UOW-logo

Informatics Institute of Technology

Department of Computing

Software Development II Coursework Report

Module : 4COSC010C: Software Development II

Module Leader : Iresh Bandara

Date of submission : 26/07/2021

Student ID : 2020222/ w123456

Student First Name : Navindu

Student Surname : Perera

"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 : Navindu Perera

Student ID : 20200502

**Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Test Case** | **Expected Result** | **Actual Result** | **Pass/Fail** |
|  | Task 1 and Task 2 |  |  |  |
| 1 | (Booths Initialised correctly)  After program starts, 100 or VVB | Displays ‘empty’ for all booths | Displays ‘empty’ for all booths | Pass |
| 2 | (Add Patient “john” to Booth 0)  102 or APB enter “Scott” | 102 or APB  Displays “john” for booth 0 | Booth Number 0 Occupied By john | pass |
| 3 | 101 or VEB | All Booth empty except booth 0 | All Booth empty except booth 0 | pass |
| 4 | 102 or APB enter “wick” | Booth Number 5 Occupied By wick | Booth Number 5 Occupied By wick | pass |
| 5 | 102 or APB enter “Bravo” | Booth Number 1 Occupied By bravo | Booth Number 1 Occupied By bravo | pass |
| 6 | 101 or VEB | Only 2,3,4 booths empty | Only 2,3,4 booths empty | pass |
| 7 | 104 or VPS | Alphabetical Order  Bravo,john,wick | Alphabetical Order  Bravo,john,wick | pass |
| 8 | 105 or SPD | Data stored sucessfully | Data stored sucessfully | pass |
| 9 | 106 or LPD | Data Load Successfully! | Data Load Successfully! | pass |
| 10 | 107 or VRV | Remaining Vaccine Count: 147 | Remaining Vaccine Count: 147 | pass |
| 11 | 108 or AVS(added 50 vaccines) | Total Vaccine Count: 197 | Total Vaccine Count: 197 | pass |
| 12 | 102 or APB enter “ ” | Enter Your Name: | Enter Your Name: | pass |
| 13 | 102 or APB enter “6 for booth ” | Invalid Booth Number! | Invalid Booth Number! | pass |
|  | Task 3 Array version |  |  |  |
| 14 | 102 or APB  Enters 0 for AstraZeneca  And then for the first name and last name and see if they give a specific booth | Select booth 0/1 | Select booth 0/1 | pass |
| 15 | 102 or APB enters 3 | Invalid Request!  Please Request Your Vaccine | Invalid Request!  Please Request Your Vaccine | pass |
| 16 | 102 or APB enters “ ” | Enter name |  | fail |
|  | Task 3 class version |  |  |  |
| 17 | 102 or APB  Enters 0 for AstraZeneca  And then for the first name, last name. age,city,passport see if they give a specific booth | Select booth 0/1 | Select booth 0/1 | pass |
| 18 | 102 or APB Enters 1 for SinoPharm  And then for the first name, last name. and for nic puts a sting value | Give age again | Program crash | fail |
| 19 | 102 or APB Enters 1 for SinoPharm  And then for the first name, last name. and for Passport puts a sting value | Give Passport again | Program crash | fail |
|  | Task 4 |  |  |  |
| 20 | 102 or APB adds 6 pateints with details from0-5 adds the 7 patient | Would you like to stay little longer, we'll give you a booth as soon as one of our guests leaves?  (1)Yes / (2)No : | Would you like to stay little longer, we'll give you a room as soon as one of our guests leaves?  (1)Yes / (2)No : | pass |
| 21 | 103 or RPB: removes a patient in booth 0 | Patient john Removed From Booth 0  Vaccine Used: AstraZeneca  Patient john Added to Booth 0  Vaccine Requested: AstraZeneca | Patient john Removed From Booth 0  Vaccine Used: AstraZeneca  Patient john Added to Booth 0  Vaccine Requested: AstraZeneca | pass |

**Discussion**

For my test cases I started from the beginning and giving the entries the program expects then to check the validations in task 1 I tried other ways to see if the validation works Tried if the add patient and remove patient works smoothly trying to think of what’s the worst way a person would try to enter his/her details and tested it for task 1 and 2 the validation worked well

For task 3(array and class)- in the course work specification, they ask for extra details like age, NIC, surname, passport and vaccinetype.so created the program where they ask for what vaccine we wanna request and assign a booth for that specific vaccine that worked well

For task 4- since 2 booths are assigned to each vaccine type if 3rd patient is added they will be asked if they want to stay in waiting room or not if yes, they will be sent to a waiting list and if a patient from that requested vaccine booth leaves the 3rd patient will be added to the booth automatically

File handling in each tasks worked smoothly (loading and saving). saved under each project file with the task number.txt

For task 5 the GUI -patient is given to fill a small form, the form is requesting them to fill name, last name, age, City, NIC and vaccine type if these fields are not filled the fields that are not filed will turn red until its filed after pressing the button “Generate Receipt ” it will shift to a new window named Receipt showing the details entered with the date and time of generating the receipt

**Code :**

Task 1

Code:-

package CWD;  
  
import java.io.File;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.Arrays;  
  
import java.util.Scanner;  
  
public class Task1 {  
 private static final String[] *currentVBooth* = new String[6];  
  
 private static int *Vaccinecount* = 150;  
  
 public static void main(String[] args) {  
  
 *viewVBooth*(*currentVBooth*);  
  
  
 label:  
 while (true){  
  
 if (*Vaccinecount* <= 20){  
 System.*out*.println("Waring! Vaccine Count Reached The Limit of " + *Vaccinecount*);  
 }  
  
 Scanner x = new Scanner(System.*in*);  
 System.*out*.println();  
 System.*out*.println("Enter Following Numbers/KeyWords To View Details!");  
 System.*out*.println();  
  
 System.*out*.println(  
 "100 or VVB: View all Vaccination Booths"+"\n"+  
 "101 or VEB: View all Empty Booths"+"\n"+  
 "102 or APB: Add Patient to a Booth"+"\n"+  
 "103 or RPB: Remove Patient from a Booth"+"\n"+  
 "104 or VPS: View Patients Sorted in alphabetical order"+"\n"+  
 "105 or SPD: Store Program Data into file"+"\n"+  
 "106 or LPD: Load Program Data from file"+"\n"+  
 "107 or VRV: View Remaining Vaccinations"+"\n"+  
 "108 or AVS: Add Vaccinations to the Stock"+"\n"+  
 "999 or EXT: Exit the Program"  
  
 );  
 System.*out*.println();  
 System.*out*.print("Please Enter The Desired Number: ");  
 String number = x.nextLine().toLowerCase();  
  
 switch (number) {  
 case "100":  
 case "vvb":  
 *viewVBooth*(*currentVBooth*);  
 break;  
 case "101":  
 case "veb":  
 *emptyVBooths*(*currentVBooth*);  
 break;  
 case "102":  
 case "apb":  
 *AddBooth*(*currentVBooth*);  
 break;  
 case "103":  
 case "rpb":  
 *removeFromBooth*(*currentVBooth*);  
 break;  
 case "104":  
 case "vps":  
 *sortBooth*(*currentVBooth*);  
 break;  
 case "105":  
 case "spd":  
 *storeData*(*currentVBooth*);  
 break;  
 case "106":  
 case "lpd":  
 *ReadData*();  
 break;  
 case "107":  
 case "vrv":  
 System.*out*.println("Remaining Vaccine Count: " + *Vaccinecount*);  
 break;  
 case "108":  
 case "avs":  
 Scanner r = new Scanner(System.*in*);  
 System.*out*.print("Adding Vaccine Count: ");  
 int add = r.nextInt();  
 *Vaccinecount* = *Vaccinecount* + add;  
 System.*out*.println(add + " Vaccines Added To Store");  
 System.*out*.println("Total Vaccine Count: " + *Vaccinecount*);//adding vaccine to the vaccine stock(150)  
 break;  
 case "999":  
 case "ext":  
 System.*out*.println("Program End!");  
 break label;  
 default:  
 System.*out*.println("Invalid Input Please Follow Given Instructions");  
 break;  
 }  
  
 }  
  
 }  
 //METHOD INITIALISE  
 public static void viewVBooth(String[] Vaccinebooth){  
 for (int i = 0; i < Vaccinebooth.length; ++i){//parameter to check if we have exceeded the booth count(used in all the methods)  
 if (Vaccinebooth[i] == null){  
 Vaccinebooth[i] = "Booth Empty";  
 }  
 System.*out*.println(i + " - " + Vaccinebooth[i]);  
  
  
 }  
 }  
 //METHOD to View Booth  
 public static void emptyVBooths(String[] emptyBooth){  
 for (int i = 0; i < emptyBooth.length; ++i){  
 if (emptyBooth[i].equals("Booth Empty")){  
 System.*out*.println(i + " - " + emptyBooth[i]);  
 }  
 }  
 if (!emptyBooth[0].equals("Booth Empty") && !emptyBooth[1].equals("Booth Empty") && !emptyBooth[2].equals("Booth Empty") && !emptyBooth[3].equals("Booth Empty") &&  
 !emptyBooth[4].equals("Booth Empty") && !emptyBooth[5].equals("Booth Empty")){  
 System.*out*.println("No Empty Booths Available!");  
 }  
  
 }  
 //METHOD to show the available booths  
 public static void AddBooth(String[] addBooth){  
 boolean innerLoop = true;  
 while (innerLoop) {  
 Scanner y = new Scanner(System.*in*);  
 System.*out*.print("Enter Your Name: ");  
 String name = y.nextLine().trim();  
 if (name.length() > 0) {//if the field is empty "Enter your name" will be repeated  
 System.*out*.print("Please Enter Booth Number(0-5) or (6) to exit: ");  
 int num = y.nextInt();  
 if (num == 6) {  
 innerLoop = false;  
 } else {  
 System.*out*.print("Please Enter Your Name:");  
  
