LAB # 1

Introduction to Eclipse IDE

Lab Task:

Write the java code with the help of the given class diagram of Hospital Management System.

CODE:

USER CLASS

```java
package Lab_01;
public class User{
    public void choice() {
        System.out.println("User making choices");
    }
}
```

RECEPTIONIST CLASS

```java
package Lab_01;
import java.util.ArrayList;
import java.util.List;
public class Receptionist extends User {
    int R_id;
    String R_name;
    Billing billing;
    List<Patient> patients;

    public Receptionist(int R_id, String R_name, Billing billing) {
        this.R_id = R_id;
        this.R_name = R_name;
        this.billing = billing;
        patients = new ArrayList<>();
    }

    public void getchoice() {
        choice();
    }

    public void Giveappoint() {
        System.out.println("Receptionist " + R_name + " (ID: " + R_id + ") is giving appointments.");
    }
    public void GenerateBill() {
        System.out.println("Receptionist " + R_name + " (ID: " + R_id + ") is generating a bill: " + billing);
    }
}
```

ONE-TO-MANY-RELATION

A **single Receptionist** can handle **multiple Patients**, represented by the List<Patient> inside the Receptionist class plus Receptionist Class having the aggretion relationship with the Billing class creating Billing object outside the class in Main separately it will use Billing but work independetly

PATIENTCLASS

Adding **Patient** in **Patient_List** which is in Receptionist Class plus Patient Having composition relation with the Billing class creating billing object inside the Patient class

```java
package Lab_01;
public class Patient extends User {
    Doctor D_name;
    Receptionist R_name;
    String P_name;
    int P_id;
    Billing billing;
    public Patient(String P_name, int P_id, Receptionist R_name, Doctor D_name, int bill_no, double bill_amount) {
        this.P_name = P_name;
        this.P_id = P_id;
        this.R_name = R_name;
        this.D_name = D_name;
        R_name.patients.add(this);
        D_name.patients.add(this);
        billing = new Billing(bill_no, bill_amount);
    }
    public void generatechoice() {
        choice();
    }
    public void PayBill() {
        System.out.println("Patient name: " + P_name + " Paying bill: " + billing);
    }
    @Override
    public String toString() {
        return "Patient Name: " + P_name + ", ID: " + P_id;
    }
}
```

DOCTORCLASS

MANY TO ONE RELATION

Many **Patients** can be treated by a single **Doctor**, represented by the List<Patient> inside the Doctor class.

```java
package Lab_01;
import java.util.ArrayList;
import java.util.List;
public class Doctor extends User {
    int D_id;
    String D_name;
    List<Patient> patients;

    public Doctor(int D_id, String D_name) {
        this.D_id = D_id;
        this.D_name = D_name;
        patients = new ArrayList<>();
    }

    public void get_User() {
        choice();
    }

    public void CheckPatient() {
        System.out.println("Many to One Relationship: one Doctor many Patients");
        for (Patient p : patients) {
            System.out.println("Doctor name: " + D_name + " → Patient: " + p);
        }
    }
}
```

Adding the Patient in Many Patient in Patient List which are to be manage By The one Doctor which is Doctor class

```java
package Lab_01;
public class Patient extends User {
 D_name.patients.add(this);
```

BILLINGCLASS

```java
package Lab_01;
public class Billing {
    int bill_no;
    double bill_amount;
    public Billing(int bill_no, double bill_amount) {
        this.bill_no = bill_no;
        this.bill_amount = bill_amount;
    }
    @Override
    public String toString() {
        return "Bill No: " + bill_no + ", Amount: " + bill_amount;
    }
}
```

MAINCLASS

```java
package Lab_01;
public class Hospital_main_Class {
    public static void main(String[] args) {
        Billing bill1 = new Billing(10, 5000);
        Receptionist rec1 = new Receptionist(101, "Muneeb", bill1);
        Doctor d1 = new Doctor(102, "Muneeb");
        Patient pat1 = new Patient("Muneeb", 301, rec1, d1, 10, 2000);
        Patient pat2 = new Patient("MuneebAhmed", 302, rec1, d1, 100, 3000);
        System.out.println("Receptionist giving appointment .....");
        rec1.Giveappoint();
        System.out.println("Receptionist choices .....");
        rec1.getchoice();
        System.out.println("Receptionist generating bill for patient .....");
        rec1.GenerateBill();
        pat1.generatechoice();
        System.out.println("Patient 1 paying bill .....");
        pat1.PayBill();
        System.out.println("Patient 2 paying bill .....");
        pat2.PayBill();
        System.out.println("Patient 2 choice .....");
        pat2.generatechoice();
        System.out.println("Doctor checking patient .....");
        d1.CheckPatient();
        System.out.println("Doctor choices .....");
        d1.get_User();
    }
}
```

OUTPUT

```
Receptionist giving appointment .....
Receptionist Muneeb (ID: 101) is giving appointments.
Receptionist choices .....
User making choices
Receptionist generating bill for patient .....
Receptionist Muneeb (ID: 101) is generating a bill: Bill No: 10, Amount: 5000.0
User making choices
Patient 1 paying bill .....
Patient name: Muneeb Paying bill: Bill No: 10, Amount: 2000.0
Patient 2 paying bill .....
Patient name: MuneebAhmed Paying bill: Bill No: 100, Amount: 3000.0
Patient 2 choice .....
User making choices
Doctor checking patient .....
Many to One Relationship: one Doctor many Patients
Doctor name: Muneeb → Patient: Patient Name: Muneeb, ID: 301
Doctor name: Muneeb → Patient: Patient Name: MuneebAhmed, ID: 302
Doctor choices .....
User making choices
```

2023F-BSE130
SEC:B