pass

5	(Stock count while adding patients) When the stock reaches a count value that is a multiple of 20 102 or APB	Displays: your vaccine stock is at : <value of CVStock>	Displays: your vaccine stock is at : <value of CVStock>	Pass
6	(Stock count while adding patients) When the stock reaches 20 102 or APB	Displays : Warning Covid Vaccine stock Critically Low!!	Displays : Warning Covid Vaccine stock Critically Low!!	Pass
7	(Add patient to already occupied booth) 102 or APB	Displays : this booth is Full try again!	Displays : this booth is Full try again!	Pass
8	(Remove patient from booth) correctly input a booth number of a occupied booth 103 or RPB	Displays : Patient <Name> Has been removed from Booth	Displays : Patient <Name> Has been removed from Booth	Pass
9	(Patient List Alphabetical Sorter) 104 or VPS	Displays Names of patients in alphabetical order	Displays Names of patients in alphabetical order	Pass
10	(Saving data into a file) 105 or SPD	Creates a Corona.txt file and stores vaccine count in first line and all patient names in following lines	Creates a Corona.txt file and stores vaccine count in first line and all patient names in following lines	Pass
11	(Loading the Data back into program) 106 or LPD	Reads the Corona.txt file created in 10 and assigns the values line by line to appropriate places	Reads the Corona.txt file created in 10 and assigns the values line by line to appropriate places	Pass
12	(Loading the Data back into program) Entering 100 or VVB after loading file	Displays the Data loaded	Displays the Data loaded	Pass
13	(View all Vaccines) 107 or VRV	Displays the Vaccine count	Displays the Vaccine count	Pass
14	(Add Vaccine) 107 or VRV after adding vaccine using 108or AVS	Displays Updated Vaccine Count	Displays Updated Vaccine Count	Pass

Task 3 (Array Version)

No	Test Case	Expected Result	Actual Result	Pass/Fail
1	(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'is Free' for all booths	Displays 'is Free' for all booths	Pass
2	(Add Patient first name and Surname and select a Vaccination type) 102 or APB and enter "Speedy" for first name and "Potato" and 1 for vaccination type	Displays "Speedy Potato" for booth 0 (when 100 or VVB)	Displays "Speedy Potato" for booth 0 (when 100 or VVB)	Pass
3	(View All empty Booths) 101 or VEB	Displays ONLY empty booths	Displays ONLY empty booths	Pass
4	(Add Patient 7 to any booth after adding 6 patients) 102 or APB	Displays : warning unable to add more patients!	Displays : warning unable to add more patients!	Pass
5	(Stock count while adding patients) When the stock reaches a count value that is a multiple of 20 102 or APB	Displays: your vaccine stock is at : <value of CVStock>	Displays: your vaccine stock is at : <value of CVStock>	Pass
6	(Stock count while adding patients) When the stock reaches 20 102 or APB	Displays : Warning Covid Vaccine stock Critically Low!!	Displays : Warning Covid Vaccine stock Critically Low!!	Pass
7	(Add patient to vaccine type that's been taken over by 2 people) 102 or APB	Displays : Sorry there aren't any vacant booths that can administer<Vaccine>	Displays : Sorry there aren't any vacant booths that can administer<Vaccine>	Pass
8	(Remove patient from booth) correctly input a booth number of a occupied booth 103 or RPB	Displays : Patient <Name> Has been removed from Booth	Displays : Patient <Name> Has been removed from Booth	Pass

9	(Patient List Alphabetical Sorter) 104 or VPS	Displays FULL Names of patients in alphabetical order	Displays FULL Names of patients in alphabetical order	Pass
10	(Saving data into a file) 105 or SPD	Creates a Corona.txt file and stores vaccine count in first line and all patient details in following lines	Creates a Corona.txt file and stores vaccine count in first line and all patient details in following lines	Pass
11	(Loading the Data back into program) 106 or LPD	Reads the Corona.txt file created in 10 and assigns the values line by line to appropriate places of each patient	Reads the Corona.txt file created in 10 and assigns the values line by line to appropriate places of each patient	Pass
12	(Loading the Data back into program) Entering 100 or VVB after loading file	Displays the Data loaded	Displays the Data loaded	Pass
13	(View all Vaccines) 107 or VRV	Displays the Vaccine count	Displays the Vaccine count	Pass
14	(Add Vaccine) 107 or VRV after adding vaccine using 108or AVS	Displays Updated Vaccine Count	Displays Updated Vaccine Count	Pass

Task 3 (Classes Version)

No	Test Case	Expected Result	Actual Result	Pass/Fail
1	(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'is Free' for all booths	Displays 'is Free' for all booths	Pass
2	(Add Patient details and vaccine type) 102 or APB	Displays patient name and booth and allows for more details of same patient to be viewed by opting to	Displays patient name and booth and allows for more details of same patient to be viewed	Pass

		see more details (when 100 or VVB)	by opting to see more details (when 100 or VVB)	
3	(View All empty Booths) 101 or VEB	Displays ONLY empty booths	(View All empty Booths) 101 or VEB	Pass
4	(Add Patient 7 to any booth after adding 6 patients) 102 or APB	Displays : warning unable to add more patients!	Displays : warning unable to add more patients!	Pass
5	(Stock count while adding patients) When the stock reaches a count value that is a multiple of 20 102 or APB	Displays: your vaccine stock is at : <value of CVStock>	Displays: your vaccine stock is at : <value of CVStock>	Pass
6	(Stock count while adding patients) When the stock reaches 20 102 or APB	Displays : Warning Covid Vaccine stock Critically Low!!	Displays : Warning Covid Vaccine stock Critically Low!!	Pass
7	(Add patient to vaccine type that's been taken over by 2 people) 102 or APB	Displays : Sorry there aren't any vacant booths that can administer<Vaccine>	Displays : Sorry there aren't any vacant booths that can administer<Vaccine>	Pass
8	(Remove patient from booth) correctly input a booth number of a occupied booth 103 or RPB	Does this : all Patient details Have been removed from Booth	Does this : all Patient details Have been removed from Booth	Pass
9	(Patient List Alphabetical Sorter) 104 or VPS	Displays FULL Names of patients in alphabetical order	Displays FULL Names of patients in alphabetical order	Pass
10	(Saving data into a file) 105 or SPD	Creates a Corona.txt file and stores vaccine count in first line and all patient details in following lines	Creates a Corona.txt file and stores vaccine count in first line and all patient details in following lines	Pass
11	(Loading the Data back into program) 106 or LPD	Reads the Corona.txt file created in 10 and assigns the values	Reads the Corona.txt file created in 10 and assigns the	Pass

		of patient details line by line to appropriate places of each patient	values of patient details line by line to appropriate places of each patient	
12	(Loading the Data back into program) Entering 100 or VVB after loading file	Displays the Data loaded	Displays the Data loaded	Pass
13	(View all Vaccines) 107 or VRV	Displays the Vaccine count	Displays the Vaccine count	Pass
14	(Add Vaccine) 107 or VRV after adding vaccine using 108or AVS	Displays Updated Vaccine Count	Displays Updated Vaccine Count	Pass

Task 4 (Linked Lists)

No	Test Case	Expected Result	Actual Result	Pass/Fail
1	(Booths Initialised correctly) After program starts, 100 or VVB	Displays 'is Free' for all booths	Displays 'is Free' for all booths	Pass
2	(Add Patient details and vaccine type) 102 or APB	Displays patient name and booth and allows for more details of same patient to be viewed by opting to see more details (when 100 or VVB)	Displays patient name and booth and allows for more details of same patient to be viewed by opting to see more details (when 100 or VVB)	Pass
3	(View All empty Booths) 101 or VEB	Displays ONLY empty booths	Displays ONLY empty booths	Pass
4	(Stock count while adding patients) When the stock reaches a count value that is a multiple of 20	Displays: your vaccine stock is at : <value of CVStock>	Displays: your vaccine stock is at : <value of CVStock>	Pass

	102 or APB			
5	(Stock count while adding patients) When the stock reaches 20 102 or APB	Displays : Warning Covid Vaccine stock Critically Low!!	Displays : Warning Covid Vaccine stock Critically Low!!	Pass
6	(Add patient to filled up booths) 102 or APB	Displays : sorry <name> no vacant booths and adds them to the appropriate vaccine waiting list	Displays : sorry <name> no vacant booths and adds them to the appropriate vaccine waiting list	Pass
7	(Remove patient from booth) correctly input a booth number of a occupied booth 103 or RPB	Does this : all Patient details Have been removed from Booth if there is people in waiting list for that vaccine type booth add them into the booth	Does this : all Patient details Have been removed from Booth if there is people in waiting list for that vaccine type booth add them into the booth	Pass
8	(Patient List Alphabetical Sorter) 104 or VPS	Displays FULL Names of patients in alphabetical order	Displays FULL Names of patients in alphabetical order	Pass
9	(Saving data into a file) 105 or SPD	Creates a Corona.txt file and stores vaccine count in first line and all patient details in following lines Creates AstraZeneca.txt, Pfizer.txt and Sinopharm.txt for each type of vaccine waiting list and stores waiting list data	Creates a Corona.txt file and stores vaccine count in first line and all patient details in following lines Creates AstraZeneca.txt, Pfizer.txt and Sinopharm.txt for each type of vaccine waiting list and stores waiting list data	Pass
10	(Loading the Data back into program) 106 or LPD	Reads the Corona.txt file created in 9 and assigns the values of patient details line	Reads the Corona.txt file created in 9 and assigns the values of patient details line by line to	Pass

		by line to appropriate places of each patient Reads the AstraZeneca.txt, Pfizer.txt and Sinopharm.txt for each type of vaccine waiting list and restores the data into the linked lists	appropriate places of each patient Reads the AstraZeneca.txt, Pfizer.txt and Sinopharm.txt for each type of vaccine waiting list and restores the data into the linked lists	
11	(Loading the Data back into program) Entering 100 or VVB after loading file	Displays the Data loaded	Displays the Data loaded	Pass
12	(View all Vaccines) 107 or VRV	Displays the Vaccine count	Displays the Vaccine count	Pass
13	(Add Vaccine) 107 or VRV after adding vaccine using 108or AVS	Displays Updated Vaccine Count	Displays Updated Vaccine Count	Pass
14	Removing patient after loading data from all files	Should automatically restore patients waiting lists and add them to booth on removal and remove 1 vaccine from stock	Should automatically restore patients waiting lists and add them to booth on removal and remove 1 vaccine from stock	Pass

Discussion

While I was starting to code the coursework I started off with the given example code from the course work documentation. I checked the code out and implemented a menu first with “if , else if, else” statements and made sure the validation was good. I then moved on to making each function 1 by 1 and making sure all of them had proper validation so that there would be no loopholes in the code.

In the “view all empty booths” method I made sure to filter out occupied booths.

In the “add patient to booth” method I made sure if all booths are full to return booth full message instead of asking for a “user input” directly this was upgraded in task 4 to add them directly to a waiting list rather than turning them away. I made sure every properly added patient would result in a reduction of vaccine stock.

The “remove patient from booth” method worked on a system where it would show occupied booths and request for a booth number to be emptied. This was slowly upgrade in task 2, 3, and 4 where it finally had the code for waiting list assigning and vaccine stock reduction.