 }  
 System.*out*.print("Booth Number(0-5): ");  
 int BoothNumber = y.nextInt();  
  
 if (BoothNumber <= 5 && BoothNumber >= 0) {  
 if (addBooth[BoothNumber].equals("Booth Empty")) {  
 addBooth[BoothNumber] = name;  
 System.*out*.println("Booth Number " + BoothNumber + " Occupied By " + name);  
 *Vaccinecount* = *Vaccinecount* - 1;  
 System.*out*.println("Remaining Vaccine Count: " + *Vaccinecount*);  
 innerLoop = false;  
 } else if (!addBooth[BoothNumber].equals("Booth Empty")) {  
 System.*out*.println("Already Occupied!");  
 *viewVBooth*(*currentVBooth*);  
 }  
  
 } else {  
 System.*out*.println("Invalid Booth Number!");  
 }  
 }  
 }  
  
  
 }  
 //METHOD to remove Patient from a booth  
 public static void removeFromBooth(String[] removeFromBooth) {  
 Scanner r = new Scanner(System.*in*);  
 boolean innerRemove = true;  
 while (innerRemove){  
 System.*out*.println("Enter Booth Number to Remove(0-5)");  
 int patient = r.nextInt();  
  
 if (patient <= 5 && patient >= 0){  
 if (!removeFromBooth[patient].equals("Booth Empty")){  
 System.*out*.println(removeFromBooth[patient] + " Removed!");  
 removeFromBooth[patient] = "Booth Empty";  
 innerRemove = false;  
 }  
 else {  
 System.*out*.println("This Booth is Already Empty!");  
 *viewVBooth*(*currentVBooth*);  
 }  
 }  
 else {  
 System.*out*.println("Invalid Number");  
 }  
 }  
  
 }  
 //METHOD to sort pateints in alphebetical Order  
 public static void sortBooth(String[] sortBooth){  
 String[] sorting = Arrays.*copyOfRange*(sortBooth, 0, sortBooth.length);  
 int j = 0;  
 String temp;  
 for (int i = 0; i < sorting.length ; i++){  
 for (j = 0; j < sorting.length ; ++j){  
 if(sorting[j].charAt(0)>sorting[i].charAt(0)){  
 temp=sorting[i];  
 sorting[i]=sorting[j];  
 sorting[j]=temp;  
 }  
 }  
 }  
 for(int k=0;k<sorting.length ;k++){  
 if (!sorting[k].equals("Booth Empty")){  
 System.*out*.println(sorting[k]);  
 }  
  
 }  
  
 }  
 //METHOD to store data in a Text file called Task1 in the project folder  
 public static void storeData(String [] storeData ){  
  
 try {  
 FileWriter myFile = new FileWriter("Task1.txt");  
 for(int x =0 ; x < storeData.length ; x++){  
 myFile.write(" -------------------------- Booth Details --------------------------" + "\n");  
 myFile.write("Name : " + storeData[x] + "\n");  
 myFile.write("Booth Number: " + x + "\n");  
 myFile.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" + "\n");  
 myFile.write("\n\n ");  
 }  
 myFile.write("Remaining Vaccine Count: " + Integer.*toString*(*Vaccinecount*));  
 myFile.close();  
 System.*out*.println("Data Successfully Stored");  
 } catch (IOException e) {  
 System.*out*.println("An Error Occurred");  
 e.printStackTrace();  
 }  
  
 }  
 //METHOD to view data inside the java program that are taken from the Task1.txt file  
 public static void ReadData(){  
 try {  
 File myFile = new File("Task1.txt");  
 Scanner readFile = new Scanner(myFile) ;  
 while (readFile.hasNextLine()){  
 String data = readFile.nextLine();  
 System.*out*.println(data);  
 System.*out*.println();  
 }  
 readFile.close();  
 System.*out*.println("Data Load Successfully! ");  
 } catch (IOException e) {  
 System.*out*.println("An Error Occurred");  
 e.printStackTrace();  
 }  
 }  
  
}

Task 2 (Class version)

Code:-

Driver class

package CWD;  
  
public class Driver {  
 public static void main(String[] args) {  
 VaccinationCenter doThis = new VaccinationCenter();  
 doThis.Vaccineprogram();  
 }  
}

Vaccination Center Class

package CWD;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class VaccinationCenter {  
 Booth[] Vpatients = new Booth[6];  
 Booth myVBooth = new Booth();  
  
 Scanner x = new Scanner(System.*in*);  
  
 public void Vaccineprogram() {  
 for (int x = 0; x < Vpatients.length; x++) {  
 Vpatients[x] = new Booth();  
 }  
 *initialise*(Vpatients);  
 while (true) {  
  
 System.*out*.println();  
 System.*out*.println("Enter Following Numbers/KeyWords To View Details!");  
 System.*out*.println();  
  
 System.*out*.println(  
 "100 or VVB: View all Vaccination Booths" + "\n" +  
 "101 or VEB: View all Empty Booths" + "\n" +  
 "102 or APB: Add Patient to a Booth" + "\n" +  
 "103 or RPB: Remove Patient from a Booth" + "\n" +  
 "104 or VPS: View Patients Sorted in alphabetical order" + "\n" +  
 "105 or SPD: Store Program Data into file" + "\n" +  
 "106 or LPD: Load Program Data from file" + "\n" +  
 "107 or VRV: View Remaining Vaccinations" + "\n" +  
 "108 or AVS: Add Vaccinations to the Stock" + "\n" +  
 "999 or EXT: Exit the Program"  
  
 );  
 System.*out*.println();  
 System.*out*.print("Please Enter Respective Number: ");  
 String number = x.nextLine().toLowerCase().trim();  
 System.*out*.println();  
 switch (number) {  
 case "100":  
 case "vvb":  
 *viewVBooth*(Vpatients);  
 break;  
 case "101":  
 case "veb":  
 *emptyVBooth*(Vpatients);  
 break;  
 case "102":  
 case "apb":  
 AddBooth(Vpatients);  
 break;  
 case "103":  
 case "rpb":  
 *removeFromBooth*(Vpatients);  
 break;  
 case "104":  
 case "vps":  
 *sortBooth*(Vpatients);  
 break;  
 case "105":  
 case "spd":  
 storeData(Vpatients);  
 break;  
 case "106":  
 case "lpd":  
 *ReadData*(Vpatients);  
 break;  
 case "107":  
 case "vrv":  
 System.*out*.println("Remaining Vaccine Count: " + myVBooth.getCount());  
 break;  
 case "108":  
 case "avs":  
 addVaccine();  
 break;  
 case "999":  
 case "ext":  
 System.*out*.println("Program End!");  
 break;  
 default:  
 System.*out*.println("Invalid Input Please Follow Given Instructions");  
 break;  
 }  
  
 }  
  
 }  
  
 //METHOD INITIALISE  
 private static void initialise(Booth[] patientRef) {  
 for (int x = 0; x < patientRef.length; x++) { //parameter to check if we have exceeded the booth count(used in all the methods)  
 patientRef[x].setName("Booth Empty");  
  
 }  
 }  
 //METHOD View Booth  
 public static void viewVBooth(Booth[] patientRef) {  
 for (int x = 0; x < patientRef.length; ++x) {  
 if (patientRef[x].getName().equals("Booth Empty")) {  
 System.*out*.println("Booth " + x + " is empty");  
 } else {  
 System.*out*.println("Booth " + x + " Already Occupied By " + patientRef[x].getName());  
 }  
 }  
 }  
 //METHOD to show the available booths  
 public static void emptyVBooth(Booth[] patientsRef) {  
 int count = 0;  
 for (int x = 0; x < patientsRef.length; ++x) {  
 if (patientsRef[x].getName().equals("Booth Empty")) {  
 System.*out*.println("Booth " + x + " is empty");  
 count = count + 1;  
 }  
 }  
 if (count == 0) {  
 System.*out*.println("No Empty Booths Available!");  
 }  
 }  
 //METHOD to add patients and assign a booth  
 public void AddBooth(Booth[] patientRef) {  
 boolean adding = true;  
 int count = 0;  
 while (adding) {  
 for (int x = 0; x < patientRef.length; ++x) {  
 if (!patientRef[x].getName().equals("Booth Empty")) {  
 count = count + 1;  
 }  
 }  
 if (count == 6) {  
 System.*out*.println("All Booths Are Filled");  
 adding = false;  
 } else {  
 Scanner y = new Scanner(System.*in*);  
 System.*out*.print("Please Enter Your Name: ");  
 String name = y.nextLine().trim();  
 if (name.length() > 0) {  
 System.*out*.print("Please Enter Booth Number(0-5) or (6) to exit: ");  
 int num = y.nextInt();  
 if (num == 6) {  
 adding = false;  
 } else {  
 System.*out*.print("Please Enter Your Name:");  
  
 }  
 if (num <= 5 && num >= 0) {  
 if (!patientRef[num].getName().equals("Booth Empty")) {  
 System.*out*.println("Booth Already Occupied");  
 count = 0;  
 } else {  
 patientRef[num].setName(name);  
 System.*out*.println(name + " Added to Booth Number " + num);  
 myVBooth.adding(-1);  
 adding = false;  
 }  
 } else {  
 if (num != 6) {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 }  
 }  
  
