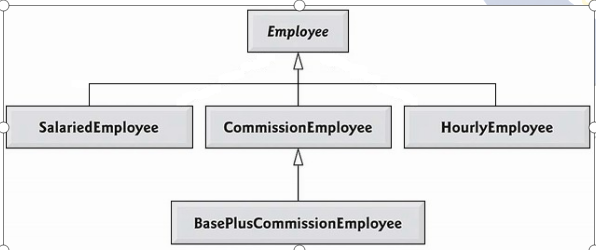
**LAB # 08**

**Task # 01:** Create a payroll system using **classes**, **inheritance** and **polymorphism**

Four types of employees paid weekly

1. **Salaried employees**: fixed salary irrespective of hours
2. **Hourly employees**: 40 hours salary and overtime (> 40 hours)
3. **Commission employees**: paid by a percentage of sales
4. **Base-plus-commission employees**: base salary and a percentage of sales

The information know about each employee is his/her first name, last name and national identity card number. The reset depends on the type of employee.



**Solution:**

**CLASS EMPLOYEE**

public class employee {

String firstname;

String lastname;

String cnic;

public employee() {}

public employee(String firstname, String lastname, String cnic) {

this.firstname = firstname;

this.lastname = lastname;

this.cnic = cnic;}

public void setCnic(String cnic) {

this.cnic = cnic;}

public void setFirstname(String firstname) {

this.firstname = firstname;}

public void setLastname(String lastname) {

this.lastname = lastname;}

public String getFirstname() {

return firstname;}

public String getLastname() {

return lastname;}

public String getCnic() {

return cnic;}

@Override

public String toString(){

return ("\nFirst Name : "+ firstname+"\n "+"Last Name : "+lastname+"\n"+"CNIC : "+cnic+"\n ");}

double earning(){

return 0.0;}

**CLASS SALARIED EMPLOYEE**

public class SalariedEmployee extends employee{

private double weeklysalary;

public void setWeeklysalary(double weeklysalary) {

if (weeklysalary>=0) {

this.weeklysalary = weeklysalary;

}else{

System.out.println("Salary Cn't Be Negative !!");

}}

public SalariedEmployee(double weeklysalary) {

this.weeklysalary = weeklysalary;}

public SalariedEmployee() {}

public SalariedEmployee(String firstname, String lastname, String cnic,double weeklySalary) {

super( firstname, lastname, cnic);

this.weeklysalary=weeklySalary; }

@Override

public String toString() {

return "\n "+"Weekly Salary : "+weeklysalary+"\n"+super.toString(); }

@Override

double earning() {

return weeklysalary; }}

**CLASS HOURLY EMPLOYEE**

public class HourlyEmployee extends employee {

double wage;

double hour;

public void setWage(double wage) {

if (wage >= 0) {

this.wage = wage;

} else {

System.out.println("WAGE Can't Be Negative !!");

}}

public void setHour(double hour) {

if (hour >= 0) {

this.hour = hour;

} else {

System.out.println("HOUR Can't Be Negative !!");

}}

public HourlyEmployee() {}

public HourlyEmployee(String firstname, String lastname, String cnic, double wage, double hour) {

super(firstname, lastname, cnic);

this.wage = wage;

this.hour = hour; }

@Override

public String toString() {

return "HOURLY EMPLOYEE " + "\n " + super.toString();}

@Override

double earning() {

if (hour <= 40) {

return wage \* hour;

} else {

return 40 \* wage + (hour - 40) \* wage \* 1.5;

}}}

**CLASS COMMISSION EMPLOYEE**

public class CommissionEmployee extends employee{

double grossSales ;

double commissionRate ;

public void setGrossSales(double grossSales) {

if (grossSales >= 0) {

this.grossSales = grossSales;

} else {

System.out.println("Gross Sales Can't Be Negative !!");}}

public void setCommissionRate(double commissionRate) {

if (commissionRate >= 0) {

this.commissionRate = commissionRate;

} else {

System.out.println("Commission Rates Can't Be Negative !!");

}}

public CommissionEmployee() {}

public CommissionEmployee(double grossSales, double commissionRate,String firstname, String lastname, String cnic) {

super(firstname, lastname, cnic);

this.grossSales = grossSales;

this.commissionRate = commissionRate;}

@Override

public String toString() {

return "\nCommission Employee: "+super.toString(); }

@Override

double earning() {

return grossSales \* commissionRate;}

**CLASS BASE PLUS COMMISSION EMPLOYEE**

public class BasePlusCommissionEmployee extends CommissionEmployee {

double basesalary;

public double getBasesalary() {

return basesalary;}

public void setBasesalary(double basesalary) {

if (basesalary >= 0) {

this.basesalary = basesalary;

} else {

System.out.println("Base salary Can't Be Negative !!");}}

public BasePlusCommissionEmployee() {}

public BasePlusCommissionEmployee(String firstName, String lastName, String CNIC, double grossSale, double commissionRate, double basesalary) {

super(grossSale, commissionRate, firstName, lastName, CNIC);

this.basesalary = basesalary;}

@Override

public String toString() {

return "\nBase plus Commission employee: " + super.toString();}

@Override

double earning() {

return basesalary + super.earning();}}

**MAIN METHOD**

public class Polymorphism {

public static void main(String[] args) {

employee firstEmployee = new SalariedEmployee("Usman", "Ali", "111-11-1111", 800.00);

employee secondEmployee = new CommissionEmployee(10000, 0.06, "Atif", "Aslam", "222-22-2222");

employee thirdEmployee = new BasePlusCommissionEmployee("Rana", "Naseeb", "333-33-3333", 5000, 0.04, 300);

employee fourthEmployee = new HourlyEmployee("Renson", "Isaac", "444-44-4444", 16.75, 40);

