BELL HOSPITAL MANAGEMENT SYSTEM

BY: LES PAUL RANALAN

This program was created using the Eclipse IDE application and was solely based on specific requirements mentioned within the Software Engineering Module No. 4201.

The requirements are to create 4 separate entities that fall under the hospital management system to be created in Java; that being for patients, doctors, appointments, and medicine.

Below is the detailed explanation of the whole program, discussing the classes, operations (what the user can do within the program), a case diagram and a UML diagram for easier understanding of the whole program design, and finally a series of different outputs seen through screenshots.

CLASSES

Setters and getters were primarily utilized for each classes as this is required for handling and storing data. Through this method, it is a lot more efficient and easier to update or set values to variables. Not only that, the getters function is useful as it can return specific values from specific variables to the main method. An example to this would be to call the getPatientID() method in the main class and it would return all of the registration ID numbers of patients that are stored.

toString() method was also utilized in each classes which transforms an object to a String data type. This is useful as it can return the information as a String to the main method to print in the console, as well the possibility of String formatting which uses the % symbol as a format specifier. In this case, String formatting was used to create an invisible table of columns and rows.

Note that a case diagram can be found below for visualization.

1. Patient

The setters and getters method were initialized under this class which comprises of patient information such as registration id number, insurance id number, gender, age, name, symptoms, home address, and contact numbers.

Only toString() method and registration id number are affiliated with a getter method as this is how to identify a specific patient.

2. Doctor

Setters and getters method were initialized under this class which comprises of doctor information such as staff id number, specialization, name, and contact number.

Only toString() method and staff ID number are affiliated with a getter method as this is how to identify a specific doctor.

3. Appointment

Setters and getters method were initialized under this class which comprises of appointment information such as room number, registration id number of patient, staff id number of doctor, date, and time.

Only toString() method, room number, date, registration ID number of patient, and staff ID number of doctor are affiliated with a getter method as this is how to identify a specific appointment via its room number. Not only that, this specific class alone links 2 classes; patient and doctor hence the getAppointmentDoctorID() and getAppointmentPatientID() method which coincides with registration ID number of patient and staff ID number of doctor.

4. Medicine

Setters and getters method were initialized under this class which comprises of medicine information such medicine ID number, name of medicine, strength or weight of medicine, expiry date, manufacturing date, and other key information that would be printed on a paper such as instructions, definition of medicine, lists of side effects, important notices, etc.

Only toString() method, manufacturing date, expiry date, and medicine ID number are affiliated with a getter method as this is how to identify a specific medicine via its medicine ID number.

5. bell_hospital_main

This is the class where all of the main operations, file handling, data storing are mostly done. Under this class, there are four separate methods which are patient_records(), doctor_records(), appointment_records(), and medicine_records(). All of which coincides with their respective classes with setters and getters under that class in a separate .java file.

Each method creates and initializes an encrypted .txt file to be used for data storage. The method also has the function to read and write data to its own .txt file or other .txt files under a different name if specified.
Each method initializes an ArrayList, ListIterator, and Iterator; this is to efficiently perform data manipulation and storage such as changing data or adding new data.
Each method has its own sets of operations, all of which are to be explained in the following chapters.
Each method initializes an ObjectOutputStream and an ObjectInputStream as this is essential for updating and retrieving data from .txt file. Hence the keywords input and output.

No setters and getters are initialized under this class. Instead, numerous scanners are initialized in the first portion of the main method along with two String variables were initialized to make specific texts **bold**. The scanners to be used in operations later on. The next portion is where the main menu is located within a try catch function. An if statement is utilized for the main menu, having 6 options which executes that specific method if selected by the user.

Note that the try catch function in the main menu is universal. An example to this would be if the user is out of the main menu and proceeds to the medicine records and the user accidentally typed a character instead of an integer, the program will directly execute the catch block from the main menu. The catch block reads, “INVALID INPUT. PLEASE ENTER AN INTEGER. RESTART PROGRAM TO TRY AGAIN.”

OPERATIONS

There are a total of 7 operations that the user can do when using the hospital management system. However, some methods or records have only 6, this is because only the appointment_record() method has an additional operation which is Access.

1. Add

When the user executes the Add operation, a series of prompts will execute, asking the user to enter data for that specific record. Afterwards, the program will add a new object, together with the data entered by the user, to an ArrayList. The program will then store that ArrayList to a .txt file.

An example to this would be adding a new entry to the patient records, which will require entering the patient's name, age, home address, contact number, etc. Afterwards, it will create a new Patient Object with all the entered data. The program will then write that data in a .txt file and can be viewed via console.

2. Refresh

When the user executes the Refresh operation, the program will rebuild the record list. This is to check if any changes or updates have been made to the system. This can also be useful when there is a bug or an error is present.

3. Access

NOTE: ACCESS OPERATION IS ONLY FOUND IN APPOINTMENT RECORDS.

When the user executes the Access operation, the program will send a series of prompts, asking the user to enter specific data from another record. If the entered data matches the same data within the current record and the other record, it will print both data. This is to show the link between two or more records and to display two or more valuable information at the same time without leaving the current record.

An example to this would be the in the appointment_records() method wherein one appointment requires both a doctor and a patient. If the program were to display all three data in a report format, it would require the Access operation. This then prompts the user to enter the registration ID number of patient and the staff ID number of doctor to print the data of all three records.

4. Search

When the user executes the Search operation, the program will send a prompt, asking the user to enter a specific data. If the entered data matches with one of the entries in the record, it will print that entry and display it to the user.

An example to this would be displaying key information of a specific doctor with only their staff ID number. The first step would be to enter the staff ID number of the unknown doctor. The program will then display the information of that specific doctor whether it be their name or specialization.

5. Update

When the user executes the Update operation, the program will send a prompt, asking the user to enter a specific data. If the entered data matches with one of the entries in the record, it will modify that specific entry by asking the user to enter updated data or NEW data. Afterwards, it will recreate an Object with the NEW data that the user entered and puts it in an ArrayList. The program then writes that ArrayList to a .txt file.

An example to this would be changing the name of a specific patient. The first step would be to enter the registration ID number of that patient and then change the value of the name. Afterwards, save changes to the record.

6. Delete

When the user executes the Delete operation, the program will send a prompt, asking the user to enter a specific data. If the entered data matches with one of the entries in the record, the program will ERASE that specific entry or ArrayList. Afterwards, the program will write the changes in a .txt file.

An example to this would be deleting a medicine entry. The first step would be top enter the medicine ID number and then the program will do the rest; save changes and updates data to .txt files.

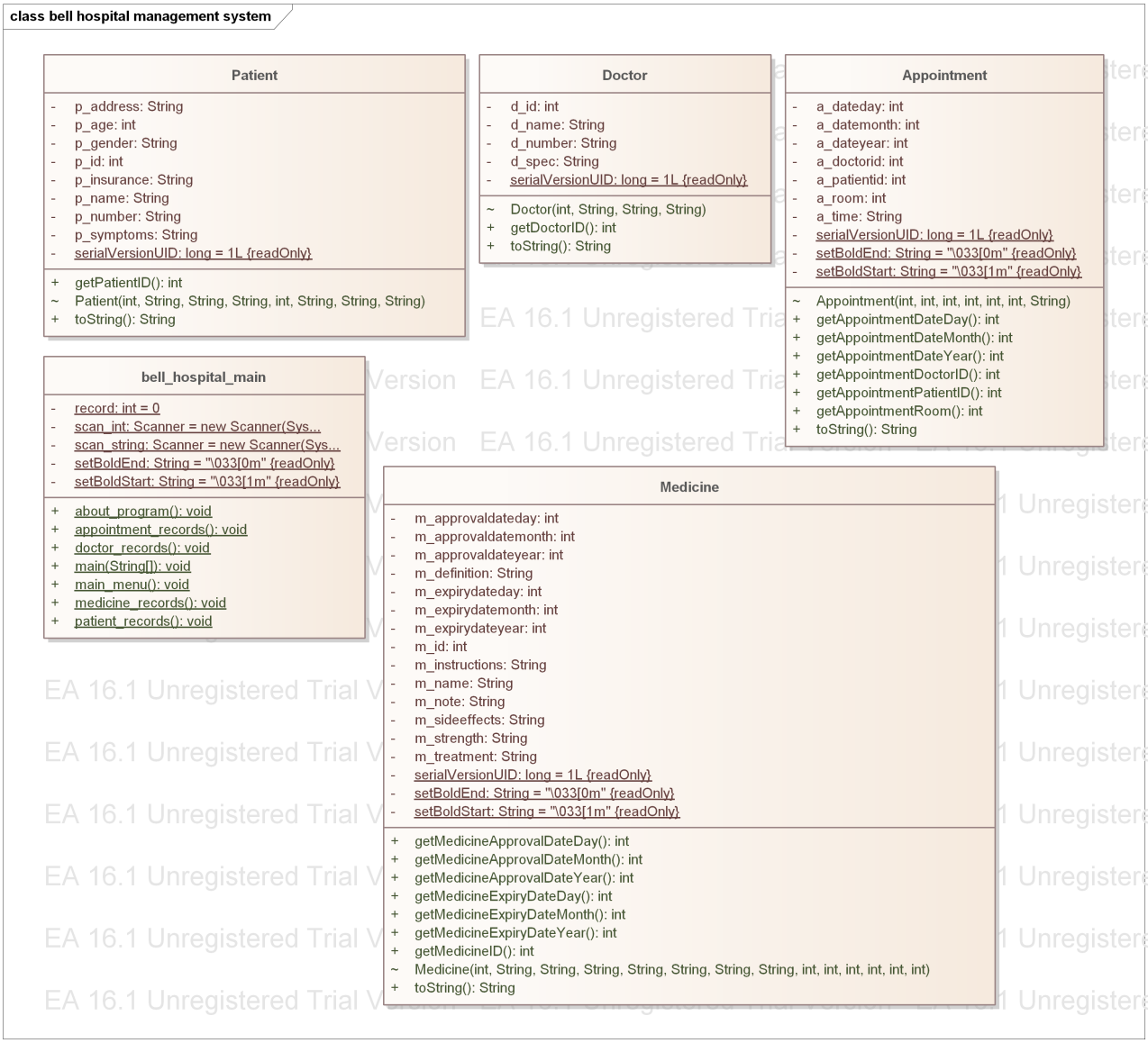
7. Cancel

NOTE: CANCEL OPERATION IS ONLY FOUND WHILE PROGRAM IS ASKING USER FOR INPUTS.

When the user executes the Cancel operation, the program will cancel the prompts and return back to the respective menu of the current record.

This is useful if the user types the wrong information on accident.

UML DIAGRAM



As seen above, the UML diagram is used to visualize the design of this particular code.

You might notice that there are no arrows or connections between each classes. This is because each class or method has their own sets of syntaxes, variables, and values. One reason is also is that each method corresponds directly to their class counterpart and does not interact with any other classes. i.e.: patient records method only specifically interacts specifically with class Patient and not any other classes.

OUTPUTS

Below are the numerous outputs in the program showcased through screenshots. Each record has its own chapter for easier navigation. If any further explanation are needed, it will be listed under the screenshot.

Main Menu

```
System.out.println(new String(new char[70]).replace("\0", "\r\n"));

System.out.println("   +-----" + setBoldStart + "BELL" + setBoldEnd + "-" + setBoldStart + "HOSPITAL"
        + setBoldEnd + "-" + setBoldStart + "MANAGEMENT" + setBoldEnd + "-----+" + "\n   |"
        + "                                |"));
System.out.println("   |   1. patient records             |");
System.out.println("   |   2. doctor records              |");
System.out.println("   |   3. appointment records         |");
System.out.println("   |   4. medicine records            |" + "\n   |                                  | ");
System.out.println(
        "   |   5. about program" + "              |" + "\n   |" + "                                |");
System.out.println(
        "   |   0. exit" + "                       |" + "\n   |" + "                                |");
System.out.println("   +---------------------------------+" + "\n");
System.out.print(setBoldStart + "   enter record: " + setBoldEnd);

main_menu = scan_int.nextInt();

// executes program to go to patient records
if (main_menu == 1) {
    patient_records();
}

// executes program to go to doctor records
if (main_menu == 2) {
    doctor_records();
}

// executes program to go to appointment records
if (main_menu == 3) {
    appointment_records();
}

// executes program to go to medicine records
if (main_menu == 4) {
    medicine_records();
}

// executes program to go to about program
if (main_menu == 5) {
    about_program();
}
```

```
+-----BELL-HOSPITAL-MANAGEMENT-----+
|                                   |
|   1. patient records              |
|   2. doctor records               |
|   3. appointment records          |
|   4. medicine records             |
|                                   |
|   5. about program                |
|                                   |
|   0. exit                         |
|                                   |
+-----------------------------------+

enter record:
```

Figure 1: Main menu of program

```java
// exits the program entirely
if (main_menu == 0) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "    GOODBYE!\n" + setBoldEnd);
    Thread.sleep(1500);
    System.exit(0);
}
```

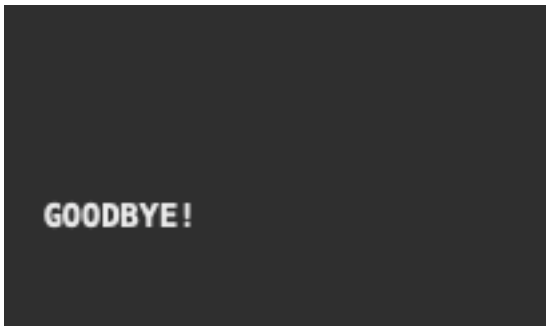


Figure 2: Output when user enters 0 in main menu
As seen above, the program displays “GOODBYE!” and exits the program entirely.

Patient Records

```java
// patient record method
@SuppressWarnings("unchecked")
public static void patient_records() throws Exception {
    record = 1;
    ArrayList<Patient> p_collection = new ArrayList<Patient>(); // initializes a new array list from class patient

    File file_patient = new File("patient.txt"); // creates new text file named patient.txt
    ObjectOutputStream p_oos = null; // initializes object output stream for class patient
    ObjectInputStream p_ois = null; // initializes object input stream for class patient

    if (file_patient.isFile()) { // if file patient.txt is found
        p_ois = new ObjectInputStream(new FileInputStream(file_patient)); // initializes new file input stream for
                                                                          // reading patient.txt
        p_collection = (ArrayList<Patient>) p_ois.readObject(); // reads data from patient.txt and imports it to
                                                                // patient array list
        p_ois.close(); // closes the object input stream for class patient
    }

    int p_menu; // declares patient choice for operation menu as an integer
    int p_search = 0; // declares patient search as initially 0, which means search user did not
                      // execute search function yet

    do {
        Iterator<Patient> p_iteration = p_collection.iterator(); // initializes patient iterator from patient array
                                                                 // list
        ListIterator<Patient> p_list = null; // initializes list iterator from patient class

        if (p_search == 0) { // if search operation is active, clear menu
            System.out.println(new String(new char[70]).replace("\0", "\r\n"));

            System.out.printf(
                    " +-----------------------------------------------------------------------------------------------------------------------------------------------------------+"
                            + "\n | " + "%-20s%-20s%-30s%-10s%-10s%-30s%-25s%-100s%n",
                    setBoldStart + "REGISTRATION ID", "    INSURANCE ID", "    NAME", "    GENDER", "    AGE",
                    "    SYMPTOMS / FINDINGS", "    CONTACT NUMBER",
                    "    HOME ADDRESS                                                                                                                                          |"
                            + setBoldEnd);

            while (p_iteration.hasNext()) { // while loop for returning true if patient iteration has more elements
                Patient p_e = p_iteration.next(); // returns next element in patient iteration
                System.out.println(p_e); // prints all elements of patient iteration
            }

            System.out.printf(
                    " +-----------------------------------------------------------------------------------------------------------------------------------------------------------+");

        }
```

```java
System.out.println("\n" + "\n" + "\n");
System.out.println("    +-----" + setBoldStart + "PATIENT" + setBoldEnd + "-" + setBoldStart + "RECORDS"
        + setBoldEnd + "-----+" + "\n    |" + "                                |");
System.out.println("    |    1. add" + "                |");
System.out.println("    |    2. refresh" + "            |");
System.out.println("    |    3. search" + "             |          enter " + setBoldStart + "000" + setBoldEnd
        + " to cancel operation.");

System.out.println("    |    4. update" + "             |");
System.out.println("    |    5. delete" + "             |" + "\n    |" + "                                |");
System.out.println("    |    0. back" + "               |" + "\n    |" + "                                |");
System.out.println("    +-------------------------+" + "\n");
System.out.print(setBoldStart + "    enter operation: " + setBoldEnd);
p_menu = scan_int.nextInt();
```

```
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+
| REGISTRATION ID     INSURANCE ID        NAME                      GENDER    AGE       SYMPTOMS / FINDINGS          CONTACT NUMBER          HOME ADDRESS
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1001                ASDB534             Les Paul Ranalan          Male      18        Headache, possibly fever     +971 52 161 1925        Jamal Abdul Nasser, Sharjah
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1002                KNBH1237            Godfrey Richard           Male      71        Chest pain                   +971 56 875 9735        Al Nahda, Sharjah
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1003                JNB765              Chas Cooley               Male      16        Vomitting                    +971 54 786 8536        Ras Al Khaimah
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1004                KMNB9785            Ena Hays                  Female    48        Acute confusion              +971 50 198 6583        Deira, Dubai
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1004                YUTN875             Nancy Maynard             Female    83        Abdominal pain               +971 55 385 7638        Mirdrif, Dubai
+-----------------------------------------------------------------------------------------------------------------------------------------------------------+


  +-----PATIENT-RECORDS-----+
  |                         |
  |    1. add               |
  |    2. refresh           |
  |    3. search            |          enter 000 to cancel operation.
  |    4. update            |
  |    5. delete            |
  |                         |
  |    0. back              |
  |                         |
  +-------------------------+

  enter operation:
```

Figure 3: Menu of Patient records and entry list.

Add

```
case 1: // add
    System.out.print(
            "\n" + "   enter " + setBoldStart + "registration id" + setBoldEnd + " of patient: \t\t");
    int p_id1 = scan_int.nextInt();
    if (p_id1 == 000) { // if scanner reads "000" it will automatically cancel the operation and returns
                        // to the menu.
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        patient_records();
    }

    System.out.print("   enter " + setBoldStart + "insurance id" + setBoldEnd + " of patient: \t\t");
    String p_insurance = scan_string.nextLine();
    if (p_insurance.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        patient_records();
    }

    System.out.print("   enter " + setBoldStart + "name" + setBoldEnd + " of patient: \t\t\t");
    String p_name = scan_string.nextLine();
    if (p_name.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        patient_records();
    }

    System.out.print("   enter " + setBoldStart + "gender" + setBoldEnd + " of patient: \t\t\t");
    String p_gender = scan_string.nextLine();
    if (p_gender.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        patient_records();
    }

    System.out.print("   enter " + setBoldStart + "age" + setBoldEnd + " of patient: \t\t\t");
    int p_age = scan_int.nextInt();
    if (p_age == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        patient_records();
    }
```

```
System.out.print("   enter " + setBoldStart + "symptoms / findings" + setBoldEnd + " of patient: \t");
String p_symptoms = scan_string.nextLine();
if (p_symptoms.equals("000")) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
    Thread.sleep(1500);
    patient_records();
}

System.out.print("   enter " + setBoldStart + "home address" + setBoldEnd + " of patient: \t\t");
String p_address = scan_string.nextLine();
if (p_address.equals("000")) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
    Thread.sleep(1500);
    patient_records();
}

System.out.print("   enter " + setBoldStart + "contact number" + setBoldEnd + " of patient: \t\t");
String p_number = scan_string.nextLine();
if (p_number.equals("000")) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
    Thread.sleep(1500);
    patient_records();
}

System.out.println(new String(new char[70]).replace("\0", "\r\n"));
System.out.println(setBoldStart + "   PATIENT RECORD ADDED." + setBoldEnd + "\n");
Thread.sleep(1500);

p_collection
        .add(new Patient(p_id1, p_insurance, p_name, p_gender, p_age, p_symptoms, p_address, p_number)); // adds all data of each variables entered by the user to a new patient object

p_search = 0; // patient search is not executed yet by user

p_oos = new ObjectOutputStream(new FileOutputStream(file_patient)); // initializes a new file output
                                                                    // stream for writing data on
                                                                    // patient.txt
p_oos.writeObject(p_collection); // updates or stores the data of array list onto patient.txt
p_oos.close(); // closes the object output stream for class patient
break;
```

```
+-----PATIENT-RECORDS------+
|                          |
|    1. add                |
|    2. refresh            |
|    3. search             |          enter 000 to cancel operation.
|    4. update             |
|    5. delete             |
|                          |
|    0. back               |
|                          |
+--------------------------+

enter operation: 1

enter registration id of patient:           1005
enter insurance id of patient:              YUTNB875
enter name of patient:                      Nancy Maynard
enter gender of patient:                    Female
enter age of patient:                       83
enter symptoms / findings of patient:       Abdominal pain
enter home address of patient:              Mirdif, Dubai
enter contact number of patient:            +971 55 385 7638
```

Figure 4: Output when user enters 1 in Patient records menu which is Add operation.
As seen above, the user is adding a new entry with the registration ID number 1005 under the name Nancy Maynard.

```
PATIENT RECORD ADDED.
```

Figure 5: Output when Add operation is done correctly.