 }  
 }  
 }  
 }  
 //METHOD to remove Patient from a booth  
 public static void removeFromBooth(Booth[] patientRef) {  
 boolean removing = true;  
 int count = 0;  
 while (removing) {  
 for (int x = 0; x < patientRef.length; ++x) {  
 if (patientRef[x].getName().equals("Booth Empty")) {  
 count = count + 1;  
 }  
 }  
 if (count == 6) {  
 System.*out*.println("All Booths Are Empty No Patients to Remove");  
 removing = false;  
 } else {  
 Scanner y = new Scanner(System.*in*);  
 System.*out*.println("Please Enter Booth Number(0-5) or (6) to exit: ");  
 int num = y.nextInt();  
 if (num == 6) {  
 removing = false;  
 }  
 if (num <= 5 && num >= 0) {  
 if (patientRef[num].getName().equals("Booth Empty")) {  
 System.*out*.println("Booth Already Empty");  
 count = 0;  
 } else {  
 System.*out*.println(patientRef[num].getName() + " Removed from Booth Number " + num);  
 patientRef[num].setName("Booth Empty");  
 removing = false;  
 }  
 } else {  
 if (num != 6) {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 }  
 }  
 }  
 }  
 }  
 //METHOD to sort pateints in alphebetical Order  
 public static void sortBooth(Booth[] patientRef) {  
 String[] sorting = new String[patientRef.length];  
 for (int r = 0; r < patientRef.length; ++r) {  
 sorting[r] = patientRef[r].getName();  
 }  
 int j = 0;  
 String temp;  
 for (int i = 0; i < sorting.length; i++) {  
 for (j = i + 1; j < sorting.length; ++j) {  
 if (sorting[i].compareTo(sorting[j]) > 0) {  
 temp = sorting[i];  
 sorting[i] = sorting[j];  
 sorting[j] = temp;  
 }  
 }  
 }  
 int count = 0;  
 for (int k = 0; k < sorting.length; k++) {  
 if (!sorting[k].equals("Booth Empty")) {  
 System.*out*.println(sorting[k]);  
 count = count + 1;  
 }  
 }  
 if (count == 0) {  
 System.*out*.println("All Booths Are Empty");  
 }  
 }  
 //METHOD to store data in a Text file called Task2 in the project folder  
 private void storeData(Booth[] patientRef) {  
 try {  
 FileWriter myFile = new FileWriter("Task2.txt");  
 for (int x = 0; x < patientRef.length; x++) {  
 myFile.write("Name : " + patientRef[x].getName() + "\n");  
 myFile.write("Booth Number: " + (x) + "\n");  
 myFile.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" + "\n");  
 myFile.write("\n\n ");  
 }  
 myFile.write("Remaining Vaccine Count: " + Integer.*toString*(myVBooth.getCount()));  
 myFile.close();  
 System.*out*.println("Data Successfully Stored");  
 } catch (IOException e) {  
 System.*out*.println("An error occurred");  
 e.printStackTrace();  
 }  
 }  
 //METHOD to view data inside the java program that are taken from the Task2.txt file  
 private static void ReadData(Booth[] patientRef) {  
 try {  
 File myFile = new File("Task2.txt");  
 Scanner reader = new Scanner(myFile);  
 while (reader.hasNextLine()) {  
 String data = reader.nextLine();  
 System.*out*.println(data);  
 }  
 reader.close();  
 } catch (FileNotFoundException e) {  
 System.*out*.println("An Error occurred.");  
 e.printStackTrace();  
 }  
 }  
 //METHOD to add vaccines to the main stock(150 starting stock)  
 public void addVaccine() {  
 Scanner adder = new Scanner(System.*in*);  
 System.*out*.print("Adding to the Vaccine count: ");  
 int addCount = adder.nextInt();  
 myVBooth.adding(addCount);  
 }  
  
  
}

Booth class

package CWD;  
  
public class Booth {  
 private String Name;  
 private int Count = 150;  
  
 public Booth (){  
 Name = "";  
 }  
 public void setName(String name) {  
 Name = name;  
 }  
 public String getName() {  
 return Name;  
 }  
 public void adding(int count) {  
 Count = Count + count;  
 System.*out*.println("Total Vaccine Count: " + Count);  
 }  
 public int getCount() {  
 return Count;  
 }  
}

Task 3 (Array version)

Code:-

package CWD;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
  
  
import java.util.Scanner;  
public class Task3 {  
  
 private static int *vaccineCount* = 150;  
 public static void main(String[] args) {  
  
 String [] booth = new String[6];  
 String [] firstName = new String[6];  
 String [] surName = new String[6];  
 String [] vaccine = new String[6];  
 String [] request = {"AstraZeneca","Sinopharm","Pfizer"};  
  
 for (int i = 0; i < booth.length; ++i){  
 booth[i] = "-";  
 firstName[i] = "-";  
 surName[i] = "-";  
 vaccine[i] = "-";  
 }  
 *initialise*(booth);  
  
 label:  
 while (true){  
  
 Scanner x = new Scanner(System.*in*);  
 System.*out*.println();  
 System.*out*.println("Enter Following Numbers/KeyWords To View Details!");  
 System.*out*.println();  
  
 System.*out*.println(  
 "100 or VVB: View all Vaccination Booths"+"\n"+  
 "101 or VEB: View all Empty Booths"+"\n"+  
 "102 or APB: Add Patient to a Booth"+"\n"+  
 "103 or RPB: Remove Patient from a Booth"+"\n"+  
 "104 or VPS: View Patients Sorted in alphabetical order"+"\n"+  
 "105 or SPD: Store Program Data into file"+"\n"+  
 "106 or LPD: Load Program Data from file"+"\n"+  
 "107 or VRV: View Remaining Vaccinations"+"\n"+  
 "108 or AVS: Add Vaccinations to the Stock"+"\n"+  
 "999 or EXT: Exit the Program"  
  
 );  
 System.*out*.print("Please Enter Respective Number: ");  
 String number = x.nextLine().toLowerCase();  
  
 switch (number) {  
 case "100":  
 case "vvb":  
 *viewVBooth*(booth,firstName,surName,vaccine);  
 break;  
 case "101":  
 case "veb":  
 *emptyVBooth*(booth);  
 break;  
 case "102":  
 case "apb":  
 *AddBooth*(booth,firstName,surName,vaccine,request);  
 break;  
 case "103":  
 case "rpb":  
 *removeFromBooth*(booth,firstName,surName,vaccine);  
 break;  
 case "104":  
 case "vps":  
 *sortBooth*(booth,firstName,surName);  
 break;  
 case "105":  
 case "spd":  
 *storeData*(booth,firstName,surName,vaccine);  
 break;  
 case "106":  
 case "lpd":  
 *readData*();  
 break;  
 case "107":  
 case "vrv":  
 System.*out*.println("Remaining Vaccine Count: " + *vaccineCount*);//Remaining vaccine count  
 break;  
 case "108":  
 case "avs":  
 Scanner r = new Scanner(System.*in*);  
 System.*out*.print("Adding Vaccine Count: ");  
 int add = r.nextInt();  
 *vaccineCount* = *vaccineCount* + add;  
 System.*out*.println(add + " Vaccines Added To Store");  
 System.*out*.println("Total Vaccine Count: " + *vaccineCount*);  
 break;  
 case "999":  
 case "ext":  
 System.*out*.println("Program End!");  
 break label;  
 default:  
 System.*out*.println("Invalid Input Please Follow Given Instructions");  
 break;  
 }  
 }  
  
 }  
 //Initializing Method  
 private static void initialise(String[] patientRef) {//parameter to check if we have exceeded the booth count(used in all the methods)  
 for (int x = 0; x < patientRef.length; x++) {  
 patientRef[x] = "Booth Empty";  
  
 }  
 }  
 //METHOD to View Booth  
 private static void viewVBooth(String [] patientRef, String[] firstName, String[] surName, String[] vaccine) {  
 for (int x = 0; x < patientRef.length; x++) {  
 if (patientRef[x].equals("Booth Empty")) {  
 System.*out*.println("Booth " + x + " is Empty");  
 } else {  
 System.*out*.println();  
 System.*out*.println("Booth " + x + " Already Occupied By " + patientRef[x]);  
 System.*out*.println("Name : " + firstName[x]+" "+ surName[x] );  
 System.*out*.println();  
 System.*out*.println("Vaccine : " + vaccine[x]);  
 System.*out*.println();  
 }  
 }  
 }  
 //METHOD to show the available booths  
 private static void emptyVBooth(String[] patientRef) {  
 int count = 0;  
 for (int x = 0; x < patientRef.length; x++) {  
 if (patientRef[x].equals("Booth Empty")) {  
 System.*out*.println("Booth " + x + " is Empty");  
 count = count + 1;  
 }  
 }  
 if (count == 0){  
 System.*out*.println("No Empty Booths Available");  
 }  
 }  
 //METHOD to remove Patient from a booth  
 public static void AddBooth(String [] patientRef, String[] firstName, String[] surName, String[] vaccine, String[] request){  
 boolean adding = true;  
 int count = 0;  
 int count2 = 0;  
 Scanner input = new Scanner(System.*in*);  
 while (adding){  
 for (int x = 0; x < patientRef.length; ++x){  
 if (!patientRef[x].equals("Booth Empty")){  
 count2 = count2 + 1;  
 }  
 }  
 if (count2 == 6){  
 System.*out*.println("All Booths Are Filled!");  
 adding = false;  
 }  
 else {  
 System.*out*.print("Please Request Your Vaccine\nAstraZeneca(0)\nSinoPharm(1)\nPfizer(2):" );  
 int order = input.nextInt();  
 if (order >= 0 && order <= 2){  
 for (int i = 0; i < vaccine.length; ++i){  
 if (vaccine[i].equals(request[order])){  
 count = count + 1;  
 }  
 }  
 if (count == 2){  
 System.*out*.println("No Empty Booths Available for the Requested Vaccine!");  
 adding = false;  
 }  
 else {  
 System.*out*.println("First Name: ");  
 String fname = input.next();  
 System.*out*.println("SurName: ");  
 String lname = input.next();  
 if (order == 0){  
 System.*out*.println("Booth No 0/1 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 0 || boothNum == 1 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!patientRef[boothNum].equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
 count2 = 0;  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ request[order]);  
 patientRef[boothNum] = fname;  
 firstName[boothNum] = fname;  
 surName[boothNum] = lname;  
 vaccine[boothNum] = request[order];  
 *vaccineCount* = *vaccineCount* - 1;  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 count2 = 0;  
 }  
 }  
 else if (order == 1){  
 System.*out*.println("Booth No 2/3 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 2 || boothNum == 3 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!patientRef[boothNum].equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
 count2 = 0;  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ request[order]);  
 patientRef[boothNum] = fname;  
 firstName[boothNum] = fname;  
 surName[boothNum] = lname;  
 vaccine[boothNum] = request[order];  
 *vaccineCount* = *vaccineCount* - 1;  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 count2 = 0;  
 }  
 }  
 else if (order == 2){  
 System.*out*.println("Booth No 4/5 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 4 || boothNum == 5 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!patientRef[boothNum].equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
 count2 = 0;  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ request[order]);  
 patientRef[boothNum] = fname;  
 firstName[boothNum] = fname;  
 surName[boothNum] = lname;  
 vaccine[boothNum] = request[order];  
 *vaccineCount* = *vaccineCount* - 1;  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 count2 = 0;  
 }  
 }  
 }  
 }  
 else {  
 System.*out*.println("Invalid Request!");  
 count = 0;  
 count2 = 0;  
 }  
 }  
 }  
 }  
 //METHOD to remove Patient from a booth  
 public static void removeFromBooth(String [] patientRef, String[] firstName, String[] surName, String[] vaccine){  
 int count = 0;  
 boolean removing = true;  
 Scanner rmv = new Scanner(System.*in*);  
 while (removing){  
 for (int x = 0; x < patientRef.length; ++x){  
 if (patientRef[x].equals("Booth Empty")){  
 count = count + 1;  
 }  
 }  
 if (count == 6){  
 System.*out*.println("All Booths Are Empty No Patients To Remove");  
 removing = false;  
 }  
 else {  
 System.*out*.print("Please Enter Booth Number (0-5) to Remove or (6) to exit: ");  
 int remove = rmv.nextInt();  
 if (remove >= 0 && remove <= 6){  
 if (remove == 6){  
 System.*out*.println("Exited");  
 removing = false;  
 }  
 else {  
 if (patientRef[remove].equals("Booth Empty")){  
 System.*out*.println("Booth Already Empty ");  
 count = 0;  
 }  
 else {  
 System.*out*.println("Patient " + patientRef[remove] + " Removed From Booth " + remove );  
 System.*out*.println("Vaccine Used: " + vaccine[remove]);  
 patientRef[remove] = "Booth Empty";  
 firstName[remove] = "-";  
 surName[remove] = "-";  
 vaccine[remove] = "-";  
 removing = false;  
 }  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 }  
 }  
 }  
 }  
 //METHOD to sort pateints in alphebetical Order  
 private static void sortBooth(String [] patientRef, String[] firstName, String[] surName){  
 String [] sorting = new String[patientRef.length];  
 for (int i = 0; i < patientRef.length; i++) {  
 sorting[i] = firstName[i] + " " + surName[i];  
 }  
 for(int i = 0; i < sorting.length; i++){  
 for (int j = i + 1;j < sorting.length; j++){  
 if (sorting[i].compareTo(sorting[j]) > 0 ){  
 String temp = sorting[i];  
 sorting[i] = sorting[j];  
 sorting[j] = temp;  
 }  
 }  
 }  
 for (int i = 0; i < sorting.length; i++){  
 if(!sorting[i].equals("Booth Empty") && !sorting[i].equals("- -")) {  
 System.*out*.println(sorting[i] );  
 }  
  
 }  
  
 }  
 //METHOD to store data in a Text file called Task3 in the project folder  
 private static void storeData(String [] patientRef,String[] firstName,String[] surName,String[] vaccine){  
 try{  
 FileWriter myFile = new FileWriter("Task3.txt");  
 for(int x =0 ; x < patientRef.length; x++){  
 myFile.write(" -------------------------- Booth Details --------------------------" + "\n");  
 myFile.write("Name : " + patientRef[x] + "\n");  
 myFile.write("Booth Number: " + x + "\n");  
 myFile.write(" First Name : " + firstName[x] + "\n" + " SurName : " + surName[x]+ "\n" + " Vaccine : " + vaccine[x] + "\n");  
 myFile.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" + "\n");  
 myFile.write("\n\n ");  
 }  
 myFile.write("Remaining Vaccine Count: " + Integer.*toString*(*vaccineCount*));  
 myFile.close();  
 System.*out*.println("Data Successfully Stored");  
 }catch (IOException e){  
 System.*out*.println("An error occurred");  
 e.printStackTrace();  
 }  
 }  
 //METHOD to view data inside the java program that are taken from the Task3.txt file  
 private static void readData(){  
 try{  
 File myFile = new File("Task3.txt");  
 Scanner reader = new Scanner(myFile);  
 while (reader.hasNextLine()){  
 String data = reader.nextLine();  
 System.*out*.println(data);  
 }  
 reader.close();  
 }catch (FileNotFoundException e){  
 System.*out*.println("An Error Occurred.");  
 e.printStackTrace();  
 }  
 }  
 public static void addVaccine(int addCount){  
 boolean adding = true;  
 Scanner adder = new Scanner(System.*in*);  
 while (adding){  
 System.*out*.println("Enter Adding Vaccine Count or 0 to exit: ");  
 int add = adder.nextInt();  
 if (add == 0){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (add > 0){  
 addCount = addCount + add;  
 System.*out*.println("Vaccines Added: " + add);  
 System.*out*.println("Total Count: " + addCount);  
 adding = false;  
 }  
 else {  
 System.*out*.println("Invalid Entry!");  
 }  
 }  
 }  
}

Task 3 (Class version)

Code:-

Driver class

package CWD;  
  
public class Driver {  
 public static void main(String[] args) {  
 VaccinationCenter doThis = new VaccinationCenter();  
 doThis.Vaccineprogram();  
 }  
}

Booth class

package CWD;  
  
public class Booth {  
 private String Name;  
 private int Count = 150;  
  
 public Booth (){  
 Name = "";  
 }  
 public void setName(String name) {  
 Name = name;  
 }  
 public String getName() {  
 return Name;  
 }  
 public void adding(int count) {  
 Count = Count + count;  
 }  
 public int getCount() {  
 return Count;  
 }  
}

Patient class

package CWD;  
  
public class Patient {  
 private String FirstName;  
 private String SurName;  
 private int Age;  
 private String City;  
 private int Passport;  
 private String Vaccine;  
 private String Astra = "AstraZeneca";  
 private String Sino = "Sinopharm";  
 private String Pfizer = "Pfizer";  
  
 public Patient (){  
 FirstName = "";  
 SurName = "";  
 Age = 0;  
 City = "";  
 Passport = 0;  
 Vaccine = "";  
 }  
 public void setFirstName(String firstName) {  
 FirstName = firstName;  
 }  
 public void setSurName(String surName) {  
 SurName = surName;  
 }  
 public void setAge(int age) {  
 Age = age;  
 }  
 public void setCity(String city) {  
 City = city;  
 }  
 public void setPassport(int passport){Passport = passport;}  
 public void setVaccine(String vaccine) {  
 Vaccine = vaccine;  
 }  
 public String getFirstName() {  
 return FirstName;  
 }  
 public String getSurName() {  
 return SurName;  
 }  
 public int getAge() {  
 return Age;  
 }  
 public String getCity() {  
 return City;  
 }  
 public int getPassport(){return Passport;}  
 public String getVaccine() {  
 return Vaccine;  
 }  
 public String setRequest(int req){  
 if (req == 0){  
 return Astra;  
 }  
 else if (req == 1){  
 return Sino;  
 }  
 else {  
 return Pfizer;  
 }  
 }  
  
  
}

Vaccination class

package CWD;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class VaccinationCenter {  
 Booth [] boothRef = new Booth[6];  
 Patient [] patientRef = new Patient[6];;  
 Booth myVBooth = new Booth();  
 Patient myPatient = new Patient();  
  
 Scanner x = new Scanner(System.*in*);  
  
 public void Vaccineprogram() {  
 for (int x = 0; x < boothRef.length; x++) {  
 boothRef[x] = new Booth();  
 patientRef[x] = new Patient();  
 }  
 *initialise*(boothRef);  
 while (true) {  
  
 System.*out*.println();  
 System.*out*.println("Enter Following Numbers/KeyWords To View Details!");  
 System.*out*.println();  
  
 System.*out*.println(  
 "100 or VVB: View all Vaccination Booths" + "\n" +  
 "101 or VEB: View all Empty Booths" + "\n" +  
 "102 or APB: Add Patient to a Booth" + "\n" +  
 "103 or RPB: Remove Patient from a Booth" + "\n" +  
 "104 or VPS: View Patients Sorted in alphabetical order" + "\n" +  
 "105 or SPD: Store Program Data into file" + "\n" +  
 "106 or LPD: Load Program Data from file" + "\n" +  
 "107 or VRV: View Remaining Vaccinations" + "\n" +  
 "108 or AVS: Add Vaccinations to the Stock" + "\n" +  
 "999 or EXT: Exit the Program"  
  
 );  
 System.*out*.println();  
 System.*out*.print("Please Enter Respective Number: ");  
 String number = x.nextLine().toLowerCase().trim();  
 System.*out*.println();  
 switch (number) {  
 case "100":  
 case "vvb":  
 viewVBooth(boothRef,patientRef);  
 break;  
 case "101":  
 case "veb":  
 emptyVBooth(boothRef);  
 break;  
 case "102":  
 case "apb":  
 AddBooth(boothRef,patientRef);  
 break;  
 case "103":  
 case "rpb":  
 *removeFromBooth*(boothRef,patientRef);  
 break;  
 case "104":  
 case "vps":  
 *sortBooth*(boothRef,patientRef);  
 break;  
 case "105":  
 case "spd":  
 *storeData*(boothRef,patientRef);  
 break;  
 case "106":  
 case "lpd":  
 *ReadData*();  
 break;  
 case "107":  
 case "vrv":  
 System.*out*.println("Remaining Vaccine Count: " + myVBooth.getCount());  
 break;  
 case "108":  
 case "avs":  
 addVaccine();  
 break;  
 case "999":  
 case "ext":  
 System.*out*.println("Program End!");  
 break;  
 default:  
 System.*out*.println("Invalid Input Please Follow Given Instructions");  
 break;  
 }  
  