The “Patients sorter method” would always copy out the names/ full names of the main CovidBooth array… and add it to a clone array called CloneCoronaBooth and then I would do a bubble sort code to sort them alphabetically. The use of a clone array helped me avoid messing with any of the original data within the main array

For the “Store program data” method I directly dumped raw data into text files. This was upgraded in task 4 to have 3 extra text files where all patient data of waiting list was dumped and number of patients in waiting list was stored at the top of "vaccine type waiting list".txt file so that the next function can load it without any errors.

For the “Load Program Data” method I simply reversed the logic of the previous method and ensured all the raw data was properly reassigned back into their proper places including the data from waiting lists in part 4. I had to use a lot of validation to ensure the files get loaded safely.

Code :

Task 1:

Task1.java

```
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;

public class Task1{
    public static String[] CovidBooth = new String[6]; //Array that holds Names o
f patients in all 6 booths
    public static int boothNum = 0;
    public static int CVStock = 150;  // Variable to hold Vaccination Stock

    public static void main(String[] args) {

        for (int x = 0; x < 6; x++)  //initializing the primitive array with all
empty values
            CovidBooth[x] = "e";

        PrintMenu();
    }

    public static void PrintMenu() {  // Print Menu method
        Scanner input = new Scanner(System.in);
        Boolean Boothchecker = true;
        while (Boothchecker) {

            System.out.print("\n======================== MENU ==================
==========");
            System.out.print("\n100 or VVB: View all Vaccination Booths");
            System.out.print("\n101 or VEB: View all Empty Booths");
            System.out.print("\n102 or APB: Add Patient to a Booth");
            System.out.print("\n103 or RPB: Remove Patient from a Booth");
            System.out.print("\n104 or VPS: View Patients Sorted in alphabetical
order ");
            System.out.print("\n105 or SPD: Store Program Data into file");
            System.out.print("\n106 or LPD: Load Program Data from file");
            System.out.print("\n107 or VRV: View Remaining Vaccinations");
            System.out.print("\n108 or AVS: Add Vaccinations to the Stock");
            System.out.print("\n999 or EXT: Exit the Program");
```

```
            System.out.print("\n==========================================================
==========");

            System.out.print("\nEnter your Input : ");
            String userInput = input.nextLine();

            if (userInput.equals("100") || userInput.equalsIgnoreCase("VVB")) {
                System.out.print("\nEntered input is either 100 or VVB!\n ");
                ViewVBooth();
            } else if (userInput.equals("101") || userInput.equalsIgnoreCase("VEB
")) {
                System.out.print("\nEntered input is either 101 or VEB!\n");
                ViewEBooth();
            } else if (userInput.equals("102") || userInput.equalsIgnoreCase("APB
")) {
                System.out.print("\nEntered input is either 102 or APB!\n");
                AddP2Booth(input);
            } else if (userInput.equals("103") || userInput.equalsIgnoreCase("RPB
")) {
                System.out.print("\nEntered input is either 103 or RPB!\n");
                WipeP4rmBooth(input);
            } else if (userInput.equals("104") || userInput.equalsIgnoreCase("VPS
")) {
                System.out.print("\nEntered input is either 104 or VPS!\n");
                Sorter();
            } else if (userInput.equals("105") || userInput.equalsIgnoreCase("SPD
")) {
                System.out.print("\nEntered input is either 105 or SPD!\n");
                StorePrgrmData();
            } else if (userInput.equals("106") || userInput.equalsIgnoreCase("LPD
")) {
                System.out.print("\nEntered input is either 106 or LPD!\n");
                LoadPrgrmData();
            } else if (userInput.equals("107") || userInput.equalsIgnoreCase("VRV
")) {
                System.out.print("\nEntered input is either 107 or VRV!\n");
                ViewRCV();
            } else if (userInput.equals("108") || userInput.equalsIgnoreCase("AVS
")) {
                System.out.print("\nEntered input is either 108 or AVS!\n");
                AddCVStock(input);
            } else if (userInput.equals("999") || userInput.equalsIgnoreCase("EXT
")) {
                System.out.print("\nThank you For Trying the Application! \n");
                Boothchecker = false;
```

```
            } else {
                System.out.print("\nThe entered input is Invalid!\n");
            }
        }
    }

    // View all Vaccination Booths method
    public static void ViewVBooth() {
        System.out.print("\nDisplaying all Vaccination Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].equals("e")) {
                System.out.println("Booth Number " + x + " is Free");
            } else {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x]);
            }
        }
    }

    // View all Empty Booths method
    public static void ViewEBooth() {
        System.out.print("\nDisplaying all Empty Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].equals("e"))
                System.out.println("Booth Number " + x + " is Free");
        }
    }

    // Add Patient to a Booth method
    public static void AddP2Booth(Scanner input) {
        int OccupiCount = 0; //Variable to count how many booths are fully occupi
ed
        for (int x = 0; x < 6; x++) {

            if (!CovidBooth[x].equals("e")){
                OccupiCount +=1;
            }
        }
        if (OccupiCount==6){ //condition to say unable to add patients if all 6 b
ooths occupied
            System.out.print("\nWARNING!: unable to add more patients!\nSeems Lik
e all Booths are Full! Try again later when a Booth is Free");
            return;
        }
```

```
        System.out.print("\nEnter patient Name : ");
        String PName = input.nextLine();

        Boolean Boothchecker = true;
        while (Boothchecker) {

            System.out.print("\nDisplaying all Empty Booths: \n");
            for (int x = 0; x < 6; x++) {
                if (CovidBooth[x].equals("e"))
                    System.out.println("Booth Number " + x + " is Free");
            }

            System.out.print("\nWhere Would u like to Assign " + PName + " to : "
);
            int BoothIndex = Integer.parseInt(input.nextLine());

            if (BoothIndex >= CovidBooth.length) {
                System.out.print("\nThe entered Index is Out of Range!: \n");
                continue;
            } else {
                if (CovidBooth[BoothIndex].equals("e")) {
                    CovidBooth[BoothIndex] = PName;
                    Boothchecker = false;
                    System.out.print(PName + " has Been Added to the Booth Number
 " + BoothIndex + "!");
                    CVStock--;
                    if (CVStock%20==0){
                        System.out.print("\nyour Vaccine Stock is at : " + CVStoc
k);           // triggers every multiple of 20
                    }
                    if (CVStock==20){
                        System.out.print("\nWARNING COVID Vaccine Stock Criticall
y Low!!!\n"); // triggers when stock reaches 20
                    }

                } else {
                    System.out.print("\nThis booth is occupied try again!\n");
                    continue;
                }
            }
        }

    }

    // Remove Patient from a Booth method
```

```
    public static void WipeP4rmBooth(Scanner input) {
        System.out.print("\nDisplaying all Occupied Booths: \n");  // Prints all
occupied booths with their booth numbers
        for (int x = 0; x < 6; x++) {
            if (!CovidBooth[x].equals("e")) {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x]);
            }
        }

        Boolean Boothchecker = true;
        while (Boothchecker) {
            System.out.print("\nEnter the booth number of the patient that you wi
sh to remove : ");
            int BoothNum = Integer.parseInt(input.nextLine());
            if (BoothNum >= CovidBooth.length) {
                System.out.print("\nThe entered Index is Out of Range!: \n");
                continue;
            } else {
                System.out.print("\nPatient " + CovidBooth[BoothNum] + " Has been
 Removed From Booth Number " + BoothNum+"\n");
                CovidBooth[BoothNum] = "e";
                Boothchecker = false;
            }
        }
    }

    // View Patients Sorted in alphabetical order method
    public static void Sorter() {

        String[] CloneCoronaBooth = new String[6];       // cloning Booth array a
s a backup to ensure no sorted change is permanant
        for (int x = 0; x < 6; x++) {
            CloneCoronaBooth[x] = CovidBooth[x];


        }

        for (int x = 0; x < 5; x++) {                    //Bubble sort technique
            for (int y = 0; y < 5 - x; y++) {
                if (CloneCoronaBooth[y].toLowerCase().compareTo(CloneCoronaBooth[
y + 1].toLowerCase()) > 0) {

                    String Holder = CloneCoronaBooth[y];
                    CloneCoronaBooth[y] = CloneCoronaBooth[y + 1];
                    CloneCoronaBooth[y + 1] = Holder;
```

```
                }
            }
        }

        System.out.print("\nPatient Names Sorted in Alphabetical Order: \n");
        for (int x = 0; x < 6; x++) {
            if (!CloneCoronaBooth[x].equals("e")) {
                System.out.println(CloneCoronaBooth[x]);
            }
        }
    }

    // Store Program Data into file method
    public static void StorePrgrmData() {
        try (BufferedWriter bkp = new BufferedWriter(new FileWriter("Corona.txt")
)) { // taken from here : https://www.codejava.net/java-se/file-io/how-to-read-
and-write-text-file-in-java
            bkp.write(String.valueOf(CVStock));     // Stores Stock in first line
            bkp.newLine();
            for (int x = 0; x < 6; x++) {
                bkp.write(CovidBooth[x]);
                bkp.newLine();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Dumped all data to Corona.txt!\n");

    }

    // Load Program Data from file method
    public static void LoadPrgrmData() {
        try {
            Scanner Loader = new Scanner(new FileInputStream("Corona.txt")); // t
aken from here : https://www.codejava.net/java-se/file-io/how-to-read-and-write-
text-file-in-java
            CVStock = Integer.parseInt(Loader.nextLine());
            for (int x = 0; x < 6; x++) {
                CovidBooth[x] = Loader.nextLine();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Loaded all data from Corona.txt!\n");
```

```
    }

    // View Remaining Vaccinations method
    public static void ViewRCV() {
        System.out.print("\nTotal Number of Covid Vaccines Left : " + CVStock);
    }

    // Add Vaccinations to the Stock
    public static void AddCVStock(Scanner input) {
        System.out.print("\nEnter Amount of Vaccines to add to stock : ");
        int CVNewStock = Integer.parseInt(input.nextLine());
        CVStock += CVNewStock;
        System.out.print("\nTotal Number of Covid Vaccines has been Updated to :
" + CVStock);
    }

}
```

Task 2:

VaccinationCenter.java

```
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;

public class VacinationCenter {
    public static Booth[] CovidBooth = new Booth[6]; //Array of booth type that h
olds Names of patients in all 6 booths
    public static int boothNum = 0;
    public static int CVStock = 150; // Variable to hold Vaccination Stock

    public static void main(String[] args) {

        for (int x = 0; x < 6; x++)  //initializing the booth type array with new
 empty booths
            CovidBooth[x] = new Booth("e");

        PrintMenu();
    }
```

```
    public static void PrintMenu() { // Print Menu method
        Scanner input = new Scanner(System.in);
        Boolean Boothchecker = true;
        while (Boothchecker) {

            System.out.print("\n======================== MENU ===================
==========");
            System.out.print("\n100 or VVB: View all Vaccination Booths");
            System.out.print("\n101 or VEB: View all Empty Booths");
            System.out.print("\n102 or APB: Add Patient to a Booth");
            System.out.print("\n103 or RPB: Remove Patient from a Booth");
            System.out.print("\n104 or VPS: View Patients Sorted in alphabetical
order ");
            System.out.print("\n105 or SPD: Store Program Data into file");
            System.out.print("\n106 or LPD: Load Program Data from file");
            System.out.print("\n107 or VRV: View Remaining Vaccinations");
            System.out.print("\n108 or AVS: Add Vaccinations to the Stock");
            System.out.print("\n999 or EXT: Exit the Program");
            System.out.print("\n==============================================
==========");

            System.out.print("\nEnter your Input : ");
            String userInput = input.nextLine();

            if (userInput.equals("100") || userInput.equalsIgnoreCase("VVB")) {
                System.out.print("\nEntered input is either 100 or VVB!\n ");
                ViewVBooth();
            } else if (userInput.equals("101") || userInput.equalsIgnoreCase("VEB
")) {
                System.out.print("\nEntered input is either 101 or VEB!\n");
                ViewEBooth();
            } else if (userInput.equals("102") || userInput.equalsIgnoreCase("APB
")) {
                System.out.print("\nEntered input is either 102 or APB!\n");
                AddP2Booth(input);
            } else if (userInput.equals("103") || userInput.equalsIgnoreCase("RPB
")) {
                System.out.print("\nEntered input is either 103 or RPB!\n");
                WipeP4rmBooth(input);
            } else if (userInput.equals("104") || userInput.equalsIgnoreCase("VPS
")) {
                System.out.print("\nEntered input is either 104 or VPS!\n");
                Sorter();
            } else if (userInput.equals("105") || userInput.equalsIgnoreCase("SPD
")) {
```

```
                System.out.print("\nEntered input is either 105 or SPD!\n");
                StorePrgrmData();
            } else if (userInput.equals("106") || userInput.equalsIgnoreCase("LPD
")) {
                System.out.print("\nEntered input is either 106 or LPD!\n");
                LoadPrgrmData();
            } else if (userInput.equals("107") || userInput.equalsIgnoreCase("VRV
")) {
                System.out.print("\nEntered input is either 107 or VRV!\n");
                ViewRCV();
            } else if (userInput.equals("108") || userInput.equalsIgnoreCase("AVS
")) {
                System.out.print("\nEntered input is either 108 or AVS!\n");
                AddCVStock(input);
            } else if (userInput.equals("999") || userInput.equalsIgnoreCase("EXT
")) {
                System.out.print("\nThank you For Trying the Application! \n");
                Boothchecker = false;
            } else {
                System.out.print("\nThe entered input is Invalid!\n");
            }
        }
    }

    // View all Vaccination Booths method
    public static void ViewVBooth() {
        System.out.print("\nDisplaying all Vaccination Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].getPatientName().equals("e")) {
                System.out.println("Booth Number " + x + " is Free");
            } else {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x].getPatientName());
            }
        }
    }

    // View all Empty Booths method
    public static void ViewEBooth() {
        System.out.print("\nDisplaying all Empty Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].getPatientName().equals("e"))
                System.out.println("Booth Number " + x + " is Free");
        }
    }
```

```
    // Add Patient to a Booth method
    public static void AddP2Booth(Scanner input) {
        int OccupiCount = 0; //Variable to count how many booths are fully occupi
ed
        for (int x = 0; x < 6; x++) {

            if (!CovidBooth[x].getPatientName().equals("e")) {
                OccupiCount += 1;
            }
        }
        if (OccupiCount == 6) { //condition to say unable to add patients if all
6 booths occupied
            System.out.print(
                    "\nWARNING!: unable to add more patients!\nSeems Like all Boo
ths are Full! Try again later when a Booth is Free");
            return;
        }

        System.out.print("\nEnter patient Name : ");
        Booth PName = new Booth(input.nextLine()); // input Name in Variable of t
he Booth type

        Boolean Boothchecker = true;
        while (Boothchecker) {
            System.out.print("\nDisplaying all Empty Booths: \n");
            for (int x = 0; x < 6; x++) {
                if (CovidBooth[x].getPatientName().equals("e"))
                    System.out.println("Booth Number " + x + " is Free");
            }
            System.out.print("\nWhere Would u like to Assign " + PName.getPatient
Name() + " to : ");
            int BoothIndex = Integer.parseInt(input.nextLine());
            if (BoothIndex >= CovidBooth.length) {
                System.out.print("\nThe entered Index is Out of Range!: \n");
                continue;
            } else {
                if (CovidBooth[BoothIndex].getPatientName().equals("e")) {
                    CovidBooth[BoothIndex] = PName;
                    Boothchecker = false;
                    System.out
                            .print(PName.getPatientName() + " has Been Added to t
he Booth Number " + BoothIndex + "!");
                    CVStock--;
                    if (CVStock % 20 == 0) {
```

```
                    System.out.print("\nyour Vaccine Stock is at : " + CVStoc
k);           // triggers every multiple of 20
                    }
                    if (CVStock == 20) {
                        System.out.print("\nWARNING COVID Vaccine Stock Criticall
y Low!!!\n"); // triggers when stock reaches 20
                    }

                } else {
                    System.out.print("\nThis booth is occupied try again!\n");
                    continue;
                }
            }
        }

    }

    // Remove Patient from a Booth method
    public static void WipeP4rmBooth(Scanner input) {
        System.out.print("\nDisplaying all Occupied Booths: \n"); // Prints all o
ccupied booths with their booth numbers
        for (int x = 0; x < 6; x++) {
            if (!CovidBooth[x].getPatientName().equals("e")) {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x].getPatientName());
            }
        }

        Boolean Boothchecker = true;
        while (Boothchecker) {
            System.out.print("\nEnter the booth number of the patient that you wi
sh to remove : ");
            int BoothNum = Integer.parseInt(input.nextLine());
            if (BoothNum >= CovidBooth.length) {
                System.out.print("\nThe entered Index is Out of Range!: \n");
                continue;
            } else {
                System.out.print("\nPatient " + CovidBooth[BoothNum].getPatientNa
me() + " Has been Removed From Booth Number " + BoothNum + "\n");
                CovidBooth[BoothNum].setPatientName("e");
                Boothchecker = false;
            }
        }
    }
```

```
    // View Patients Sorted in alphabetical order method
    public static void Sorter() {

        String[] CloneCoronaBooth = new String[6];  // cloning Booth array as a b
ackup to ensure no sorted change is permanant
        for (int x = 0; x < 6; x++) {
            CloneCoronaBooth[x] = CovidBooth[x].getPatientName();

        }
        for (int x = 0; x < 5; x++) {               //Bubble sort technique
            for (int y = 0; y < 5 - x; y++) {
                if (CloneCoronaBooth[y].toLowerCase().compareTo(CloneCoronaBooth[
y + 1].toLowerCase()) > 0) {

                    String Holder = CloneCoronaBooth[y];
                    CloneCoronaBooth[y] = CloneCoronaBooth[y + 1];
                    CloneCoronaBooth[y + 1] = Holder;
                }
            }
        }
        System.out.print("\nPatient Names Sorted in Alphabetical Order: \n");
        for (int x = 0; x < 6; x++) {
            if (!CloneCoronaBooth[x].equals("e")) {
                System.out.println(CloneCoronaBooth[x]);
            }
        }

    }

    // Store Program Data into file method
    public static void StorePrgrmData() {
        try (BufferedWriter bkp = new BufferedWriter(new FileWriter("Corona.txt")
)) { // taken from here : https://www.codejava.net/java-se/file-io/how-to-read-
and-write-text-file-in-java
            bkp.write(String.valueOf(CVStock));    // Stores Stock in first line
            bkp.newLine();
            for (int x = 0; x < 6; x++) {
                bkp.write(CovidBooth[x].getPatientName());
                bkp.newLine();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Dumped all data to Corona.txt!\n");
```

```
    }

    // Load Program Data from file method
    public static void LoadPrgrmData() {
        try {
            Scanner Loader = new Scanner(new FileInputStream("Corona.txt")); // t
aken from here : https://www.codejava.net/java-se/file-io/how-to-read-and-write-
text-file-in-java
            CVStock = Integer.parseInt(Loader.nextLine());
            for (int x = 0; x < 6; x++) {
                CovidBooth[x].setPatientName(Loader.nextLine());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Loaded all data from Corona.txt!\n");
    }

    // View Remaining Vaccinations method
    public static void ViewRCV() {
        System.out.print("\nTotal Number of Covid Vaccines Left : " + CVStock);
    }

    // Add Vaccinations to the Stock
    public static void AddCVStock(Scanner input) {
        System.out.print("\nEnter Amount of Vaccines to add to stock : ");
        int CVNewStock = Integer.parseInt(input.nextLine());
        CVStock += CVNewStock;
        System.out.print("\nTotal Number of Covid Vaccines has been Updated to :
" + CVStock);
    }

}
```

Booth.java

```
public class Booth {
    private String patientName; // Primitive type Variable inside booth class

    public String getPatientName() { // Getter
        return patientName;
    }

    public void setPatientName(String x) { //Setter
        patientName = x;
```

```
    }

    public Booth(String PN) { // Constructor
        patientName = PN;

    }
}
```

Task 3 (Array Version):