Refresh

```
case 2: // refresh
    patient_records(); // executes patient records
    p_search = 0;
    break;
```

```
+-----------------+--------------+-------------------+--------+-----+-------------------------+----------------+-------------------------+
| REGISTRATION ID | INSURANCE ID | NAME              | GENDER | AGE | SYMPTOMS / FINDINGS     | CONTACT NUMBER | HOME ADDRESS            |
|-----------------|--------------|-------------------|--------|-----|-------------------------|----------------|-------------------------|
| 1001            | ASDB534      | Les Paul Ranalan  | Male   | 18  | Headache, possibly fever| +971 52 161 1925| Jamal Abdul Nasser, Sharjah|
| 1002            | KNBH1237     | Godfrey Richard   | Male   | 71  | Chest pain              | +971 56 875 9735| Al Nahda, Sharjah       |
| 1003            | JNB765       | Chas Cooley       | Male   | 16  | Vomitting               | +971 54 786 8536| Ras Al Khaimah          |
| 1004            | KMNB9785     | Ena Hays          | Female | 48  | Acute confusion         | +971 50 198 6583| Deira, Dubai            |
| 1005            | YUTNB875     | Nancy Maynard     | Female | 83  | Abdominal pain          | +971 55 385 7638| Mirdif, Dubai           |
+-----------------+--------------+-------------------+--------+-----+-------------------------+----------------+-------------------------+


  +-----PATIENT-RECORDS-----+
  |                         |
  |   1. add                |
  |   2. refresh            |
  |   3. search             |          enter 000 to cancel operation.
  |   4. update             |
  |   5. delete             |
  |                         |
  |   0. back               |
  |                         |
  +-------------------------+

  enter operation:
```

Figure 6: Output when user enters 2 in the Patient records menu which is refresh operation. As seen above, the entry list got updated and an entry for registration ID number 1005 with the name Nancy Maynard was added.

Search

```
case 3: // search
    boolean found = false;
    System.out
            .print("\n" + "   enter " + setBoldStart + "registration id" + setBoldEnd + " of patient: \t");
    int p_id = scan_int.nextInt();
    if (p_id == 000) { // if scanner reads "000", it will automatically cancel the operation and go
                       // back to the menu
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        patient_records();
    }

    p_iteration = p_collection.iterator(); // assigns patient iteration variable as patient array list as an
                                           // iterator
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.printf(
            " +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+"
                    + "\n | " + "%-20s%-20s%-30s%-10s%-10s%-30s%-25s%-100s%n",
            setBoldStart + "REGISTRATION ID", "    INSURANCE ID", "    NAME", "    GENDER", "    AGE",
            "    SYMPTOMS / FINDINGS", "    CONTACT NUMBER",
            "    HOME ADDRESS                                                                                                                                                                |"
                    + setBoldEnd);
    while (p_iteration.hasNext()) { // while loop for returning true if patient iteration has more or has
                                    // next elements
        Patient p_e = p_iteration.next(); // assigns p_e as patient iteration with next element

        if (p_e.getPatientID() == p_id) { // if the next element of patient iteration matches the entered
                                          // patient id
            System.out.println(p_e); // prints the array list of matching patient id
            found = true; // changes boolean found to true
            p_search = 1; // changes value of variable patient search to 1. this means that we are in the
                          // search operation.
        }

    }

    System.out.printf(
            " +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+");

    if (found != true) { // if patient id entered by user is not found, it will display an error and
                         // returns to menu.
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   PATIENT RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }

    System.out.println("\n" + "\n" + "\n");
    break;
```

```
+-----PATIENT-RECORDS-----+
|                         |
|    1. add               |
|    2. refresh           |
|    3. search            |          enter 000 to cancel operation.
|    4. update            |
|    5. delete            |
|                         |
|    0. back              |
|                         |
+-------------------------+

enter operation: 3

enter registration id of patient:    1001
```

Figure 7: Output when user enters 3 in Patient records menu which is Search operation. As seen above, the user enters the registration ID number to be searched.

Figure 8: Output when Search operation is done correctly.
As seen above, the program displays data of specific patient whose registration ID is 1001.

Update

```
case 4: // update
    if (file_patient.isFile()) { // if text file patient.txt is found
        p_ois = new ObjectInputStream(new FileInputStream(file_patient)); // initializes new file input
                                                                          // stream for reading data in
                                                                          // patient.txt
        p_collection = (ArrayList<Patient>) p_ois.readObject(); // reads the data from patient.txt and
                                                                // imports data to array list of patient
        p_ois.close(); // closes the object input stream for class patient
    }

    found = false; // declares boolean found as initially false.

    p_iteration = p_collection.iterator(); // assigns patient iteration as patient array list as an iterator
    p_list = p_collection.listIterator(); // assigns patient list as patient array list as a LIST iterator

    System.out.print("   enter " + setBoldStart + "registration id" + setBoldEnd + " of patient: ");
    p_id = scan_int.nextInt();
    if (p_id == 000) { // if scanner reads "000", it will automatically cancel the operation and return
                       // to menu.
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        patient_records();
    }

    while (p_list.hasNext()) { // while the patient list iterator has next element, it will return true
        Patient p_e = p_list.next(); // assigns p_e as the patient list iterator having next element

        if (p_e.getPatientID() == p_id) { // if patient id is matching with a patient object, the program
                                          // will then ask the user to enter new values to be entered in
                                          // the patient object
            System.out.print("\n" + "   enter " + setBoldStart + "NEW registration id" + setBoldEnd
                    + " of patient: \t");
            p_id1 = scan_int.nextInt();
            if (p_id1 == 000) { // if scanner detects "000", it will automatically stop the operation and
                                // returns to menu.
                System.out.println(new String(new char[70]).replace("\0", "\r\n"));
                System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
                Thread.sleep(1500);
                patient_records();
            }

            System.out.print(
                    "   enter " + setBoldStart + "NEW insurance id" + setBoldEnd + " of patient: \t\t");
            p_insurance = scan_string.nextLine();
            if (p_insurance.equals("000")) {
                System.out.println(new String(new char[70]).replace("\0", "\r\n"));
                System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
                Thread.sleep(1500);
                patient_records();
            }
```

```
            p_list.set(new Patient(p_id1, p_insurance, p_name, p_gender, p_age, p_symptoms, p_address,
                    p_number)); // sets the specific patient object with the new values entered by the user
            found = true; // changes boolean found to true
        }
    }

    if (found) { // if patient id is found and is matching to the patient id entered by the user,
                 // it will return true
        p_oos = new ObjectOutputStream(new FileOutputStream(file_patient)); // initializes a new file output
                                                                            // stream from for reading
                                                                            // patient.txt
        p_oos.writeObject(p_collection); // updates or writes the data of array list onto patient.txt
        p_oos.close(); // closes the object output stream for class patient
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println("   " + setBoldStart + "PATIENT RECORD UPDATED." + setBoldEnd);
        Thread.sleep(1500);
    }

    else { // if patient id is not found or is not matching with patient id entered by the
           // user, it will display error and returns to menu.
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   PATIENT RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }
    break;
```

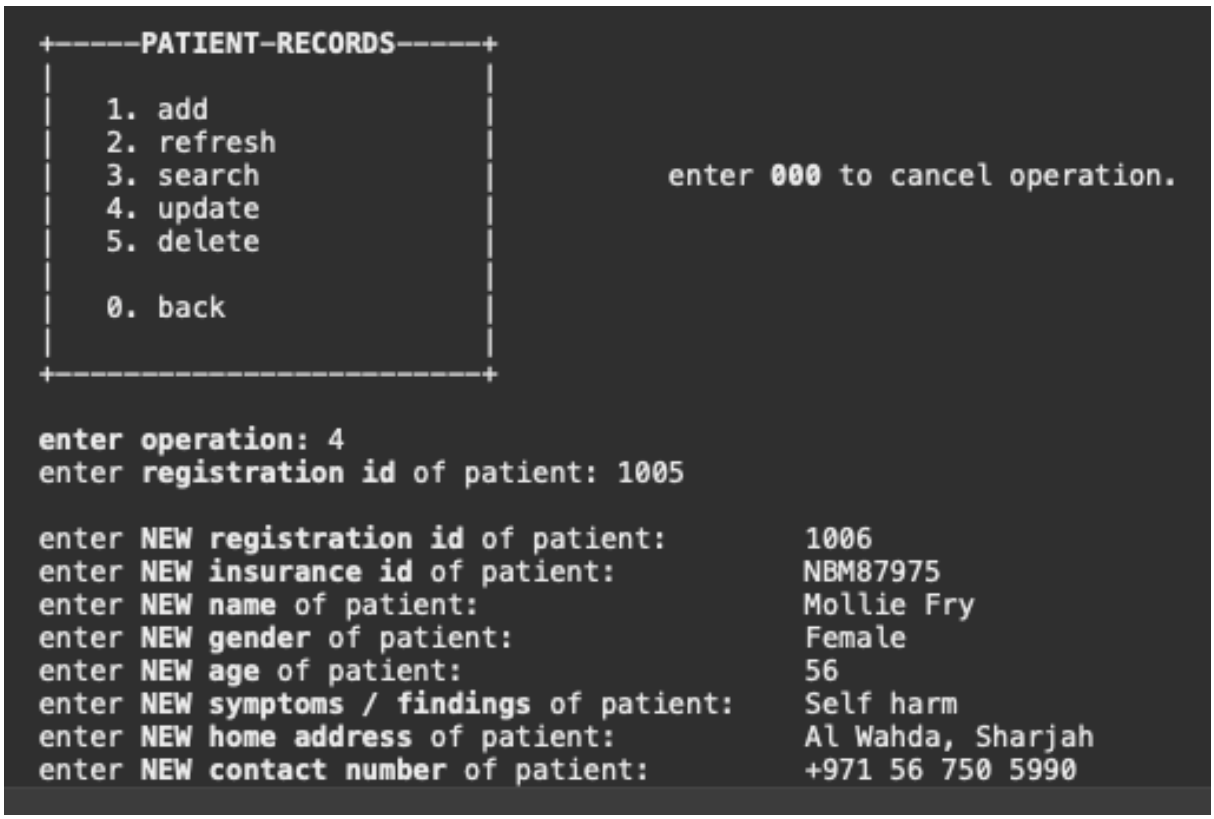


Figure 9: Output when user enters 4 in Patient records menu which is Update operation.
As seen above, the entry with the registration ID 1005 is being updated with new data, such as replacing the old name for a new name which is Mollie Fry, and etc.

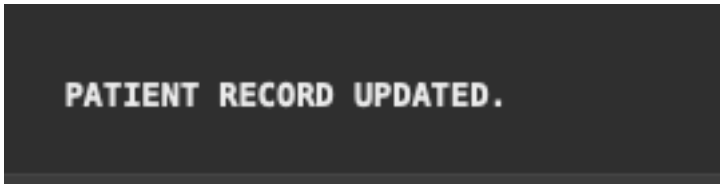


Figure 10: Output when Update operation is done correctly.

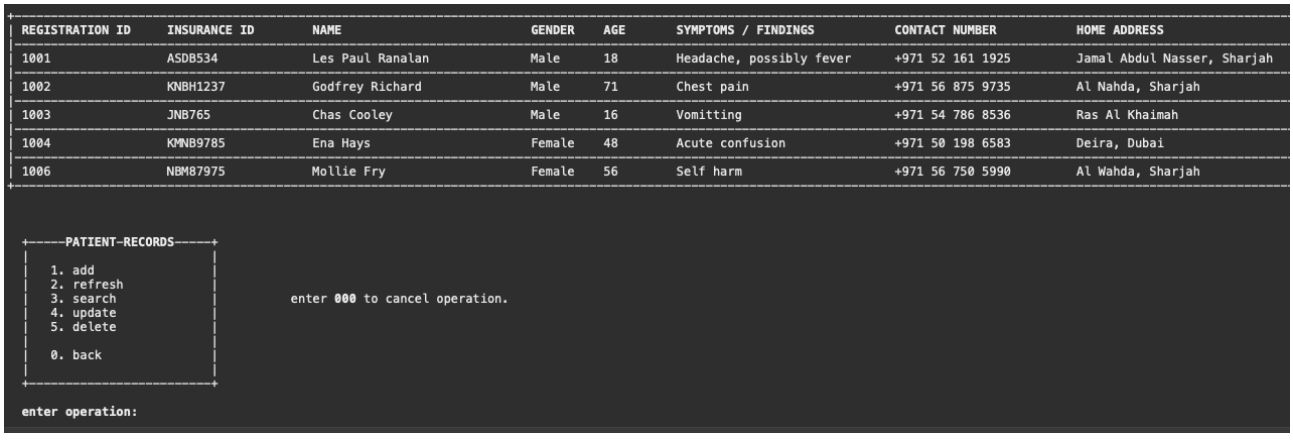


Figure 11: Output of entry list after Update operation.
As seen above, the old entry which is registration ID number 1005 is not longer existent and got replaced by the new entry registration ID number 1006 under the name Mollie Fry.

Delete

```
case 5: // delete
    found = false;

    System.out
            .print("\n" + "   enter " + setBoldStart + "registration id" + setBoldEnd + " of patient: \t");
    p_id = scan_int.nextInt();
    if (p_id == 000) { // if scanner detects "000", it will automatically stop the operation and return
                       // to menu.
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        patient_records();
    }

    p_iteration = p_collection.iterator(); // assigns patient iteration as the patient array list as an
                                           // iterator.
    while (p_iteration.hasNext()) { // while condition for scanning data in patient array list
        Patient p_e = p_iteration.next(); // assigns p_e as patient iteration with next element.

        if (p_e.getPatientID() == p_id) { // if condition for matching patient id
            p_iteration.remove(); // removes patient array list of matching patient id
            found = true; // changes boolean found to true.
            p_search = 1;
            System.out.println(new String(new char[70]).replace("\0", "\r\n"));
            p_search = 0;
            System.out.println(setBoldStart + "   PATIENT RECORD DELETED." + setBoldEnd);
            Thread.sleep(1500);
        }
    }

    if (found != true) { // if patient id is not found or is not matching with patient id entered by
                         // user, it will display error and return to menu.
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        p_search = 0;
        System.out.println(setBoldStart + "   PATIENT RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }

    p_oos = new ObjectOutputStream(new FileOutputStream(file_patient)); // initializes a new object output
                                                                        // stream
    p_oos.writeObject(p_collection); // updates and writes the data of array list onto patient.txt
    p_oos.close(); // closes the object output stream of class patient
    break;
```

```
+-----PATIENT-RECORDS------+
|                          |
|   1. add                 |
|   2. refresh             |
|   3. search              |          enter 000 to cancel operation.
|   4. update              |
|   5. delete              |
|                          |
|   0. back                |
|                          |
+--------------------------+

enter operation: 5

enter registration id of patient:    1006
```

Figure 12: Output when user enters 5 in the Patient records menu which is Delete operation. As seen above, the user enters specific registration ID number from an entry to delete.

```
PATIENT RECORD DELETED.
```

Figure 13: Output when Delete operation is done correctly.

```
+-----------------+---------------+-------------------+--------+-----+-------------------------+-----------------+-------------------------+
| REGISTRATION ID | INSURANCE ID  | NAME              | GENDER | AGE | SYMPTOMS / FINDINGS     | CONTACT NUMBER  | HOME ADDRESS            |
|-----------------+---------------+-------------------+--------+-----+-------------------------+-----------------+-------------------------|
| 1001            | ASDB534       | Les Paul Ranalan  | Male   | 18  | Headache, possibly fever| +971 52 161 1925| Jamal Abdul Nasser, Sharjah |
|-----------------+---------------+-------------------+--------+-----+-------------------------+-----------------+-------------------------|
| 1002            | KNBH1237      | Godfrey Richard   | Male   | 71  | Chest pain              | +971 56 875 9735| Al Nahda, Sharjah       |
|-----------------+---------------+-------------------+--------+-----+-------------------------+-----------------+-------------------------|
| 1003            | JNB765        | Chas Cooley       | Male   | 16  | Vomitting               | +971 54 786 8536| Ras Al Khaimah          |
|-----------------+---------------+-------------------+--------+-----+-------------------------+-----------------+-------------------------|
| 1004            | KMNB9785      | Ena Hays          | Female | 48  | Acute confusion         | +971 50 198 6583| Deira, Dubai            |
+-----------------+---------------+-------------------+--------+-----+-------------------------+-----------------+-------------------------+


  +-----PATIENT-RECORDS-----+
  |                         |
  |   1. add                |
  |   2. refresh            |
  |   3. search             |          enter 000 to cancel operation.
  |   4. update             |
  |   5. delete             |
  |                         |
  |   0. back               |
  |                         |
  +-------------------------+

  enter operation:
```

Figure 14: Output of updated entry list after Delete operation.
As seen above, the registration ID number 1006 is no longer existent as it has been deleted.

Doctor Records

```java
@SuppressWarnings("unchecked")
public static void doctor_records() throws Exception {
    record = 2;
    ArrayList<Doctor> d_collection = new ArrayList<Doctor>();

    File file_doctor = new File("doctor.txt");
    ObjectOutputStream d_oos = null;
    ObjectInputStream d_ois = null;

    if (file_doctor.isFile()) {
        d_ois = new ObjectInputStream(new FileInputStream(file_doctor));
        d_collection = (ArrayList<Doctor>) d_ois.readObject();
        d_ois.close();
    }

    int d_menu;
    int d_search = 0;

    do {
        Iterator<Doctor> d_iteration = d_collection.iterator();
        ListIterator<Doctor> d_list = null;

        if (d_search == 0) {
            System.out.println(new String(new char[70]).replace("\0", "\r\n"));

            System.out.printf(
                    " +-------------------------------------------------------------------------------------------------------------------------------+"
                            + "\n | " + "%-30s%-30s%-30s%-30s%n",
                    setBoldStart + "STAFF ID", "    DOCTOR NAME", "    SPECIALIZATION",
                    "    CONTACT NUMBER               |" + setBoldEnd);

            while (d_iteration.hasNext()) {
                Doctor d_e = d_iteration.next();
                System.out.println(d_e);
            }

            System.out.printf(
                    " +-------------------------------------------------------------------------------------------------------------------------------+");

        }
```

```java
System.out.println("\n" + "\n" + "\n");
System.out.println("    +-----" + setBoldStart + "DOCTOR" + setBoldEnd + "-" + setBoldStart + "RECORDS"
        + setBoldEnd + "-----+" + "\n    |" + "                          |");
System.out.println("    |    1. add" + "               |");
System.out.println("    |    2. refresh" + "           |");
System.out.println("    |    3. search" + "            |          enter " + setBoldStart + "000" + setBoldEnd
        + " to cancel operation.");
System.out.println("    |    4. update" + "            |");
System.out.println("    |    5. delete" + "            |" + "\n    |" + "                          |");
System.out.println("    |    0. back" + "              |" + "\n    |" + "                          |");
System.out.println("    +--------------------------+" + "\n");
System.out.print(setBoldStart + "    enter operation: " + setBoldEnd);
d_menu = scan_int.nextInt();
```

```
+-------------------------------------------------------------------------------------------------------------------------------+
| STAFF ID                      DOCTOR NAME                    SPECIALIZATION                 CONTACT NUMBER                  |
|-------------------------------------------------------------------------------------------------------------------------------|
| 9001                          Case Lowler                    Pathology                      +971 56 876 8623                |
|-------------------------------------------------------------------------------------------------------------------------------|
| 9002                          Kaden Gerstner                 General Physician              +971 54 490 1347                |
|-------------------------------------------------------------------------------------------------------------------------------|
| 9003                          Haylee Burnum                  Surgery                        +971 52 147 9745                |
|-------------------------------------------------------------------------------------------------------------------------------|
| 9004                          Spencer Wishum                 Dentistry                      +971 50 235 9745                |
|-------------------------------------------------------------------------------------------------------------------------------|
| 9005                          Angel Carfagno                 Opthalmology                   +971 55 975 8123                |
+-------------------------------------------------------------------------------------------------------------------------------+


  +------DOCTOR-RECORDS------+
  |                          |
  |    1. add                |
  |    2. refresh            |
  |    3. search             |          enter 000 to cancel operation.
  |    4. update             |
  |    5. delete             |
  |                          |
  |    0. back               |
  |                          |
  +--------------------------+

  enter operation:
```

Figure 15: Menu of Doctor records and entry list.

Add

```
case 1: // add
    System.out.print("\n" + "   enter " + setBoldStart + "staff id" + setBoldEnd + " of doctor: \t\t");
    int d_id1 = scan_int.nextInt();
    if (d_id1 == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        doctor_records();
    }

    System.out.print("   enter " + setBoldStart + "name" + setBoldEnd + " of doctor: \t\t");
    String d_name = scan_string.nextLine();
    if (d_name.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        doctor_records();
    }

    System.out.print("   enter " + setBoldStart + "specialization" + setBoldEnd + " of doctor: \t");
    String d_spec = scan_string.nextLine();
    if (d_spec.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        doctor_records();
    }

    System.out.print("   enter " + setBoldStart + "contact number" + setBoldEnd + " of doctor: \t");
    String d_number = scan_string.nextLine();
    if (d_number.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        doctor_records();
    }
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   DOCTOR RECORD ADDED." + setBoldEnd);
    Thread.sleep(1500);

    d_collection.add(new Doctor(d_id1, d_name, d_spec, d_number));

    d_search = 0;

    d_oos = new ObjectOutputStream(new FileOutputStream(file_doctor));
    d_oos.writeObject(d_collection);
    d_oos.close();
    break;
```

```
+-----DOCTOR-RECORDS-----+
|                        |
|   1. add               |
|   2. refresh           |
|   3. search            |          enter 000 to cancel operation.
|   4. update            |
|   5. delete            |
|                        |
|   0. back              |
|                        |
+------------------------+

enter operation: 1

enter staff id of doctor:             9006
enter name of doctor:                 Ellen Feezor
enter specialization of doctor:       Psychology
enter contact number of doctor:       +971 54 913 1786
```

Figure 16: Output when user enters 1 in Doctor records menu which is Add operation.
As seen above, the user is adding a new entry with the staff ID number 9006 under the name Ellen Feezor.

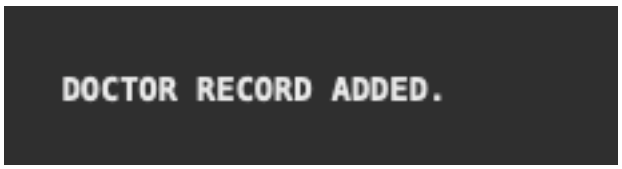


Figure 17: Output when Add operation is done correctly.