 }  
 }  
 //METHOD INITIALISE  
 private static void initialise(Booth[] boothRef) {  
 for (int x = 0; x < boothRef.length; x++) {//parameter to check if we have exceeded the booth count(used in all the methods)  
 boothRef[x].setName("Booth Empty");  
 }  
 }  
 //METHOD View Booth  
 private void viewVBooth(Booth [] boothRef,Patient[] patientRef) {  
 for (int x = 0; x < boothRef.length; x++) {  
 if (boothRef[x].getName().equals("Booth Empty")) {  
 System.*out*.println("Booth " + x + " is Empty");  
 } else {  
 System.*out*.println("Booth " + x + " Already Occupied By " + boothRef[x].getName());  
 System.*out*.println("First Name : " + patientRef[x].getFirstName());  
 System.*out*.println("SurName : " + patientRef[x].getSurName());  
 System.*out*.println("Age : " + patientRef[x].getAge());  
 System.*out*.println("City : " + patientRef[x].getCity());  
 System.*out*.println("Passport No/ID : " + patientRef[x].getPassport());  
 System.*out*.println("Vaccine : " + patientRef[x].getVaccine());  
 System.*out*.println();  
 }  
 }  
 }  
 //METHOD to show the available booths  
 private void emptyVBooth(Booth[] boothRef) {  
 int count = 0;  
 for (int x = 0; x < boothRef.length; x++) {  
 if (boothRef[x].getName().equals("Booth Empty")) {  
 System.*out*.println("Booth " + x + " is Empty");  
 count = count + 1;  
 }  
 }  
 if (count == 0){  
 System.*out*.println("No Empty Booths Available");  
 }  
 }  
 //METHOD to add patients and assign a booth  
 public void AddBooth(Booth [] boothRef,Patient[] patientRef){  
 boolean adding = true;  
 int count = 0;  
 int count2 = 0;  
 Scanner input = new Scanner(System.*in*);  
 while (adding){  
 for (int x = 0; x < boothRef.length; ++x){  
 if (!boothRef[x].getName().equals("Booth Empty")){  
 count2 = count2 + 1;  
 }  
 }  
 if (count2 == 6){  
 System.*out*.println("All Booths Are Filled!");  
 adding = false;  
 }  
 else {  
 System.*out*.print("Please Request Your Vaccine AstraZeneca(0)/SinoPharm(1)/Pfizer(2): ");  
 int order = input.nextInt();  
 if (order >= 0 && order <= 2){  
 for (int i = 0; i < boothRef.length; ++i){  
 if (patientRef[i].getVaccine().equals(myPatient.setRequest(order))){  
 count = count + 1;  
 }  
 }  
 if (count == 2){  
 System.*out*.println("No Empty Booths Available for the Requested Vaccine!");  
 adding = false;  
 }  
 else {  
 Scanner y = new Scanner(System.*in*);  
 System.*out*.print("First Name: ");  
 String fname = input.next().trim();  
 System.*out*.print("SurName: ");  
 String lname = input.next();  
 System.*out*.print("Age: ");  
 int age = input.nextInt();  
 System.*out*.print("City: ");  
 String city = input.next();  
 System.*out*.print("Passport/ID: ");  
 int passID = input.nextInt();  
 if (order == 0){// if they put 0 AstraZeneca  
 System.*out*.println("Booth No 0/1 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 0 || boothNum == 1 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!boothRef[boothNum].getName().equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
 count2 = 0;  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ myPatient.setRequest(order));  
 boothRef[boothNum].setName(fname);  
 patientRef[boothNum].setFirstName(fname);  
 patientRef[boothNum].setSurName(lname);  
 patientRef[boothNum].setAge(age);  
 patientRef[boothNum].setCity(city);  
 patientRef[boothNum].setPassport(passID);  
 patientRef[boothNum].setVaccine(myPatient.setRequest(order));  
 myVBooth.adding(-1);  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 count2 = 0;  
 }  
 }  
 else if (order == 1){// if they put 1 SinoPharm  
 System.*out*.println("Booth No 2/3 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 2 || boothNum == 3 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!boothRef[boothNum].getName().equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
 count2 = 0;  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ myPatient.setRequest(order));  
 boothRef[boothNum].setName(fname);  
 patientRef[boothNum].setFirstName(fname);  
 patientRef[boothNum].setSurName(lname);  
 patientRef[boothNum].setAge(age);  
 patientRef[boothNum].setCity(city);  
 patientRef[boothNum].setPassport(passID);  
 patientRef[boothNum].setVaccine(myPatient.setRequest(order));  
 myVBooth.adding(-1);  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 count2 = 0;  
 }  
 }  
 else if (order == 2){// if they put 2 Pfizer  
 System.*out*.println("Booth No 4/5 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 4 || boothNum == 5 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!boothRef[boothNum].getName().equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
 count2 = 0;  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ myPatient.setRequest(order));  
 boothRef[boothNum].setName(fname);  
 patientRef[boothNum].setFirstName(fname);  
 patientRef[boothNum].setSurName(lname);  
 patientRef[boothNum].setAge(age);  
 patientRef[boothNum].setCity(city);  
 patientRef[boothNum].setPassport(passID);  
 patientRef[boothNum].setVaccine(myPatient.setRequest(order));  
 myVBooth.adding(-1);  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 count2 = 0;  
 }  
 }  
 }  
 }  
 else {  
 System.*out*.println("Invalid Request!");  
 count = 0;  
 count2 = 0;  
 }  
 }  
 }  
 }  
 //METHOD to remove Patient from a booth  
 public static void removeFromBooth(Booth [] boothRef,Patient[] patientRef){  
 int count = 0;  
 boolean removing = true;  
 Scanner rmv = new Scanner(System.*in*);  
 while (removing){  
 for (int x = 0; x < boothRef.length; ++x){  
 if (boothRef[x].getName().equals("Booth Empty")){  
 count = count + 1;  
 }  
 }  
 if (count == 6){  
 System.*out*.println("All Booths Are Empty No Patients To Remove");  
 removing = false;  
 }  
 else {  
 System.*out*.print("Please Enter Booth Number (0-5) to Remove or (6) to exit: ");  
 int remove = rmv.nextInt();  
 if (remove >= 0 && remove <= 6){  
 if (remove == 6){  
 System.*out*.println("Exited");  
 removing = false;  
 }  
 else {  
 if (boothRef[remove].getName().equals("Booth Empty")){  
 System.*out*.println("Booth Already Empty ");  
 count = 0;  
 }  
 else {  
 System.*out*.println("Patient " + boothRef[remove].getName() + " Removed From Booth " + remove );  
 System.*out*.println("Vaccine Used: " + patientRef[remove].getVaccine());  
 boothRef[remove].setName("Booth Empty");  
 patientRef[remove].setFirstName("");  
 patientRef[remove].setSurName("");  
 patientRef[remove].setAge(0);  
 patientRef[remove].setCity("");  
 patientRef[remove].setPassport(0);  
 patientRef[remove].setVaccine("");  
 removing = false;  
 }  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 }  
 }  
 }  
 }  
 //METHOD to sort pateints in alphebetical Order  
 private static void sortBooth(Booth [] boothRef, Patient[] patientRef){  
 String [] sorting = new String[boothRef.length];  
 for (int i = 0; i < boothRef.length; i++) {  
 sorting[i] = patientRef[i].getFirstName() + " " + patientRef[i].getSurName();  
 }  
 for(int i = 0; i < sorting.length; i++){  
 for (int j = i + 1;j < sorting.length; j++){  
 if (sorting[i].compareTo(sorting[j]) > 0 ){  
 String temp = sorting[i];  
 sorting[i] = sorting[j];  
 sorting[j] = temp;  
 }  
 }  
 }  
 for (int i = 0; i < sorting.length; i++){  
 if(!sorting[i].equals("Booth Empty") && !sorting[i].equals(" ")) {  
 System.*out*.println(sorting[i] );  
 }  
 }  
 }  
 //METHOD to store data in a Text file called Task3-Class in the project folder  
 private static void storeData(Booth [] boothRef,Patient[] patientRef){  
 try{  
 FileWriter myFile = new FileWriter("Task3-Class.txt");  
 for(int x =0 ; x < boothRef.length; x++){  
 myFile.write(" -------------------------- Booth Details --------------------------" + "\n");  
 myFile.write("Name : " + boothRef[x].getName() + "\n");  
 myFile.write("Booth Number: " + x + "\n");  
 myFile.write(" First Name : " + patientRef[x].getFirstName() + "\n" + " SurName : " + patientRef[x].getSurName()+ "\n" + " Vaccine : " + patientRef[x].getVaccine() + "\n");  
 myFile.write(" Age : " + patientRef[x].getAge() + "\n" + " City : " + patientRef[x].getCity()+ "\n" + " Passport/ID : " + patientRef[x].getPassport() + "\n");  
 myFile.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" + "\n");  
 myFile.write("\n\n ");  
 }  
 myFile.close();  
 System.*out*.println("Data Successfully Stored");  
 }catch (IOException e){  
 System.*out*.println("An error occurred");  
 e.printStackTrace();  
 }  
 }  
 //METHOD to view data inside the java program that are taken from the Task3-Class.txt file  
 private static void ReadData(){  
 try{  
 File myFile = new File("Task3-Class.txt");  
 Scanner reader = new Scanner(myFile);  
 while (reader.hasNextLine()){  
 String data = reader.nextLine();  
 System.*out*.println(data);  
 }  
 reader.close();  
 }catch (FileNotFoundException e){  
 System.*out*.println("An Error Occurred.");  
 e.printStackTrace();  
 }  
 }  
 //METHOD to add vaccines to the main stock(150 starting stock)  
 public void addVaccine(){  
 boolean adding = true;  
 Scanner adder = new Scanner(System.*in*);  
 while (adding){  
 System.*out*.println("Enter Adding Vaccine Count or 0 to exit: ");  
 int add = adder.nextInt();  
 if (add == 0){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (add > 0){  
 myVBooth.adding(add);  
 System.*out*.println("Vaccines Added: " + add);  
 System.*out*.println("Total Count: " + myVBooth.getCount());  
 adding = false;  
 }  
 else {  
 System.*out*.println("Invalid Entry!");  
 }  
 }  
 }  
}

Task 4

Code:-

Driver class

package CWD;  
  
public class Driver {  
 public static void main(String[] args) {  
 VaccinationCenter doThis = new VaccinationCenter();  
 doThis.Vaccineprogram();  
 }  
}

Booth class

package CWD;  
  
public class Booth {  
 private String Name;  
 private int Count = 150;  
  
 public Booth (){  
 Name = "";  
 }  
 public void setName(String name) {  
 Name = name;  
 }  
 public String getName() {  
 return Name;  
 }  
 public void adding(int count) {  
 Count = Count + count;  
 }  
 public int getCount() {  
 return Count;  
 }  
}

Patient class

package CWD;  
  
public class Patient {  
 private String FirstName;  
 private String SurName;  
 private int Age;  
 private String City;  
 private int Passport;  
 private String Vaccine;  
 private String Astra = "AstraZeneca";  
 private String Sino = "Sinopharm";  
 private String Pfizer = "Pfizer";  
  
 public Patient (){  
 FirstName = "";  
 SurName = "";  
 Age = 0;  
 City = "";  
 Passport = 0;  
 Vaccine = "";  
 }  
 public void setFirstName(String firstName) {  
 FirstName = firstName;  
 }  
 public void setSurName(String surName) {  
 SurName = surName;  
 }  
 public void setAge(int age) {  
 Age = age;  
 }  
 public void setCity(String city) {  
 City = city;  
 }  
 public void setPassport(int passport){Passport = passport;}  
 public void setVaccine(String vaccine) {  
 Vaccine = vaccine;  
 }  
 public String getFirstName() {  
 return FirstName;  
 }  
 public String getSurName() {  
 return SurName;  
 }  
 public int getAge() {  
 return Age;  
 }  
 public String getCity() {  
 return City;  
 }  
 public int getPassport(){return Passport;}  
 public String getVaccine() {  
 return Vaccine;  
 }  
 public String setRequest(int req){  
 if (req == 0){  
 return Astra;  
 }  
 else if (req == 1){  
 return Sino;  
 }  
 else {  
 return Pfizer;  
 }  
 }  
  
  
}

Vaccination class

package CWD;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.LinkedList;  
import java.util.List;  
import java.util.Scanner;  
  
public class VaccinationCenter {  
 Booth [] boothRef = new Booth[6];  
 Patient [] patientRef = new Patient[6];;  
 Booth myVBooth = new Booth();  
 Patient myPatient = new Patient();  
 LinkedList<String> waiting = new LinkedList<String>();  
 LinkedList<String> waitingVaccine = new LinkedList<String>();  
 LinkedList<String> wSurName = new LinkedList<String>();  
 LinkedList<Integer> wAge = new LinkedList<Integer>();  
 LinkedList<String> wCity = new LinkedList<String>();  
 LinkedList<Integer> wPassID = new LinkedList<Integer>();  
  
 Scanner x = new Scanner(System.*in*);  
  
 public void Vaccineprogram() {  
 for (int x = 0; x < boothRef.length; x++) {  
 boothRef[x] = new Booth();  
 patientRef[x] = new Patient();  
 }  
 *initialise*(boothRef);  
 while (true) {  
  
 System.*out*.println();  
 System.*out*.println("Enter Following Numbers/KeyWords To View Details!");  
 System.*out*.println();  
  
 System.*out*.println(  
 "100 or VVB: View all Vaccination Booths" + "\n" +  
 "101 or VEB: View all Empty Booths" + "\n" +  
 "102 or APB: Add Patient to a Booth" + "\n" +  
 "103 or RPB: Remove Patient from a Booth" + "\n" +  
 "104 or VPS: View Patients Sorted in alphabetical order" + "\n" +  
 "105 or SPD: Store Program Data into file" + "\n" +  
 "106 or LPD: Load Program Data from file" + "\n" +  
 "107 or VRV: View Remaining Vaccinations" + "\n" +  
 "108 or AVS: Add Vaccinations to the Stock" + "\n" +  
 "999 or EXT: Exit the Program"  
  
 );  
 System.*out*.println();  
 System.*out*.print("Please Enter Respective Number: ");  
 String number = x.nextLine().toLowerCase().trim();  
 System.*out*.println();  
 switch (number) {  
 case "100":  
 case "vvb":  
 viewVBooth(boothRef,patientRef);  
 break;  
 case "101":  
 case "veb":  
 emptyVBooth(boothRef);  
 break;  
 case "102":  
 case "apb":  
 AddBooth(boothRef,patientRef);  
 break;  
 case "103":  
 case "rpb":  
 removeFromBooth(boothRef,patientRef);  
 break;  
 case "104":  
 case "vps":  
 *sortBooth*(boothRef,patientRef);  
 break;  
 case "105":  
 case "spd":  
 *storeData*(boothRef,patientRef);  
 break;  
 case "106":  
 case "lpd":  
 *ReadData*();  
 break;  
 case "107":  
 case "vrv":  
 System.*out*.println("Remaining Vaccine Count: " + myVBooth.getCount());  
 break;  
 case "108":  
 case "avs":  
 addVaccine();  
 break;  
 case "999":  
 case "ext":  
 System.*out*.println("Program End!");  
 break;  
 default:  
 System.*out*.println("Invalid Input Please Follow Given Instructions");  
 break;  
 }  
  
 }  
 }  
 //METHOD INITIALISE  
 private static void initialise(Booth[] boothRef) {  
 for (int x = 0; x < boothRef.length; x++) {  
 boothRef[x].setName("Booth Empty");  
 }  
 }  
 //METHOD View Booth  
 private void viewVBooth(Booth [] boothRef,Patient[] patientRef) {  
 for (int x = 0; x < boothRef.length; x++) {  
 if (boothRef[x].getName().equals("Booth Empty")) {  
 System.*out*.println("Booth " + x + " is Empty");  
 } else {  
 System.*out*.println("Booth " + x + " Already Occupied By " + boothRef[x].getName());  
 System.*out*.println("First Name : " + patientRef[x].getFirstName());  
 System.*out*.println("SurName : " + patientRef[x].getSurName());  
 System.*out*.println("Age : " + patientRef[x].getAge());  
 System.*out*.println("City : " + patientRef[x].getCity());  
 System.*out*.println("Passport No/ID : " + patientRef[x].getPassport());  
 System.*out*.println("Vaccine : " + patientRef[x].getVaccine());  
 System.*out*.println();  
 }  
 }  
 }  
 //METHOD to show the available booths  
 private void emptyVBooth(Booth[] boothRef) {  
 int count = 0;  
 for (int x = 0; x < boothRef.length; x++) {  
 if (boothRef[x].getName().equals("Booth Empty")) {  
 System.*out*.println("Booth " + x + " is Empty");  
 count = count + 1;  
 }  
 }  
 if (count == 0){  
 System.*out*.println("No Empty Booths Available");  
 }  
 }  
 //METHOD to add patients and assign a booth  
 public void AddBooth(Booth [] boothRef, Patient[] patientRef){  
 boolean adding = true;  
 int count = 0;  
 int count2 = 0;  
 Scanner input = new Scanner(System.*in*);  
 while (adding){  
 System.*out*.print("Please Request Your Vaccine AstraZeneca(0)/SinoPharm(1)/Pfizer(2): ");  
 int order = input.nextInt();  
 if (order >= 0 && order <= 2){  
 for (int i = 0; i < boothRef.length; ++i){  
 if (patientRef[i].getVaccine().equals(myPatient.setRequest(order))){  
 count = count + 1;  
 }  
 }  
 if (count == 2){  
 boolean fullLoop = true;  
 while (fullLoop){  
 System.*out*.println("No Empty Booths Available for Requested Vaccine. ");  
 System.*out*.println("Would you like to stay little longer, we'll give you a booth as soon as one of our guests leaves? ");  
 System.*out*.print("(1)Yes / (2)No : ");  
 String wait = input.next();  
 if (wait.equals("1")){  
 System.*out*.print("First Name: ");  
 String fname = input.next();  
 System.*out*.print("SurName: ");  
 String lname = input.next();  
 System.*out*.print("Age: ");  
 int age = input.nextInt();  
 System.*out*.print("City: ");  
 String city = input.next();  
 System.*out*.print("Passport/ID: ");  
 int passID = input.nextInt();  
 System.*out*.println(fname + " Added to The Waiting List");  
 System.*out*.println("Requested Vaccine: " + myPatient.setRequest(order));  
 waiting.add(fname);  
 wSurName.add(lname);  
 wAge.add(age);  
 wCity.add(city);  
 wPassID.add(passID);  
 waitingVaccine.add(myPatient.setRequest(order));  
 fullLoop = false;  
 adding = false;  
 }  
 else if (wait.equals("2")){  
 System.*out*.println("Thank You For Reaching Us Please Try Again Later!");  
 fullLoop = false;  
 adding = false;  
 }  
 else {  
 System.*out*.println("Invalid Number!");  
 count = 0;  
 }  
  
 }  
 }  
 else {  
 System.*out*.print("First Name: ");  
 String fname = input.next();  
 System.*out*.print("SurName: ");  
 String lname = input.next();  
 System.*out*.print("Age: ");  
 int age = input.nextInt();  
 System.*out*.print("City: ");  
 String city = input.next();  
 System.*out*.print("Passport/ID: ");  
 int passID = input.nextInt();  
 if (order == 0){// if they put 0 AstraZeneca  
 System.*out*.println("Booth No 0/1 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 0 || boothNum == 1 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!boothRef[boothNum].getName().equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ myPatient.setRequest(order));  
 boothRef[boothNum].setName(fname);  
 patientRef[boothNum].setFirstName(fname);  
 patientRef[boothNum].setSurName(lname);  
 patientRef[boothNum].setAge(age);  
 patientRef[boothNum].setCity(city);  
 patientRef[boothNum].setPassport(passID);  
 patientRef[boothNum].setVaccine(myPatient.setRequest(order));  
 myVBooth.adding(-1);  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
  
