LABORATORY MANUAL

OBJECT ORIENTED PROGRAMMING (OOP)

Submitted by

AADARSH MEHDI | 00314965

Instructor

Sir. Rozi Khan
Lab Engineer, Department of Computer Science,
National University of Sciences & Technology,
Balochistan Campus (NBC), Quetta.

National University of Sciences & Technology (NUST)
Balochistan Campus (NBC), Department of Computer Science

LAB 04:

TASK 1(Classes):

INPUT:

Abstract Class: Employee

```
public abstract class Employee {
    private String firstName;
    private String lastName;
    private String socialSecurityNumber;
    private int day, month, year;
    private Date birthDate=new Date(day,getBirthMonth(),year);

    public Employee(String first, String last, String ssn, int d, int m, int y)
    {
       setFirstName(first);
       setLastName(last);
       setSocialSecurityNumber(ssn);
       day=d;
       setBirthMonth(m);
       year =y;

    }

    public void setFirstName(String f)
    {
        firstName = f;
    }
    public String getFirstName()
    {
        return firstName;
    }
```

```
    public void setLastName(String l)
    {
        lastName = l;

    }
    public String getLastName()
    {
        return lastName;
    }
    public void setSocialSecurityNumber(String s)
    {
        socialSecurityNumber = s;
    }
    public String getSocialSecurityNumber()
    {
        return socialSecurityNumber;
    }
    public void setBirthMonth(int m) {
        month=m;
    }
    public int getBirthMonth() {
        return month;
    }

    public int getCalenderMonth() {
        return birthDate.getCurrentMonth();
    }
    public String toString()
    {
        return String.format("%s %s\n%s: %s", getFirstName(), getLastName(), "" +
                "Social-Security Number", getSocialSecurityNumber());
    }

    public abstract double earnings();

}
```

Class: BasePlusCommissionEmployee

```java
public class BasePlusCommissionEmployee extends CommissionEmployee {
    private double baseSalary;
    public BasePlusCommissionEmployee(String first, String last, String ssn,
            double sales, double rate, int d, int m, int y, double bSalary) {
        super(first, last, ssn, sales, rate, d, m ,y);
        setBaseSalary(bSalary);

    }
    public void setBaseSalary(double salary)
    {
        if(salary > 0.0 )
        {
            baseSalary = salary;
        }
        else
            throw new IllegalArgumentException("Salary must be greater than 0.0");
    }
    public double getBaseSalary()
    {
        return baseSalary;
    }

    @Override
    public double earnings() {
        if (getCalenderMonth()==getBirthMonth()) {
            return (getBaseSalary() + super.earnings()+100.00);
        }
        else {
            return getBaseSalary() + super.earnings();
        }
    }

    @Override
    public String toString() {
        return String.format("%s: %s\n%s: $%,.2f","Base Salaried", super.toString(), "Base Salary", getBaseSalary());
    }
}
```

Class: CommissionEmployee:

```
public class CommissionEmployee extends Employee {
    private double grossSales;
    private double commissionRate;

    public CommissionEmployee(String first, String last, String ssn, double sales, double rate,int d, int m, int y) {

        super(first, last, ssn,d,m , y );
        setCommissionRate(rate);
        setGrossSales(sales);

    }
    public void setGrossSales(double gSales)
    {
        if(gSales > 0.0)
        {
            grossSales = gSales;
        }
        else
            throw new IllegalArgumentException("Gross sales must be greater than 0.0");

    }
    public double getGrossSales()
    {
        return grossSales;
    }
```

```
   public void setCommissionRate(double cRate)
   {
       if(cRate > 0.0 )
       {
           commissionRate = cRate;
       }
       else
           throw new IllegalArgumentException("Commission rate cannot be 0.0 ");
   }
   public double getCommissionRate()
   {
       return commissionRate;
   }

   @Override
   public double earnings() {
       if (getCalenderMonth()==getBirthMonth()) {
           return (getGrossSales() * getCommissionRate()+100.00);
       }
       else {
           return getGrossSales() * getCommissionRate();
       }
   }
```

```
    @Override
    public String toString() {
        return String.format("%s: %s\n%s: $%,.2f; %s: %,.2f", "Commission Employee",
                super.toString(), "Gross Sales", getGrossSales(), "Commission rate", getCommissionRate());
    }
}
```

