

**PROJECT REPORT**

**JAVA PROGRAMMING FUNDAMENTALS**

**(EBDS22ET2)**

2024-2025(EVEN SEMESTER)

**DEPARTMENT OF B.Tech DS & AI (E&T)**

**COURSE :** B-TECH CSE-DS (AI)

**YEAR/SEM/SEC :** 1st YEAR / 2nd SEM / AB1

**PROJECT TITLE :** TO-DO TASK TRACKER



**BONAFIDE CERTIFICATE**

**JAVA PROGRAMMING FUNDAMENTALS**

**DEPARTMENT OF B.Tech DS & AI (E&T)**

Certified that this project report **“ TO-DO TASK TRACKER ”** is confirmed work of **YOKESHWARI.K , SANJANA.D , THENMOZHI.D**  I-year B-Tech CSE- DS(AI) in **JAVA PROGRAMMING FUNDAMENTALS (EBDS22ET2)** who carried out the project work under the supervision

Signature of Lab-in-Charge Signature of Head of Dept

Submitted for the Practical Examination held on

Internal Examiner External Examiner

ABSTRACT

Project objective:

The To-Do Task Tracker for Patient Information is a Java-based project developed to help manage and organize health-related tasks efficiently. The main purpose of this project is to provide a structured way to track various activities associated with patient details such as patient ID, patient name, Description about the problem, Admission date, Doctor name.

Key features of this Project:

* Add Task
* View Task
* Update Task
* Delete Task
* Search Task
* Mark as Complete

Concepts involved---

**Conditional statements:**

If and if-else: In this program to check whether the information about the patient found or deleted.

CONTINUE - It is used inside loops when you want to skip over part of the code under certain conditions (like invalid input).

SWITCH - The user's input is stored in choice.

The switch statement executes different code blocks depending on the choice.

Each case handles one specific operation (add, view, delete, exit).

BREAK - Break statements in this program used to exit the switch case after the competition of certain operation.

RETURN - Returning the value to the call function, i.e. Patient info. Then gets terminated and exit from the program.

**Arraylist** - Dynamic list used to store multiple patients.

**Class** - Patient class which stores the info of patient about Patient Id, Description, AdmittedDate, Doctor name, status.

nextLine() - A method in the Scanner class used to read an entire line of input, including indentation, until the user Enters or passes to next line.

**OOPS** - Runtime Polymorphism

Overriding --A subclass provides its own version of a method that is already defined in its superclass. Represents the specific information related to patient such as admitted date.

**Looping statements** - Used to repeat the certain task of Representing patient info in this program— the user to add, view, or update patients again and again until they choose to exit.

**INTRODUCTION TO JAVA**

Java is a high-level, object-oriented, platform-independent programming language developed by Sun Microsystems in 1995 (now owned by Oracle Corporation). It is designed to be simple, secure, and robust, enabling developers to write code once and run it anywhere.

**Key features of JAVA:**

* Simple
* Object-Oriented
* Platform Independent
* Secure
* Robust
* Multithreaded
* High Performance
* Distributed

**Components of JAVA:**

1. Java Programming Language

Object-oriented programming language. It supports features like:

* Classes and Objects
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation

**2**. Java Development Kit (JDK)

The JDK is a complete software development kit used to develop Java applications.

**3.** Java Runtime Environment (JRE)

JRE provides the libraries, Java Virtual Machine (JVM), and other components to run Java applications.

**4.** Java Virtual Machine (JVM)

The JVM is the engine that runs Java bytecode on any platform.

Provides platform independence (Write Once, Run Anywhere)

**5.** Java API (Application Programming Interface)

* Handle I/O
* Work with data structures
* Build GUIs
* Perform networking
* Access databases

**6.** Java Packages for

* Code organization
* Reusability
* Access protection

**Java Language Syntax and structure for writing Java programs.**

**JVM Runs bytecode and makes Java platform-independent.**

**JRE Environment for running Java programs.**

**JDK Full kit for developing Java applications.**

**Java API Library of pre-written classes and interfaces.**

**Packages Grouping of related classes and interfaces.**

**TO-DO-TASK-TRACKING**

**AIM**:

To design and develop a Java-based To-Do List Tracking system for managing patient information efficiently, enabling users to add, update, view, and delete patient-related tasks such as Patient ID, Patient name, description, admission and doctor name using user-friendly Interface.

**ALGORITHM:**

1. **Initialization of the program:**

Starting with an empty list, infoList to store patient data information.

1. **Loop**:

Continuous display options to the user:

For the following inputs,

* Add patient
* Update patient
* List all patients
* Delete patient
* Exit

After Reading the user input and executing the main method.