Task3_1.java

```
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;

public class Task3_1 {
    public static String[] CovidBooth = new String[6];   //Array that holds full
Names (first and 2nd ) of patients in all 6 booths
    public static String[] PatientFName = new String[6]; //Array that holds FIRST
 Names of patients in all 6 booths
    public static String[] PatientSName = new String[6]; //Array that holds Surna
mes of patients in all 6 booths
    public static String[] VaccineType = new String[6];  // Array that holds the
vaccine type adminsterd in all 6 booths
    public static int boothNum = 0;
    public static int CVStock = 150;                     // Variable to hold Vacc
ination Stock

    public static void main(String[] args) {

        for (int x = 0; x < 6; x++) { //initializing the primitive arrays with al
l empty values
            CovidBooth[x] = "e";
            PatientFName[x] = "e";
            PatientSName[x] = "e";
            VaccineType[x] = "e";
        }

        PrintMenu();
    }
```

```
    public static void PrintMenu() { // Print Menu method
        Scanner input = new Scanner(System.in);
        Boolean Boothchecker = true;
        while (Boothchecker) {

            System.out.print("\n======================== MENU ===================
==========");
            System.out.print("\n100 or VVB: View all Vaccination Booths");
            System.out.print("\n101 or VEB: View all Empty Booths");
            System.out.print("\n102 or APB: Add Patient to a Booth");
            System.out.print("\n103 or RPB: Remove Patient from a Booth");
            System.out.print("\n104 or VPS: View Patients Sorted in alphabetical
order ");
            System.out.print("\n105 or SPD: Store Program Data into file");
            System.out.print("\n106 or LPD: Load Program Data from file");
            System.out.print("\n107 or VRV: View Remaining Vaccinations");
            System.out.print("\n108 or AVS: Add Vaccinations to the Stock");
            System.out.print("\n999 or EXT: Exit the Program");
            System.out.print("\n==============================================
==========");

            System.out.print("\nEnter your Input : ");
            String userInput = input.nextLine();

            if (userInput.equals("100") || userInput.equalsIgnoreCase("VVB")) {
                System.out.print("\nEntered input is either 100 or VVB!\n ");
                ViewVBooth();
            } else if (userInput.equals("101") || userInput.equalsIgnoreCase("VEB
")) {
                System.out.print("\nEntered input is either 101 or VEB!\n");
                ViewEBooth();
            } else if (userInput.equals("102") || userInput.equalsIgnoreCase("APB
")) {
                System.out.print("\nEntered input is either 102 or APB!\n");
                AddP2Booth(input);
            } else if (userInput.equals("103") || userInput.equalsIgnoreCase("RPB
")) {
                System.out.print("\nEntered input is either 103 or RPB!\n");
                WipeP4rmBooth(input);
            } else if (userInput.equals("104") || userInput.equalsIgnoreCase("VPS
")) {
                System.out.print("\nEntered input is either 104 or VPS!\n");
                Sorter();
            } else if (userInput.equals("105") || userInput.equalsIgnoreCase("SPD
")) {
```

```
                System.out.print("\nEntered input is either 105 or SPD!\n");
                StorePrgrmData();
            } else if (userInput.equals("106") || userInput.equalsIgnoreCase("LPD
")) {
                System.out.print("\nEntered input is either 106 or LPD!\n");
                LoadPrgrmData();
            } else if (userInput.equals("107") || userInput.equalsIgnoreCase("VRV
")) {
                System.out.print("\nEntered input is either 107 or VRV!\n");
                ViewRCV();
            } else if (userInput.equals("108") || userInput.equalsIgnoreCase("AVS
")) {
                System.out.print("\nEntered input is either 108 or AVS!\n");
                AddCVStock(input);
            } else if (userInput.equals("999") || userInput.equalsIgnoreCase("EXT
")) {
                System.out.print("\nThank you For Trying the Application! \n");
                Boothchecker = false;
            } else {
                System.out.print("\nThe entered input is Invalid!\n");
            }
        }
    }

     // View all Vaccination Booths method
    public static void ViewVBooth() {
        System.out.print("\nDisplaying all Vaccination Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].equals("e")) {
                System.out.println("Booth Number " + x + " is Free\n");
            } else {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x]);
                System.out.println("          | Vaccination type : " + VaccineTyp
e[x] + "\n");
            }
        }
    }

    // View all Empty Booths method
    public static void ViewEBooth() {
        System.out.print("\nDisplaying all Empty Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].equals("e"))
                System.out.println("Booth Number " + x + " is Free");
```

```
        }
    }

    // Add Patient to a Booth method
    public static void AddP2Booth(Scanner input) {
        int OccupiCount = 0; //Variable to count how many booths are fully occupi
ed
        for (int x = 0; x < 6; x++) {

            if (!CovidBooth[x].equals("e")){
                OccupiCount +=1;
            }
        }
        if (OccupiCount==6){ //condition to say unable to add patients if all 6 b
ooths occupied
            System.out.print("\nWARNING!: unable to add more patients!\nSeems Lik
e all Booths are Full! Try again later when a Booth is Free");
            return;
        }

        System.out.print("\nEnter patient First Name : ");
        String PName = input.nextLine();

        System.out.print("\nEnter patient Surname: ");
        String SName = input.nextLine();

        Boolean VaccineChecker = true;
        while(VaccineChecker){ //while loop to get vaccine type and auto assign a
 booth accordingly
            System.out.print("\n==================== Vaccine type ==============
=====");
            System.out.print("\nEnter 1 for : AstraZeneca");
            System.out.print("\nEnter 2 for : Sinopharm");
            System.out.print("\nEnter 3 For : Pfizer");
            System.out.print("\n================================================
=====\n");

            System.out.print("\nEnter your Input : ");
            String UserVaccine = input.nextLine();

            if (UserVaccine.equals("1")) {

                if (CovidBooth[0].equals("e")){
                    System.out.print("\nYou have Requested for AstraZeneca!\n");
                    VaccineType[0] = "AstraZeneca";
```

```
                    CovidBooth[0] = PName + " " + SName;
                    PatientFName[0] = PName;
                    PatientSName[0] = SName;
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 0 and will be Adminsterd the AstraZeneca vaccine!\n");
                }
                else if(CovidBooth[1].equals("e")){

                    System.out.print("\nYou have Requested for AstraZeneca!\n");
                    VaccineType[1] = "AstraZeneca";
                    CovidBooth[1] = PName + " " + SName;
                    PatientFName[1] = PName;
                    PatientSName[1] = SName;
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 1 and will be Adminsterd the AstraZeneca vaccine!\n");
                }else {
                    System.out.print("Sorry there aren't any vacant booths that c
an administer AstraZeneca\n");
                    continue;
                }
                VaccineChecker = false;
            } else if (UserVaccine.equals("2")) {
                if (CovidBooth[2].equals("e")){
                    System.out.print("\nYou have Requested for Sinopharm! \n");
                    VaccineType[2] = "Sinopharm";
                    CovidBooth[2] = PName + " " + SName;
                    PatientFName[2] = PName;
                    PatientSName[2] = SName;
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 2 and will be Adminsterd the Sinopharm vaccine!\n");
                }
                else if(CovidBooth[3].equals("e")){
                    System.out.print("\nYou have Requested for Sinopharm! \n");
                    VaccineType[3] = "Sinopharm";
                    CovidBooth[3] = PName + " " + SName;
                    PatientFName[3] = PName;
                    PatientSName[3] = SName;
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 3 and will be Adminsterd the Sinopharm vaccine!\n");
                }else {
                    System.out.print("Sorry there aren't any vacant booths that c
an administer Sinopharm\n");
                    continue;
                }
                VaccineChecker = false;
```

```
            } else if (UserVaccine.equals("3")) {
                if (CovidBooth[4].equals("e")){
                    System.out.print("\nYou have Requested for Pfizer! \n");
                    VaccineType[4] = "Pfizer";
                    CovidBooth[4] = PName + " " + SName;
                    PatientFName[4] = PName;
                    PatientSName[4] = SName;
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 4 and will be Adminsterd the Pfizer vaccine!\n");
                }
                else if(CovidBooth[5].equals("e")){
                    System.out.print("\nYou have Requested for Pfizer! \n");
                    VaccineType[5] = "Pfizer";
                    CovidBooth[5] = PName + " " + SName;
                    PatientFName[5] = PName;
                    PatientSName[5] = SName;
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 5 and will be Adminsterd the Pfizer vaccine!\n");
                }else {
                    System.out.print("Sorry there aren't any vacant booths that c
an administer Pfizer\n");
                    continue;
                }
                VaccineChecker = false;
            } else {
                System.out.print("\nThe entered input is Invalid! try again\n");
            }
        }
        CVStock--;
        if (CVStock%20==0){
            System.out.print("\nyour Vaccine Stock is at : " + CVStock);
   // triggers every multiple of 20
        }
        if (CVStock==20){
            System.out.print("\nWARNING COVID Vaccine Stock Critically Low!!!\n")
;  // triggers when stock reaches 20
        }

    }

    // Remove Patient from a Booth method
    public static void WipeP4rmBooth(Scanner input) {
        System.out.print("\nDisplaying all Occupied Booths: \n");   // Prints all
 occupied booths with their booth numbers
        for (int x = 0; x < 6; x++) {
```

```
            if (!CovidBooth[x].equals("e")) {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x]);
            }
        }

        Boolean Boothchecker = true;
        while (Boothchecker) {
            System.out.print("\nEnter the booth number of the patient that you wi
sh to remove : ");
            int BoothNum = Integer.parseInt(input.nextLine());
            if (BoothNum >= CovidBooth.length) {
                System.out.print("\nThe entered Index is Out of Range!: \n");
                continue;
            } else {
                System.out.print(
                        "\nPatient " + CovidBooth[BoothNum] + " Has been Removed
From Booth Number " + BoothNum + "\n");
                CovidBooth[BoothNum] = "e";
                PatientFName[BoothNum] = "e";
                PatientSName[BoothNum] = "e";
                VaccineType[BoothNum] = "e";
                Boothchecker = false;
            }
        }
    }

    // View Patients Sorted in alphabetical order method
    public static void Sorter() {

        String[] CloneCoronaBooth = new String[6];     // cloning Booth array as
a backup to ensure no sorted change is permanant
        for (int x = 0; x < 6; x++) {
            CloneCoronaBooth[x] = CovidBooth[x];

        }
        for (int x = 0; x < 5; x++) {               //Bubble sort technique
            for (int y = 0; y < 5 - x; y++) {
                if (CloneCoronaBooth[y].toLowerCase().compareTo(CloneCoronaBooth[
y + 1].toLowerCase()) > 0) {

                    String Holder = CloneCoronaBooth[y];
                    CloneCoronaBooth[y] = CloneCoronaBooth[y + 1];
                    CloneCoronaBooth[y + 1] = Holder;
                }
```

```
            }
        }
        System.out.print("\nPatient Names Sorted in Alphabetical Order: \n");
        for (int x = 0; x < 6; x++) {
            if (!CloneCoronaBooth[x].equals("e")) {
                System.out.println(CloneCoronaBooth[x]);
            }
        }

    }

    // Store Program Data into file method
    public static void StorePrgrmData() {
        try (BufferedWriter bkp = new BufferedWriter(new FileWriter("Corona.txt")
)) { // taken from here : https://www.codejava.net/java-se/file-io/how-to-read-
and-write-text-file-in-java
            bkp.write(String.valueOf(CVStock));    // Stores Stock in first line
            bkp.newLine();
            for (int x = 0; x < 6; x++) {
                bkp.write(CovidBooth[x]);
                bkp.newLine();
                bkp.write(PatientFName[x]);
                bkp.newLine();
                bkp.write(PatientSName[x]);
                bkp.newLine();
                bkp.write(VaccineType[x]);
                bkp.newLine();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Dumped all data to Corona.txt!\n");

    }

    // Load Program Data from file method
    public static void LoadPrgrmData() {
        try {
            Scanner Loader = new Scanner(new FileInputStream("Corona.txt")); // t
aken from here : https://www.codejava.net/java-se/file-io/how-to-read-and-write-
text-file-in-java
            CVStock = Integer.parseInt(Loader.nextLine());
            for (int x = 0; x < 6; x++) {
                CovidBooth[x] = Loader.nextLine();
```

```
                PatientFName[x] = Loader.nextLine();
                PatientSName[x] = Loader.nextLine();
                VaccineType[x] = Loader.nextLine();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Loaded all data from Corona.txt!\n");
    }

    // View Remaining Vaccinations method
    public static void ViewRCV() {
        System.out.print("\nTotal Number of Covid Vaccines Left : " + CVStock);
    }

    // Add Vaccinations to the Stock
    public static void AddCVStock(Scanner input) {
        System.out.print("\nEnter Amount of Vaccines to add to stock : ");
        int CVNewStock = Integer.parseInt(input.nextLine());
        CVStock += CVNewStock;
        System.out.print("\nTotal Number of Covid Vaccines has been Updated to :
" + CVStock);
    }

}
```

Task 3 (Class Version):

VacinationCenter.java

```
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;

public class VacinationCenter {
    public static Booth[] CovidBooth = new Booth[6]; //Array of booth type that h
olds Names of patients in all 6 booths
    public static int boothNum = 0;
    public static int CVStock = 150; // Variable to hold Vaccination Stock
```

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

        for (int x = 0; x < 6; x++)  //initializing the booth type array with new
 Patients per Booth with all data set to empty
            CovidBooth[x] = new Booth(new Patient("e", "e", 0, "e", "e", "e"));