Class: HourlyEmployee:

```java
public class HourlyEmployee extends Employee {
    private double wage;
    private double hours;
    public HourlyEmployee(String first, String last, String ssn, double wages, double hour, int d, int m , int y) {

        super(first, last, ssn, d, m , y);
        setHours(hour);
        setWage(wages);

    }
    public void setWage(double hourlyWage)
    {
        if(hourlyWage >= 0.0)
        {
            wage = hourlyWage;
        }
        else
            throw new IllegalArgumentException("Wages must be greater than or equal to 0.0");
    }
    public double getWage()
    {
        return wage;
    }
    public void setHours(double hour)
    {
        if(hour >= 0.0 && hour <= 168)
        {
            hours = hour;
        }
        else
            throw new IllegalArgumentException("Hours must be between 0 - 168");
    }
    public double getHours()
    {
        return hours;
    }

    @Override
    public double earnings() {
        if(getHours() <= 40)
        {
            if (getCalenderMonth()==getBirthMonth()) {
                return (getWage()*getHours())+100.00;
            }
            else {
                return getWage() * getHours();
            }
        }
        else
        {
            if (getCalenderMonth()==getBirthMonth()) {
                return (40 * getWage() + (getHours() - 40) * getWage() * 1.5)+100.000;
            }
            else {
                return 40 * getWage() + (getHours() - 40) * getWage() * 1.5;
            }
        }
    }

    @Override
    public String toString() {
        return String.format("Hourly Employee: %s\n%s: $%,.2f; %s: %,.2f", super.toString(),
                "Wages", getWage(), "Hours worked", getHours());
    }
}
```

Class:SalariedEmployee:

```
public class SalariedEmployee extends Employee {

    private double weeklySalary;
    public SalariedEmployee(String first, String last, String ssn, double week, int d, int m, int y) {
        super(first, last, ssn, d, m , y);
        setWeeklySalary(week);
    }
    public void setWeeklySalary(double week)
    {
        if(week > 0.0)
        {
            weeklySalary = week;
        }
        else
            throw new IllegalArgumentException("Weekly Salary must be greater than 0.0");

    }
    public double getWeeklySalary()
    {
        return weeklySalary;
    }
    @Override
    public String toString()
    {
        return String.format("%s: %s\n%s: $%,.2f",
                "Salaried Employee", super.toString(), "Weekly Salary", getWeeklySalary());
    }
```

```
    @Override
    public double earnings() {
        if (getCalenderMonth()==getBirthMonth()) {
            return (getWeeklySalary()+100.00);
        }
        else {
            return getWeeklySalary();
        }
    }
}
```

Class: Date:

```java
import java.util.Calendar;
public class Date {
    private int month;
    private int day;
    private int year;
    private int currentMonth = 1 + Calendar.getInstance().get(Calendar.MONTH);
    private static final int [] daysPerMonth = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    public Date(int theMonth, int theDay, int theYear)
    {
        month = checkMonth(theMonth);
        year = theYear;
        day = checkDay(theDay);
    }
    public int checkMonth(int testMonth)
    {
        if(testMonth <= 0 && testMonth > 12)
        {
            throw new IllegalArgumentException("Month value must be in range 0f 1 - 12");

        }
        else
        {
            return testMonth;
        }
    }
```

```java
    public int checkDay(int testDay)
    {
        if(testDay <= 0 && testDay > daysPerMonth[month])
        {
            throw new IllegalArgumentException("Day is not valid for this Month & Year");
        }

        if(month == 2 && day == 29 && (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0)))
        {
            return testDay;
        }
        else {
            return testDay;
        }
    }
    public int getCurrentMonth() {
        return currentMonth;
    }
    public String toString()
    {
        return String.format("%d/%d/%d ", month, day, year);
    }
    public int getMonth() {
        return month;
    }


}
```

National University of Sciences & Technology (NUST)
Balochistan Campus (NBC), Department of Computer Science

TASK 2(Classes):

INPUT:

Class: Pieceworkers:

```
public class PieceWorker extends Employee{
    private double wage;
    private int pieces;
    public PieceWorker(String first, String last, String ssn, double wageperprice, int piece, int d, int m, int y) {
        super(first, last, ssn, d, m, y);
        setWage(wageperprice);
        setPieces(pieces);
    }
    public void setWage(double wpp) {
        wage=wpp;
    }
    public double getWage() {
        return wage;
    }
    public void setPieces(int p) {
        pieces=p;
    }
    public int getPieces() {
        return pieces;
    }

    @Override
    public double earnings() {
        if(getCalenderMonth()==getBirthMonth()) {
            return (getPieces()*getWage()+100.00);
        }
        return getPieces()*getWage();
    }
}
```

Test Class:

Class: PayRollTest:

```
public class PayRollTest {
    public static void main(String[] args) {
        //Task 1 and Task2 is shown demonstrated here:
        Employee emptype []=new Employee[5];
        emptype[0]=new BasePlusCommissionEmployee("Mustafa", "Ali", "123456", 450, 0.9, 8, 2,2000, 80.00);
        emptype[1]=new CommissionEmployee("Javed", "Akbar", "1258465", 140, 1.2, 2, 3, 2001);
        emptype[2]=new HourlyEmployee("Aadarsh Mehdi", "Hyderi", "85551512", 50, 40, 2, 9, 2000);
        emptype[3]=new SalariedEmployee("Kashif ", "Hyderi", "14782205520", 500, 19, 2,1999);
        emptype[4]=new PieceWorker("Missem ", "Raza", "1475224494", 120.00, 7, 2, 8, 2006);        //Task 2 implementation
        for (int i =0; i<5; i++) {
            System.out.printf("%s\n%s: $%, .2f", emptype[i], "Earned", emptype[i].earnings());
            System.out.println();
        }


    }
}
```

OUTPUT for Both Tasks**:**