Refresh

```
case 2: // refresh
    doctor_records();
    d_search = 0;
    break;
```

```
+-----------------------------------------------------------------------------------------------------+
| STAFF ID                  DOCTOR NAME               SPECIALIZATION            CONTACT NUMBER        |
|-----------------------------------------------------------------------------------------------------|
| 9001                      Case Lowler               Pathology                 +971 56 876 8623      |
|-----------------------------------------------------------------------------------------------------|
| 9002                      Kaden Gerstner            General Physician         +971 54 490 1347      |
|-----------------------------------------------------------------------------------------------------|
| 9003                      Haylee Burnum             Surgery                   +971 52 147 9745      |
|-----------------------------------------------------------------------------------------------------|
| 9004                      Spencer Wishum            Dentistry                 +971 50 235 9745      |
|-----------------------------------------------------------------------------------------------------|
| 9005                      Angel Carfagno            Opthalmology              +971 55 975 8123      |
|-----------------------------------------------------------------------------------------------------|
| 9006                      Ellen Feezor              Psychology                +971 54 913 1786      |
+-----------------------------------------------------------------------------------------------------+


  +-----DOCTOR-RECORDS-----+
  |                        |
  |   1. add               |
  |   2. refresh           |
  |   3. search            |          enter 000 to cancel operation.
  |   4. update            |
  |   5. delete            |
  |                        |
  |   0. back              |
  |                        |
  +------------------------+

  enter operation:
```

Figure 18: Output when user enters 2 in Doctor records menu which is Refresh operation. As seen above, the entry list is updated and a new entry can be seen which is staff ID number 9006 under the name Ellen Feezor.

Search

```
case 3: // search
    boolean found = false;

    System.out.print("\n" + "   enter " + setBoldStart + "staff id" + setBoldEnd + " of doctor: \t");
    int d_id = scan_int.nextInt();
    if (d_id == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        doctor_records();
    }

    d_iteration = d_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.printf(
            " +-------------------------------------------------------------------------------------------------------------------------------------------------------------+"
                    + "\n | " + "%-30s%-30s%-30s%-30s%n",
            setBoldStart + "STAFF ID", "    DOCTOR NAME", "    SPECIALIZATION",
            "    CONTACT NUMBER                |" + setBoldEnd);
    while (d_iteration.hasNext()) {
        Doctor d_e = d_iteration.next();

        if (d_e.getDoctorID() == d_id) {
            d_search = 1;
            System.out.println(d_e);
            found = true;
        }
    }

    System.out.printf(
            " +-------------------------------------------------------------------------------------------------------------------------------------------------------------+");

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));

        d_search = 0;
        System.out.println(setBoldStart + "   DOCTOR RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }
    break;
```

```
+-----DOCTOR-RECORDS-----+
|                        |
|    1. add              |
|    2. refresh          |
|    3. search           |          enter 000 to cancel operation.
|    4. update           |
|    5. delete           |
|                        |
|    0. back             |
|                        |
+------------------------+

enter operation: 3

enter staff id of doctor:    9001
```

Figure 19: Output when the user enters 3 in Doctor records menu which is Search operation. As seen above, the user enters a staff ID number of a specific entry to search.

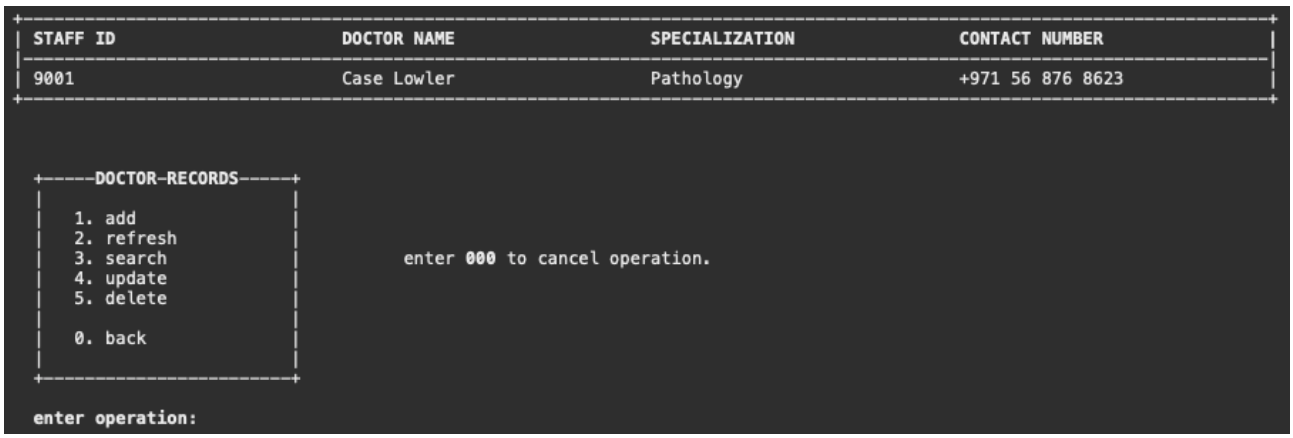


Figure 20: Output when Search operation is done correctly.
As seen above, the program displayed information of a specific doctor under staff ID number 9001.

Update

```
case 4: // update
    if (file_doctor.isFile()) {
        d_ois = new ObjectInputStream(new FileInputStream(file_doctor));
        d_collection = (ArrayList<Doctor>) d_ois.readObject();
        d_ois.close();
    }

    found = false;

    d_iteration = d_collection.iterator();
    d_list = d_collection.listIterator();

    System.out.print("   enter " + setBoldStart + "staff id" + setBoldEnd + " of doctor: ");
    d_id = scan_int.nextInt();
    if (d_id == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        doctor_records();
    }

    while (d_list.hasNext()) {
        Doctor d_e = d_list.next();

        if (d_e.getDoctorID() == d_id) {
            System.out.print(
                    "\n" + "   enter " + setBoldStart + "NEW staff id" + setBoldEnd + " of doctor: \t\t");
            d_id1 = scan_int.nextInt();
            if (d_id1 == 000) {
                System.out.println(new String(new char[70]).replace("\0", "\r\n"));
                System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
                Thread.sleep(1500);
                doctor_records();
            }

            System.out.print("   enter " + setBoldStart + "NEW name" + setBoldEnd + " of doctor: \t\t\t");
            d_name = scan_string.nextLine();
            if (d_name.equals("000")) {
                System.out.println(new String(new char[70]).replace("\0", "\r\n"));
                System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
                Thread.sleep(1500);
                doctor_records();
            }
```

```
            d_list.set(new Doctor(d_id1, d_name, d_spec, d_number));
            found = true;
        }
    }

    if (found) {
        d_oos = new ObjectOutputStream(new FileOutputStream(file_doctor));
        d_oos.writeObject(d_collection);
        d_oos.close();
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println("    " + setBoldStart + "DOCTOR RECORD UPDATED." + setBoldEnd);
        Thread.sleep(1500);
    }

    else {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   DOCTOR RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }
    break;
```

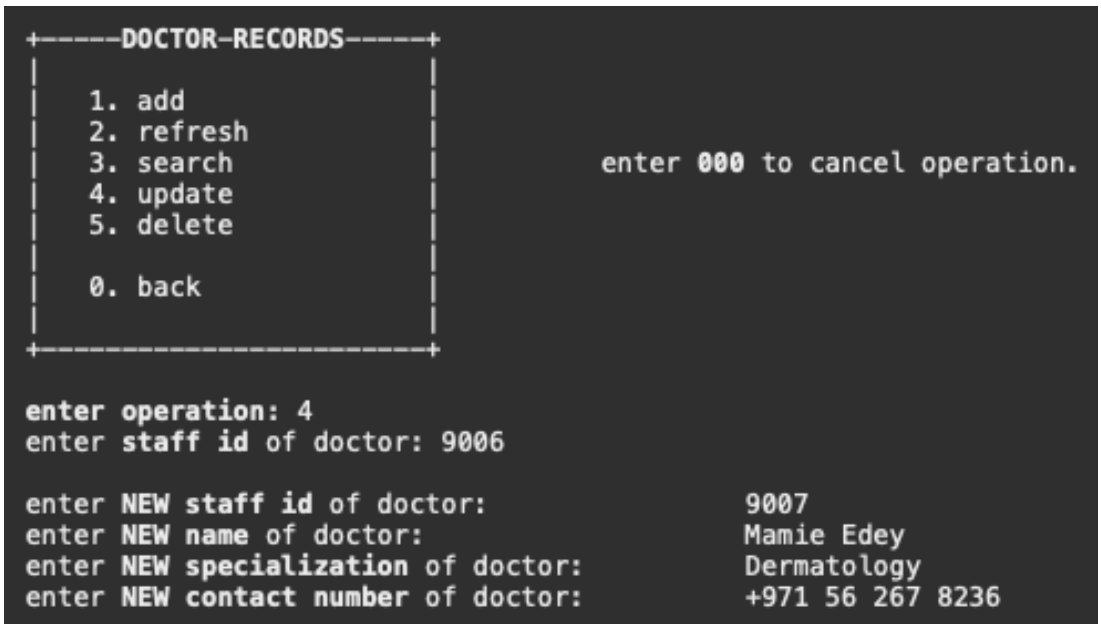


Figure 21: Output when user enters 4 in Doctor records menu which is Update operation.
As seen above, the data of staff ID number 9006 is being updated or replaced with new data entered by user.

Figure 22: Output when Update operation is done correctly.

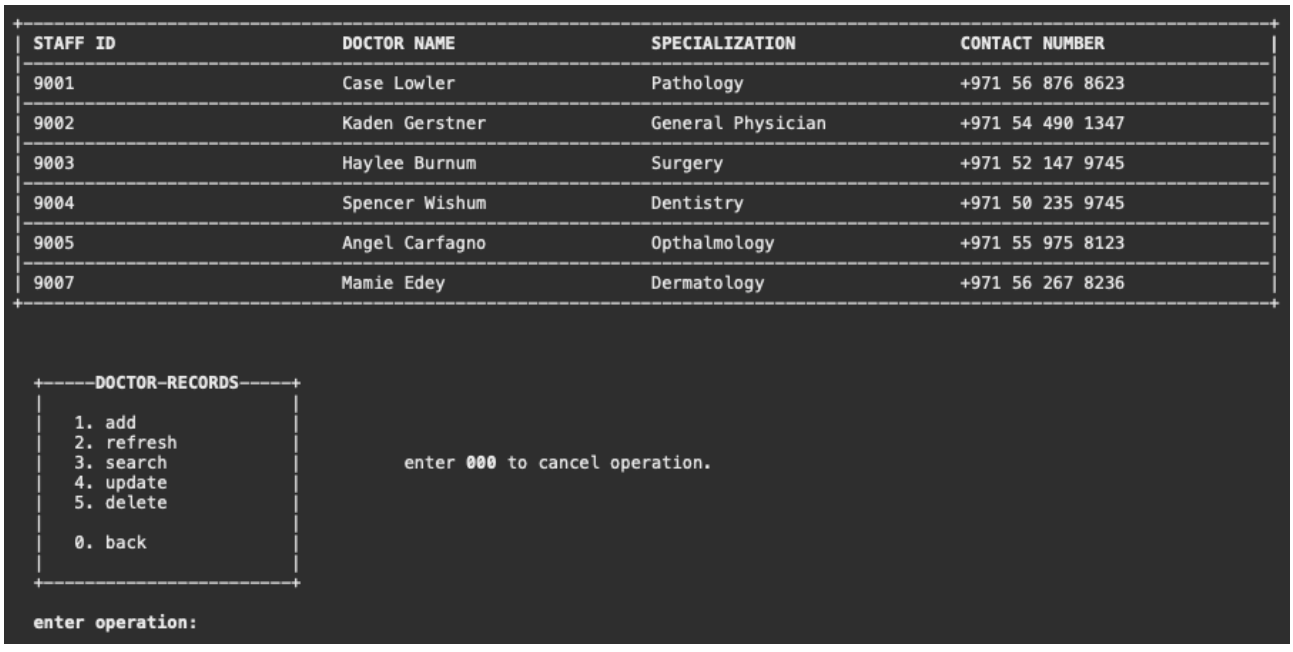


Figure 23: Output of entry list after Update operation.
As seen above, the old entry with the staff ID number 9006 is no longer existent as it was replaced by a new entry with a staff ID number of 9007 under the new name Mamie Edey.

Delete

```
case 5: // delete
    found = false;

    System.out.print("\n" + "   enter " + setBoldStart + "staff id" + setBoldEnd + " of doctor: \t");
    d_id = scan_int.nextInt();
    if (d_id == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        doctor_records();
    }

    d_iteration = d_collection.iterator();

    while (d_iteration.hasNext()) {
        Doctor d_e = d_iteration.next();

        if (d_e.getDoctorID() == d_id) {
            d_iteration.remove();
            found = true;

            System.out.println(new String(new char[70]).replace("\0", "\r\n"));

            d_search = 0;
            System.out.println(setBoldStart + "   DOCTOR RECORD DELETED." + setBoldEnd);
            Thread.sleep(1500);
        }
    }

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));

        d_search = 0;
        System.out.println(setBoldStart + "   DOCTOR RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }

    d_oos = new ObjectOutputStream(new FileOutputStream(file_doctor));
    d_oos.writeObject(d_collection);
    d_oos.close();
    break;
```

```
+-----DOCTOR-RECORDS-----+
|                        |
|   1. add               |
|   2. refresh           |
|   3. search            |          enter 000 to cancel operation.
|   4. update            |
|   5. delete            |
|                        |
|   0. back              |
|                        |
+------------------------+

enter operation: 5

enter staff id of doctor:    9007
```

Figure 24: Output when user enters 5 in Doctor records menu which is Delete operation. As seen above, the user is entering a specific staff ID number of an entry to delete.

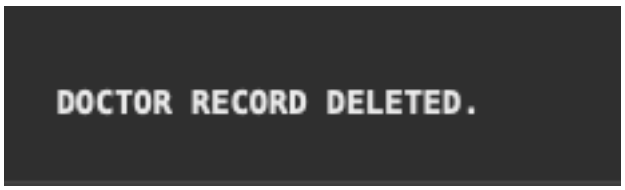


Figure 25: Output when Delete operation is done correctly.

Figure 26: Output of entry list after Delete operation.
As seen above, the entry for staff ID number 9007 is no longer existent as it was deleted.

Appointment Records

```
@SuppressWarnings("unchecked")
public static void appointment_records() throws Exception {
    record = 3;
    ArrayList<Appointment> a_collection = new ArrayList<Appointment>();
    ListIterator<Appointment> a_list = null;

    ArrayList<Patient> p_collection = new ArrayList<Patient>(); // initializes an array list of class patient
    ArrayList<Doctor> d_collection = new ArrayList<Doctor>(); // initializes an array list of class doctor

    Iterator<Patient> p_iteration = p_collection.iterator(); // initializes an iterator from class patient array
                                                             // list
    Iterator<Doctor> d_iteration = d_collection.iterator(); // initializes an iterator from class doctor array list

    File file_appointment = new File("appointment.txt");
    ObjectOutputStream a_oos = null;
    ObjectInputStream a_ois = null;

    if (file_appointment.isFile()) {
        a_ois = new ObjectInputStream(new FileInputStream(file_appointment));
        a_collection = (ArrayList<Appointment>) a_ois.readObject();
        a_ois.close();
    }

    /*
     * the following code below is used to import and read patient.txt and
     * doctor.txt in the java program. this then links all three classes altogether:
     * appointments, doctor, and patient. this is VERY essential to the access
     * operation which is case 3.
     */
    File file_patient = new File("patient.txt");
    ObjectInputStream p_ois = null;

    if (file_patient.isFile()) {
        p_ois = new ObjectInputStream(new FileInputStream(file_patient));
        p_collection = (ArrayList<Patient>) p_ois.readObject();
        p_ois.close();
    }

    File file_doctor = new File("doctor.txt");
    ObjectInputStream d_ois = null;

    if (file_doctor.isFile()) {
        d_ois = new ObjectInputStream(new FileInputStream(file_doctor));
        d_collection = (ArrayList<Doctor>) d_ois.readObject();
        d_ois.close();
    }

    int a_menu;
    boolean found = false;
    int a_search = 0;
```

```
do {

    Iterator<Appointment> a_iteration = a_collection.iterator();

    if (a_search == 0) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        while (a_iteration.hasNext()) {
            Appointment a_e = a_iteration.next();
            System.out.println(a_e);

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

    System.out.println("\n" + "\n" + "\n");
    System.out.println("   +-----" + setBoldStart + "APPOINTMENT" + setBoldEnd + "-" + setBoldStart + "RECORDS"
            + setBoldEnd + "-----+" + "\n    |" + "                                |");
    System.out.println("    |   1. add" + "                  |");
    System.out.println("    |   2. refresh" + "              |");
    System.out.println("    |   3. access" + "               |");
    System.out.println("    |   4. search" + "               |        enter " + setBoldStart + "000"
            + setBoldEnd + " to cancel operation.");
    System.out.println("    |   5. update" + "               |");
    System.out.println("    |   6. delete" + "               |" + "\n   |" + "                           |");
    System.out.println("    |   0. back" + "                 |" + "\n   |" + "                           |");
    System.out.println("    +-------------------------------+" + "\n");
    System.out.print(setBoldStart + "   enter operation: " + setBoldEnd);
    a_menu = scan_int.nextInt();
```

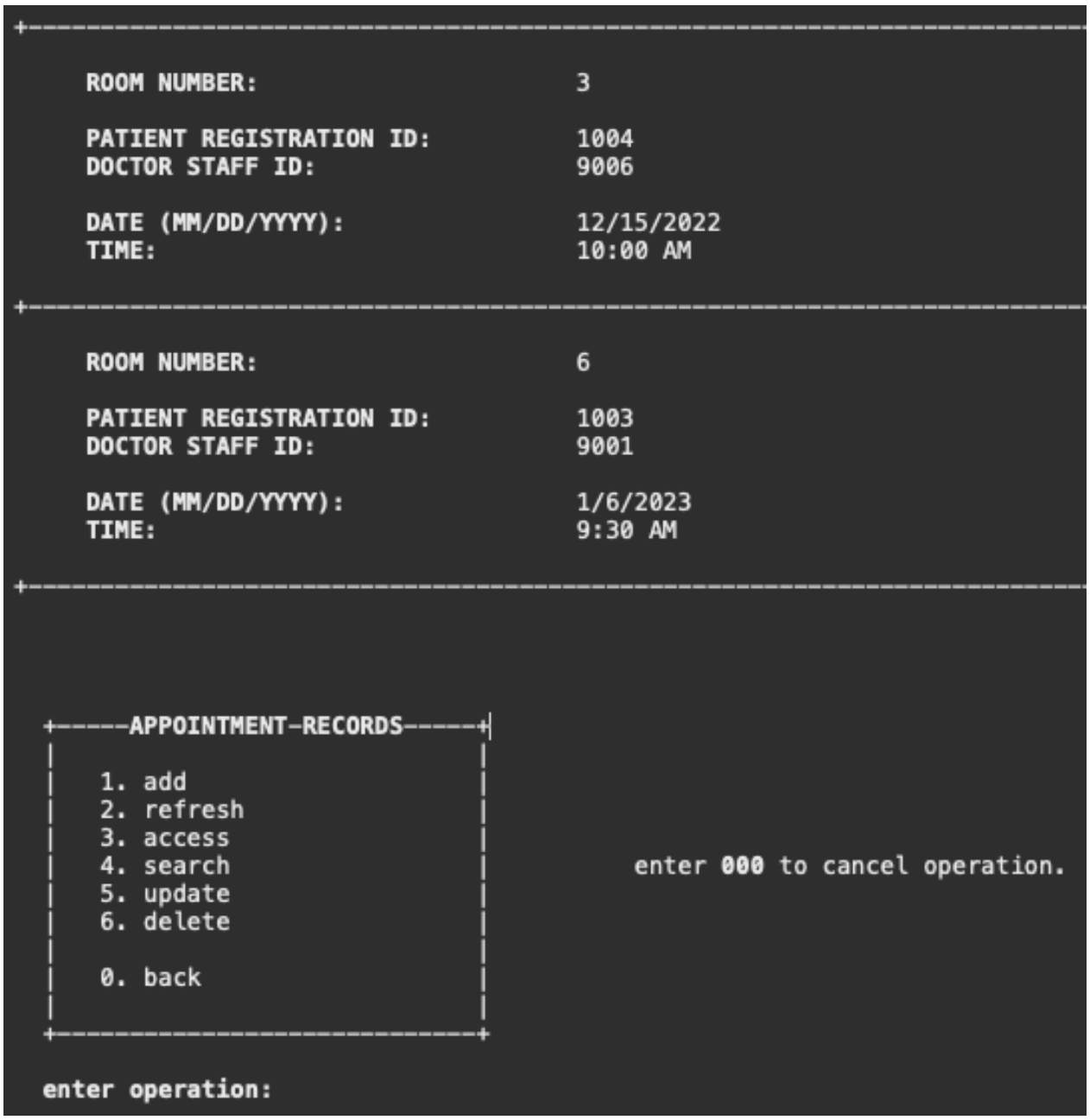
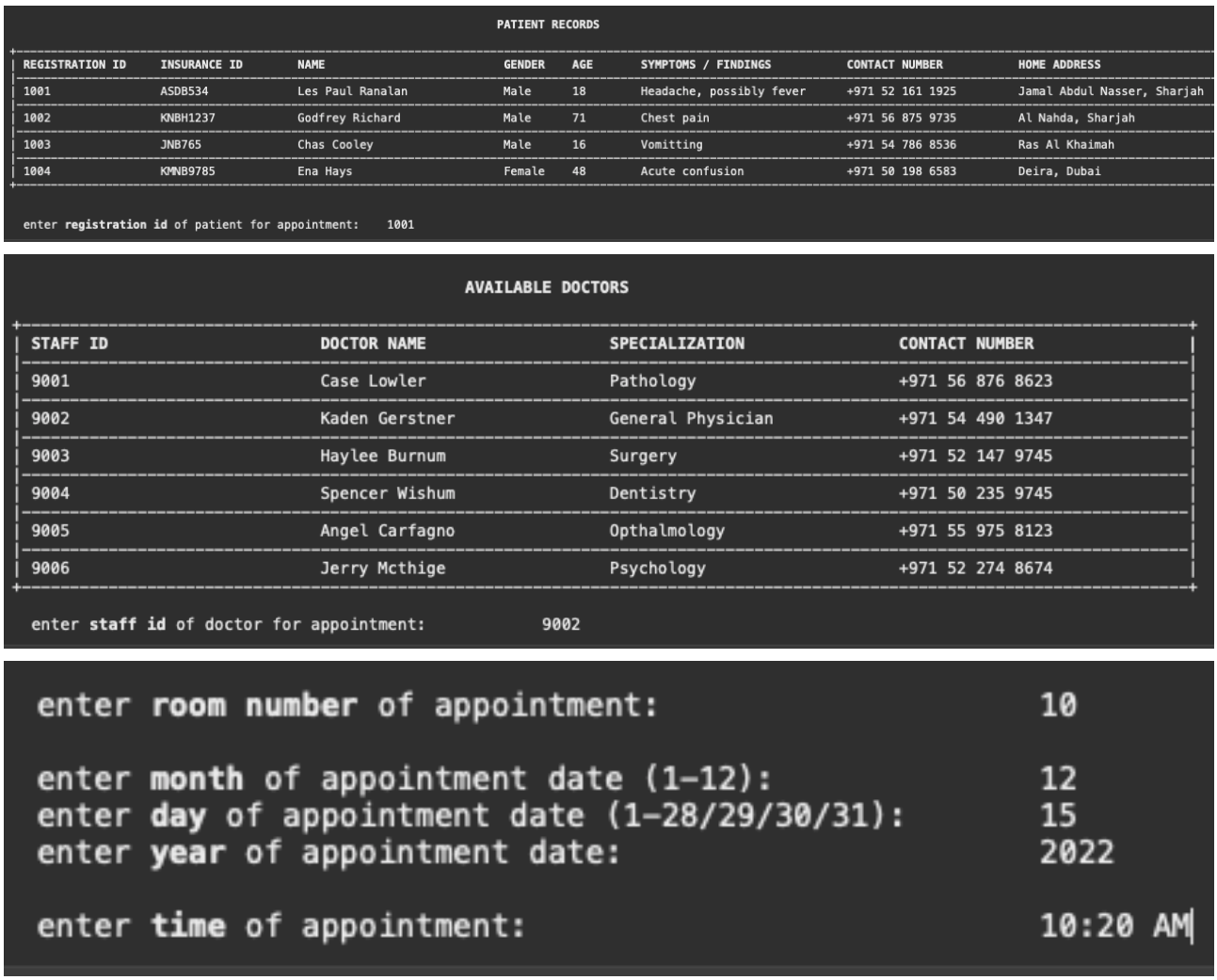


Figure 27: Menu of Appointment records and entry list.

Add

```java
case 1: // add
    p_iteration = p_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.printf(setBoldStart + "\t\t\t\t\t\t\t\t\t\tPATIENT RECORDS" + setBoldEnd
            + "\n\n +------------------------------------------------------------------------------------------------------------------------------------
            + "\n | " + "%-20s%-20s%-30s%-10s%-10s%-30s%-25s%-100s%n", setBoldStart + "REGISTRATION ID",
            "    INSURANCE ID", "    NAME", "    GENDER", "    AGE", "    SYMPTOMS / FINDINGS",
            "    CONTACT NUMBER",
            "    HOME ADDRESS                                                                   |"
                + setBoldEnd);
    while (p_iteration.hasNext()) {
        Patient p_e = p_iteration.next();
        System.out.println(p_e);
    }
    System.out.println(
            " +------------------------------------------------------------------------------------------------------------------------------------

    System.out.print("\n" + "   enter " + setBoldStart + "registration id" + setBoldEnd
            + " of patient for appointment: \t");
    int a_patientid1 = scan_int.nextInt();
    if (a_patientid1 == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    d_iteration = d_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.printf(setBoldStart + "\t\t\t\t\t\t\tAVAILABLE DOCTORS" + setBoldEnd
            + "\n\n +----------------------------------------------------------------------------------------------------------------------------------+"
            + "\n | " + "%-30s%-30s%-30s%-30s%n", setBoldStart + "STAFF ID", "    DOCTOR NAME",
            "    SPECIALIZATION", "    CONTACT NUMBER                    |" + setBoldEnd);
    while (d_iteration.hasNext()) {
        Doctor d_e = d_iteration.next();
        System.out.println(d_e);
    }
    System.out.println(
            " +----------------------------------------------------------------------------------------------------------------------------+\n");

    System.out.print(
            "   enter " + setBoldStart + "staff id" + setBoldEnd + " of doctor for appointment: \t\t");
    int a_doctorid1 = scan_int.nextInt();
    if (a_doctorid1 == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }
```

```java
System.out.println(new String(new char[70]).replace("\0", "\r\n"));
System.out.println(setBoldStart + "   APPOINTMENT RECORD ADDED." + setBoldEnd);
Thread.sleep(1500);

a_collection.add(new Appointment(a_room1, a_patientid1, a_doctorid1, a_datemonth1, a_dateday1,
        a_dateyear1, a_time));

a_search = 0;

a_oos = new ObjectOutputStream(new FileOutputStream(file_appointment));
a_oos.writeObject(a_collection);
a_oos.close();
break;
```



PATIENT RECORDS

REGISTRATION ID	INSURANCE ID	NAME	GENDER	AGE	SYMPTOMS / FINDINGS	CONTACT NUMBER	HOME ADDRESS
1001	ASDB534	Les Paul Ranalan	Male	18	Headache, possibly fever	+971 52 161 1925	Jamal Abdul Nasser, Sharjah
1002	KNBH1237	Godfrey Richard	Male	71	Chest pain	+971 56 875 9735	Al Nahda, Sharjah
1003	JNB765	Chas Cooley	Male	16	Vomitting	+971 54 786 8536	Ras Al Khaimah
1004	KMNB9785	Ena Hays	Female	48	Acute confusion	+971 50 198 6583	Deira, Dubai

enter **registration id** of patient for appointment: 1001

AVAILABLE DOCTORS

STAFF ID	DOCTOR NAME	SPECIALIZATION	CONTACT NUMBER
9001	Case Lowler	Pathology	+971 56 876 8623
9002	Kaden Gerstner	General Physician	+971 54 490 1347
9003	Haylee Burnum	Surgery	+971 52 147 9745
9004	Spencer Wishum	Dentistry	+971 50 235 9745
9005	Angel Carfagno	Opthalmology	+971 55 975 8123
9006	Jerry Mcthige	Psychology	+971 52 274 8674

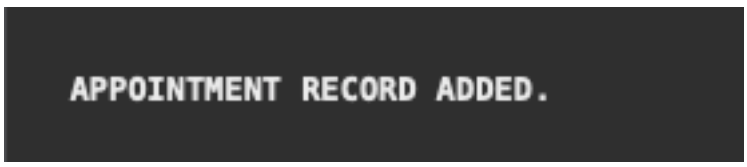
enter **staff id** of doctor for appointment: 9002

enter **room number** of appointment: 10

enter **month** of appointment date (1-12): 12
enter **day** of appointment date (1-28/29/30/31): 15
enter **year** of appointment date: 2022

enter **time** of appointment: 10:20 AM

Figures 28: Output when user enters 1 in Appointment records menu which is Add operation. As seen above, when adding a new entry for an appointment, the program prints both patient and doctor records and asks the user which patient the appointment is for and who is the doctor for that appointment along with other important information.



APPOINTMENT RECORD ADDED.

Figure 29: Output when Add operation is done correctly.

Refresh

```
case 2: // refresh
    a_search = 0;
    appointment_records();
    break;
```

```
+-------------------------------------------------------------------------------

    ROOM NUMBER:                    3

    PATIENT REGISTRATION ID:        1004
    DOCTOR STAFF ID:                9006

    DATE (MM/DD/YYYY):              12/15/2022
    TIME:                           10:00 AM

+-------------------------------------------------------------------------------

    ROOM NUMBER:                    6

    PATIENT REGISTRATION ID:        1003
    DOCTOR STAFF ID:                9001

    DATE (MM/DD/YYYY):              1/6/2023
    TIME:                           9:30 AM

+-------------------------------------------------------------------------------

    ROOM NUMBER:                    10

    PATIENT REGISTRATION ID:        1001
    DOCTOR STAFF ID:                9002

    DATE (MM/DD/YYYY):              12/15/2022
    TIME:                           10:20 AM

+-------------------------------------------------------------------------------



  +-----APPOINTMENT-RECORDS-----+
  |                             |
  |   1. add                    |
  |   2. refresh                |
  |   3. access                 |
  |   4. search                 |          enter 000 to cancel operation.
  |   5. update                 |
  |   6. delete                 |
  |                             |
  |   0. back                   |
  |                             |
  +-----------------------------+

  enter operation:
```

Figure 30: Output when the user enters 2 in Appointment records menu which is Refresh operation.

As seen above, the entry list is updated and a new entry with room number 10, patient registration ID number 1001, and doctor staff ID number 9002 can be seen.

Access

```
case 3: // access
    // initializes 3 booleans for finding doctor, patient, and appointment. all 3
    // are initially false.
    boolean doctor_found = false;
    boolean patient_found = false;
    boolean appointment_found = false;
    a_search = 1;

    /*
     * the follow code below is then asking the user to input the registration id of
     * patient and doctor staff id.
     */
    System.out.print("\n" + "   enter " + setBoldStart + "registration id" + setBoldEnd
            + " of patient in appointment: \t");
    int a_patientid = scan_int.nextInt();
    if (a_patientid == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    System.out.print(
            "   enter " + setBoldStart + "staff id" + setBoldEnd + " of doctor in appointment: \t\t");
    int a_doctorid = scan_int.nextInt();
    if (a_doctorid == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    a_iteration = a_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (a_iteration.hasNext()) {
        Appointment a_e = a_iteration.next();

        /*
         * the following code below is the program checking if the patient id entered by
         * the user is matching to the patient id and found in the array list of class
         * appointment. if patient id matches, it will print patient information of that
         * specific patient along with the appointment details above.
         */
        if (a_e.getAppointmentPatientID() == a_patientid) {
            System.out.println(a_e);
            appointment_found = true;
            System.out.println(
                    " +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+\n");
        }
    }
```

```
p_iteration = p_collection.iterator();
while (p_iteration.hasNext()) {
    Patient p_e = p_iteration.next();

    if (p_e.getPatientID() == a_patientid) {
        patient_found = true;
        System.out.printf(
                " \n +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+"
                        + "\n | " + "%-20s%-20s%-30s%-10s%-10s%-30s%-25s%-100s%n",
                setBoldStart + "REGISTRATION ID", "    INSURANCE ID", "    NAME", "    GENDER",
                "    AGE", "    SYMPTOMS / FINDINGS", "    CONTACT NUMBER",
                "    HOME ADDRESS                                                                                                                                                  |"
                        + setBoldEnd);
        System.out.println(p_e);
        System.out.println(
                " +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+");
    }
}

/*
 * the following code below is the program checking if the doctor id entered by
 * the user is matching to the doctor id and found in the array list of class
 * appointment. if doctor id matches, it will print doctor information of that
 * specific doctor along with the appointment details above.
 */
d_iteration = d_collection.iterator();
while (d_iteration.hasNext()) {
    Doctor d_e = d_iteration.next();

    if (d_e.getDoctorID() == a_doctorid) {
        doctor_found = true;
        System.out.printf(
                "\n +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+"
                        + "\n | " + "%-30s%-30s%-30s%-30s%n",
                setBoldStart + "STAFF ID", "    DOCTOR NAME", "    SPECIALIZATION",
                "    CONTACT NUMBER                |" + setBoldEnd);
        System.out.println(d_e);
        System.out.println(
                " +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+");
    }
}
/*
 * the following code below is when any of the ids are not matching with the
 * ones found in appointment records. it will print an error message.
 */
if (appointment_found != true) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println("\n   APPOINTMENT WITH PATIENT " + setBoldStart + "REGISTRATION ID NUMBER '"
            + a_patientid + setBoldEnd + "' AND DOCTOR " + setBoldStart + "STAFF ID NUMBER '"
            + a_doctorid + setBoldEnd + "' IS NOT REGISTERED IN APPOINTMENT RECORDS.");
}
```

```
if (patient_found != true) {
    System.out.println("\n   PATIENT " + setBoldStart + "REGISTRATION ID NUMBER '" + a_patientid
            + setBoldEnd + "' IS NOT REGISTERED IN PATIENT RECORDS.");
}

if (doctor_found != true) {
    System.out.println("\n   DOCTOR " + setBoldStart + "STAFF ID NUMBER '" + a_doctorid + setBoldEnd
            + "' IS NOT REGISTERED IN DOCTOR RECORDS.");
}

break;
```

Figure 31: Output when user enters 3 in Appointment records menu which is Access operation. As seen above, the user enters the patient registration ID number and the doctor staff ID number of an appointment in order for the Access operation to work.

Figure 32: Output when Access operation is done correctly.
As seen above, all three entries containing important information are displayed in one window; that being the data under patient registration ID number 1001 and the data under the doctor staff ID number 9002.

Search

```
case 4: // search
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));

    int a_search_menu;
    a_search = 1;

    do {

        System.out.println("\n" + "\n" + "\n");

        System.out.println("    +-----" + setBoldStart + "APPOINTMENT" + setBoldEnd + "-" + setBoldStart
                + "RECORDS" + setBoldEnd + "-----+" + "\n    |                                |");
        System.out.println(
                "    |    1. search via doctor id" + "    |" + "\n    |                                |");
        System.out.println("    |    2. search via month" + "        |");
        System.out.println("    |    3. search via year        |" + "          enter " + setBoldStart + "000"
                + setBoldEnd + " to cancel operation.");
        System.out.println("    |    4. search via date" + "         |\n    |                                |");
        System.out.println("    |    0. back                   |" + "\n    |                                |");
        System.out.println("    +------------------------------+" + "\n");
        System.out.print(setBoldStart + "    enter operation: " + setBoldEnd);

        a_search_menu = scan_int.nextInt();
```

```
+-----APPOINTMENT-RECORDS-----+
|                              |
|    1. search via doctor id   |
|                              |
|    2. search via month       |
|    3. search via year        |          enter 000 to cancel operation.
|    4. search via date        |
|                              |
|    0. back                   |
|                              |
+------------------------------+

enter operation:
```

Figure 33: Output when user enters 4 in the Appointment records MAIN menu which is Search operation.

As seen above, the user is taken to a sub menu within the Search operation. In this sub menu, the user can search under different conditions such as the date or the doctor staff ID number.

Search via Doctor ID

```
case 1: // search via doctor id
    found = false;

    System.out.print("\n" + "   enter " + setBoldStart + "staff id" + setBoldEnd
            + " of doctor in appointment: \t");

    a_doctorid = scan_int.nextInt();
    if (a_doctorid == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    a_iteration = a_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));

    while (a_iteration.hasNext()) {
        Appointment a_e = a_iteration.next();

        if (a_e.getAppointmentDoctorID() == a_doctorid) {
            System.out.println(a_e);
            found = true;
        }
    }

    System.out.println(
            " +-------------------------------------------------------------------------------------------------------------------------------------------------------------------+");

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   APPOINTMENT RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        found = false;
    }
    break;
```

```
+-----APPOINTMENT-RECORDS------+
|                              |
|    1. search via doctor id   |
|                              |
|    2. search via month       |
|    3. search via year        |        enter 000 to cancel operation.
|    4. search via date        |
|                              |
|    0. back                   |
|                              |
+------------------------------+

enter operation: 1

enter staff id of doctor in appointment:     9001
```

Figure 34: Output when user enters 1 in Appointment records SUB menu which is Search via doctor ID operation.
As seen above, the user entered doctor staff ID number 9001 to display current appointments with that specific doctors.

Figure 35: Output when Search via doctor ID operation is done correctly.
As seen above, the program displayed current appointments for that specific doctor under the staff ID number 9001.

Search via Month

```
case 2: // search via month
    System.out.print("\n" + "    enter " + setBoldStart + "month" + setBoldEnd
            + " of appointment (1-12): \t");
    int a_datemonth = scan_int.nextInt();
    if (a_datemonth == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "    OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    a_iteration = a_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (a_iteration.hasNext()) {
        Appointment a_e = a_iteration.next();

        if (a_e.getAppointmentDateMonth() == a_datemonth) {
            System.out.println(a_e);
            found = true;
        }
    }

    System.out.println(
            " +-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+");

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "    APPOINTMENT RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        found = false;
    }
    break;
```

```
+-----APPOINTMENT-RECORDS-----+
|                             |
|   1. search via doctor id   |
|                             |
|   2. search via month       |
|   3. search via year        |          enter 000 to cancel operation.
|   4. search via date        |
|                             |
|   0. back                   |
|                             |
+-----------------------------+

enter operation: 2

enter month of appointment (1-12):    12
```

Figure 36: Output when user enters 2 in Appointment records SUB menu which is Search via month operation. As seen above, the user is searching for appointments that are in the 12th month which is the month of December.

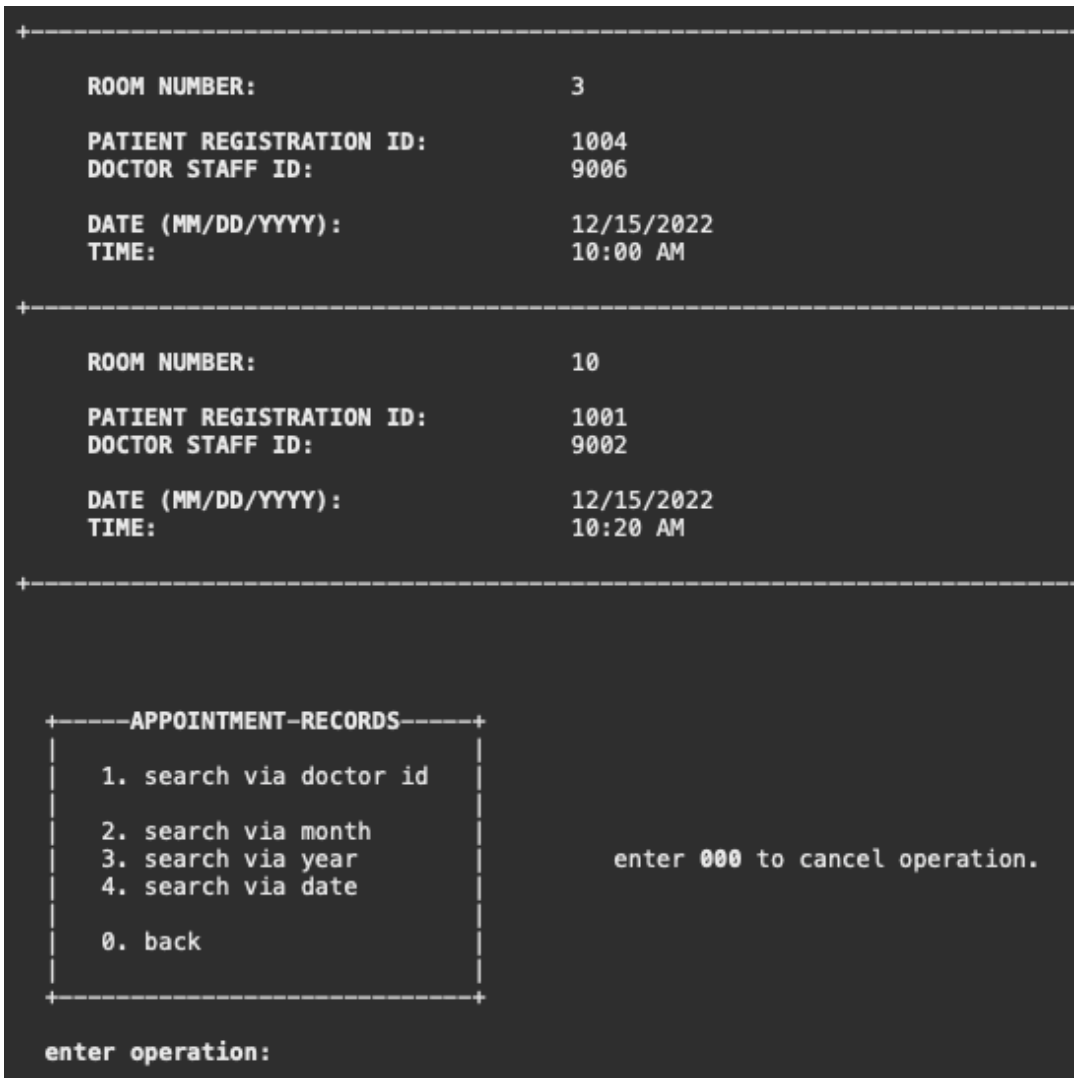


Figure 37: Output when Search via month is done correctly. As seen above, the program displayed all of the appointed that were assigned to the 12th month which is the month of December.

Search via Year

```
case 3: // search via year
    System.out.print("\n   enter " + setBoldStart + "year" + setBoldEnd + " of appointment:  \t\t");
    int a_dateyear = scan_int.nextInt();
    if (a_dateyear == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    a_iteration = a_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (a_iteration.hasNext()) {
        Appointment a_e = a_iteration.next();

        if (a_e.getAppointmentDateYear() == a_dateyear) {
            System.out.println(a_e);
            found = true;
        }
    }

    System.out.println(
            " +-------------------------------------------------------------------------------------------------------------------------------------------------------------------+");

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   APPOINTMENT RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        found = false;
    }
    break;
```

```
+-----APPOINTMENT-RECORDS-----+
|                             |
|   1. search via doctor id   |
|                             |
|   2. search via month       |
|   3. search via year        |          enter 000 to cancel operation.
|   4. search via date        |
|                             |
|   0. back                   |
|                             |
+-----------------------------+

enter operation: 3

enter year of appointment:           2023
```

Figure 38: Output when user enters 3 in Appointment records SUB menu which is Search via year operation.
As seen above, the user is searching for appointments that are in the year 2023.