        PrintMenu();
    }

    public static void PrintMenu() { // Print Menu method
        Scanner input = new Scanner(System.in);
        Boolean Boothchecker = true;
        while (Boothchecker) {

            System.out.print("\n======================== MENU ===================
==========");
            System.out.print("\n100 or VVB: View all Vaccination Booths");
            System.out.print("\n101 or VEB: View all Empty Booths");
            System.out.print("\n102 or APB: Add Patient to a Booth");
            System.out.print("\n103 or RPB: Remove Patient from a Booth");
            System.out.print("\n104 or VPS: View Patients Sorted in alphabetical
order ");
            System.out.print("\n105 or SPD: Store Program Data into file");
            System.out.print("\n106 or LPD: Load Program Data from file");
            System.out.print("\n107 or VRV: View Remaining Vaccinations");
            System.out.print("\n108 or AVS: Add Vaccinations to the Stock");
            System.out.print("\n999 or EXT: Exit the Program");
            System.out.print("\n================================================
==========");

            System.out.print("\nEnter your Input : ");
            String userInput = input.nextLine();

            if (userInput.equals("100") || userInput.equalsIgnoreCase("VVB")) {
                System.out.print("\nEntered input is either 100 or VVB!\n ");
                ViewVBooth(input);
            } else if (userInput.equals("101") || userInput.equalsIgnoreCase("VEB
")) {
                System.out.print("\nEntered input is either 101 or VEB!\n");
                ViewEBooth();
            } else if (userInput.equals("102") || userInput.equalsIgnoreCase("APB
")) {
                System.out.print("\nEntered input is either 102 or APB!\n");
                AddP2Booth(input);
```

```
            } else if (userInput.equals("103") || userInput.equalsIgnoreCase("RPB
")) {
                System.out.print("\nEntered input is either 103 or RPB!\n");
                WipeP4rmBooth(input);
            } else if (userInput.equals("104") || userInput.equalsIgnoreCase("VPS
")) {
                System.out.print("\nEntered input is either 104 or VPS!\n");
                Sorter();
            } else if (userInput.equals("105") || userInput.equalsIgnoreCase("SPD
")) {
                System.out.print("\nEntered input is either 105 or SPD!\n");
                StorePrgrmData();
            } else if (userInput.equals("106") || userInput.equalsIgnoreCase("LPD
")) {
                System.out.print("\nEntered input is either 106 or LPD!\n");
                LoadPrgrmData();
            } else if (userInput.equals("107") || userInput.equalsIgnoreCase("VRV
")) {
                System.out.print("\nEntered input is either 107 or VRV!\n");
                ViewRCV();
            } else if (userInput.equals("108") || userInput.equalsIgnoreCase("AVS
")) {
                System.out.print("\nEntered input is either 108 or AVS!\n");
                AddCVStock(input);
            } else if (userInput.equals("999") || userInput.equalsIgnoreCase("EXT
")) {
                System.out.print("\nThank you For Trying the Application! \n");
                Boothchecker = false;
            } else {
                System.out.print("\nThe entered input is Invalid!\n");
            }
        }
    }

    // View all Vaccination Booths method
    public static void ViewVBooth(Scanner input) {
        int OccupiCount = 0;
        for (int x = 0; x < 6; x++) {

            if (!CovidBooth[x].getPatientX().getmeFST().equals("e")) {
                OccupiCount += 1;
            }
        }
        showAllPatients();
        if (OccupiCount >0) {
```

```
            Boolean Pdloop = true;
            while (Pdloop){
                System.out.print("\nEnter a booth number to view more details abo
ut a patient \nOR Enter 99 to go back to the main menu: ");
                int PUInput = Integer.parseInt(input.nextLine());
                if (PUInput >= 0 && PUInput<6){
                    if(!CovidBooth[PUInput].getPatientX().getmeFST().equals("e"))
{
                        displayPatient(PUInput);
                        System.out.println("\nWould you like to check another Pat
ient ? \n");
                        showAllPatients();
                        continue;
                    }else{
                        System.out.println("\nSorry booth number " + PUInput + "
is Vacant! and therefore doesnt have any patient details to display!");
                        continue;
                    }
                }else if (PUInput == 99){
                    System.out.println("\nReturning to back to menu");
                    Pdloop= false;
                }else{
                    System.out.println("\nInvalid Input Enterd! Try again pwease
^_^\n");
                    continue;
                }
            }

        }

    }

    public static void showAllPatients(){ //displays all booths wether they are f
ree or occupied
        System.out.print("\nDisplaying all Vaccination Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].getPatientX().getmeFST().equals("e")) {
                System.out.println("Booth Number " + x + " is Free");
            } else {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x].getPatientX().getmeFST() + " " + CovidBooth[x].getPatientX().getmeSUR
());
            }
        }
    }
```

```
    public static void displayPatient(int x){ // displays all details of a patien
t of requested index method to be called in the ViewVBooth() method
        System.out.println("=================== Booth "+x+" ==================")
;
        System.out.println("First Name              : "+CovidBooth[x].getPatientX
().getmeFST());
        System.out.println("Surname                 : "+CovidBooth[x].getPatientX
().getmeSUR());
        System.out.println("Age                     : "+CovidBooth[x].getPatientX
().getmePA());
        System.out.println("City                    : "+CovidBooth[x].getPatientX
().getmePee());
        System.out.println("Nic / Passport Number   : "+CovidBooth[x].getPatientX
().getmePNICPN());
        System.out.println("Vaccination type        : "+CovidBooth[x].getPatientX
().getmePVacc());
        System.out.println("================================================");
    }

    // View all Empty Booths method
    public static void ViewEBooth() {
        System.out.print("\nDisplaying all Empty Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].getPatientX().getmeFST().equals("e"))
                System.out.println("Booth Number " + x + " is Free");
        }
    }

    // Add Patient to a Booth method
    public static void AddP2Booth(Scanner input) {
        int OccupiCount = 0; //Variable to count how many booths are fully occupi
ed
        for (int x = 0; x < 6; x++) {

            if (!CovidBooth[x].getPatientX().getmeFST().equals("e")) {
                OccupiCount += 1;
            }
        }
        if (OccupiCount == 6) { //condition to say unable to add patients if all
6 booths occupied
            System.out.print("\nWARNING!: unable to add more patients!\nSeems Lik
e all Booths are Full! Try again later when a Booth is Free");
            return;
        }
```

```
        System.out.print("\nEnter patient First Name : ");
        String PName = input.nextLine();
        System.out.print("\nEnter patient Surname : ");
        String SName = input.nextLine();
        System.out.print("\nEnter patient Age : ");
        int Age = Integer.parseInt(input.nextLine());
        System.out.print("\nEnter patient City : ");
        String City = input.nextLine();
        System.out.print("\nEnter patient NIC/Passport Number : ");
        String NIC = input.nextLine();

        Boolean Boothchecker = true;
        while (Boothchecker) {          // get vaccine type and auto assign patien
ts to proper booths
            System.out.print("\n==================== Vaccine type ==============
=====");
            System.out.print("\nEnter 1 for : AstraZeneca");
            System.out.print("\nEnter 2 for : Sinopharm");
            System.out.print("\nEnter 3 For : Pfizer");
            System.out.print("\n================================================
=====\n");

            System.out.print("\nEnter your Input : ");
            String VType = input.nextLine();

            if (VType.equals("1")){
                if (CovidBooth[0].getPatientX().getmeFST().equals("e")){
                    CovidBooth[0].setPatientX(new Patient(PName, SName, Age, City
, NIC, "AstraZeneca"));
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 0 and will be Adminsterd the AstraZeneca vaccine!\n");
                    Boothchecker = false;
                }else if (CovidBooth[1].getPatientX().getmeFST().equals("e")){
                        CovidBooth[1].setPatientX(new Patient(PName, SName, Age
, City, NIC, "AstraZeneca"));
                        System.out.print(PName+ " "+ SName + " has Been Added t
o the Booth Number 1 and will be Adminsterd the AstraZeneca vaccine!\n");
                        Boothchecker = false;
                }else{
                    System.out.print("Sorry there aren't any vacant booths that c
an administer AstraZeneca Try Another Vaccine Maybe ?\n");
                    continue;
                }
            }else if (VType.equals("2")){
```

```
                if(CovidBooth[2].getPatientX().getmeFST().equals("e")){
                    CovidBooth[2].setPatientX(new Patient(PName, SName, Age, City,
 NIC, "Sinopharm"));
                    System.out.print(PName+ " "+ SName + " has Been Added to the B
ooth Number 2 and will be Adminsterd the Sinopharm vaccine!\n");
                    Boothchecker=false;
                }else if (CovidBooth[3].getPatientX().getmeFST().equals("e")){
                            CovidBooth[3].setPatientX(new Patient(PName, SName, Age
, City, NIC, "Sinopharm"));
                            System.out.print(PName+ " "+ SName + " has Been Added t
o the Booth Number 3 and will be Adminsterd the Sinopharm vaccine!\n");
                            Boothchecker=false;
                }else{
                    System.out.print("Sorry there aren't any vacant booths that c
an administer Sinopharm Try Another Vaccine Maybe ?\n");
                    continue;
                }
            }else if (VType.equals("3")){
                if (CovidBooth[4].getPatientX().getmeFST().equals("e")){
                    CovidBooth[4].setPatientX(new Patient(PName, SName, Age, City
, NIC, "Pfizer"));
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 4 and will be Adminsterd the Pfizer vaccine!\n");
                    Boothchecker=false;
                }else if (CovidBooth[5].getPatientX().getmeFST().equals("e")){
                            CovidBooth[5].setPatientX(new Patient(PName, SName, Age
, City, NIC, "Pfizer"));
                            System.out.print(PName+ " "+ SName + " has Been Added t
o the Booth Number 5 and will be Adminsterd the Pfizer vaccine!\n");
                            Boothchecker=false;
                }else {
                    System.out.print("Sorry there aren't any vacant booths that c
an administer Pfizer Try Another Vaccine Maybe ?\n");
                    continue;
                }
            }else {
                System.out.print("Pls enter a Valid Vaccine Type Input");
            }
        }
        CVStock--; // reduce stock here
        if (CVStock % 20 == 0) {
            System.out.print("\nyour Vaccine Stock is at : " + CVStock);
    // triggers every multiple of 20
        }
        if (CVStock == 20) {
```

```
            System.out.print("\nWARNING COVID Vaccine Stock Critically Low!!!\n")
;   // triggers when stock reaches 20
        }
    }

    // Remove Patient from a Booth method
    public static void WipeP4rmBooth(Scanner input) {
        System.out.print("\nDisplaying all Occupied Booths: \n");  // Prints all
occupied booths with their booth numbers
        for (int x = 0; x < 6; x++) {
            if (!CovidBooth[x].getPatientX().getmeFST().equals("e")) {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x].getPatientX().getmeFST()+ " " + CovidBooth[x].getPatientX().getmeSUR(
));
            }
        }

