LOGICIEL DE PAIE UTILISANT LE POLYMORPHISME

Ce TP a pour but d'exploiter les classes, les méthodes abstraites et le polymorphisme (en Java à l'aide de L'IDE Eclipse) pour effectuer des calculs de paie en fonction du type d'employé. Nous utilisons pour ce faire une superclasse abstraite ***Employe.***

Les sous-classes de la classe ***Employe*** seront :

- ***Patron*** payé à salaire hebdomadaire fixe.
- ***TravailleurCommission*** rémunéré selon un traitement de base auquel s'ajoute un pourcentage lié aux ventes.
- ***TravailleurPiece*** rétribué selon le nombre de pièces produites.
- ***TravailleurHoraire*** qui reçoit sa paie selon le nombre d'heures travaillées, y compris des heures supplémentaires.

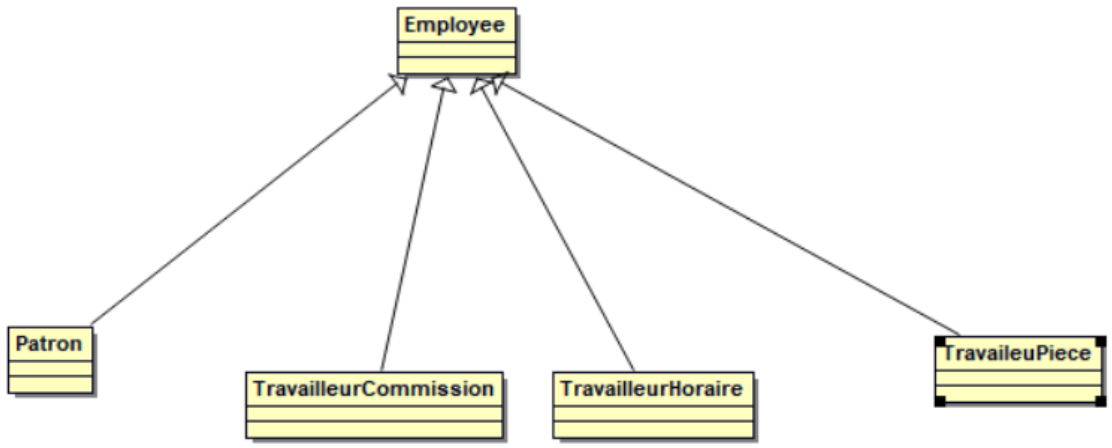

Diagramme de classes

Implémentation des classes

Classe Employe :

```java
package Ppatron;
import Pemploye.Employe;

public final class Patron extends Employe {


    private double salaireHebdo;

    public Patron(String n, String p, double sal) {
        super(n, p);/* super est utilisé pour accéder à la variable n et p de la classe parente Employe*/
        salaireHebdo = sal;

    }


    public void setSalaireHebdo(double salaireHebdo) {
        this.salaireHebdo = salaireHebdo;/* dans une méthode, on a un paramètre ayant le même nom qu'un
        attribut de la classe dont la méthode fait partie, on peut désigner explicitement l'attribut grâce à this*/
    }

     public double gains() {
         return salaireHebdo;
     }

     public String toString() {
         return "Patron: " + super.toString(); /*Le mot-clé « super » est
          utilisé pour appeler la méthode toString de la classe parent qui est Employe*/
     }

}
```

Classe Patron :

```java
package Pemploye;

public abstract class Employe {

    private String nomFamille;
    private String prenom;

    public Employe(String nom, String pre) {
    nomFamille = nom;
    prenom = pre;
    }

    public String getNomFamille() {
        return nomFamille;
    }

    public void setNomFamille(String nomFamille) {
        this.nomFamille = nomFamille;
    }

    public String getPrenom() {
        return prenom;
    }

    public void setPrenom(String prenom) {
        this.prenom = prenom;
    }

    public String toString() {
        return prenom +' '+ nomFamille;
    }

    public abstract double gains();



}