1. **Add Patient Information:**

Prompt the user to (name, description, admission date, doctor, and condition/status).

Create an Info object and add it to the list.

1. **Update Patient Information:**

If Patient Id is found, display current values and proceeding user for new

Value. Updating the fields accordingly.

**5. List All Patient Information:**

Print all Info objects in the list.

Then Display message if the list is empty.

1. **Delete Patient Information:**

Delete/remove the object from the list, if it is found.

1. **Terminate the program:**

Breaking the loop and terminate the application.

SOURCE CODE:

**package** com.mgr.research.hospital.todo;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**class** Information {

**public** **int** patientId;

**public** String patientName;

**public** String description;

**public** String admittedDate;

**public** String doctorName;

**public** CriticalStatus status;

**public** **enum** CriticalStatus {

***UNDETERMINED***, ***GOOD***, ***SERIOUS***, ***CRITICAL***;

}

**public** Information(**int** patientId, String patientName, String description, String admittedDate, String doctorName,

CriticalStatus status) {

**this**.patientId = patientId;

**this**.patientName = patientName;

**this**.description = description;

**this**.admittedDate = admittedDate;

**this**.doctorName = doctorName;

**this**.status = CriticalStatus.*valueOf*(status.name());

}

@Override

**public** String toString() {

**return** "Information : { " + "patientId= " + patientId + ", patientName= '" + patientName + '\''

+ ", description= '" + description + '\'' + ", admittedDate= '" + admittedDate + '\''

+ ", doctorName= '" + doctorName + '\'' + ", status= '" + status + '\'' + '}';

}

}

**public** **class** ToDoList {

**public** **static** Scanner *scanner* = **new** Scanner(System.***in***);

**public** **static** List<Information> *infoList* = **new** ArrayList<>();

**public** **static** **int** *id* = 1;

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

**while** (**true**) {

System.***out***.println("\n\*\*\*\*\*\*\*\* Patient To-Do List Tracker \*\*\*\*\*\*\*\*");

System.***out***.println("1. Add Patient Information");

System.***out***.println("2. Update Patient Information");

System.***out***.println("3. List All Patient Information");

System.***out***.println("4. Delete Patient Information");

System.***out***.println("5. Close the List Tracker");

System.***out***.print("Enter your choice: ");

**int** option = *scanner*.nextInt();

*scanner*.nextLine();

**switch** (option) {

**case** 1:

*addPatientInfo*();

**break**;

**case** 2:

*updatePatientInfo*();

**break**;

**case** 3:

*listPatientInfo*();

**break**;

**case** 4:

*deletePatientInfo*();

**break**;

**case** 5:

System.***out***.println("\*\*\*\*\*\*\*\* Closing To-Do List \*\*\*\*\*\*\*\*");

**return**;

**default**:

System.***out***.println("Entered invalid option. Please select option from 1 to 5.");

**break**;

}

}

}

**public** **static** **void** addPatientInfo() {

System.***out***.println("Adding a new information :-");

System.***out***.print("Enter Patient Name: ");

String patient = *scanner*.nextLine();

System.***out***.print("Enter Description: ");

String description = *scanner*.nextLine();

System.***out***.print("Enter Admitted Date (DD-MM-YYYY): ");

String date = *scanner*.nextLine();

System.***out***.print("Enter Doctor Name: ");

String doctor = *scanner*.nextLine();

System.***out***.print("Provide Patient's condition (UNDETERMINED, GOOD, SERIOUS, CRITICAL): ");

String condition = *scanner*.nextLine().toUpperCase();

Information info = **new** Information(*id*++, patient, description, date, doctor,

Information.CriticalStatus.*valueOf*(condition));

*infoList*.add(info);

System.***out***.println("\*\*\*\* Information Saved \*\*\*\*");

}

**public** **static** **void** updatePatientInfo() {

System.***out***.println("Updating a information :-");

System.***out***.println("If no changes needed, just press Enter.");

System.***out***.print("Enter Patient ID to update: ");

String id = *scanner*.nextLine();

Information infoToUpdate = **null**;

**for** (Information info : *infoList*) {

**if** (String.*valueOf*(info.patientId).equals(id)) {

infoToUpdate = info;

**break**;

}

}

**if** (infoToUpdate == **null**) {

System.***out***.println("No information found with the given Patient ID.");

**return**;

}

System.***out***.print("Enter new Patient Name (current info: " + infoToUpdate.patientName + "): ");

String patient = *scanner*.nextLine();

**if** (!patient.isEmpty())

infoToUpdate.patientName = patient;

System.***out***.print("Enter new Description (current info: " + infoToUpdate.description + "): ");

