**Certainly! The provided Java code represents a simple Hospital Management System and consists of the HospitalManagementSystem class. Let's go through the code line by line with explanations:**

package HospitalManagementSystem;

**This line declares the package name in which the HospitalManagementSystem class resides.**

import java.sql.\*;

import java.util.Scanner;

**These import statements bring in the necessary classes from the java.sql package for working with JDBC (Java Database Connectivity) and the Scanner class for user input.**

public class HospitalManagementSystem {

**This line starts the definition of the HospitalManagementSystem class.**

private static final String url = "jdbc:mysql://localhost:3306/hospital";

private static final String username = "root";

private static final String password = "karan@321";

**These lines declare constants for the database URL, username, and password.**

public static void main(String[] args){

**This line starts the main method, which serves as the entry point of the program.**

try{

Class.forName("com.mysql.cj.jdbc.Driver");

}catch (ClassNotFoundException e){

e.printStackTrace();

}

**This block loads the JDBC driver for MySQL. This is necessary in older versions of Java; in recent versions, it's not required, and the driver is loaded automatically.**

Scanner scanner = new Scanner(System.in);

**This line creates a Scanner object to read user input.**

try{

Connection connection = DriverManager.getConnection(url,username,password);

Patient patient = new Patient(connection,scanner);

Doctor doctor = new Doctor(connection);

**In this block, a database connection is established using the provided URL, username, and password. Instances of the Patient and Doctor classes are created and passed the database connection and Scanner object.**

while(true){

System.out.println("HOSPITAL SYSTEM MANAGEMENT");

// Menu options

System.out.println("1. Add patient");

System.out.println("2. View Patients");

System.out.println("3. View Doctors");

System.out.println("4. Book Appointment");

System.out.println("5. Exit");

System.out.println("Enter Your Choice");

int choice = scanner.nextInt();

switch(choice){

// Menu options handling

}

}

**This block initiates an infinite loop that displays a menu for the hospital system. It prompts the user to enter a choice and uses a switch statement to perform actions based on the selected option.**

}catch(SQLException e){

e.printStackTrace();

}

}

**This block catches any SQLException that might occur during database connection or interaction and prints the stack trace.**

public static void bookAppointment(Patient patient, Doctor doctor, Connection connection, Scanner scanner){

**This line starts the definition of the bookAppointment method, which is used to book an appointment between a patient and a doctor.**

System.out.print("Enter Patient Id: ");

int patientId = scanner.nextInt();

System.out.print("Enter Doctor Id: ");

int doctorId = scanner.nextInt();

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

String appointmentDate = scanner.next();

**These lines prompt the user to enter the patient ID, doctor ID, and appointment date.**

if(patient.getPatientById(patientId) && doctor.getDoctorById(doctorId)){

if(checkDoctorAvailability(doctorId, appointmentDate, connection)){

String appointmentQuery = "INSERT INTO appointments(patient\_id,doctor\_id,appointment\_date) VALUES(?,?,?)";

try{

PreparedStatement preparedStatement = connection.prepareStatement(appointmentQuery);

preparedStatement.setInt(1, patientId);

preparedStatement.setInt(2, doctorId);

preparedStatement.setString(3, appointmentDate);

int rowAffected = preparedStatement.executeUpdate();

if(rowAffected>0){

System.out.println("Appointment Booked");

}else{

System.out.println("Failed to book Appointment");

}

}catch(SQLException e){

e.printStackTrace();

}

}else{

System.out.println("Doctor not available on this date");

}

**In this block, it checks if both the patient and doctor with the specified IDs exist. If they do, it then checks if the doctor is available on the specified appointment date using the checkDoctorAvailability method. If the doctor is available, it inserts a new appointment record into the database.**

}else{

System.out.println("Neither Doctor nor Patient Exist");

}

}

**This block handles the case where either the patient or the doctor with the specified IDs does not exist.**

public static boolean checkDoctorAvailability(int doctorId , String appointmentDate , Connection connection){

**This line starts the definition of the checkDoctorAvailability method, which checks if a doctor is available on a specified appointment date.**

String query = "SELECT COUNT(\*) FROM appointments WHERE doctor\_id = ? AND appointment\_date = ?";

try{

PreparedStatement preparedStatement = connection.prepareStatement(query);

preparedStatement.setInt(1, doctorId);

preparedStatement.setString(2, appointmentDate);

ResultSet resultSet = preparedStatement.executeQuery();

if(resultSet.next()){

int count = resultSet.getInt(1);

if(count==0){

return true;

}else{

return false;

}

}

}catch(SQLException e){

e.printStackTrace();

}

return false;

}

}

**In this block, a SQL SELECT query is prepared and executed to count the number of appointments for a specified doctor on a given date. If the count is zero, it means the doctor is available, and the method returns true; otherwise, it returns false. Any SQL exception is caught and printed in the catch block. If an exception occurs, the method also returns false.**