// polymorphism: calling toString() and earning() on Employee’s reference

System.out.println(firstEmployee);

System.out.println("Earning :" + firstEmployee.earning());

System.out.println(secondEmployee);

System.out.println("Earning :" + secondEmployee.earning());

System.out.println(thirdEmployee);

// performing downcasting to access & raise base salary

BasePlusCommissionEmployee currentEmployee

= (BasePlusCommissionEmployee) thirdEmployee;

double oldBaseSalary = currentEmployee.getBasesalary();

System.out.println("old base salary: " + oldBaseSalary);

currentEmployee.setBasesalary(1.10 \* oldBaseSalary);

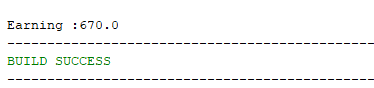
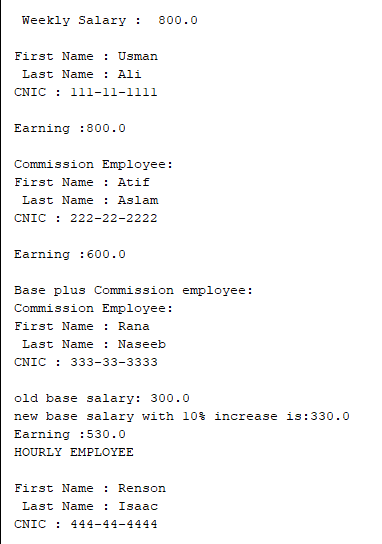
System.out.println("new base salary with 10% increase is:" + currentEmployee.getBasesalary());

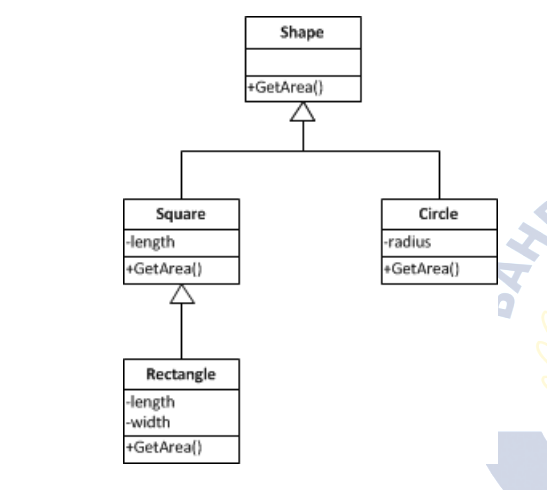
System.out.println("Earning :" + thirdEmployee.earning());

System.out.println(fourthEmployee);

System.out.println("Earning :" + fourthEmployee.earning());}}

**Output:**

****

**Task # 02:** 2. You have to implement the following diagram including some attributes and other functions:

**Solution:**

**CLASS SHAPE**

public class Shape {

public double GetArea()

{

double area=0;

return area;

}

}

**CLASS SQUARE**

public class square extends Shape{

protected double length;

public square(double length)

{

this.length=length; }

public double getLength() {

return length;}

public void setLength(double length) {

this.length=length; }

public double GetArea()

{

double area=length\*length;

return area;}

public String toString()

{

return "Length of one side: "+getLength();

}}

**CLASS CIRCLE**

public class circle extends Shape{

private double radius;

public circle(int radius)

{

this.radius=radius; }

public double getRadius() {

return radius;}

public void setRadius(double radius) {

this.radius=radius; }

public double GetArea()

{

double area=3.14\*radius\*radius;

return area;}

public String toString()

{

return "Radius: "+getRadius()+" Area: "+GetArea();}}

**CLASS RECTANGLE**

public class rectangle extends square{

private double width;

public rectangle(double length,double width)

{

super(length);

this.width=width;

}

public void setWidth(int width) {

this.width = width;}

public double getWidth() {

return width; }

public double GetArea()

{

double area=length\*width;

return area; }

@Override

public String toString() {

super.toString();

return "rectangle " + "lenght=" + getLength() +" rectangle" + " width=" + getWidth() ;} }

**MAIN METHOD**

public class Task2 {

public static void main(String[] args) {

System.out.println("---------AREA OF SQUARE---------");

square s = new square(10);

System.out.println("Area IS : "+s.GetArea());

System.out.println("===================================");

System.out.println("---------AREA OF CIRCLE---------");

circle c = new circle(12);

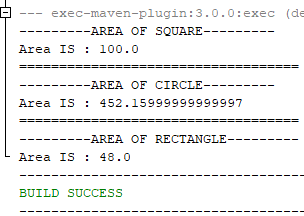
System.out.println("Area IS : "+c.GetArea());

System.out.println("===================================");

System.out.println("---------AREA OF RECTANGLE---------");

rectangle r = new rectangle(12, 4);

System.out.println("Area IS : "+r.GetArea());}}

**Output:**