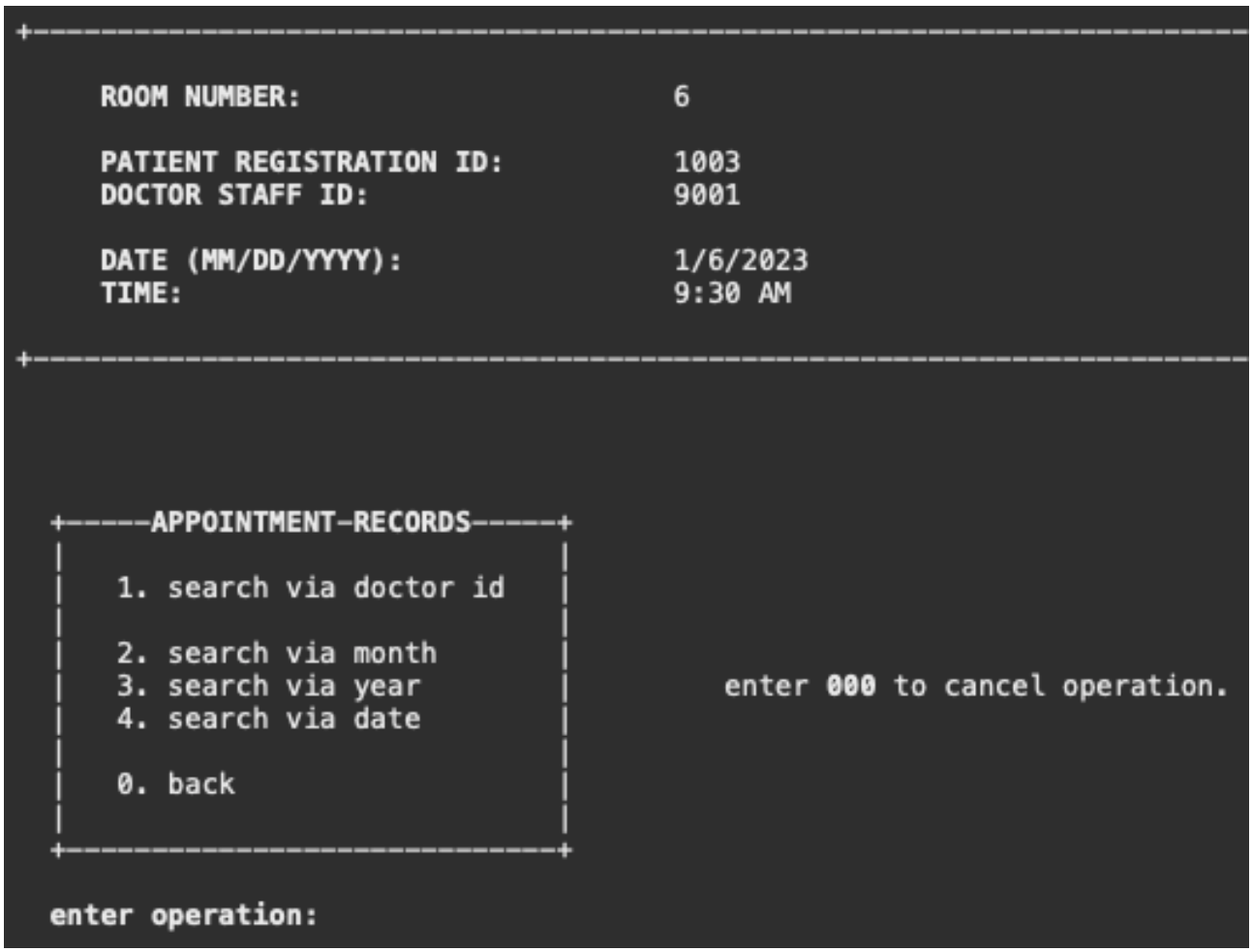


Figure 39: Output when Search via year operation is done correctly.
As seen above, the program printed appointments that are dated on the year 2023.

Search via Date

```java
case 4: // search via date
    System.out.print("\n   enter " + setBoldStart + "month" + setBoldEnd
            + " of appointment date (1-12): \t\t");
    a_datemonth = scan_int.nextInt();
    if (a_datemonth == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    System.out.print("   enter " + setBoldStart + "day" + setBoldEnd
            + " of appointment date (1-28/29/30/31): \t");
    int a_dateday = scan_int.nextInt();
    if (a_dateday == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    System.out.print(
            "   enter " + setBoldStart + "year" + setBoldEnd + " of appointment date: \t\t\t");
    a_dateyear = scan_int.nextInt();
    if (a_dateyear == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    a_iteration = a_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (a_iteration.hasNext()) {
        Appointment a_e = a_iteration.next();

        if (a_e.getAppointmentDateMonth() == a_datemonth && a_e.getAppointmentDateDay() == a_dateday
                && a_e.getAppointmentDateYear() == a_dateyear) {
            System.out.println(a_e);
            found = true;
        }
    }

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

```java
if (found != true) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   APPOINTMENT RECORD NOT FOUND." + setBoldEnd);
    Thread.sleep(1500);
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    found = false;
}
break;
```

```
+-----APPOINTMENT-RECORDS-----+
|                             |
|   1. search via doctor id   |
|                             |
|   2. search via month       |
|   3. search via year        |         enter 000 to cancel operation.
|   4. search via date        |
|                             |
|   0. back                   |
|                             |
+-----------------------------+

enter operation: 4

enter month of appointment date (1-12):               12
enter day of appointment date (1-28/29/30/31):        25
enter year of appointment date:                       2022
```

Figure 40: Output when user enters 4 in Appointment records SUB menu which is Search via date operation. As seen above, the user is searching an appointment with a specific date.

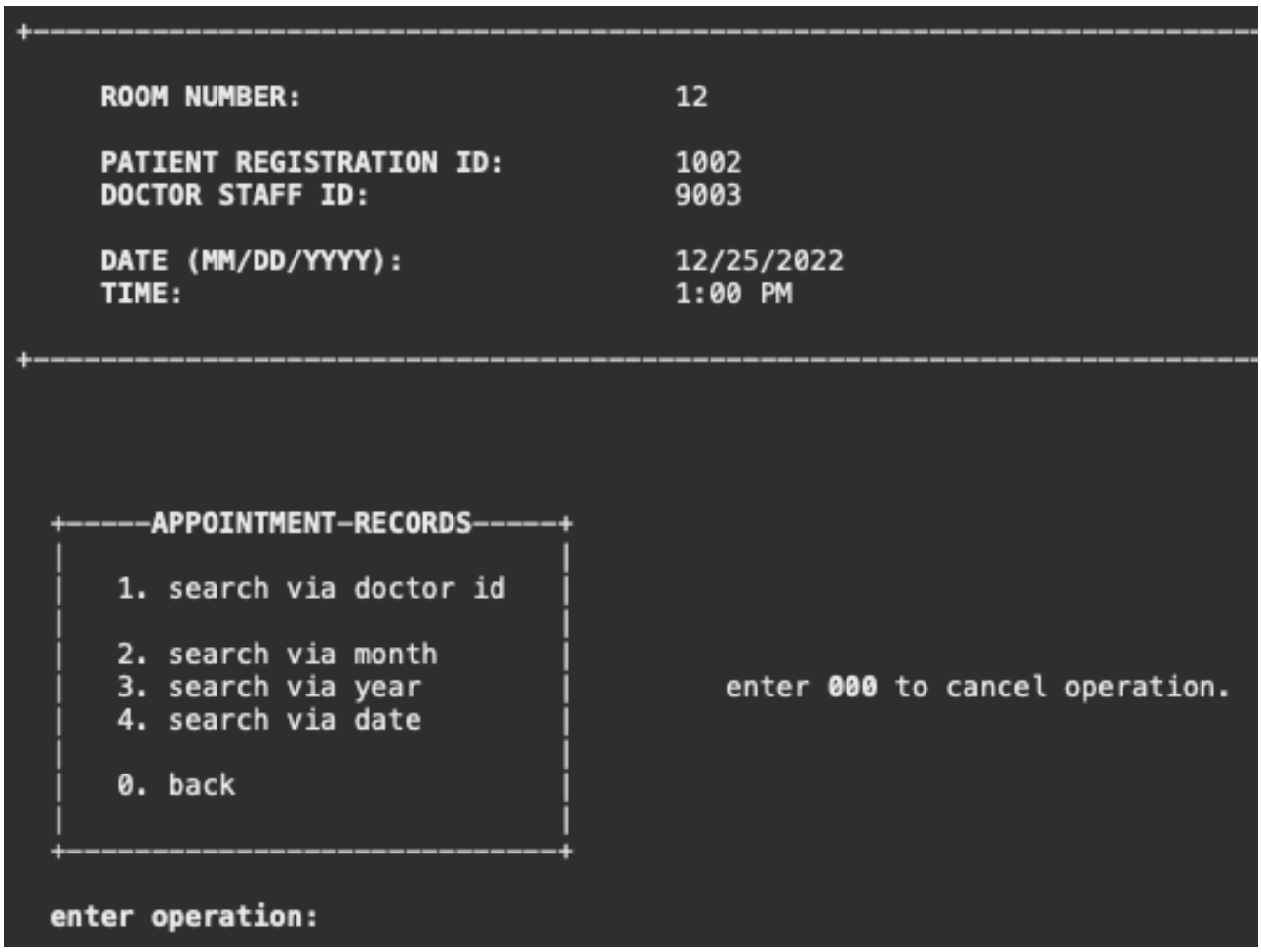


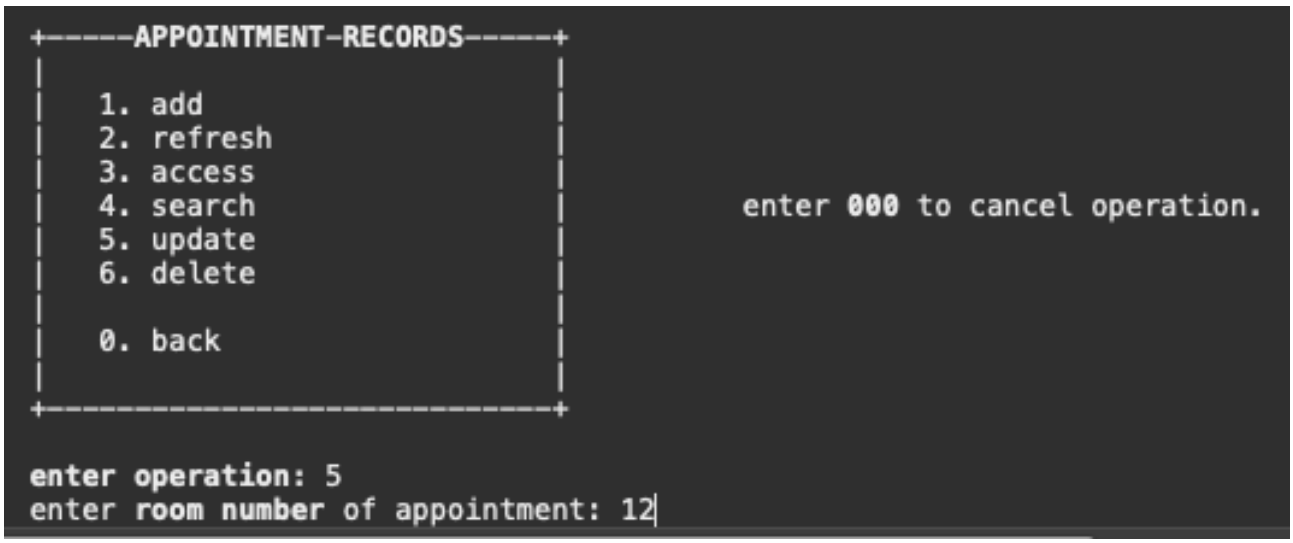
Figure 41: Output when Search via date operation is done correctly.
As seen above, the program displayed an appointment assigned on a specific date which is 12/25/2022 or December 25 of 2022.

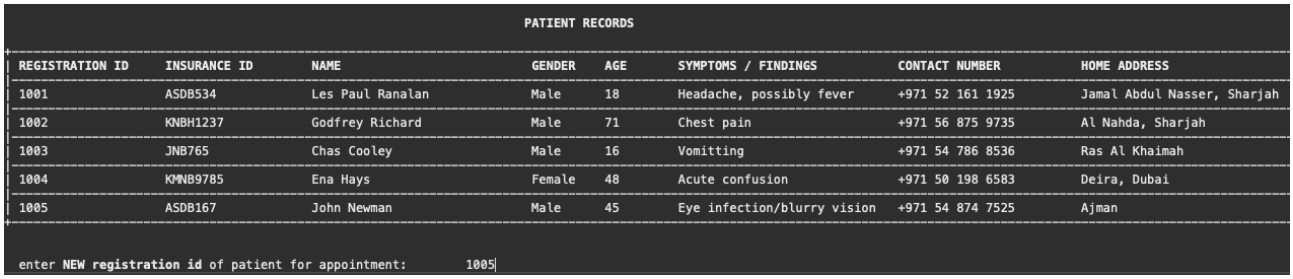
Update

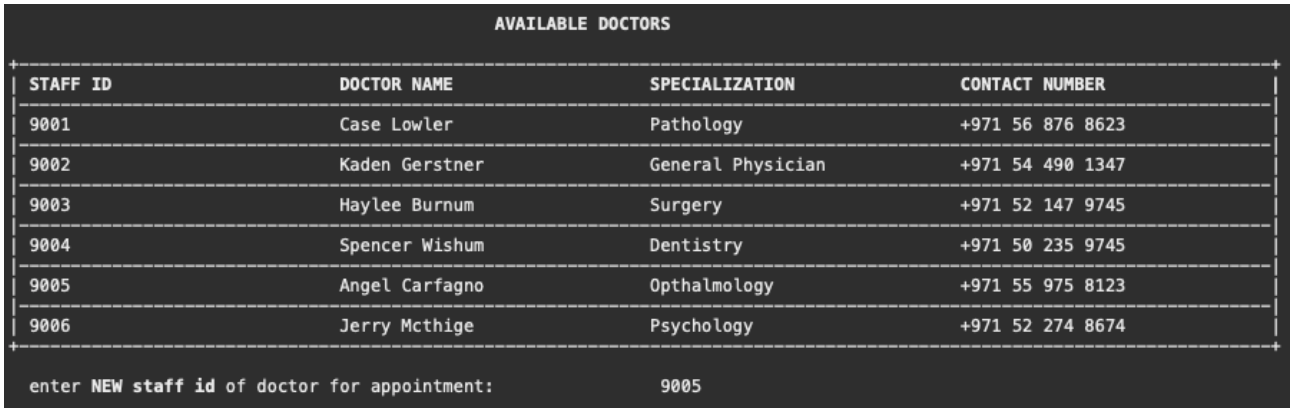
```
case 5: // update
    if (file_appointment.isFile()) {
        a_ois = new ObjectInputStream(new FileInputStream(file_appointment));
        a_collection = (ArrayList<Appointment>) a_ois.readObject();
        a_ois.close();
    }

    found = false;

    a_iteration = a_collection.iterator();
    a_list = a_collection.listIterator();

    System.out.print("   enter " + setBoldStart + "room number" + setBoldEnd + " of appointment: ");
    int a_room = scan_int.nextInt();
    if (a_room == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    while (a_list.hasNext()) {
        Appointment a_e = a_list.next();

        if (a_e.getAppointmentRoom() == a_room) {
            p_iteration = p_collection.iterator();
            System.out.println(new String(new char[70]).replace("\0", "\r\n"));
            System.out.printf(setBoldStart + "\t\t\t\t\t\t\t\t\t\tPATIENT RECORDS" + setBoldEnd
                    + "\n\n +----------------------------------------------------------------------------------------------------------------------------------------------------------------+"
                    + "\n | " + "%-20s%-20s%-30s%-10s%-10s%-30s%-25s%-100s%n",
                    setBoldStart + "REGISTRATION ID", "    INSURANCE ID", "    NAME", "    GENDER",
                    "    AGE", "    SYMPTOMS / FINDINGS", "    CONTACT NUMBER",
                    "    HOME ADDRESS                                                                                                                                                  |"
                            + setBoldEnd);
            while (p_iteration.hasNext()) {
                Patient p_e = p_iteration.next();
                System.out.println(p_e);
            }
            System.out.println(
                    " +----------------------------------------------------------------------------------------------------------------------------------------------------------------+");

            System.out.print("\n" + "   enter " + setBoldStart + "NEW registration id" + setBoldEnd
                    + " of patient for appointment: \t");
            a_patientid1 = scan_int.nextInt();
            if (a_patientid1 == 000) {
                System.out.println(new String(new char[70]).replace("\0", "\r\n"));
                System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
                Thread.sleep(1500);
                appointment_records();
            }
```

```
d_iteration = d_collection.iterator();
System.out.println(new String(new char[70]).replace("\0", "\r\n"));
System.out.printf(setBoldStart + "\t\t\t\t\t\t\tAVAILABLE DOCTORS" + setBoldEnd
        + "\n\n +-----------------------------------------------------------------------------------------------------------------------------------------------+"
        + "\n | " + "%-30s%-30s%-30s%-30s%n", setBoldStart + "STAFF ID", "    DOCTOR NAME",
        "    SPECIALIZATION", "    CONTACT NUMBER                |" + setBoldEnd);
while (d_iteration.hasNext()) {
    Doctor d_e = d_iteration.next();
    System.out.println(d_e);
}
System.out.println(
        " +-----------------------------------------------------------------------------------------------------------------------------------------------+\n");

System.out.print("   enter " + setBoldStart + "NEW staff id" + setBoldEnd
        + " of doctor for appointment: \t\t");
a_doctorid1 = scan_int.nextInt();
if (a_doctorid1 == 000) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
    Thread.sleep(1500);
    appointment_records();
}

System.out.println(new String(new char[70]).replace("\0", "\r\n"));
System.out.print("   enter " + setBoldStart + "NEW room number" + setBoldEnd
        + " of appointment: \t\t\t");
a_room1 = scan_int.nextInt();
if (a_room1 == 000) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
    Thread.sleep(1500);
    appointment_records();
}

System.out.print("\n   enter " + setBoldStart + "NEW month" + setBoldEnd
        + " of appointment date (1-12): \t\t\t");
a_datemonth1 = scan_int.nextInt();
if (a_datemonth1 == 000) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
    Thread.sleep(1500);
    appointment_records();
}
```

```
            a_list.set(new Appointment(a_room1, a_patientid1, a_doctorid1, a_datemonth1, a_dateday1,
                    a_dateyear1, a_time));
            found = true;
        }
    }

    if (found) {
        a_oos = new ObjectOutputStream(new FileOutputStream(file_appointment));
        a_oos.writeObject(a_collection);
        a_oos.close();
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println("    " + setBoldStart + "APPOINTMENT RECORD UPDATED." + setBoldEnd);
        Thread.sleep(1500);
    }

    else {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "    APPOINTMENT RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }
    break;
```







Figures 42: Output when user enters 5 in Appointment records MAIN menu which is Update operation.
As seen above, the user is trying to update an appointment in room number 12.

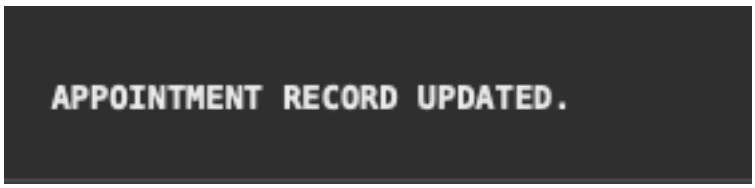


Figure 43: Output when Update operation is done correctly.

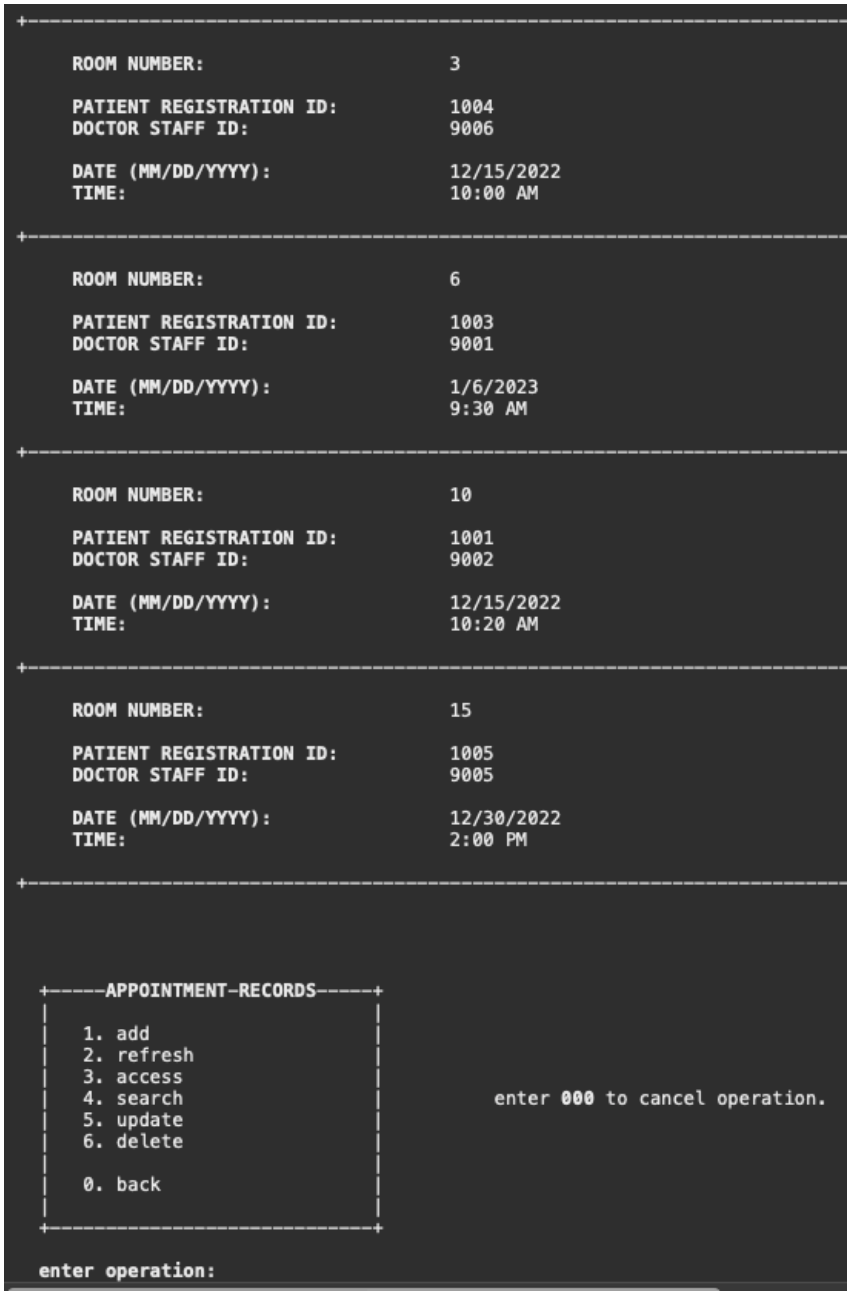


Figure 44: Output of updated entry list of Appoint records after Update operation.
As seen above, the appointment assigned to room number 12 is no longer existent as it was changed or updated to be assigned to room number 15.

Delete