        Boolean Boothchecker = true;
        while (Boothchecker) {
            System.out.print("\nEnter the booth number of the patient that you wi
sh to remove : ");
            int BoothNum = Integer.parseInt(input.nextLine());
            if (BoothNum >= CovidBooth.length) {
                System.out.print("\nThe entered Index is Out of Range!: \n");
                continue;
            } else {
                System.out.print("\nPatient " + CovidBooth[BoothNum].getPatientX(
).getmeFST()+ " " + CovidBooth[BoothNum].getPatientX().getmeSUR() + " Has been Re
moved From Booth Number " + BoothNum + "\n");
                CovidBooth[BoothNum].setPatientX(new Patient("e", "e", 0, "e", "e
", "e"));
                Boothchecker = false;
            }
        }
    }

    // View Patients Sorted in alphabetical order method
    public static void Sorter() {

        String[] CloneCoronaBooth = new String[6];  // cloning Booth array as a b
ackup to ensure no sorted change is permanant
        for (int x = 0; x < 6; x++) {
            CloneCoronaBooth[x] = CovidBooth[x].getPatientX().getmeFST() +" "+ Co
vidBooth[x].getPatientX().getmeSUR(); // stores names in the format "(First Name)
 (Surname)"
```

```
        }
        for (int x = 0; x < 5; x++) {                //Bubble sort technique
            for (int y = 0; y < 5 - x; y++) {
                if (CloneCoronaBooth[y].toLowerCase().compareTo(CloneCoronaBooth[
y + 1].toLowerCase()) > 0) {

                    String Holder = CloneCoronaBooth[y];
                    CloneCoronaBooth[y] = CloneCoronaBooth[y + 1];
                    CloneCoronaBooth[y + 1] = Holder;
                }
            }
        }
        System.out.print("\nPatient Names Sorted in Alphabetical Order: \n");
        for (int x = 0; x < 6; x++) {
            if (!CloneCoronaBooth[x].equals("e e")) {
                System.out.println(CloneCoronaBooth[x]);
            }
        }

    }

    // Store Program Data into file method
    public static void StorePrgrmData() {
        try (BufferedWriter bkp = new BufferedWriter(new FileWriter("Corona.txt")
)) {  // taken from here : https://www.codejava.net/java-se/file-io/how-to-read-
and-write-text-file-in-java
            bkp.write(String.valueOf(CVStock)); // Stores Stock in first line
            bkp.newLine();
            for (int x = 0; x < 6; x++) {       // Stores all patient details lin
e by line
                bkp.write(CovidBooth[x].getPatientX().getmeFST());
                bkp.newLine();
                bkp.write(CovidBooth[x].getPatientX().getmeSUR());
                bkp.newLine();
                bkp.write(String.valueOf(CovidBooth[x].getPatientX().getmePA()));
                bkp.newLine();
                bkp.write(CovidBooth[x].getPatientX().getmePee());
                bkp.newLine();
                bkp.write(CovidBooth[x].getPatientX().getmePNICPN());
                bkp.newLine();
                bkp.write(CovidBooth[x].getPatientX().getmePVacc());
                bkp.newLine();
            }

        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Dumped all data to Corona.txt!\n");

    }

    // Load Program Data from file method
    public static void LoadPrgrmData() {
        try {
            Scanner Loader = new Scanner(new FileInputStream("Corona.txt")); // t
aken from here : https://www.codejava.net/java-se/file-io/how-to-read-and-write-
text-file-in-java
            CVStock = Integer.parseInt(Loader.nextLine());
            for (int x = 0; x < 6; x++) {

                String PName =Loader.nextLine();
                String SName = Loader.nextLine();
                int Age = Integer.parseInt(Loader.nextLine());
                String City = Loader.nextLine();
                String NIC = Loader.nextLine();
                String VType = Loader.nextLine();

                CovidBooth[x].setPatientX(new Patient(PName, SName, Age, City, NI
C, VType));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Loaded all data from Corona.txt!\n");
    }

    // View Remaining Vaccinations method
    public static void ViewRCV() {
        System.out.print("\nTotal Number of Covid Vaccines Left : " + CVStock);
    }

    // Add Vaccinations to the Stock
    public static void AddCVStock(Scanner input) {
        System.out.print("\nEnter Amount of Vaccines to add to stock : ");
        int CVNewStock = Integer.parseInt(input.nextLine());
        CVStock += CVNewStock;
        System.out.print("\nTotal Number of Covid Vaccines has been Updated to :
" + CVStock);
    }
```

```
}
```

Booth.java

```
public class Booth {
    private Patient PatientX; //Patient type variable inside Booth class

    public Patient getPatientX() { // Getter
        return PatientX;
    }

    public void setPatientX(Patient x) { //Setter
        PatientX = x;
    }

    public Booth(Patient PN) { // Constructor
        PatientX = PN;

    }

}
```

Patient.java

```
public class Patient{
    // Primitive type Variable inside patient class
    private String FSTName;
    private String SURName;
    private int PAge;
    private String PeeCity;
    private String PNICPN;
    private String PVaccType;

    // Getters for the above variables
    public String getmeFST(){
        return FSTName;
    }
    public String getmeSUR(){
        return SURName;
    }
    public int getmePA(){
        return PAge;
    }
    public String getmePee(){
        return PeeCity;
    }
    public String getmePNICPN(){
```

```
        return PNICPN;
    }
    public String getmePVacc(){
        return PVaccType;
    }

    //Setters for the above variables
    public void setmeFST(String x){
        FSTName = x;
    }
    public void setmeSUR(String y){
        SURName = y;
    }
    public void setmePA(int s){
        PAge = s;
    }
    public void setmePee(String z){
        PeeCity = z;
    }
    public void setmePNICPN(String q){
        PNICPN = q;
    }
    public void setmePVacc(String d){
        PVaccType = d;
    }

    public Patient(String PFN){
        FSTName = PFN;

    }

    // Constructor
    public Patient(String Name, String Surname, int Age, String City, String ID,
String VaccineType){
        FSTName= Name ;
        SURName= Surname ;
        PAge= Age ;
        PeeCity= City;
        PNICPN= ID;
        PVaccType = VaccineType;
    }

}
```

Task 4 :

VacinationCenter.java

```
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.util.LinkedList;

public class VacinationCenter {
    public static Booth[] CovidBooth = new Booth[6]; //Array of booth type that h
olds Names of patients in all 6 booths
    public static int boothNum = 0;
    public static int CVStock = 150; // Variable to hold Vaccination Stock

    public static LinkedList<Booth> WaitingRoomVT1 = new LinkedList<Booth>();  //
 Linked list for AstraZeneca waiting list
    public static LinkedList<Booth> WaitingRoomVT2 = new LinkedList<Booth>();  //
 Linked list for Sinopharm waiting list
    public static LinkedList<Booth> WaitingRoomVT3 = new LinkedList<Booth>();  //
 Linked list for Pfizer waiting list

    public static void main(String[] args) {

        for (int x = 0; x < 6; x++) //initializing the booth type array with new
Patients per Booth with all data set to empty
            CovidBooth[x] = new Booth(new Patient("e", "e", 0, "e", "e", "e"));

        PrintMenu();
    }

    public static void PrintMenu() { // Print Menu method
        Scanner input = new Scanner(System.in);
        Boolean Boothchecker = true;
        while (Boothchecker) {

            System.out.print("\n======================== MENU ==================
==========");
            System.out.print("\n100 or VVB: View all Vaccination Booths");
            System.out.print("\n101 or VEB: View all Empty Booths");
            System.out.print("\n102 or APB: Add Patient to a Booth");
            System.out.print("\n103 or RPB: Remove Patient from a Booth");
            System.out.print("\n104 or VPS: View Patients Sorted in alphabetical
order ");
```

```
            System.out.print("\n105 or SPD: Store Program Data into file");
            System.out.print("\n106 or LPD: Load Program Data from file");
            System.out.print("\n107 or VRV: View Remaining Vaccinations");
            System.out.print("\n108 or AVS: Add Vaccinations to the Stock");
            System.out.print("\n999 or EXT: Exit the Program");
            System.out.print("\n==============================================
==========");

            System.out.print("\nEnter your Input : ");
            String userInput = input.nextLine();

            if (userInput.equals("100") || userInput.equalsIgnoreCase("VVB")) {
                System.out.print("\nEntered input is either 100 or VVB!\n ");
                ViewVBooth(input);
            } else if (userInput.equals("101") || userInput.equalsIgnoreCase("VEB
")) {
                System.out.print("\nEntered input is either 101 or VEB!\n");
                ViewEBooth();
            } else if (userInput.equals("102") || userInput.equalsIgnoreCase("APB
")) {
                System.out.print("\nEntered input is either 102 or APB!\n");
                AddP2Booth(input);
            } else if (userInput.equals("103") || userInput.equalsIgnoreCase("RPB
")) {
                System.out.print("\nEntered input is either 103 or RPB!\n");
                WipeP4rmBooth(input);
            } else if (userInput.equals("104") || userInput.equalsIgnoreCase("VPS
")) {
                System.out.print("\nEntered input is either 104 or VPS!\n");
                Sorter();
            } else if (userInput.equals("105") || userInput.equalsIgnoreCase("SPD
")) {
                System.out.print("\nEntered input is either 105 or SPD!\n");
                StorePrgrmData();
            } else if (userInput.equals("106") || userInput.equalsIgnoreCase("LPD
")) {
                System.out.print("\nEntered input is either 106 or LPD!\n");
                LoadPrgrmData();
            } else if (userInput.equals("107") || userInput.equalsIgnoreCase("VRV
")) {
                System.out.print("\nEntered input is either 107 or VRV!\n");
                ViewRCV();
            } else if (userInput.equals("108") || userInput.equalsIgnoreCase("AVS
")) {
                System.out.print("\nEntered input is either 108 or AVS!\n");
```

```
                AddCVStock(input);
            } else if (userInput.equals("999") || userInput.equalsIgnoreCase("EXT
")) {
                System.out.print("\nThank you For Trying the Application! \n");
                Boothchecker = false;
            } else {
                System.out.print("\nThe entered input is Invalid!\n");
            }
        }
    }

    // View all Vaccination Booths method
    public static void ViewVBooth(Scanner input) {
        int OccupiCount = 0;
        for (int x = 0; x < 6; x++) {

            if (!CovidBooth[x].getPatientX().getmeFST().equals("e")) {
                OccupiCount += 1;
            }
        }
        showAllPatients();
        if (OccupiCount >0) {
            Boolean Pdloop = true;
            while (Pdloop){
                System.out.print("\nEnter a booth number to view more details abo
ut a patient \nOR Enter 99 to go back to the main menu: ");
                int PUInput = Integer.parseInt(input.nextLine());
                if (PUInput >= 0 && PUInput<6){
                    if(!CovidBooth[PUInput].getPatientX().getmeFST().equals("e"))
{
                        displayPatient(PUInput);
                        System.out.println("\nWould you like to check another Pat
ient ? \n");
                        showAllPatients();
                        continue;
                    }else{
                        System.out.println("\nSorry booth number " + PUInput + "
is Vacant! and therefore doesnt have any patient details to display!");
                        continue;
                    }
                }else if (PUInput == 99){
                    System.out.println("\nReturning to back to menu");
                    Pdloop= false;
                }else{
```

```
                    System.out.println("\nInvalid Input Enterd! Try again pwease
^_^\n");
                    continue;
                }
            }