 }  
 }  
 else if (order == 1){// if they put 1 SinoPharm  
 System.*out*.println("Booth No 2/3 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 2 || boothNum == 3 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!boothRef[boothNum].getName().equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ myPatient.setRequest(order));  
 boothRef[boothNum].setName(fname);  
 patientRef[boothNum].setFirstName(fname);  
 patientRef[boothNum].setSurName(lname);  
 patientRef[boothNum].setAge(age);  
 patientRef[boothNum].setCity(city);  
 patientRef[boothNum].setPassport(passID);  
 patientRef[boothNum].setVaccine(myPatient.setRequest(order));  
 myVBooth.adding(-1);  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
  
 }  
 }  
 else if (order == 2){// if they put 2 Pfizer  
 System.*out*.println("Booth No 4/5 to add or (6) to exit: ");  
 int boothNum = input.nextInt();  
 if (boothNum == 4 || boothNum == 5 || boothNum == 6){  
 if (boothNum == 6){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (!boothRef[boothNum].getName().equals("Booth Empty")){  
 System.*out*.println("Booth Already Occupied By Someone!");  
 count = 0;  
  
 }  
 else {  
 System.*out*.println("Booth No " + boothNum + " Occupied By " + fname);  
 System.*out*.println("Vaccine: "+ myPatient.setRequest(order));  
 boothRef[boothNum].setName(fname);  
 patientRef[boothNum].setFirstName(fname);  
 patientRef[boothNum].setSurName(lname);  
 patientRef[boothNum].setAge(age);  
 patientRef[boothNum].setCity(city);  
 patientRef[boothNum].setPassport(passID);  
 patientRef[boothNum].setVaccine(myPatient.setRequest(order));  
 myVBooth.adding(-1);  
 adding = false;  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
  
 }  
 }  
 }  
 }  
 else {  
 System.*out*.println("Invalid Request!");  
 count = 0;  
  
 }  
 }  
 }  
 //METHOD to remove Patient from a booth  
 public void removeFromBooth(Booth [] boothRef,Patient[] patientRef){  
 int count = 0;  
 int count2 = 0;  
 boolean removing = true;  
 Scanner rmv = new Scanner(System.*in*);  
 while (removing){  
 for (int x = 0; x < boothRef.length; ++x){  
 if (boothRef[x].getName().equals("Booth Empty")){  
 count = count + 1;  
 }  
 }  
 if (count == 6){  
 System.*out*.println("All Booths Are Empty No Patients To Remove");  
 removing = false;  
 }  
 else {  
 System.*out*.print("Please Enter Booth Number (0-5) to Remove or (6) to exit: ");  
 int remove = rmv.nextInt();  
 if (remove >= 0 && remove <= 6){  
 if (remove == 6){  
 System.*out*.println("Exited");  
 removing = false;  
 }  
 else {  
 if (boothRef[remove].getName().equals("Booth Empty")){  
 System.*out*.println("Booth Already Empty ");  
 count = 0;  
 }  
 else {  
 for (int j = 0;j < waiting.size(); ++j){  
 if (patientRef[remove].getVaccine().equals(waitingVaccine.get(j))){  
 count2 = count2 + 1;  
 }  
 }  
 if (count2 > 0){  
 for (int k = 0; k < waitingVaccine.size(); ++k){//Waiting room used to put a patient of its requested vaccine only  
 if (patientRef[remove].getVaccine().equals(waitingVaccine.get(k))){  
 System.*out*.println("Patient " + boothRef[remove].getName() + " Removed From Booth " + remove );  
 System.*out*.println("Vaccine Used: " + patientRef[remove].getVaccine());  
 System.*out*.println();  
 System.*out*.println("Patient " + waiting.get(k) + " Added to Booth " + remove );  
 System.*out*.println("Vaccine Requested: " + patientRef[remove].getVaccine());  
  
 boothRef[remove].setName(waiting.get(k));  
 patientRef[remove].setFirstName(waiting.get(k));  
 patientRef[remove].setSurName(wSurName.get(k));  
 patientRef[remove].setAge(wAge.get(k));  
 patientRef[remove].setCity(wCity.get(k));  
 patientRef[remove].setPassport(wPassID.get(k));  
 patientRef[remove].setVaccine(waitingVaccine.get(k));  
 myVBooth.adding(-1);  
  
 waiting.remove(k);  
 wSurName.remove(k);  
 wAge.remove(k);  
 wCity.remove(k);  
 wPassID.remove(k);  
 waitingVaccine.remove(k);  
 removing = false;  
 break;  
 }  
 }  
 }  
 else if (count2 == 0){  
 System.*out*.println("Patient " + boothRef[remove].getName() + " Removed From Booth " + remove );  
 System.*out*.println("Vaccine Used: " + patientRef[remove].getVaccine());  
 boothRef[remove].setName("Booth Empty");  
 patientRef[remove].setFirstName("");  
 patientRef[remove].setSurName("");  
 patientRef[remove].setAge(0);  
 patientRef[remove].setCity("");  
 patientRef[remove].setPassport(0);  
 patientRef[remove].setVaccine("");  
 myVBooth.adding(-1);  
 removing = false;  
 }  
 }  
 }  
 }  
 else {  
 System.*out*.println("Invalid Booth Number");  
 count = 0;  
 }  
 }  
 }  
 }  
 //METHOD to sort pateints in alphebetical Order  
 private static void sortBooth(Booth [] boothRef, Patient[] patientRef){  
 String [] sorting = new String[boothRef.length];//creating a new array to avoid the change in a Original array  
 for (int i = 0; i < boothRef.length; i++) {  
 sorting[i] = patientRef[i].getFirstName() + " " + patientRef[i].getSurName();  
 }  
 for(int i = 0; i < sorting.length; i++){  
 for (int j = i + 1;j < sorting.length; j++){  
 if (sorting[i].compareTo(sorting[j]) > 0 ){  
 String temp = sorting[i];  
 sorting[i] = sorting[j];  
 sorting[j] = temp;  
 }  
 }  
 }  
 for (int i = 0; i < sorting.length; i++){  
 if(!sorting[i].equals("Booth Empty") && !sorting[i].equals(" ")) {  
 System.*out*.println(sorting[i] );  
 }  
 }  
 }  
 //METHOD to store data in a Text file called Task2 in the project folder  
 private static void storeData(Booth [] boothRef,Patient[] patientRef){  
 try{  
 FileWriter myFile = new FileWriter("Task4.txt");  
 for(int x =0 ; x < boothRef.length; x++){  
 myFile.write(" -------------------------- Booth Details --------------------------" + "\n");  
 myFile.write("Name : " + boothRef[x].getName() + "\n");  
 myFile.write("Booth Number: " + x + "\n");  
 myFile.write(" First Name : " + patientRef[x].getFirstName() + "\n" + " SurName : " + patientRef[x].getSurName()+ "\n" + " Vaccine : " + patientRef[x].getVaccine() + "\n");  
 myFile.write(" Age : " + patientRef[x].getAge() + "\n" + " City : " + patientRef[x].getCity()+ "\n" + " Passport/ID : " + patientRef[x].getPassport() + "\n");  
 myFile.write("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" + "\n");  
 myFile.write("\n\n ");  
 }  
 myFile.close();  
 System.*out*.println("Data Successfully Stored");  
 }catch (IOException e){  
 System.*out*.println("An error occurred");  
 e.printStackTrace();  
 }  
 }  
 //METHOD to view data inside the java program that are taken from the Task2.txt file  
 private static void ReadData(){  
 try{  
 File myFile = new File("Task4.txt");  
 Scanner reader = new Scanner(myFile);  
 while (reader.hasNextLine()){  
 String data = reader.nextLine();  
 System.*out*.println(data);  
 }  
 reader.close();  
 }catch (FileNotFoundException e){  
 System.*out*.println("An Error Occurred.");  
 e.printStackTrace();  
 }  
 }  
 //METHOD to add vaccines to the main stock(150 starting stock)  
 public void addVaccine(){  
 boolean adding = true;  
 Scanner adder = new Scanner(System.*in*);  
 while (adding){  
 System.*out*.println("Enter Adding Vaccine Count or 0 to exit: ");  
 int add = adder.nextInt();  
 if (add == 0){  
 System.*out*.println("Exited");  
 adding = false;  
 }  
 else if (add > 0){  
 myVBooth.adding(add);  
 System.*out*.println("Vaccines Added: " + add);  
 System.*out*.println("Total Count: " + myVBooth.getCount());  
 adding = false;  
 }  
 else {  
 System.*out*.println("Invalid Entry!");  
 }  
 }  
 }  
}

Task 5

Code:-

Main.java

package application;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.stage.Stage;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
  
  
public class Main extends Application {  
  
 @Override  
 public void start(Stage stage) {  
 try {  
  
 Parent root = FXMLLoader.*load*(getClass().getResource("Scene1.fxml"));  
 Scene scene = new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
  
