**OBJECT ORIENTED PROGRAMMING**

**LAB# 07 TASKS**

## **Polymorphism in Java**

1.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.



**Step by Step Guidelines**

**Step 1: Define Employee Class**

* Being the base class, Employee class contains the common behavior. Add firstName, lastName and CNIC as attributes of type String
* Provide getter & setters for each attribute
* Write default & parameterized constructors
* Override **toString**() method as shown below
public String **toString**( ) {

return firstName + “ ” + lastName + “ CNIC# ” + CNIC ;

}

* Define **earning()** method as shown below
 public double earnings( ) {
 return 0.00;

}

Step 2: Define SalariedEmployee Class

* Extend this class from Employee class.
* Add **weeklySalary** as an attribute of type double
* Provide **getter** & **setters** for this attribute. Make sure that **weeklySalary** never sets to **negative** value. (use if)
* Write **default** & **parameterize** constructor. Don’t forget to call default & parameterize constructors of Employee class.
* Override **toString**() method as shown below
public String toString( ) {

 return “\nSalaried employee: ” + super.toString();

}

* Override **earning**() method to implement class specific behavior as shown below

public double **earnings**( ) {

 return weeklySalary;

}

Step 3: Define HourlyEmployee Class

* Extend this class from Employee class.
* Add wage and **hours** as attributes of type double
* Provide **getter** & **setters** for these attributes. Make sure that **wage** and **hours** never set to a negative value.
* Write default & parameterize constructor. Don’t forget to call default & parameterize constructors of Employee class.
* Override **toString**() method as shown below

public String **toString**( ) {

return “\nHourly employee: ” + super.toString();

}

* Override **earning**() method to implement class specific behaviour as shown below

public double **earnings**( ) {

if (hours <= 40){

return wage \* hours;

}

else{

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

}

}

Step 4: Define CommissionEmployee Class

* Extend this class form Employee class.
* Add **grossSales** and **commissionRate** as attributes of type double
* Provide **getter** & setters for these attributes. Make sure that grossSales and commissionRate never set to a negative value.
* Write default & parameterize constructor. Don’t forget to call default & parameterize constructors of Employee class.
* Override **toString**() method as shown below
public String **toString**( ) {

return “\nCommission employee: ” + super.toString();

}

* Override **earning**() method to implement class specific behaviour as shown below

public double **earnings**( ) {

return grossSales \* commisionRate;

}

Step 5: Define BasePlusCommissionEmployee Class

* Extend this class form **CommissionEmployee** class not from Employee class. Why? Think on it by yourself
* Add **baseSalary** as an attribute of type double
* Provide **getter** & **setters** for these attributes. Make sure that **baseSalary** never sets to negative value.
* Write default & parameterize constructor. Don’t forget to call default & parameterize constructors of Employee class.
* Override **toString**() method as shown below
public String toString( ) {

return “\nBase plus Commission employee: ” + super.toString();

}

* Override **earning**() method to implement class specific behaviour as shown below

public double **earnings**( ) {

return baseSalary + super.earning();

 }

Step 6: Putting it all Together

public class PayRollSystemTest {

 public static void main (String [] args) {

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

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

 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(firstEmployee.earnings());

 System.out.println(secondEmployee);

 System.out.println(secondEmployee.earnings());

 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(thirdEmployee.earnings() );

 System.out.println(fourthEmployee);

 System.out.println(fourthEmployee.earnings() );

 } // end main

} // end class

1. You have to implement the following diagram including some attributes and other functions:

