| Experiment No. 4 |
| --- |
| Classes and Objects |
| Date of Performance:26/07/24 |
| Date of Submission:02/08/24 |

**Aim:-** Implementation of concepts of classes and objects in object-oriented programming

**Objective:-** Implement a simple hospital management system in Java that manages patients, doctors, and appointments. The system should include three main classes: Patient, Doctor, and Appointment. The Patient class should have attributes like patientID, name, age, gender, and contactNumber, with a method displayPatientDetails() to display patient details. The Doctor class should include doctorID, name, specialization, and contactNumber attributes, and a method displayDoctorDetails() to show doctor details. The Appointment class should consist of appointmentID, appointmentDate, doctorRequired and appointmentTime, along with a method displayAppointmentDetails() to display the appointment details along with the patient and doctor details. The Main class should create instances of Patient, Doctor, and Appointment, and use the respective methods to display their details.

**Theory:-**  A class is a user defined blueprint or prototype from which objects are created.  It represents the set of properties i.e. members and methods that are common to all objects of one type. In general, class declarations can include these components, in order:

1. Modifiers : A class can be public or has default access.
2. class keyword: class keyword is used to create a class.
3. Class name: The name should begin with a initial letter (capitalized by convention).
4. Superclass(if any): The name of the class’s parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
5. Interfaces(if any): A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
6. Body: The class body surrounded by braces, { }.

A OBJECT is a basic unit of Object Oriented Programming and represents the real life entities.  A typical Java program creates many objects, which interact by invoking methods. An object consists of :

1. State : It is represented by attributes of an object. It also reflects the properties of an object.
2. Behavior : It is represented by methods of an object. It also reflects the response of an object with other objects.
3. Identity : It gives a unique name to an object and enables one object to interact with other objects.

**Code:-**

public class Appointment {

private String appointmentID;

private String appointmentDate;

private Doctor doctorRequired;

private String appointmentTime;

private Patient patient;

public Appointment(String appointmentID, String appointmentDate, Doctor doctorRequired, String appointmentTime, Patient patient) {

this.appointmentID = appointmentID;

this.appointmentDate = appointmentDate;

this.doctorRequired = doctorRequired;

this.appointmentTime = appointmentTime;

this.patient = patient;

}

public void displayAppointmentDetails() {

System.out.println("Appointment ID: " + appointmentID);

System.out.println("Appointment Date: " + appointmentDate);

System.out.println("Appointment Time: " + appointmentTime);

System.out.println("Doctor Details:");

doctorRequired.displayDoctorDetails();

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

patient.displayPatientDetails();

}

public static void main(String[] args) {

Patient patient = new Patient("P001", "Karan", 29, "Male", "555-1234");

Doctor doctor = new Doctor("D001", "Dr. Jane Doe", "Neurology", "555-5678");

Appointment appointment = new Appointment("A001", "2024-09-20", doctor, "2:00 PM", patient);

appointment.displayAppointmentDetails();

}

}

class Doctor {

private String doctorID;

private String name;

private String specialization;

private String contactNumber;

public Doctor(String doctorID, String name, String specialization, String contactNumber) {

this.doctorID = doctorID;

this.name = name;

this.specialization = specialization;

this.contactNumber = contactNumber;

}

public void displayDoctorDetails() {

System.out.println("Doctor ID: " + doctorID);

System.out.println("Name: " + name);

System.out.println("Specialization: " + specialization);

System.out.println("Contact Number: " + contactNumber);

}

}

class Patient {

private String patientID;

private String name;

private int age;

private String gender;

private String contactNumber;

public Patient(String patientID, String name, int age, String gender, String contactNumber) {

this.patientID = patientID;

this.name = name;

this.age = age;

this.gender = gender;

this.contactNumber = contactNumber;

}

public void displayPatientDetails() {

System.out.println("Patient ID: " + patientID);

System.out.println("Name: " + name);

System.out.println("Age: " + age);

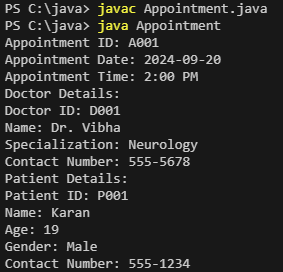
System.out.println("Gender: " + gender);

System.out.println("Contact Number: " + contactNumber);

}

}

**Output:-**

****

**Conclusion:-**  Objects in object-oriented programming help organize and manage code by grouping related data and functions together. This makes it easier to build, understand, and maintain programs. By breaking complex systems into smaller, reusable parts, objects mimic real-world entities, making it simpler to design and modify software.