```
case 6: // delete
    System.out
            .print("\n" + "   enter " + setBoldStart + "room number" + setBoldEnd + " of appointment: \t");
    a_room = scan_int.nextInt();
    if (a_room == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        appointment_records();
    }

    a_iteration = a_collection.iterator();
    while (a_iteration.hasNext()) {
        Appointment a_e = a_iteration.next();

        if (a_e.getAppointmentRoom() == a_room) {
            a_iteration.remove();
            found = true;
            a_search = 1;
            System.out.println(new String(new char[70]).replace("\0", "\r\n"));
            a_search = 0;
            System.out.println(setBoldStart + "   APPOINTMENT RECORD DELETED." + setBoldEnd);
            Thread.sleep(1500);
        }
    }

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        a_search = 0;
        System.out.println(setBoldStart + "   APPOINTMENT RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }

    a_oos = new ObjectOutputStream(new FileOutputStream(file_appointment));
    a_oos.writeObject(a_collection);
    a_oos.close();
    break;
```

```
+------APPOINTMENT-RECORDS------+
|                               |
|    1. add                     |
|    2. refresh                 |
|    3. access                  |
|    4. search                  |          enter 000 to cancel operation.
|    5. update                  |
|    6. delete                  |
|                               |
|    0. back                    |
|                               |
+-------------------------------+

enter operation: 6

enter room number of appointment:    15
```

Figure 44: Output when user enters 6 in Appointment records MAIN menu which is Delete operation.
As seen above, the user enters the appointment room number to be deleted.

```
   APPOINTMENT RECORD DELETED.
```

Figure 45: Output when Delete operation is done correctly.

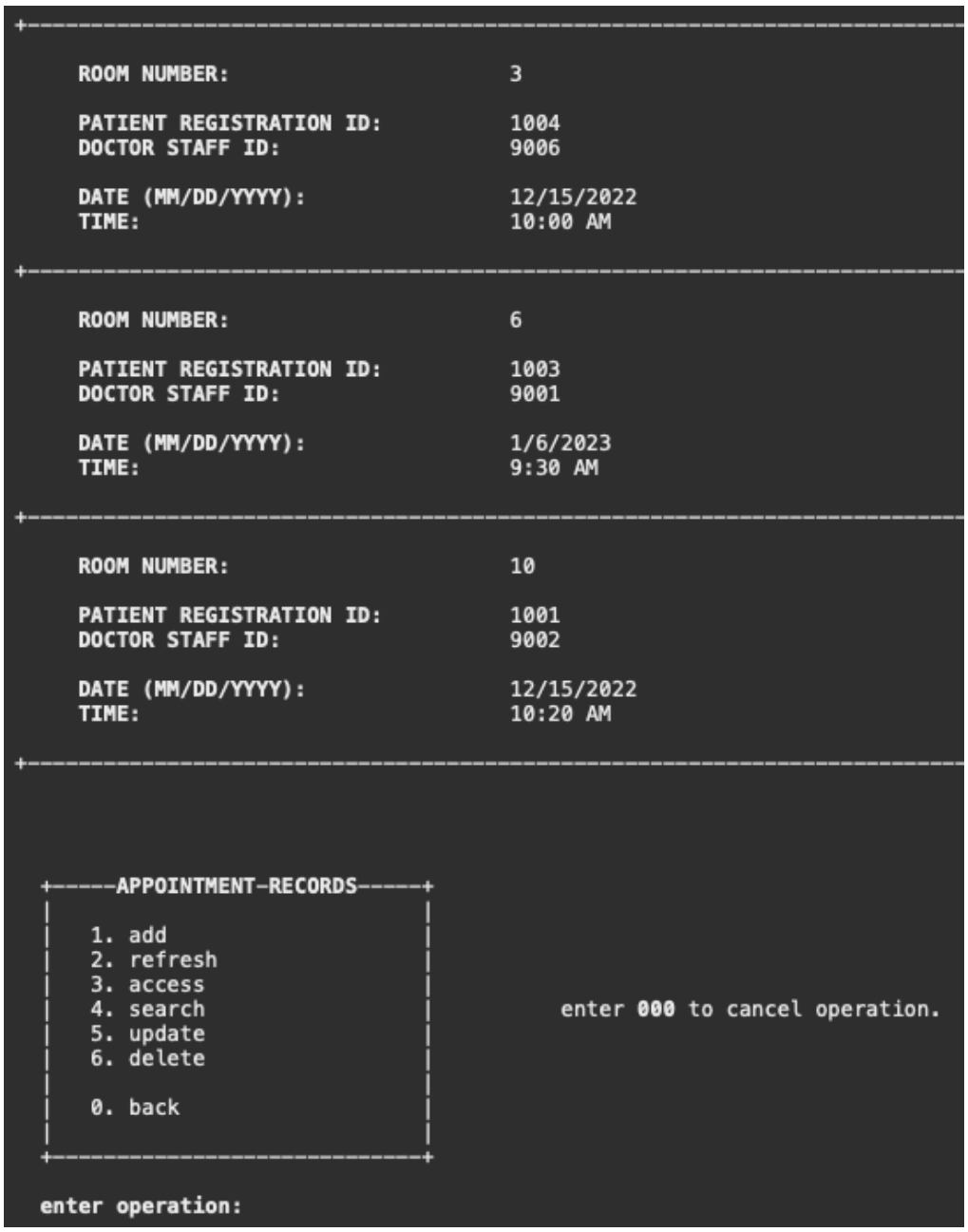


Figure 46: Output of updated entry list of Appointment records after Delete operation.
As seen above, the appointment assigned to room number 15 is no longer existent as it was deleted.

Medicine Records

```
@SuppressWarnings("unchecked")
public static void medicine_records() throws Exception {
    record = 4;
    ArrayList<Medicine> m_collection = new ArrayList<Medicine>();
    ListIterator<Medicine> m_list = null;

    File file_medicine = new File("medicine.txt");
    ObjectOutputStream m_oos = null;
    ObjectInputStream m_ois = null;

    if (file_medicine.isFile()) {
        m_ois = new ObjectInputStream(new FileInputStream(file_medicine));
        m_collection = (ArrayList<Medicine>) m_ois.readObject();
        m_ois.close();
    }

    int m_menu;
    int m_search = 0;

    do {
        Iterator<Medicine> m_iteration = m_collection.iterator();

        if (m_search == 0) {
            System.out.println(new String(new char[70]).replace("\0", "\r\n"));

            while (m_iteration.hasNext()) {
                Medicine m_e = m_iteration.next();
                System.out.println(m_e);
            }
            System.out.println(
                    "\n +-----------------------------------------------------------------------------------------------
        }

        System.out.println("\n" + "\n" + "\n");
        System.out.println("    +-----" + setBoldStart + "MEDICINE" + setBoldEnd + "-" + setBoldStart + "RECORDS"
                + setBoldEnd + "-----+" + "\n    |" + "                                |");
        System.out.println("    |   1. add" + "                |");
        System.out.println("    |   2. refresh" + "            |");
        System.out.println("    |   3. search" + "             |          enter " + setBoldStart + "000"
                + setBoldEnd + " to cancel operation.");
        System.out.println("    |   4. update" + "             |");
        System.out.println("    |   5. delete" + "             |" + "\n    |" + "                          |");
        System.out.println("    |   0. back" + "               |" + "\n    |" + "                          |");
        System.out.println("    +--------------------------+" + "\n");
        System.out.print(setBoldStart + "    enter operation: " + setBoldEnd);
        m_menu = scan_int.nextInt();
```

Figure 47: Menu of Medicine records and entry list.

Add

```
case 1: // add
    System.out.print("\n" + "   enter " + setBoldStart + "id number" + setBoldEnd + " of medicine: \t\t\t");
    int m_id1 = scan_int.nextInt();
    if (m_id1 == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    System.out.print("   enter " + setBoldStart + "name" + setBoldEnd + " of medicine: \t\t\t\t");
    String m_name = scan_string.nextLine();
    if (m_name.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    System.out.print("   enter " + setBoldStart + "weight / mass" + setBoldEnd + " of medicine (mg): \t\t");
    String m_strength = scan_string.nextLine();
    if (m_strength.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    System.out.print("   enter " + setBoldStart + "definition" + setBoldEnd + " of medicine: \t\t\t");
    String m_definition = scan_string.nextLine();
    if (m_definition.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    System.out.print("   enter " + setBoldStart + "mitigation(s)" + setBoldEnd + " of medicine: \t\t\t");
    String m_treatment = scan_string.nextLine();
    if (m_treatment.equals("000")) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }
```

```
m_collection.add(new Medicine(m_id1, m_name, m_strength, m_definition, m_treatment, m_sideeffects,
        m_instructions, m_note, m_expirydatemonth, m_expirydateday, m_expirydateyear,
        m_approvaldatemonth1, m_approvaldateday1, m_approvaldateyear1));

System.out.println(new String(new char[70]).replace("\0", "\r\n"));
System.out.println(setBoldStart + "   MEDICINE RECORD ADDED." + setBoldEnd);
Thread.sleep(1500);

m_oos = new ObjectOutputStream(new FileOutputStream(file_medicine));
m_oos.writeObject(m_collection);
m_oos.close();

m_search = 0;
break;
```

```
+-----MEDICINE-RECORDS-----+
|                          |
|   1. add                 |
|   2. refresh             |
|   3. search              |          enter 000 to cancel operation.
|   4. update              |
|   5. delete              |
|                          |
|   0. back                |
|                          |
+--------------------------+

enter operation: 1

enter id number of medicine:                          5005
enter name of medicine:                               Acetaminophen
enter weight / mass of medicine (mg):                 500
enter definition of medicine:                         It is a pain reliever and a fever reducer
enter mitigation(s) of medicine:                      Athritis, backache, tootheache, sore throat, colds, flu, fever
enter possible side effects of medicine:              Stomach pain, loss of appetite, tiredness, jaundice, dark-colored urine
enter instructions of medicine:                       Take medication orally with a glass of water
enter important note(s) of medicine:                  Do not take if allergic to medication. Do not take if diagnosed with severe liver disease

enter month of expiry date (1-12):                    10
enter day of expiry date (1-28/29/30/31):             19
enter year of expiry date:                            2030

enter month of manufacturing date (1-12):             12
enter day of manufacturing date (1-28/29/30/31):      15
enter year of manufacturing date:                     2022
```

Figure 48: Output when user enters 1 in Medicine records menu which is Add operation.

```
MEDICINE RECORD ADDED.
```

Figure 49: Output when Add operation is done correctly.

Refresh

```
case 2: // refresh
    medicine_records();
    m_search = 0;
    break;
```

```
    MEDICINE DEFINITION:              It is usually used together with diet to lower blood levels of bad cholesterol to increase levels of good cholesterol.
    MITIGATION:                       High cholesterol, heart attack, stroke, coronary heart disease, type II diabetes
    POSSIBLE SIDE EFFECTS:            Joint pain, stuffy nose, sore throat, diarrhear, pain in arms or legs

    MANUFACTURING DATE (MM/DD/YYYY):  8/14/2015
    EXPIRY DATE (MM/DD/YYYY):         3/2/2023

    INSTRUCTIONS:                     Take medication orally with a glass of water
    IMPORT NOTE(S):                   Do not take if pregnant or breastfeeding. Do not take if diagnosed with liver disease

+-----------------------------------------------------------------------------------------------------------------------------

    MEDICINE ID:                      5004
    MEDICINE NAME:                    Metformin (500mg)

    MEDICINE DEFINITION:              It is usually used together with diet to lower high blood sugar levels. It lowers the amount of glubose absorbed from intestines, improves insulin sensitivity.
    MITIGATION:                       Type II diabetes, pre-diabetes, polycystic ovarian syndrome, cysts in ovaries
    POSSIBLE SIDE EFFECTS:            Low blood sugar, nausea, upset stomach or diarrhea

    MANUFACTURING DATE (MM/DD/YYYY):  8/4/2025
    EXPIRY DATE (MM/DD/YYYY):         5/17/2027

    INSTRUCTIONS:                     Take medication orally with a glass of water
    IMPORT NOTE(S):                   DO not take if pregnant or breastfeeding. May develop lactic acidosis; dangerous build-up of lactic acid in the blood. Do not take if age is under 10 years old.

+-----------------------------------------------------------------------------------------------------------------------------

    MEDICINE ID:                      5005
    MEDICINE NAME:                    Acetaminophen (500mg)

    MEDICINE DEFINITION:              It is a pain reliever and a fever reducer
    MITIGATION:                       Athritis, backache, tootheache, sore throat, colds, flu, fever
    POSSIBLE SIDE EFFECTS:            Stomach pain, loss of appetite, tiredness, jaundice, dark-colored urine

    MANUFACTURING DATE (MM/DD/YYYY):  12/15/2022
    EXPIRY DATE (MM/DD/YYYY):         10/19/2030

    INSTRUCTIONS:                     Take medication orally with a glass of water
    IMPORT NOTE(S):                   Do not take if allergic to medication. Do not take if diagnosed with severe liver disease

+-----------------------------------------------------------------------------------------------------------------------------



  +-----MEDICINE-RECORDS-----+
  |                          |
  |   1. add                 |
  |   2. refresh             |
  |   3. search              |        enter 000 to cancel operation.
  |   4. update              |
  |   5. delete              |
  |                          |
  |   0. back                |
  |                          |
  +--------------------------+

  enter operation:
```

Figure 50: Output when user enters 2 in Medicine records menu which is Refresh operation. As seen above, a new entry with a medicine ID number 5005, which was added by the user earlier, can be seen.

Search

```
case 3: // search
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));

    int m_search_menu;
    m_search = 1;

    do {
        System.out.println("\n\n\n");
        System.out.println(
                "   +------------" + setBoldStart + "MEDICINE" + setBoldEnd + "-" + setBoldStart + "RECORDS"
                        + setBoldEnd + "------------+" + "\n   |                                                  |");
        System.out.println("   |   1. search via medicine id              |"
                + "\n   |                                                  |");
        System.out.println("   |   2. search via manufacturing month      |");
        System.out.println("   |   3. search via manufacturing year       |");
        System.out.println("   |   4. search via manufacturing date       |" + "          enter "
                + setBoldStart + "000" + setBoldEnd + " to cancel operation."
                + "\n   |                                                  |");
        System.out.println("   |   5. search via expiry month             |");
        System.out.println("   |   6. search via expiry year              |");
        System.out.println("   |   7. search via expiry date              |"
                + "\n   |                                                  |");
        System.out.println("   |   0. back                                |"
                + "\n   |                                                  |");
        System.out.println("   +--------------------------------------------+" + "\n");
        System.out.print(setBoldStart + "   enter operation: " + setBoldEnd);

        m_search_menu = scan_int.nextInt();
```

```
+------------MEDICINE-RECORDS------------+
|                                         |
|   1. search via medicine id             |
|                                         |
|   2. search via manufacturing month     |
|   3. search via manufacturing year      |
|   4. search via manufacturing date      |          enter 000 to cancel operation.
|                                         |
|   5. search via expiry month            |
|   6. search via expiry year             |
|   7. search via expiry date             |
|                                         |
|   0. back                               |
|                                         |
+-----------------------------------------+

enter operation:
```

Figure 51: Output when user enters 3 in Medicine records MAIN menu which is Search operation. As seen above, the program takes the user to a sub menu of Search wherein different search operations with specific conditions can be utilized.

Search via Medicine ID

```
case 1: // search via id
    boolean found = false;

    System.out.print("\n    enter " + setBoldStart + "id number" + setBoldEnd + " of medicine: \t");
    int m_id = scan_int.nextInt();
    if (m_id == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "    OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    m_iteration = m_collection.iterator();

    while (m_iteration.hasNext()) {
        Medicine m_e = m_iteration.next();

        if (m_e.getMedicineID() == m_id) {
            System.out.println(new String(new char[70]).replace("\0", "\r\n"));
            System.out.println(m_e);
            found = true;
            m_search = 1;
        }
    }

    System.out.println(
            "\n +-----------------------------------------------------------------------------------------

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "    MEDICINE NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }
    break;
```

```
+------------MEDICINE-RECORDS------------+
|                                         |
|    1. search via medicine id            |
|                                         |
|    2. search via manufacturing month    |
|    3. search via manufacturing year     |
|    4. search via manufacturing date     |          enter 000 to cancel operation.
|                                         |
|    5. search via expiry month           |
|    6. search via expiry year            |
|    7. search via expiry date            |
|                                         |
|    0. back                              |
|                                         |
+-----------------------------------------+

enter operation: 1

enter id number of medicine:          5001
```

Figure 52: Output when user enters 1 in Medicine records SUB menu which is Search via medicine ID operation.
As seen above, the user is entering the medicine ID number to be searched.

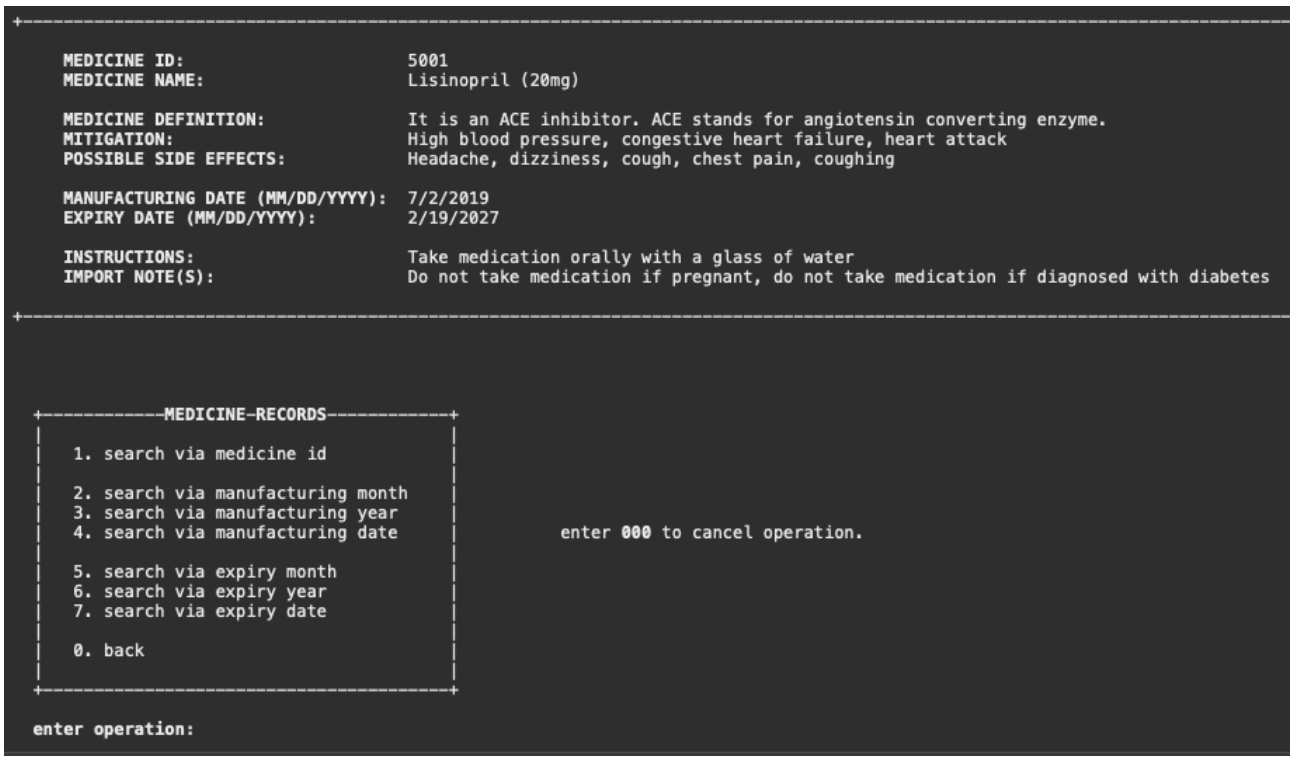


Figure 53: Output when Search via medicine ID operation is done correctly.
As seen above, the program displayed information for the specific medicine ID number 5001.

Search via Manufacturing Month

```
case 2: // search via manufacturing month
    found = false;