        }

    }

    public static void showAllPatients(){ //displays all booths wether they are f
ree or occupied
        System.out.print("\nDisplaying all Vaccination Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].getPatientX().getmeFST().equals("e")) {
                System.out.println("Booth Number " + x + " is Free");
            } else {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x].getPatientX().getmeFST() + " " + CovidBooth[x].getPatientX().getmeSUR
());
            }
        }
    }

    public static void displayPatient(int x){ // displays all details of a patien
t of requested index method to be called in the ViewVBooth() method
        System.out.println("=================== Booth "+x+" ====================")
;
        System.out.println("First Name              : "+CovidBooth[x].getPatientX
().getmeFST());
        System.out.println("Surname                 : "+CovidBooth[x].getPatientX
().getmeSUR());
        System.out.println("Age                     : "+CovidBooth[x].getPatientX
().getmePA());
        System.out.println("City                    : "+CovidBooth[x].getPatientX
().getmePee());
        System.out.println("Nic / Passport Number   : "+CovidBooth[x].getPatientX
().getmePNICPN());
        System.out.println("Vaccination type        : "+CovidBooth[x].getPatientX
().getmePVacc());
        System.out.println("================================================");
    }

    // View all Empty Booths method
    public static void ViewEBooth() {
```

```
        System.out.print("\nDisplaying all Empty Booths: \n");
        for (int x = 0; x < 6; x++) {
            if (CovidBooth[x].getPatientX().getmeFST().equals("e"))
                System.out.println("Booth Number " + x + " is Free");
        }
    }

    // Add Patient to a Booth method
    public static void AddP2Booth(Scanner input) {

        System.out.print("\nEnter patient First Name : ");
        String PName = input.nextLine();
        System.out.print("\nEnter patient Surname : ");
        String SName = input.nextLine();
        System.out.print("\nEnter patient Age : ");
        int Age = Integer.parseInt(input.nextLine());
        System.out.print("\nEnter patient City : ");
        String City = input.nextLine();
        System.out.print("\nEnter patient NIC/Passport Number : ");
        String NIC = input.nextLine();

        Boolean Boothchecker = true;
        while (Boothchecker) {           // get vaccine type and auto assign pati
ents to proper booths
            System.out.print("\n==================== Vaccine type ===============
=====");
            System.out.print("\nEnter 1 for : AstraZeneca");
            System.out.print("\nEnter 2 for : Sinopharm");
            System.out.print("\nEnter 3 For : Pfizer");
            System.out.print("\n================================================
=====\n");

            System.out.print("\nEnter your Input : ");
            String VType = input.nextLine();

            if (VType.equals("1")){
                if (CovidBooth[0].getPatientX().getmeFST().equals("e")){
                    CovidBooth[0].setPatientX(new Patient(PName, SName, Age, City
, NIC, "AstraZeneca"));
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 0 and will be Adminsterd the AstraZeneca vaccine!\n");
                    Boothchecker = false;
                }else if (CovidBooth[1].getPatientX().getmeFST().equals("e")){
                            CovidBooth[1].setPatientX(new Patient(PName, SName, Age
, City, NIC, "AstraZeneca"));
```

```
                        System.out.print(PName+ " "+ SName + " has Been Added t
o the Booth Number 1 and will be Adminsterd the AstraZeneca vaccine!\n");
                        Boothchecker = false;
                }else{
                    System.out.print("Sorry "+ PName+ " there aren't any vacant b
ooths that can administer AstraZeneca at the moment! You've been Added to the Ast
raZeneca waiting list!\n");
                    WaitingRoomVT1.addLast(new Booth (new Patient(PName, SName, A
ge, City, NIC, "AstraZeneca"))); //Waiting list code for AstraZeneca
                    Boothchecker = false;
                }
            }else if (VType.equals("2")){
                if(CovidBooth[2].getPatientX().getmeFST().equals("e")){
                   CovidBooth[2].setPatientX(new Patient(PName, SName, Age, City,
 NIC, "Sinopharm"));
                   System.out.print(PName+ " "+ SName + " has Been Added to the B
ooth Number 2 and will be Adminsterd the Sinopharm vaccine!\n");
                   Boothchecker=false;
                }else if (CovidBooth[3].getPatientX().getmeFST().equals("e")){
                        CovidBooth[3].setPatientX(new Patient(PName, SName, Age
, City, NIC, "Sinopharm"));
                        System.out.print(PName+ " "+ SName + " has Been Added t
o the Booth Number 3 and will be Adminsterd the Sinopharm vaccine!\n");
                        Boothchecker=false;
                }else{
                    System.out.print("Sorry "+ PName+ " there aren't any vacant b
ooths that can administer Sinopharm at the moment! You've been Added to the Sinop
harm waiting list!\n");
                    WaitingRoomVT2.addLast(new Booth (new Patient(PName, SName, A
ge, City, NIC, "Sinopharm"))); //Waiting list code for Sinopharm
                    Boothchecker = false;
                }
            }else if (VType.equals("3")){
                if (CovidBooth[4].getPatientX().getmeFST().equals("e")){
                    CovidBooth[4].setPatientX(new Patient(PName, SName, Age, City
, NIC, "Pfizer"));
                    System.out.print(PName+ " "+ SName + " has Been Added to the
Booth Number 4 and will be Adminsterd the Pfizer vaccine!\n");
                    Boothchecker=false;
                }else if (CovidBooth[5].getPatientX().getmeFST().equals("e")){
                        CovidBooth[5].setPatientX(new Patient(PName, SName, Age
, City, NIC, "Pfizer"));
                        System.out.print(PName+ " "+ SName + " has Been Added t
o the Booth Number 5 and will be Adminsterd the Pfizer vaccine!\n");
                        Boothchecker=false;
```

```
                }else {
                    System.out.print("Sorry "+ PName+ " there aren't any vacant b
ooths that can administer Pfizer at the moment! You've been Added to the Pfizer w
aiting list!\n");
                    WaitingRoomVT3.addLast(new Booth (new Patient(PName, SName, A
ge, City, NIC, "Pfizer"))); //Waiting list code for Pfizer
                }
            }else {
                System.out.print("Pls enter a Valid Vaccine Type Input");
            }
        }
        CVStock--; // reduce stock here
        if (CVStock % 20 == 0) {
            System.out.print("\nyour Vaccine Stock is at : " + CVStock);
  // triggers every multiple of 20
        }
        if (CVStock == 20) {
            System.out.print("\nWARNING COVID Vaccine Stock Critically Low!!!\n")
; // triggers when stock reaches 20
        }
    }

    // Remove Patient from a Booth method
    public static void WipeP4rmBooth(Scanner input) {
        System.out.print("\nDisplaying all Occupied Booths: \n");  // Prints all
occupied booths with their booth numbers
        for (int x = 0; x < 6; x++) {
            if (!CovidBooth[x].getPatientX().getmeFST().equals("e")) {
                System.out.println("Booth Number " + x + " is occupied by: " + Co
vidBooth[x].getPatientX().getmeFST()+ " " + CovidBooth[x].getPatientX().getmeSUR(
));
            }
        }

        Boolean Boothchecker = true;
        while (Boothchecker) {
            System.out.print("\nEnter the booth number of the patient that you wi
sh to remove : ");
            int BoothNum = Integer.parseInt(input.nextLine());
            if (BoothNum >= CovidBooth.length) {
                System.out.print("\nThe entered Index is Out of Range!: \n");
                continue;
            } else {
```

```
                System.out.print("\nPatient " + CovidBooth[BoothNum].getPatientX(
).getmeFST()+ " " + CovidBooth[BoothNum].getPatientX().getmeSUR() + " Has been Re
moved From Booth Number " + BoothNum + "\n");
                CovidBooth[BoothNum].setPatientX(new Patient("e", "e", 0, "e", "e
", "e"));
            }

            //Waiting list adder!
            if(BoothNum == 0 || BoothNum == 1){  //Waiting list code for adding A
straZeneca patients
                if (WaitingRoomVT1.size() > 0){
                    if (CovidBooth[0].getPatientX().getmeFST().equals("e")){
                        CovidBooth[0] = WaitingRoomVT1.getFirst();
                        WaitingRoomVT1.removeFirst();
                        System.out.println("Booth Number 0 is now occupied by: "
+ CovidBooth[0].getPatientX().getmeFST() + " " + CovidBooth[0].getPatientX().getm
eSUR());
                        CVStock--; // reduce stock here
                        Boothchecker = false;
                    }else if (CovidBooth[1].getPatientX().getmeFST().equals("e"))
{
                        CovidBooth[1] = WaitingRoomVT1.getFirst();
                        WaitingRoomVT1.removeFirst();
                        System.out.println("Booth Number 1 is now occupied by: "
+ CovidBooth[1].getPatientX().getmeFST() + " " + CovidBooth[1].getPatientX().getm
eSUR());
                        CVStock--; // reduce stock here
                        Boothchecker = false;
                    }else {
                        System.out.print("\n The Waiting List for AstraZeneca is
Empty! No New Patients have been added to Booth 0 or Booth 1! \n");
                        Boothchecker = false;
                    }
                }else{
                    Boothchecker = false;
                }
            }else if(BoothNum == 2 || BoothNum == 3){ //Waiting list code for add
ing Sinopharm patients
                if (WaitingRoomVT2.size() > 0){

                    if (CovidBooth[2].getPatientX().getmeFST().equals("e")){
                        CovidBooth[2] = WaitingRoomVT2.getFirst();
                        WaitingRoomVT2.removeFirst();
```

```
                        System.out.println("Booth Number 2 is now occupied by: "
+ CovidBooth[2].getPatientX().getmeFST() + " " + CovidBooth[2].getPatientX().getm
eSUR());
                        CVStock--; // reduce stock here
                        Boothchecker = false;
                    }else if (CovidBooth[3].getPatientX().getmeFST().equals("e"))
{
                        CovidBooth[3] = WaitingRoomVT2.getFirst();
                        WaitingRoomVT2.removeFirst();
                        System.out.println("Booth Number 3 is now occupied by: "
+ CovidBooth[3].getPatientX().getmeFST() + " " + CovidBooth[3].getPatientX().getm
eSUR());
                        CVStock--; // reduce stock here
                        Boothchecker = false;
                    }else {
                        System.out.print("\n The Waiting List for Sinopharm is Em
pty! No New Patients have been added to Booth 2 or Booth 3! \n");
                        Boothchecker = false;
                    }

                }else{
                    Boothchecker = false;
                }
            }else if(BoothNum == 4 || BoothNum == 5){ //Waiting list code for add
ing Pfizer patients
                if (WaitingRoomVT3.size() > 0){

                    if (CovidBooth[4].getPatientX().getmeFST().equals("e")){
                        CovidBooth[4] = WaitingRoomVT3.getFirst();
                        WaitingRoomVT3.removeFirst();
                        System.out.println("Booth Number 4 is now occupied by: "
+ CovidBooth[4].getPatientX().getmeFST() + " " + CovidBooth[4].getPatientX().getm
eSUR());
                        CVStock--; // reduce stock here
                        Boothchecker = false;
                    }else if (CovidBooth[5].getPatientX().getmeFST().equals("e"))
{
                        CovidBooth[5] = WaitingRoomVT3.getFirst();
                        WaitingRoomVT3.removeFirst();
                        System.out.println("Booth Number 5 is now occupied by: "
+ CovidBooth[5].getPatientX().getmeFST() + " " + CovidBooth[5].getPatientX().getm
eSUR());
                        CVStock--; // reduce stock here
                        Boothchecker = false;
                    }else {
```