String description = *scanner*.nextLine();

**if** (!description.isEmpty())

infoToUpdate.description = description;

System.***out***.print("Enter new Admitted Date (DD-MM-YYYY, current info: " + infoToUpdate.admittedDate + "): ");

String date = *scanner*.nextLine();

**if** (!date.isEmpty())

infoToUpdate.admittedDate = date;

System.***out***.print("Enter new Doctor Name (current info: " + infoToUpdate.doctorName + "): ");

String doctor = *scanner*.nextLine();

**if** (!doctor.isEmpty())

infoToUpdate.doctorName = doctor;

System.***out***.print("Provide new Patient's condition (UNDETERMINED, GOOD, SERIOUS, CRITICAL, current info: "

+ infoToUpdate.status + "): ");

String condition = *scanner*.nextLine().toUpperCase();

**if** (!condition.isEmpty())

infoToUpdate.status = Information.CriticalStatus.*valueOf*(condition);

System.***out***.println("\*\*\*\* Information Updated \*\*\*\*");

}

**public** **static** **void** deletePatientInfo() {

System.***out***.println("Deleting a information :-");

System.***out***.print("Enter Patient ID to delete: ");

**int** id = *scanner*.nextInt();

*scanner*.nextLine();

**for** (Information info : *infoList*) {

**if** (info.patientId == id) {

*infoList*.remove(info);

System.***out***.println("\*\*\*\* Information Deleted \*\*\*\*");

**break**;

} **else** {

System.***out***.println("\*\*\*\* Information Not Found \*\*\*\*");

}

}

}

**public** **static** **void** listPatientInfo() {

System.***out***.println("Listing all patient information :-");

**if** (*infoList*.isEmpty()) {

System.***out***.println("No information available.");

} **else** {

*infoList*.forEach(System.***out***::println);

}

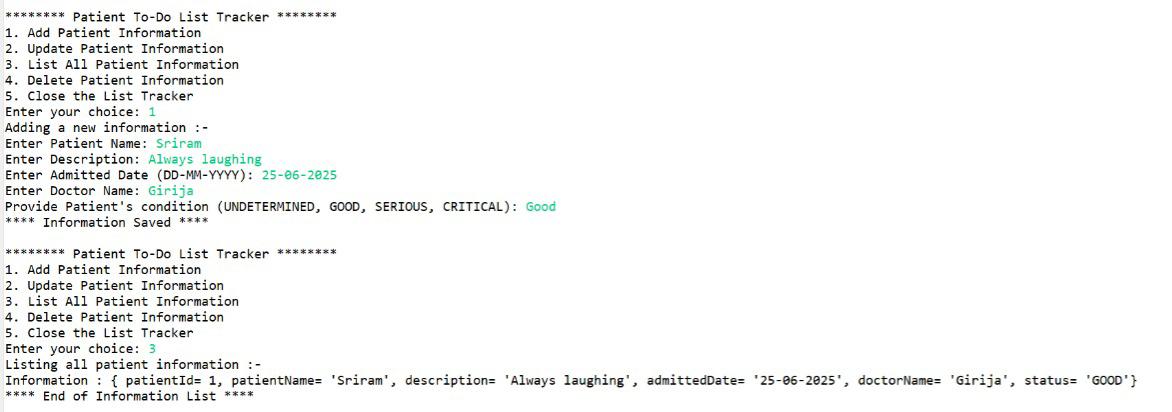
System.***out***.println("\*\*\*\* End of Information List \*\*\*\*");

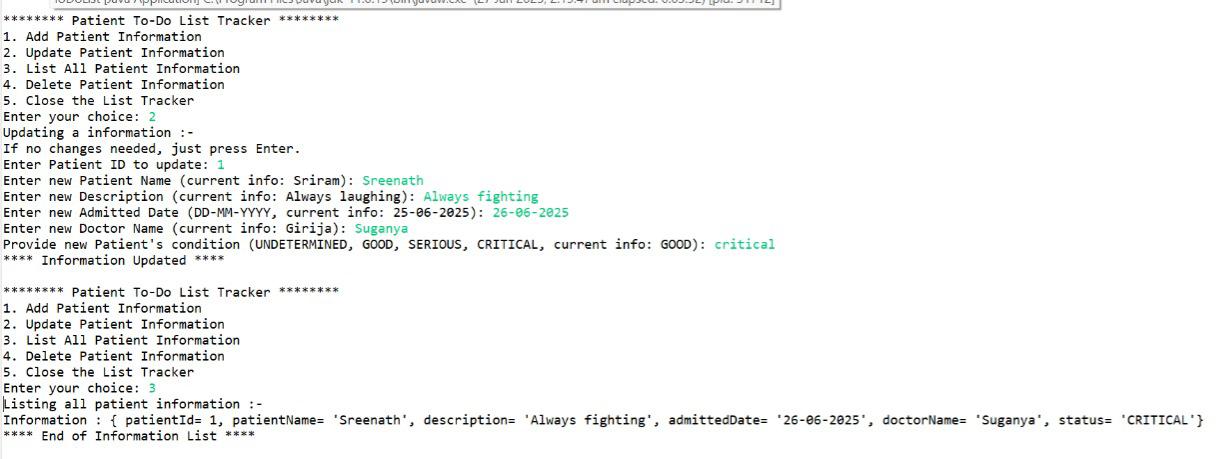
*scanner*.nextLine();