    System.out.print("\n" + "   enter " + setBoldStart + "month" + setBoldEnd
            + " of manufacturing date (1-12): \t");
    int m_approvaldatemonth = scan_int.nextInt();
    if (m_approvaldatemonth == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    m_iteration = m_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (m_iteration.hasNext()) {
        Medicine m_e = m_iteration.next();

        if (m_e.getMedicineApprovalDateMonth() == m_approvaldatemonth) {
            System.out.println(m_e);
            found = true;
            m_search = 1;
        }
    }

    System.out.println(
            "\n +-------------------------------------------------------------------------------

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   MEDICINE RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }
    break;
```

```
+-----------MEDICINE-RECORDS-----------+
|                                       |
|   1. search via medicine id           |
|                                       |
|   2. search via manufacturing month   |
|   3. search via manufacturing year    |
|   4. search via manufacturing date    |          enter 000 to cancel operation.
|                                       |
|   5. search via expiry month          |
|   6. search via expiry year           |
|   7. search via expiry date           |
|                                       |
|   0. back                             |
|                                       |
+---------------------------------------+

enter operation: 2

enter month of manufacturing date (1-12):    8
```

Figure 54: Output when user enters 2 in Medicine records SUB menu which is Search via manufacturing month operation.
As seen above, the user is entering the month of manufacturing date of medicine to be searched.

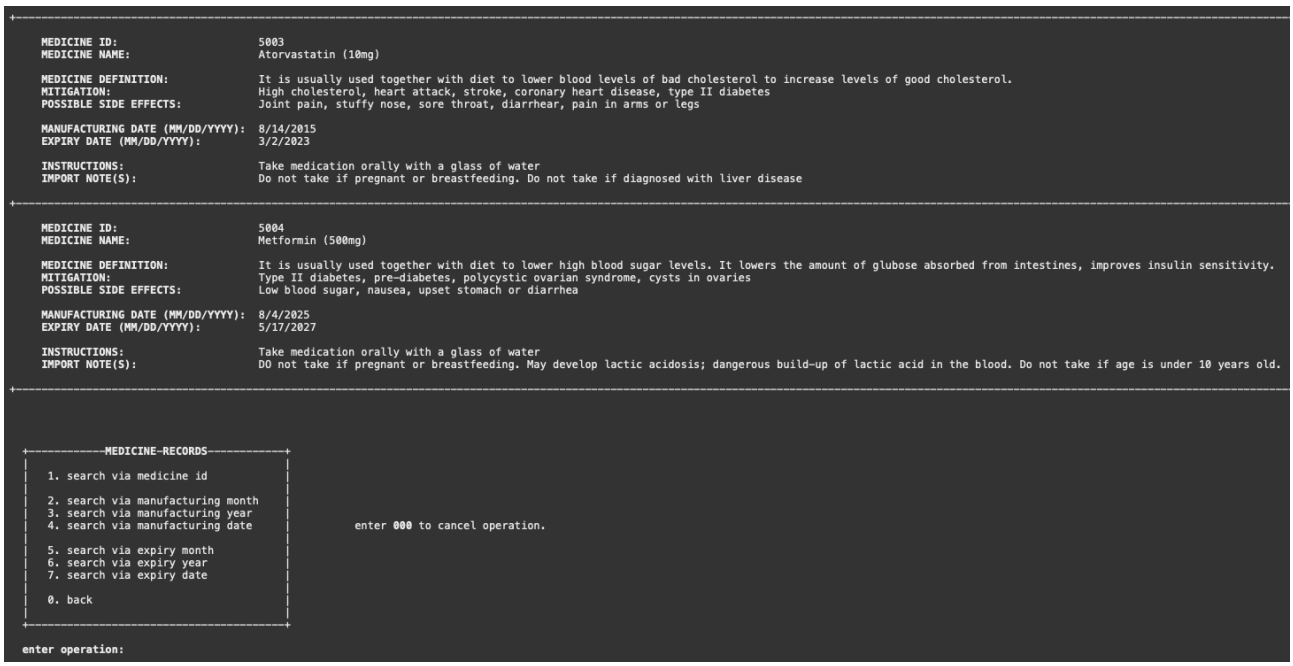


Figure 55: Output when Search via manufacturing month operation is done correctly.
As seen above, the program displayed two medicines that were manufactured on the 8th month or the month of August.

Search via Manufacturing Year

```
case 3: // search via manufacturing year
    found = false;

    System.out.print(
            "\n" + "   enter " + setBoldStart + "year" + setBoldEnd + " of manufacturing
    int m_approvaldateyear = scan_int.nextInt();
    if (m_approvaldateyear == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    m_iteration = m_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (m_iteration.hasNext()) {
        Medicine m_e = m_iteration.next();

        if (m_e.getMedicineApprovalDateYear() == m_approvaldateyear) {
            System.out.println(m_e);
            found = true;
            m_search = 1;
        }
    }

    System.out.println(
            "\n +-------------------------------------------------------------------------

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   MEDICINE RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }

    break;
```

```
+-----------MEDICINE-RECORDS-----------+
|                                       |
|   1. search via medicine id           |
|                                       |
|   2. search via manufacturing month   |
|   3. search via manufacturing year    |
|   4. search via manufacturing date    |          enter 000 to cancel operation.
|                                       |
|   5. search via expiry month          |
|   6. search via expiry year           |
|   7. search via expiry date           |
|                                       |
|   0. back                             |
|                                       |
+---------------------------------------+

enter operation: 3

enter year of manufacturing date:    2019
```

Figure 56: Output when user enters 3 in Medicine records SUB menu which is Search via manufacturing year operation.
As seen above, the user is entering the year of manufacturing date of the medicine to be searched.

Figure 57: Output when Search via manufacturing year operation is done correctly.
As seen above, the program displayed a medicine that matches with the entered year of manufacturing date by the user which is 2019.

Search via Manufacturing Date

```
case 4: // search via manufacturing date
    found = false;

    System.out.print("\n" + "   enter " + setBoldStart + "month" + setBoldEnd
            + " of manufacturing date (1-12): \t\t");
    m_approvaldatemonth = scan_int.nextInt();
    if (m_approvaldatemonth == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    System.out.print("   enter " + setBoldStart + "day" + setBoldEnd
            + " of manufacturing date (1-28/29/30/31): \t");
    int m_approvaldateday = scan_int.nextInt();
    if (m_approvaldateday == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    System.out.print(
            "   enter " + setBoldStart + "year" + setBoldEnd + " of manufacturing date: \t\t\t");
    m_approvaldateyear = scan_int.nextInt();
    if (m_approvaldateyear == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    m_iteration = m_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (m_iteration.hasNext()) {
        Medicine m_e = m_iteration.next();

        if (m_e.getMedicineApprovalDateMonth() == m_approvaldatemonth
                && m_e.getMedicineApprovalDateDay() == m_approvaldateday
                && m_e.getMedicineApprovalDateYear() == m_approvaldateyear) {
            System.out.println(m_e);
            found = true;
            m_search = 1;
        }
    }

    System.out.println(
            "\n +-------------------------------------------------------------------------------
```

```
m_iteration = m_collection.iterator();
System.out.println(new String(new char[70]).replace("\0", "\r\n"));
while (m_iteration.hasNext()) {
    Medicine m_e = m_iteration.next();

    if (m_e.getMedicineApprovalDateMonth() == m_approvaldatemonth
            && m_e.getMedicineApprovalDateDay() == m_approvaldateday
            && m_e.getMedicineApprovalDateYear() == m_approvaldateyear) {
        System.out.println(m_e);
        found = true;
        m_search = 1;
    }
}

System.out.println(
        "\n +-------------------------------------------------------------------------

if (found != true) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   MEDICINE RECORD NOT FOUND." + setBoldEnd);
    Thread.sleep(1500);
}

break;
```

```
+-----------MEDICINE-RECORDS-----------+
|                                       |
|   1. search via medicine id           |
|                                       |
|   2. search via manufacturing month   |
|   3. search via manufacturing year    |
|   4. search via manufacturing date    |          enter 000 to cancel operation.
|                                       |
|   5. search via expiry month          |
|   6. search via expiry year           |
|   7. search via expiry date           |
|                                       |
|   0. back                             |
|                                       |
+---------------------------------------+

enter operation: 4

enter month of manufacturing date (1-12):            1
enter day of manufacturing date (1-28/29/30/31):     25
enter year of manufacturing date:                    2020
```

Figure 58: Output when user enters 4 in Medicine records SUB menu which is Search via manufacturing date operation.
As seen above, the user is entering the manufacturing date of the medicine to be searched.

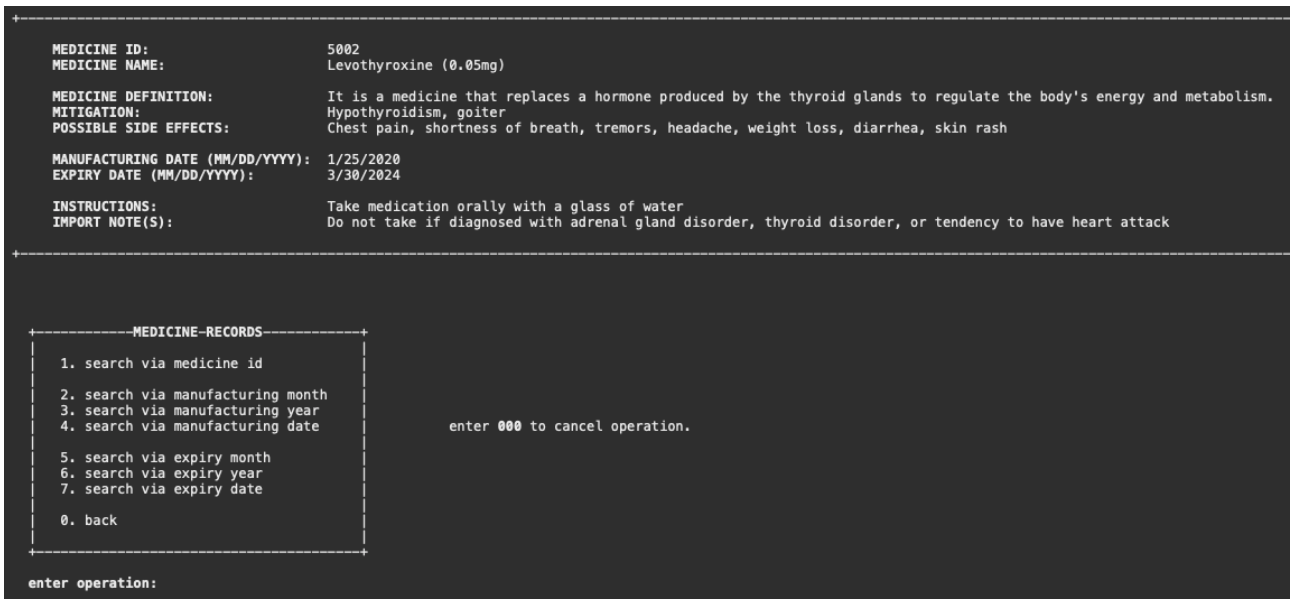


Figure 59: Output when Search via manufacturing date operation is done correctly.
As seen above, the program displayed a medicine that matches with the manufacturing date entered by the user which is 1/25/2020 or January 25 of 2020.

Search via Expiry Month

```
case 5: // search via expiry month
    found = false;

    System.out.print("\n" + "   enter " + setBoldStart + "month" + setBoldEnd
            + " of expiry date (1-12): \t");
    m_expirydatemonth = scan_int.nextInt();
    if (m_expirydatemonth == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    m_iteration = m_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (m_iteration.hasNext()) {
        Medicine m_e = m_iteration.next();

        if (m_e.getMedicineExpiryDateMonth() == m_expirydatemonth) {
            System.out.println(m_e);
            found = true;
            m_search = 1;
        }
    }

    System.out.println(
            "\n +-------------------------------------------------------------------------------

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   MEDICINE RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }

    break;
```

```
+-----------MEDICINE-RECORDS-----------+
|                                       |
|   1. search via medicine id           |
|                                       |
|   2. search via manufacturing month   |
|   3. search via manufacturing year    |
|   4. search via manufacturing date    |          enter 000 to cancel operation.
|                                       |
|   5. search via expiry month          |
|   6. search via expiry year           |
|   7. search via expiry date           |
|                                       |
|   0. back                             |
|                                       |
+---------------------------------------+

enter operation: 5

enter month of expiry date (1-12):   3
```

Figure 60: Output when user enters 5 in Medicine records SUB menu which is Search via expiry month operation.
As seen above, the user is entering the month of expiry date of medicine to be searched.

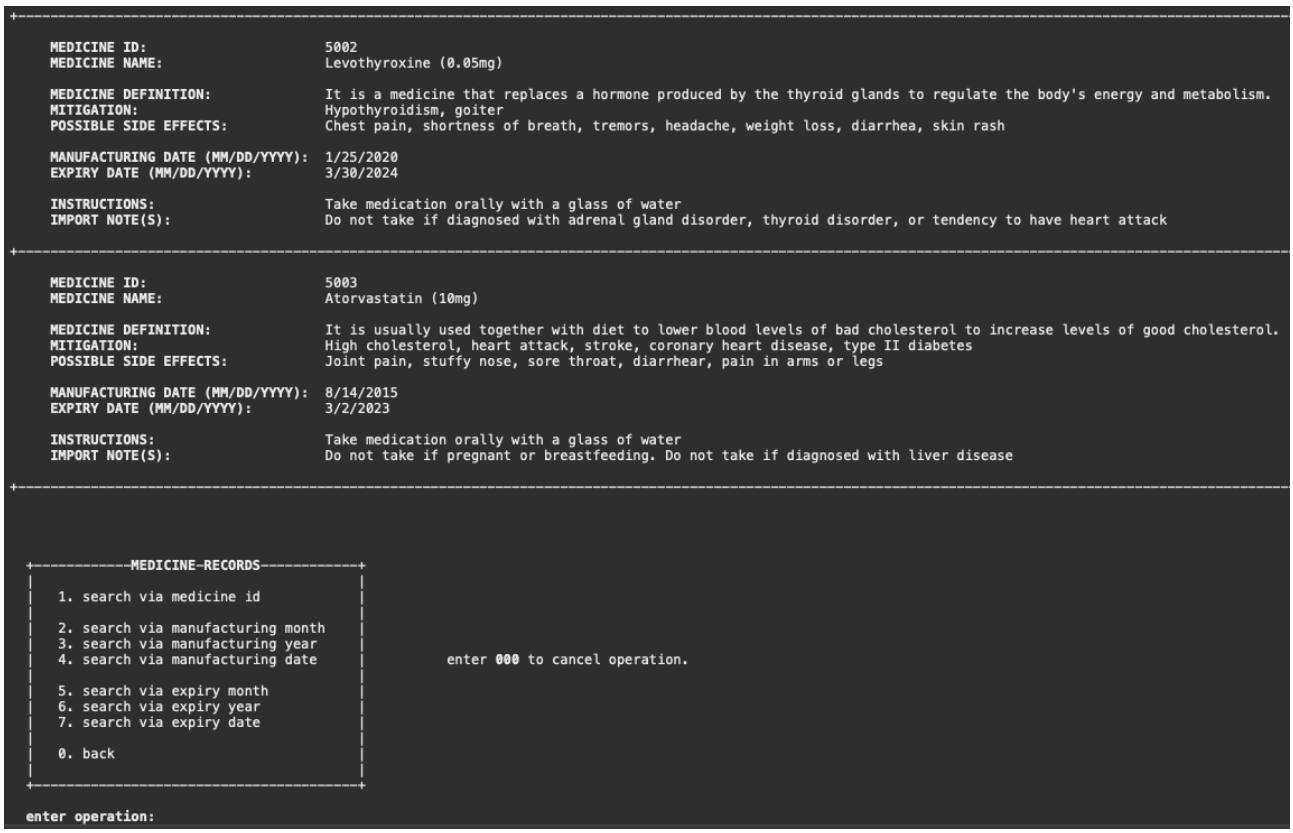


Figure 61: Output when Search via expiry month operation is done correctly.
As seen above, the program displayed two medicines that have matching month of expiry date which is the 3rd month of the month of March.

Search via Expiry Year

```
case 6: // search via expiry year
    found = false;

    System.out
            .print("\n" + "   enter " + setBoldStart + "year" + setBoldEnd + " of expiry date: \t");
    m_expirydateyear = scan_int.nextInt();
    if (m_expirydateyear == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    m_iteration = m_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (m_iteration.hasNext()) {
        Medicine m_e = m_iteration.next();

        if (m_e.getMedicineExpiryDateYear() == m_expirydateyear) {
            System.out.println(m_e);
            found = true;
            m_search = 1;
        }
    }

    System.out.println(
            "\n +-----------------------------------------------------------------------------------------

    if (found != true) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   MEDICINE RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }
    break;
```

```
+-----------MEDICINE-RECORDS-----------+
|                                       |
|   1. search via medicine id           |
|                                       |
|   2. search via manufacturing month   |
|   3. search via manufacturing year    |
|   4. search via manufacturing date    |          enter 000 to cancel operation.
|                                       |
|   5. search via expiry month          |
|   6. search via expiry year           |
|   7. search via expiry date           |
|                                       |
|   0. back                             |
|                                       |
+---------------------------------------+

enter operation: 6

enter year of expiry date:   2027
```

Figure 62: Output when user enters 6 in Medicine records SUB menu which is Search via expiry year operation.
As seen above, the user is entering the year of expiry date of medicine to be searched.

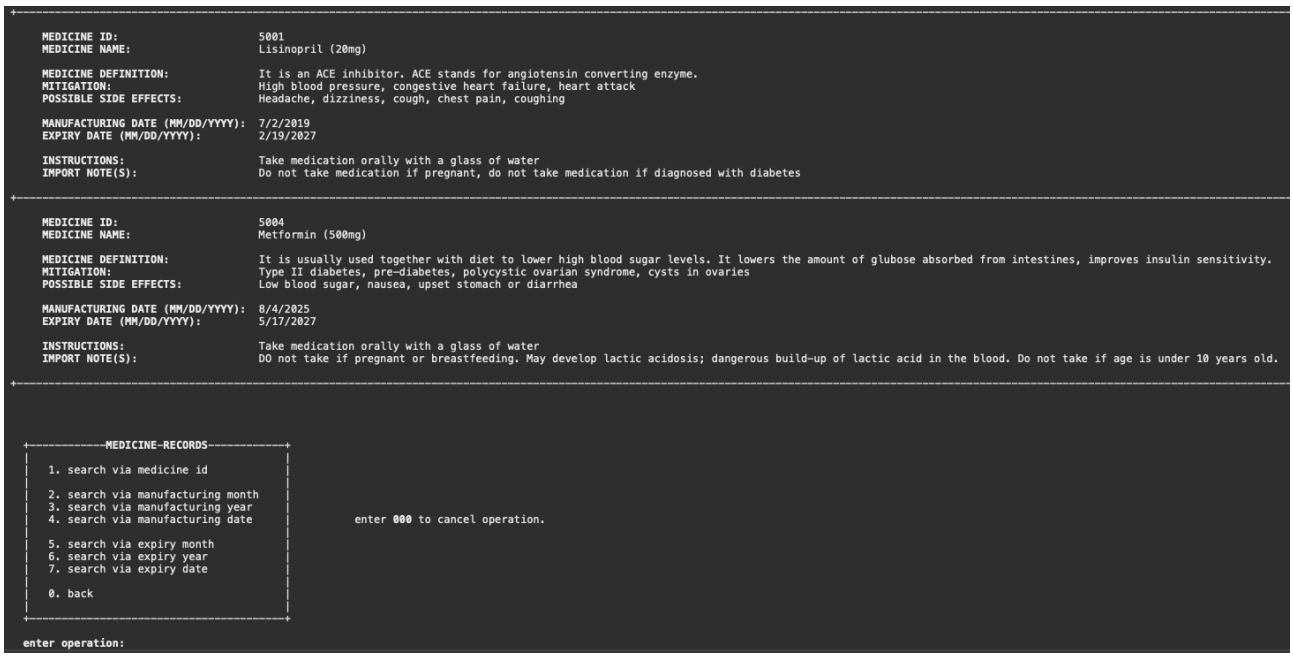


Figure 63: Output when Search via expiry year operation is done correctly.
As seen above, the program displayed two medicines that have matching year of expiry date which is the year 2027.

Search via Expiry Date

```
case 7: // search via exiry date
    found = false;

    System.out.print("\n" + "   enter " + setBoldStart + "month" + setBoldEnd
            + " of expiry date (1-12): \t\t");
    m_expirydatemonth = scan_int.nextInt();
    if (m_expirydatemonth == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    System.out.print(
            "   enter " + setBoldStart + "day" + setBoldEnd + " expiry date (1-28/29/30/31): \t");
    m_expirydateday = scan_int.nextInt();
    if (m_expirydateday == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    System.out.print("   enter " + setBoldStart + "year" + setBoldEnd + " of expiry date: \t\t\t");
    m_expirydateyear = scan_int.nextInt();
    if (m_expirydateyear == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    m_iteration = m_collection.iterator();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    while (m_iteration.hasNext()) {
        Medicine m_e = m_iteration.next();

        if (m_e.getMedicineExpiryDateMonth() == m_expirydatemonth
                && m_e.getMedicineExpiryDateDay() == m_expirydateday
                && m_e.getMedicineExpiryDateYear() == m_expirydateyear) {
            System.out.println(m_e);
            found = true;
            m_search = 1;
        }
    }

    System.out.println(
            "\n +----------------------------------------------------------------------------------------
```

```
if (found != true) {
     System.out.println(new String(new char[70]).replace("\0", "\r\n"));
     System.out.println(setBoldStart + "   MEDICINE RECORD NOT FOUND." + setBoldEnd);
     Thread.sleep(1500);
}
break;
```

Figure 64: Output when user enters 7 in Medicine records SUB menu which is Search via expiry date operation.
As seen above, the user is entering the expiry date of a particular medicine to be searched.

Figure 65: Output when Search via expiry date operation is done correctly.
As seen above, the program displayed a medicine that matches with expiry date entered by the user which is 10/19/2030 or October 19 of 2030.

Update Operation