```
                        System.out.print("\n The Waiting List for Pfizer is Empty
! No New Patients have been added to Booth 4 or Booth 5! \n");
                        Boothchecker = false;
                    }

                }else{
                    Boothchecker = false;
                }
            }else {
                System.out.print("\nChecked Waiting Lists and Added No one!: \n")
;
                Boothchecker = false;
            }
        }
        if (CVStock % 20 == 0) {
            System.out.print("\nyour Vaccine Stock is at : " + CVStock);
    // triggers every multiple of 20
        }
        if (CVStock == 20) {
            System.out.print("\nWARNING COVID Vaccine Stock Critically Low!!!\n")
;   // triggers when stock reaches 20
        }
    }

    // View Patients Sorted in alphabetical order method
    public static void Sorter() {

        String[] CloneCoronaBooth = new String[6]; // cloning Booth array as a ba
ckup to ensure no sorted change is permanant
        for (int x = 0; x < 6; x++) {
            CloneCoronaBooth[x] = CovidBooth[x].getPatientX().getmeFST() +" "+ Co
vidBooth[x].getPatientX().getmeSUR(); // stores names in the format "(First Name)
 (Surname)"
        }
        for (int x = 0; x < 5; x++) {              //Bubble sort technique
            for (int y = 0; y < 5 - x; y++) {
                if (CloneCoronaBooth[y].toLowerCase().compareTo(CloneCoronaBooth[
y + 1].toLowerCase()) > 0) {

                    String Holder = CloneCoronaBooth[y];
                    CloneCoronaBooth[y] = CloneCoronaBooth[y + 1];
                    CloneCoronaBooth[y + 1] = Holder;
                }
            }
        }
```

```
        System.out.print("\nPatient Names Sorted in Alphabetical Order: \n");
        for (int x = 0; x < 6; x++) {
            if (!CloneCoronaBooth[x].equals("e e")) {
                System.out.println(CloneCoronaBooth[x]);
            }
        }
    }

    // Store Program Data into file method
    public static void StorePrgrmData() {
        try (BufferedWriter bkp = new BufferedWriter(new FileWriter("Corona.txt")
)) { // taken from here : https://www.codejava.net/java-se/file-io/how-to-read-
and-write-text-file-in-java
            bkp.write(String.valueOf(CVStock));  // Stores Stock in first line
            bkp.newLine();
            for (int x = 0; x < 6; x++) {        // Stores all patient details li
ne by line
                bkp.write(CovidBooth[x].getPatientX().getmeFST());
                bkp.newLine();
                bkp.write(CovidBooth[x].getPatientX().getmeSUR());
                bkp.newLine();
                bkp.write(String.valueOf(CovidBooth[x].getPatientX().getmePA()));
                bkp.newLine();
                bkp.write(CovidBooth[x].getPatientX().getmePee());
                bkp.newLine();
                bkp.write(CovidBooth[x].getPatientX().getmePNICPN());
                bkp.newLine();
                bkp.write(CovidBooth[x].getPatientX().getmePVacc());
                bkp.newLine();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Dumped all Vaccination Booth data to Coro
na.txt!");

        try (BufferedWriter bkp = new BufferedWriter(new FileWriter("AstraZeneca.
txt"))) { // storing AstraZeneca waiting list data into AstraZeneca.txt
            if(WaitingRoomVT1.size() > 0){
                bkp.write(String.valueOf(WaitingRoomVT1.size()));
                bkp.newLine();

                for (int x = 0; x < WaitingRoomVT1.size(); x++) {
                    Booth TemAstra = WaitingRoomVT1.get(x);
```

```
                    bkp.write(TemAstra.getPatientX().getmeFST());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmeSUR());
                    bkp.newLine();
                    bkp.write(String.valueOf(TemAstra.getPatientX().getmePA()));
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePee());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePNICPN());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePVacc());
                    bkp.newLine();
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Dumped all AstraZeneca Waiting list data
to AstraZeneca.txt!");

        try (BufferedWriter bkp = new BufferedWriter(new FileWriter("Sinopharm.tx
t"))) { // storing Sinopharm waiting list data into Sinopharm.txt
            if(WaitingRoomVT2.size() > 0){
                bkp.write(String.valueOf(WaitingRoomVT2.size()));
                bkp.newLine();

                for (int x = 0; x < WaitingRoomVT2.size(); x++) {
                    Booth TemAstra = WaitingRoomVT2.get(x);

                    bkp.write(TemAstra.getPatientX().getmeFST());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmeSUR());
                    bkp.newLine();
                    bkp.write(String.valueOf(TemAstra.getPatientX().getmePA()));
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePee());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePNICPN());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePVacc());
                    bkp.newLine();
                }
            }
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Dumped all Sinopharm Waiting list data to
 Sinopharm.txt!");

        try (BufferedWriter bkp = new BufferedWriter(new FileWriter("Pfizer.txt")
)) {  // storing Pfizer waiting list data into Pfizer.txt
            if(WaitingRoomVT3.size() > 0){
                bkp.write(String.valueOf(WaitingRoomVT3.size()));
                bkp.newLine();

                for (int x = 0; x < WaitingRoomVT3.size(); x++) {
                    Booth TemAstra = WaitingRoomVT3.get(x);

                    bkp.write(TemAstra.getPatientX().getmeFST());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmeSUR());
                    bkp.newLine();
                    bkp.write(String.valueOf(TemAstra.getPatientX().getmePA()));
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePee());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePNICPN());
                    bkp.newLine();
                    bkp.write(TemAstra.getPatientX().getmePVacc());
                    bkp.newLine();
                }
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Dumped all Pfizer Waiting list data to Pf
izer.txt!\n");
    }

    // Load Program Data from file method
    public static void LoadPrgrmData() {
        try {
            Scanner Loader = new Scanner(new FileInputStream("Corona.txt"));  //
taken from here : https://www.codejava.net/java-se/file-io/how-to-read-and-write-
text-file-in-java
            CVStock = Integer.parseInt(Loader.nextLine());
            for (int x = 0; x < 6; x++) {
```

```
                String PName =Loader.nextLine();
                String SName = Loader.nextLine();
                int Age = Integer.parseInt(Loader.nextLine());
                String City = Loader.nextLine();
                String NIC = Loader.nextLine();
                String VType = Loader.nextLine();

                CovidBooth[x].setPatientX(new Patient(PName, SName, Age, City, NI
C, VType));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Loaded all Patient Booth data from Corona
.txt!");

        try {
            Scanner Loader = new Scanner(new FileInputStream("AstraZeneca.txt"));
   // Loading AstraZeneca waiting list data from AstraZeneca.txt
            if (Loader.hasNextLine()){
                int NumofPats = Integer.parseInt(Loader.nextLine()); //this recor
d how many people in waiting list for AstraZeneca

                for (int x = 0; x < NumofPats; x++) {

                    String PName =Loader.nextLine();
                    String SName = Loader.nextLine();
                    int Age = Integer.parseInt(Loader.nextLine());
                    String City = Loader.nextLine();
                    String NIC = Loader.nextLine();
                    String VType = Loader.nextLine();

                    WaitingRoomVT1.addLast(new Booth (new Patient(PName, SName, A
ge, City, NIC, VType)));
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Loaded all AstraZeneca waiting list data
from AstraZeneca.txt!");

        try {
            Scanner Loader = new Scanner(new FileInputStream("Sinopharm.txt"));
// Loading Sinopharm waiting list data from Sinopharm.txt
```

```
            if (Loader.hasNextLine()){
                int NumofPats = Integer.parseInt(Loader.nextLine()); //this recor
d how many people in waiting list for Sinopharm

                for (int x = 0; x < NumofPats; x++) {

                    String PName =Loader.nextLine();
                    String SName = Loader.nextLine();
                    int Age = Integer.parseInt(Loader.nextLine());
                    String City = Loader.nextLine();
                    String NIC = Loader.nextLine();
                    String VType = Loader.nextLine();

                    WaitingRoomVT2.addLast(new Booth (new Patient(PName, SName, A
ge, City, NIC, VType)));
                }

            }

        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Loaded all Sinopharm waiting list data fr
om Sinopharm.txt!");

        try {
            Scanner Loader = new Scanner(new FileInputStream("Pfizer.txt"));  //
Loading Pfizer waiting list data from Pfizer.txt
            if (Loader.hasNextLine()){
                int NumofPats = Integer.parseInt(Loader.nextLine()); //this recor
d how many people in waiting list for Pfizer

                for (int x = 0; x < NumofPats; x++) {

                    String PName =Loader.nextLine();
                    String SName = Loader.nextLine();
                    int Age = Integer.parseInt(Loader.nextLine());
                    String City = Loader.nextLine();
                    String NIC = Loader.nextLine();
                    String VType = Loader.nextLine();

                    WaitingRoomVT3.addLast(new Booth (new Patient(PName, SName, A
ge, City, NIC, VType)));
                }
            }
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        System.out.print("\nSucessfully Loaded all Pfizer waiting list data from
Pfizer.txt!\n");
    }

    // View Remaining Vaccinations method
    public static void ViewRCV() {
        System.out.print("\nTotal Number of Covid Vaccines Left : " + CVStock);
    }

    // Add Vaccinations to the Stock
    public static void AddCVStock(Scanner input) {
        System.out.print("\nEnter Amount of Vaccines to add to stock : ");
        int CVNewStock = Integer.parseInt(input.nextLine());
        CVStock += CVNewStock;
        System.out.print("\nTotal Number of Covid Vaccines has been Updated to :
" + CVStock);
    }

}
```

Booth.java

```
public class Booth {
    private Patient PatientX; //Patient type variable inside Booth class

    public Patient getPatientX() { // Getter
        return PatientX;
    }

    public void setPatientX(Patient x) { //Setter
        PatientX = x;
    }

    public Booth(Patient PN) { // Constructor
        PatientX = PN;

    }

}
```

Patient.java

```
public class Patient{
```

```java
// Primitive type Variable inside patient class
private String FSTName;
private String SURName;
private int PAge;
private String PeeCity;
private String PNICPN;
private String PVaccType;

// Getters for the above variables
public String getmeFST(){
    return FSTName;
}
public String getmeSUR(){
    return SURName;
}
public int getmePA(){
    return PAge;
}
public String getmePee(){
    return PeeCity;
}
public String getmePNICPN(){
    return PNICPN;
}
public String getmePVacc(){
    return PVaccType;
}

//Setters for the above variables
public void setmeFST(String x){
    FSTName = x;
}
public void setmeSUR(String y){
    SURName = y;
}
public void setmePA(int s){
    PAge = s;
}
public void setmePee(String z){
    PeeCity = z;
}
public void setmePNICPN(String q){
    PNICPN = q;
}
public void setmePVacc(String d){
```

```
        PVaccType = d;
    }

    public Patient(String PFN){
        FSTName = PFN;

    }

    // Constructor
    public Patient(String Name, String Surname, int Age, String City, String ID,
String VaccineType){
        FSTName= Name ;
        SURName= Surname ;
        PAge= Age ;
        PeeCity= City;
        PNICPN= ID;
        PVaccType = VaccineType;

    }

}
```