```

Classe TravailleurComission :

```java
package Ptrav;
import Pemploye.Employe;

public final class TravailleurCommission extends Employe {

    private double salFix;
    private double commis;
    private double quantite;

public TravailleurCommission(String nom, String prenom, double sal, double com, double quant )
    {
    super(nom, prenom);/* super est utilisé pour accéder à la variable nom et prenom de la classe parente Employe*/
    salFix = sal;
    commis = com;
    quantite = quant;

    }


public void setSalaire(double salFix) {
    this.salFix = salFix;/* dans une méthode, on a un paramètre ayant le même nom qu'un
    attribut de la classe dont la méthode fait partie, on peut désigner explicitement l'attribut grâce à this*/
}

public void setCommis(double commis) {
    this.commis = commis;
}

public void setQuantite(double quantite) {
    this.quantite = quantite;
}

public double gains() {
    return salFix+(commis*quantite);
}

public String toString() {
    return "Travailleur à la commission : " +super.toString();  /*Le mot-clé « super » est
      utilisé pour appeler la méthode toString de la classe parent qui est Employe*/
}

```

TravailleurPiece :

```java
package Ptravpi;
import Pemploye.Employe;

public final class TravailleurPiece extends Employe {

    private double retrib;
    private double quantpie;

    public TravailleurPiece(String nom, String prenom, double retr, double quantp) {
        super(nom, prenom);/* super est utilisé pour accéder à la variable nom et prenom de la classe parente Employe*/
        retrib = retr;
        quantpie = quantp;
    }

    public void setRetributionParPiece(double retrib) {
        this.retrib = retrib;/* dans une méthode, on a un paramètre ayant le même nom qu'un
        attribut de la classe dont la méthode fait partie, on peut désigner explicitement l'attribut grâce à this*/
    }

    public void setQuantite(double quantpie) {
        this.quantpie = quantpie;
    }

    public double gains() {
        return retrib*quantpie;
    }

    public String toString()
    {
        return "Travailleur à la pièce: " + super.toString();
    }


```

TravailleurHoraire :

```java
package Ptravhor;
import Pemploye.Employe;

public final class TravailleurHoraire extends Employe {

    private double retribHeure;
    private double nbreHeurePrest;

public TravailleurHoraire(String prenom, String nom, double retribH, double nbreH) {
    super(nom, prenom);/* super est utilisé pour accéder à la variable nom et prenom de la classe parente Employe*/
    retribHeure = retribH;
    nbreHeurePrest = nbreH;
}

public void setRetribHeure(double retribHeure) {
    this.retribHeure = retribHeure;/* dans une méthode, on a un paramètre ayant le même nom qu'un
    attribut de la classe dont la méthode fait partie, on peut désigner explicitement l'attribut grâce à this*/
}

public void setNbreHeurePrest(double nbreHeurePrest) {
    this.nbreHeurePrest = nbreHeurePrest;
}

public double gains() {
    return nbreHeurePrest*retribHeure;
}

public String toString() {
    return "Travailleur horaire : " +super.toString(); /*Le mot-clé « super » est
      utilisé pour appeler la méthode toString de la classe parent qui est Employe*/
}

}
```

Extrait de la classe Test (*qui contient la méthode Main*) :

```java
package Ptest;
import java.text.DecimalFormat;

//Packages d'extension Java.
import javax.swing.JOptionPane;
import Pemploye.*;
import Ppatron.Patron;
import Ptrav.*;
import Ptravhor.*;
import Ptravpi.*;

public class Test {



    public static void main( String args[] )
    {
       Employe unEmploye;  // référence de la superclasse
       String sortie = "";

       Patron patron = new Patron( "Jean", "Beaupré", 800.0 );

       TravailleurCommission travailleurCommission = new TravailleurCommission( "Suzanne", "Bélanger", 400.0, 3.0, 150 );

       TravailleurPiece travailleurPiece = new TravailleurPiece( "Robert", "Lévis", 2.5, 200 );

       TravailleurHoraire travailleurHoraire = new TravailleurHoraire( "Hélène", "Prince", 13.75, 40 );

       DecimalFormat precision2 = new DecimalFormat( "0.00" );

       // Référence Employe à un Patron.
       unEmploye = patron;

       sortie += unEmploye.toString() + " a droit à " +
          precision2.format( unEmploye.gains() ) + " $\n";

       // Référence Employe à un TravailleurCommission.
       unEmploye = travailleurCommission;

       sortie += unEmploye.toString() + " a droit à " +
          precision2.format( unEmploye.gains() ) + " $\n" ;
```

Résultat Final :