```
case 4: // update
    if (file_medicine.isFile()) {
        m_ois = new ObjectInputStream(new FileInputStream(file_medicine));
        m_collection = (ArrayList<Medicine>) m_ois.readObject();
        m_ois.close();
    }

    boolean found = false;

    m_iteration = m_collection.iterator();
    m_list = m_collection.listIterator();

    System.out.print("   enter " + setBoldStart + "id number" + setBoldEnd + " of medicine: ");
    int m_id = scan_int.nextInt();
    if (m_id == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    while (m_list.hasNext()) {
        Medicine m_e = m_list.next();

        if (m_e.getMedicineID() == m_id) {
            System.out.print("\n" + "   enter " + setBoldStart + "NEW id number" + setBoldEnd
                    + " of medicine: \t\t\t");
            m_id1 = scan_int.nextInt();
            if (m_id1 == 000) {
                System.out.println(new String(new char[70]).replace("\0", "\r\n"));
                System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
                Thread.sleep(1500);
                medicine_records();
            }

            System.out
                    .print("   enter " + setBoldStart + "NEW name" + setBoldEnd + " of medicine: \t\t\t\t");
            m_name = scan_string.nextLine();
            if (m_name.equals("000")) {
                System.out.println(new String(new char[70]).replace("\0", "\r\n"));
                System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
                Thread.sleep(1500);
                medicine_records();
            }
```

```
        m_list.set(new Medicine(m_id1, m_name, m_strength, m_definition, m_treatment, m_sideeffects,
                m_instructions, m_note, m_expirydatemonth, m_expirydateday, m_expirydateyear,
                m_approvaldatemonth1, m_approvaldateday1, m_approvaldateyear1));
        found = true;
    }
}

if (found) {
    m_oos = new ObjectOutputStream(new FileOutputStream(file_medicine));
    m_oos.writeObject(m_collection);
    m_oos.close();
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println("   " + setBoldStart + "MEDICINE RECORD UPDATED." + setBoldEnd);
    Thread.sleep(1500);
}

else {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   MEDICINE RECORD NOT FOUND." + setBoldEnd);
    Thread.sleep(1500);
}

break;
```

Figure 66: Output when user enters 4 in Medicine records MAIN menu which is Update operation. As seen above, the user is updating the entry for medicine ID number 5005 with NEW data.

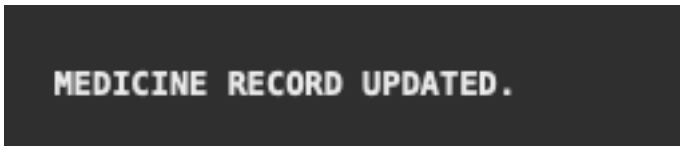


Figure 67: Output when Update operation is done correctly.

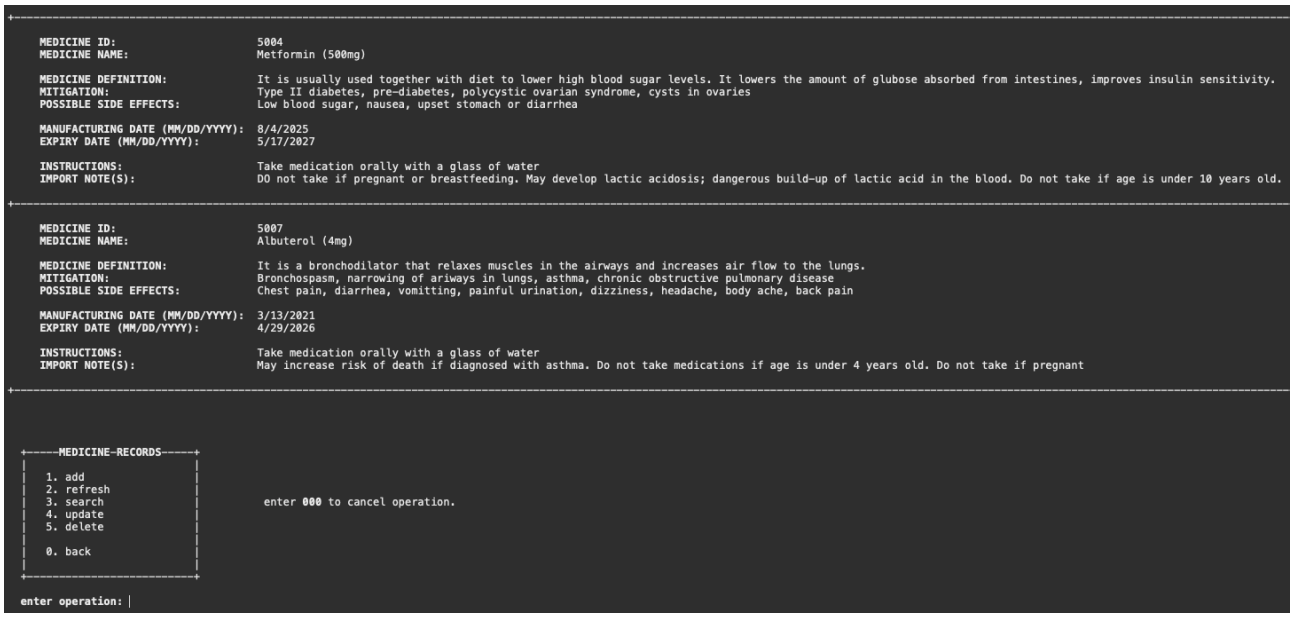


Figure 68: Output of updated entry list of Medicine records.
As seen above, the entry for medicine ID number 5005 is no longer existent as it as updated and was replaced with new medicine ID number 5007.

Delete Operation

```
case 5: // delete
    found = false;

    System.out.print("\n" + "   enter " + setBoldStart + "id number" + setBoldEnd + " of medicine: \t");
    m_id = scan_int.nextInt();
    if (m_id == 000) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   OPERATION CANCELLED." + setBoldEnd);
        Thread.sleep(1500);
        medicine_records();
    }

    m_iteration = m_collection.iterator();
    while (m_iteration.hasNext()) {
        Medicine m_e = m_iteration.next();

        if (m_e.getMedicineID() == m_id) {
            System.out.println(new String(new char[70]).replace("\0", "\r\n"));
            m_iteration.remove();
            System.out.println(setBoldStart + "   MEDICINE RECORD DELETED." + setBoldEnd);
            found = true;
            Thread.sleep(1500);
        }
    }

    if (!found) {
        System.out.println(new String(new char[70]).replace("\0", "\r\n"));
        System.out.println(setBoldStart + "   MEDICINE RECORD NOT FOUND." + setBoldEnd);
        Thread.sleep(1500);
    }

    m_oos = new ObjectOutputStream(new FileOutputStream(file_medicine));
    m_oos.writeObject(m_collection);
    m_oos.close();

    m_search = 0;
    break;
```

```
+-----MEDICINE-RECORDS------+
|                           |
|   1. add                  |
|   2. refresh              |
|   3. search               |          enter 000 to cancel operation.
|   4. update               |
|   5. delete               |
|                           |
|   0. back                 |
|                           |
+---------------------------+

enter operation: 5

enter id number of medicine:          5007
```

Figure 69: Output when enters 5 in Medicine records MAIN menu which is Delete operation.

```
  MEDICINE RECORD DELETED.
```

Figure 70: Output when Delete operation is done correctly.

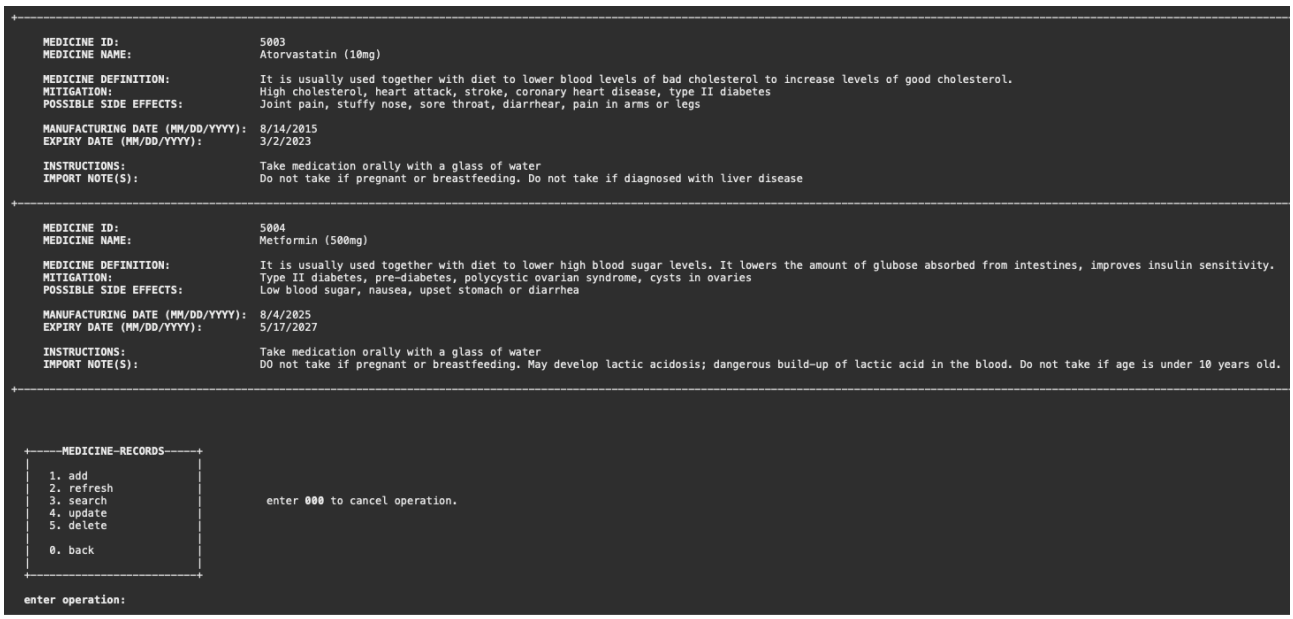


Figure 71: Output of updated entry list of Medicine records.
As seen above, the entry with medicine ID number 5007 is no longer existent as it was deleted.

OTHER OUTPUTS

These functions only exist as a design standpoint and serves no correlation to the assignment requirement. They only exist to make the program a bit more user-friendly, give more information to the user and to give quality of life improvements to the program overall.

About Program

```
public static void about_program() throws Exception {
    record = 5;
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    int about_ch;

    System.out.println("    " + setBoldStart + "MODULE TITLE:" + setBoldEnd + "  SOFTWARE ENGINEERING 4201");
    System.out.println(
            "    " + setBoldStart + "ASSIGNMENT TITLE:" + setBoldEnd + "     BELL HOSPITAL MANAGEMENT SYSTEM\n");
    System.out.println("    " + setBoldStart + "COMPLETED BY:" + setBoldEnd + "       LES PAUL RANALAN");
    System.out.println("    " + setBoldStart + "COMPLETION DATE:" + setBoldEnd + "   DEC. 6, 2022\n");
    System.out.println("    " + setBoldStart + "BUILD NUMBER:" + setBoldEnd + "  1.0\n\n\n");

    System.out.println("    " + setBoldStart + "RECORDS:" + setBoldEnd
            + "\n   as of version 1.0, the user can interact with 4 records. below are the list of all records:\n\n   1. patient records\n      - this is where all patient informations are stored.\n
            + setBoldStart + "NOTE: THE ACCESS OPERATION IS ONLY AVAILABLE FOR APPOINTMENT RECORDS." + setBoldEnd);

    System.out.println("\n\n\n    " + setBoldStart + "OPERATIONS:" + setBoldEnd
            + "\n   as of version 1.0, the user can utilize numerous operations.\n   below is the list of the essential operations in this version:\n\n   1. add\n      - add a new entry to the current
            + setBoldStart + "only available for appointment records" + setBoldEnd
            + "}\n      - display further information about an entry in the current record.\n      - i.e.: to display the patient and doctor details of a specific id number in an appointment.\n\n   4.

    System.out.println("\n\n\n");
    System.out.println("\n   +---------" + setBoldStart + "ABOUT" + setBoldEnd + "-" + setBoldStart + "PROGRAM"
            + setBoldEnd + "---------+" + "\n   |                                |");
    System.out.println("    |   0. exit                      |\n   |                                |");
    System.out.println("    +--------------------------------+");

    System.out.print(setBoldStart + "\n   enter operation: " + setBoldEnd);
    about_ch = scan_int.nextInt();
    if (about_ch == 0) {
        main_menu();
    }
}
```

```
MODULE TITLE:        SOFTWARE ENGINEERING 4201
ASSIGNMENT TITLE:    BELL HOSPITAL MANAGEMENT SYSTEM

COMPLETED BY:        LES PAUL RANALAN
COMPLETION DATE:     DEC. 6, 2022

BUILD NUMBER:        1.0



RECORDS:
as of version 1.0, the user can interact with 4 records. below are the list of all records:

1. patient records
     - this is where all patient informations are stored.
     - a single entry contains 1 patient.
     - its data is stored in a .txt or text file named 'patient.txt'.

2. doctor records
     - this is where all doctor informations are stored.
     - a single entry contains 1 doctor.
     - its data is stored in a .txt file or text file named 'doctor.txt'.

3. appointment records
     - this is where all appointment information is stored.
     - a single entry contains 1 patient, 1 doctor, and 1 appointment.
     - its data is stored in a .txt or text file named 'appointment.txt'.

4. medicine records
     - this is where all medicine informations are stored.
     - a single entry contains 1 medicine.
     - its data is stored in a .txt or text file named 'medicine.txt'.

each record is unique as it contains different data stored in a .txt file, the idea of storing data on .txt files makes
it even more convenient and fitting as the data can be linked with data from different records and the data is still present
even when the program is terminated.
however, when opening the .txt files, the data is encrypted and can only be read by java.
NOTE: THE ACCESS OPERATION IS ONLY AVAILABLE FOR APPOINTMENT RECORDS.
```

```
OPERATIONS:
as of version 1.0, the user can utilize numerous operations.
below is the list of the essential operations in this version:

1. add
     - add a new entry to the current record.
     - i.e.: to register a new patient to the hospital.

2. refresh
     - refresh or restart the entries in the current record.
     - i.e.: to rebuild the current entries to see any changes in the record.

3. access (only available for appointment records)
     - display further information about an entry in the current record.
     - i.e.: to display the patient and doctor details of a specific id number in an appointment.

4. search
     - display only a specific entry in the current record.
     - i.e.: to display information on a specific registered patient in the hosital.

5. update
     - modify a specific entry in the current records.
     - i.e.: to search a specific doctor via id number and change their specialization and contact number.

6. delete
     - remove an entry in the current record.
     - i.e.: to remove a specific medicine in the catalog using its id number.




+---------ABOUT-PROGRAM---------+
|                               |
|   0. exit                     |
|                               |
+-------------------------------+

enter operation:
```

Figures 72: Output when user enters 5 in Main menu which is About program.

Errors

```
default:
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   INVALID INPUT. PLEASE TRY AGAIN" + setBoldEnd);
    Thread.sleep(1500);
    break;
}
```

Figure 73: Output when user enters an invalid choice in any menu.

```
if (found != true) { // if patient id entered by user is not found, it will display an error and
                     // returns to menu.
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   PATIENT RECORD NOT FOUND." + setBoldEnd);
    Thread.sleep(1500);
}
```


```
if (found != true) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));

    d_search = 0;
    System.out.println(setBoldStart + "   DOCTOR RECORD NOT FOUND." + setBoldEnd);
    Thread.sleep(1500);
}
break;
```


```
if (found != true) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart + "   APPOINTMENT RECORD NOT FOUND." + setBoldEnd);
    Thread.sleep(1500);
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    found = false;
}
break;
```

```
if (found != true) {
     System.out.println(new String(new char[70]).replace("\0", "\r\n"));
     System.out.println(setBoldStart + "    MEDICINE RECORD NOT FOUND." + setBoldEnd);
     Thread.sleep(1500);
}
break;
```

```
MEDICINE RECORD NOT FOUND.
```

Figures 74: Output when program does not find any data that matches entered data.

```
*/
catch (InputMismatchException e) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println(setBoldStart
            + "    INVALID INPUT. INTEGERS ARE ONLY ACCEPTED IN THAT FIELD.\n\n    RESTARTING CURRENT RECORD..."
            + setBoldEnd);

    String x;

    if (record == 0 && scan_int.hasNext()) {
        x = scan_int.next();
        scan_int.reset();
        Thread.sleep(5000);
        main(null);
    }

    if (record == 1 && scan_int.hasNext()) {
        x = scan_int.next();
        scan_int.reset();
        Thread.sleep(5000);
        main(null);
    }

    if (record == 2 && scan_int.hasNext()) {
        x = scan_int.next();
        scan_int.reset();
        Thread.sleep(5000);
        main(null);
    }

    if (record == 3 && scan_int.hasNext()) {
        x = scan_int.next();
        scan_int.reset();
        Thread.sleep(5000);
        main(null);
    }

    if (record == 4 && scan_int.hasNext()) {
        x = scan_int.next();
        scan_int.reset();
        Thread.sleep(5000);
        main(null);
    }

    if (record == 5 && scan_int.hasNext()) {
        x = scan_int.next();
        scan_int.reset();
        Thread.sleep(5000);
        main(null);
    }
}
```

```
INVALID INPUT. INTEGERS ARE ONLY ACCEPTED IN THAT FIELD.

RESTARTING CURRENT RECORD...
```

Figure 75: Output when a user enters a character or a non-integer in an input field where integers are only accepted.

```
/*
 * the following code below is when any of the ids are not matching with the
 * ones found in appointment records. it will print an error message.
 */
if (appointment_found != true) {
    System.out.println(new String(new char[70]).replace("\0", "\r\n"));
    System.out.println("\n   APPOINTMENT WITH PATIENT " + setBoldStart + "REGISTRATION ID NUMBER '"
            + a_patientid + setBoldEnd + "' AND DOCTOR " + setBoldStart + "STAFF ID NUMBER '"
            + a_doctorid + setBoldEnd + "' IS NOT REGISTERED IN APPOINTMENT RECORDS.");
}

if (patient_found != true) {
    System.out.println("\n   PATIENT " + setBoldStart + "REGISTRATION ID NUMBER '" + a_patientid
            + setBoldEnd + "' IS NOT REGISTERED IN PATIENT RECORDS.");
}

if (doctor_found != true) {
    System.out.println("\n   DOCTOR " + setBoldStart + "STAFF ID NUMBER '" + a_doctorid + setBoldEnd
            + "' IS NOT REGISTERED IN DOCTOR RECORDS.");
}

break;
```

```
APPOINTMENT WITH PATIENT REGISTRATION ID NUMBER '1000' AND DOCTOR STAFF ID NUMBER '9000' IS NOT REGISTERED IN APPOINTMENT RECORDS.

PATIENT REGISTRATION ID NUMBER '1000' IS NOT REGISTERED IN PATIENT RECORDS.

DOCTOR STAFF ID NUMBER '9000' IS NOT REGISTERED IN DOCTOR RECORDS.
```

Figure 76: Output when Access operation does not find any matching data on entered data.

BIBLIOGRAPHY / REFERENCES

Dr. Parag Shukla (2021). *Java - Write Object and Read Object to File using Object Stream - Practical Demo (Student Object)*. *YouTube*. Available at: https://www.youtube.com/watch?v=KAWoOgKsQns&ab_channel=Dr.ParagShukla [Accessed 13 Dec. 2022].

Dr. Parag Shukla (2021). *File Handling in Java Insert, Update, Delete, Search, Sort and Display with collection in File CRUD*. *YouTube*. Available at: https://www.youtube.com/watch?v=EfS6i_jAm4g&t=1767s&ab_channel=Dr.ParagShukla [Accessed 13 Dec. 2022].

Dr. Parag Shukla (2021). *Java Collection - CRUD Operation - INSERT, UPDATE, DELETE, SEARCH and DISPLAY of Employee Collection*. *YouTube*. Available at: https://www.youtube.com/watch?v=O-XrUJj83E0&t=2s&ab_channel=Dr.ParagShukla [Accessed 13 Dec. 2022].

freeCodeCamp.org (2020). *Getters and Setters in Java Explained*. [online] freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/java-getters-and-setters/ [Accessed 13 Dec. 2022].

OmerM (2022). *Iterator/Iterable Interfaces in Java - OmerM - Medium*. [online] Medium. Available at: https://medium.com/@omerme/iterator-iterable-interfaces-in-java-d1b8dd81511d [Accessed 13 Dec. 2022].

Posa, R. (2022). *Java ListIterator - ListIterator in Java*. [online] Digitalocean.com. Available at: https://www.digitalocean.com/community/tutorials/java-listiterator [Accessed 13 Dec. 2022].

baeldung (2016). *Guide to the Java ArrayList | Baeldung*. [online] Baeldung. Available at: https://www.baeldung.com/java-arraylist [Accessed 13 Dec. 2022].

baeldung (2019). *Java toString() Method | Baeldung*. [online] Baeldung. Available at: https://www.baeldung.com/java-tostring [Accessed 13 Dec. 2022].

Nam Ha Minh (2019). *Java File IO FileInputStream and FileOutputStream Examples*. [online] Codejava.net. Available at: https://www.codejava.net/java-se/file-io/java-io-fileinputstream-and-fileoutputstream-examples [Accessed 13 Dec. 2022].

baeldung (2017). *Java String.format()*. [online] Baeldung. Available at: https://www.baeldung.com/string/format [Accessed 13 Dec. 2022].

Java2s.com. (2016). *Java Terminal Bold Output bold(String str)*. [online] Available at: http://www.java2s.com/example/java-utility-method/terminal-bold-output/bold-string-str-6d96c.html [Accessed 13 Dec. 2022].