 } catch(Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

Scene1.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.RadioButton?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.control.ToggleGroup?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="658.0" prefWidth="1139.0" style="-fx-background-color: #003366;" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.VaccineWindowController">  
 <children>  
 <Pane layoutX="-72.0" layoutY="34.0" prefHeight="71.0" prefWidth="291.0" rotate="-46.5" style="-fx-background-color: #FF4500;" />  
 <Pane layoutX="40.0" layoutY="124.0" prefHeight="71.0" prefWidth="291.0" style="-fx-background-color: #003366;" />  
 <Pane layoutX="59.0" layoutY="40.0" prefHeight="616.0" prefWidth="1046.0" style="-fx-background-color: #FFA500;">  
 <children>  
 <Label layoutX="70.0" layoutY="33.0" prefHeight="63.0" prefWidth="494.0" text="VACCINE CENTER" textAlignment="RIGHT" textFill="ORANGERED">  
 <font>  
 <Font name="Bernard MT Condensed" size="49.0" />  
 </font>  
 </Label>  
 <TextField fx:id="Boothno" layoutX="693.0" layoutY="353.0" prefHeight="25.0" prefWidth="232.0" promptText="(0-5)" />  
 </children>  
 </Pane>  
 <TextField fx:id="fname" layoutX="264.0" layoutY="201.0" prefHeight="25.0" prefWidth="232.0" promptText="First name" />  
 <TextField fx:id="sname" layoutX="752.0" layoutY="203.0" prefHeight="25.0" prefWidth="232.0" promptText="Surname" />  
 <Label layoutX="117.0" layoutY="199.0" text="First name">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <TextField fx:id="age" layoutX="264.0" layoutY="295.0" prefHeight="25.0" prefWidth="232.0" promptText="AGE" />  
 <TextField fx:id="city" layoutX="752.0" layoutY="295.0" prefHeight="25.0" prefWidth="232.0" promptText="City" />  
 <TextField fx:id="nc" layoutX="264.0" layoutY="390.0" prefHeight="25.0" prefWidth="232.0" promptText="NIC no" />  
 <Label layoutX="612.0" layoutY="201.0" text="Surname">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <Label layoutX="636.0" layoutY="293.0" text="City">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <Label layoutX="134.0" layoutY="387.0" text="NIC no">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <Label layoutX="332.0" layoutY="486.0" text="Select Vaccine">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <RadioButton fx:id="btAZ" layoutX="488.0" layoutY="492.0" mnemonicParsing="false" prefWidth="128.0" text="AstraZeneca">  
 <toggleGroup>  
 <ToggleGroup fx:id="tgV" />  
 </toggleGroup>  
 <font>  
 <Font size="15.0" />  
 </font>  
 </RadioButton>  
 <RadioButton fx:id="btS" layoutX="619.0" layoutY="492.0" mnemonicParsing="false" text="Sinopharm" toggleGroup="$tgV">  
 <font>  
 <Font size="15.0" />  
 </font>  
 </RadioButton>  
 <RadioButton fx:id="bt" layoutX="741.0" layoutY="492.0" mnemonicParsing="false" text="Pfizer" toggleGroup="$tgV">  
 <font>  
 <Font size="15.0" />  
 </font>  
 </RadioButton>  
 <Label fx:id="LbV" layoutX="399.0" layoutY="595.0" prefHeight="17.0" prefWidth="79.0" />  
 <Label fx:id="LbV2" layoutX="361.0" layoutY="612.0" prefHeight="17.0" prefWidth="79.0" />  
 <Label layoutX="611.0" layoutY="388.0" text="Booth no">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <Label fx:id="lblbooth" layoutX="488.0" layoutY="403.0">  
 <font>  
 <Font size="20.0" />  
 </font>  
 </Label>  
 <Label layoutX="163.0" layoutY="293.0" text="Age">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <Button fx:id="GenerateRecipt" layoutX="910.0" layoutY="581.0" mnemonicParsing="false" onAction="#Genertaereceipt" prefHeight="53.0" prefWidth="163.0" text="Generate Recipt" />  
 <Pane layoutX="-273.0" layoutY="79.0" prefHeight="71.0" prefWidth="291.0" rotate="-20.2" style="-fx-background-color: #003366;" />  
 <Pane layoutX="-202.0" layoutY="-40.0" prefHeight="71.0" prefWidth="291.0" style="-fx-background-color: #003366;" />  
 </children>  
</AnchorPane>

Scene2.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="825.0" prefWidth="492.0" style="-fx-background-color: #003366;" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.ReciptController">  
 <children>  
 <Pane layoutX="15.0" layoutY="11.0" prefHeight="782.0" prefWidth="457.0" style="-fx-background-color: #FFA500;">  
 <children>  
 <Label fx:id="Boothno" layoutX="167.0" layoutY="151.0" text="Booth no">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 </children>  
 </Pane>  
 <Label fx:id="fname" layoutX="67.0" layoutY="279.0" text="Firstname">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 <Label fx:id="age" layoutX="124.0" layoutY="499.0" text="Age">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 <Label fx:id="sname" layoutX="73.0" layoutY="355.0" text="Surname">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 <Label fx:id="Location" layoutX="125.0" layoutY="426.0" text="City">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 <Label fx:id="nic" layoutX="67.0" layoutY="571.0" prefHeight="45.0" prefWidth="308.0" text=" NIC NO:">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 <Label fx:id="vaccine" layoutX="76.0" layoutY="646.0" prefHeight="45.0" prefWidth="346.0" text="Vaccine Requested">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 <Label fx:id="Head" layoutX="100.0" layoutY="97.0" prefHeight="70.0" prefWidth="308.0" text="Vaccine Receipt" textFill="RED">  
 <font>  
 <Font name="System Bold" size="38.0" />  
 </font>  
 </Label>  
 <Label fx:id="date" layoutX="277.0" layoutY="27.0" text="Date">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 <Label fx:id="Time" layoutX="275.0" layoutY="60.0" text="Date">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 </children>  
</AnchorPane>

VaccineWindowController

package application;  
  
import java.io.IOException;  
import java.util.Calendar;  
import java.util.GregorianCalendar;  
import java.util.Random;  
  
  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Node;  
import javafx.scene.Parent;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.stage.Stage;  
  
public class VaccineWindowController {  
  
 @FXML  
 TextField fname;  
 @FXML  
 TextField sname;  
 @FXML  
 private TextField city;  
  
 @FXML  
 private TextField age;  
  
 @FXML  
 private TextField nc;  
  
 @FXML  
 private TextField Boothno;  
  
 @FXML  
 private RadioButton btAZ;  
  
 @FXML  
 private ToggleGroup tgV;  
  
 @FXML  
 private RadioButton btS;  
  
 @FXML  
 private RadioButton bt;  
  
 @FXML  
 private Label LbV;  
  
  
  
  
  
 private Stage stage;  
 private Scene scene;  
 private Parent root;  
  
  
  
  
 public void Genertaereceipt(ActionEvent event) throws IOException {//method when Generate Recipt is clicked  
  
  
 if (fname.getText().length() == 0) {//validation every fiwld to see if the field was left empty  
 fname.setStyle("-fx-border-color: red;-fx-border-width:4px;");//if the field is empty the textfield will get red and border will set to 3px  
  
 } else {  
 fname.setStyle(null);  
  
 if (sname.getText().length() == 0) {  
 sname.setStyle("-fx-border-color: red;-fx-border-width:4px;");  
 } else {  
 sname.setStyle(null);  
 if (age.getText().length() == 0) {  
 age.setStyle("-fx-border-color: red;-fx-border-width:4px;");  
 } else {  
 age.setStyle(null);  
 if (city.getText().length() == 0) {  
 city.setStyle("-fx-border-color: red;-fx-border-width:4px;");  
 } else {  
 city.setStyle(null);  
 if (nc.getText().length() == 0) {  
 nc.setStyle("-fx-border-color: red;-fx-border-width:4px;");  
 } else {  
 nc.setStyle(null);  
 if (Boothno.getText().length() == 0) {  
 Boothno.setStyle("-fx-border-color: red;-fx-border-width:4px;");  
 } else {  
 Boothno.setStyle(null);  
  
 String username = fname.getText();  
 String Surname = sname.getText();  
 //int a = Integer.parseInt(age.getText());  
 // LbV2.setText(Integer.toString(a));  
 String Age = age.getText();  
 String City = city.getText();  
 String NIC = nc.getText();  
 String booth = Boothno.getText();  
  
  
 String message = "";  
  
 if (btAZ.isSelected()) {  
 message += btAZ.getText() + "\n";  
 }  
 if (btS.isSelected()) {  
 message += btS.getText() + "\n";  
 }  
 if (bt.isSelected()) {  
 message += bt.getText() + "\n";  
 }  
 LbV.setText(message);  
  
  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("Scene2.fxml"));  
 root = loader.load();  
  
 ReciptController ReciptController = loader.getController();  
 ReciptController.displayName(username);  
 ReciptController.displaysname(Surname);  
 ReciptController.displaysage(Age);  
 ReciptController.displayCity(City);  
 ReciptController.displayNIC(NIC);  
 ReciptController.displaybooth(booth);  
 ReciptController.displayvaccine(message);  
 ReciptController.Clock();  
  
 //root = FXMLLoader.load(getClass().getResource("Scene2.fxml"));  
 stage = (Stage) ((Node) event.getSource()).getScene().getWindow();  
 scene = new Scene(root);  
 stage.setScene(scene);  
 stage.show();  
  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
}

ReceiptController

package application;  
  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
import org.omg.DynamicAny.DynArray;  
  
import java.util.Calendar;  
import java.util.Date;  
import java.util.GregorianCalendar;  
  
public class ReciptController {  
  
 @FXML  
 private Label fname;  
  
 @FXML  
 private Label age;  
  
 @FXML  
 private Label sname;  
  
 @FXML  
 private Label Location;  
 @FXML  
 private Label nic;  
  
 @FXML  
 private Label vaccine;  
 @FXML  
 private Label Time;  
  
 @FXML  
 private Label date;  
  
 @FXML  
 private Label Boothno;  
  
  
  
  
 public void displayName(String username) {//methods to get all the data from Vaccine window to print in receipt  
 fname.setText("Firstname: " + username);  
 }  
 public void displaysname(String Surname) {  
 sname.setText("Surname: " + Surname);  
 }  
 public void displayvaccine(String message) {  
 vaccine.setText("Vaccine: " + message);  
 }  
 public void displayCity(String city) {  
 Location.setText("City: " + city);  
 }  
 public void displayNIC(String NIC) {  
 nic.setText("NIC No: " + NIC);  
 }  
 public void displaysage(String AGE) {  
 age.setText("Age: " + AGE);  
 }  
 public void displaybooth(String Booth) {  
 Boothno.setText("Booth no: "+ Booth);  
 }  
 public void Clock(){  
 Calendar cal =new GregorianCalendar();  
 int day =cal.get(Calendar.*DAY\_OF\_MONTH*);  
 int Month =cal.get(Calendar.*MONTH*);  
 int year =cal.get(Calendar.*YEAR*);  
  
 int sec =cal.get(Calendar.*SECOND*);  
 int Min =cal.get(Calendar.*MINUTE*);  
 int hour =cal.get(Calendar.*HOUR*);  
 Time.setText("Time "+ hour+":"+Min+":"+ sec);  
 date.setText("Date "+day+":"+Month+":"+year);  
  
  
  
 }  
  
}