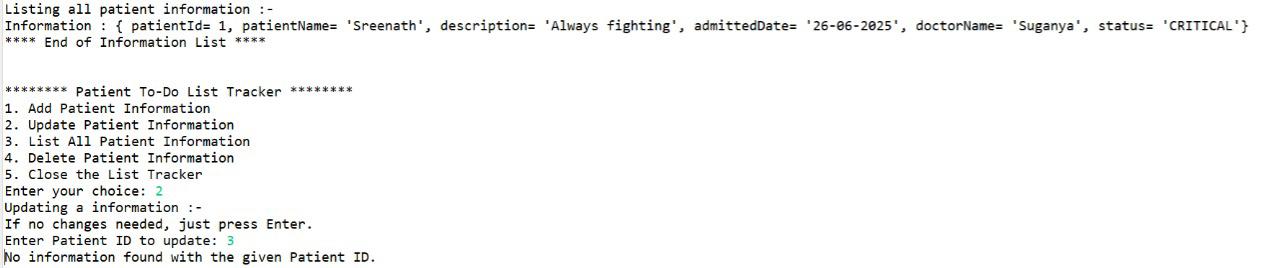
}

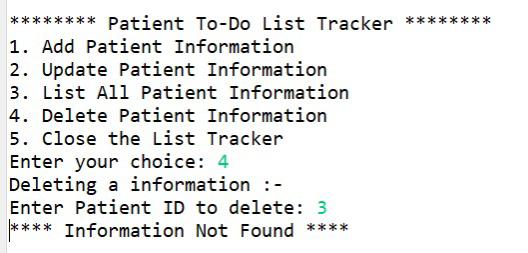
}

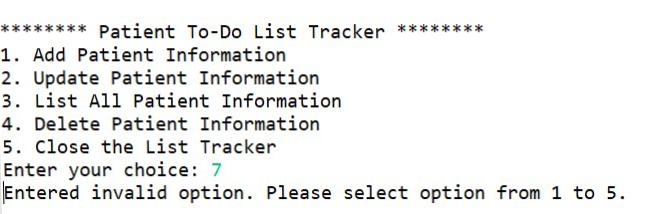
**OUTPUT:**

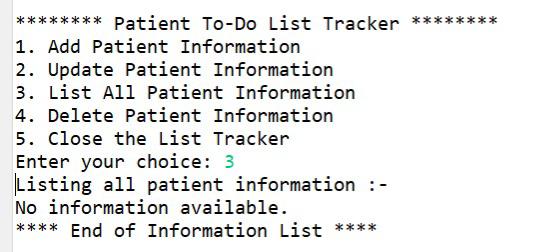
For Add & View

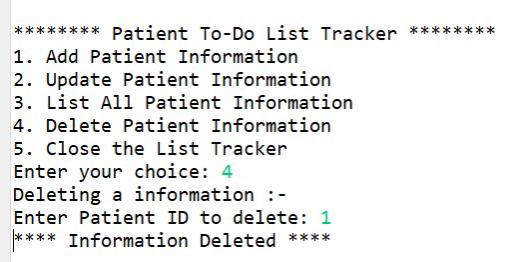
For Update & view

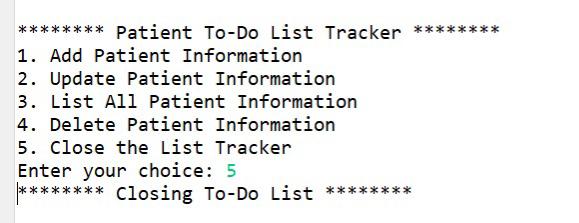
Tried update with Invalid ID

Tried to delete with Invalid ID

When entered Wrong option

No information is found

For delete

 Closing the To-do tracker

**RESULT**:

The program successfully tracks and displays patient information. The system displays each patient’s name, ID, Description, Doctor name and current status. The source code verified successfully with the Output.