JAVA TP2

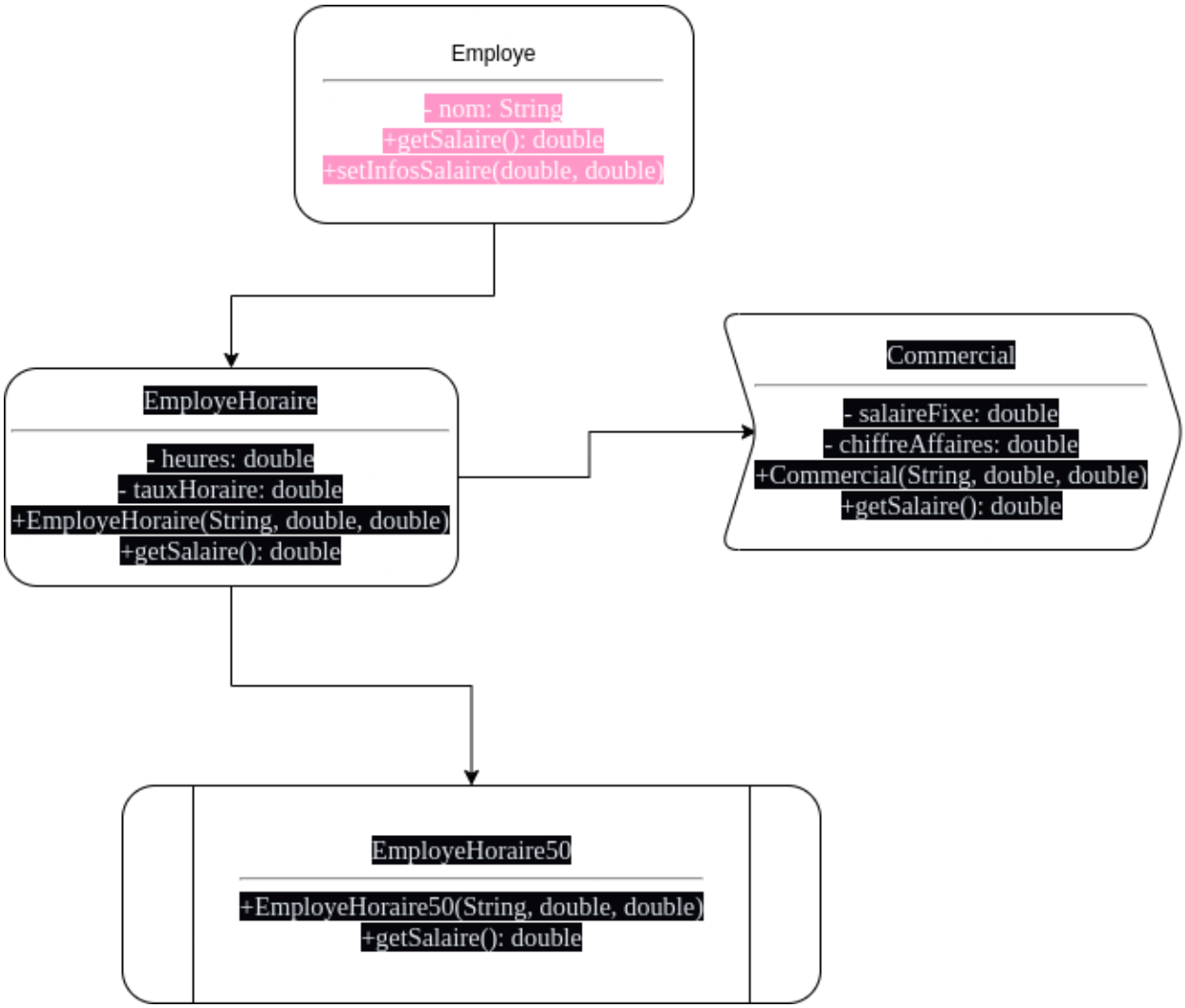
06/08/2024

—

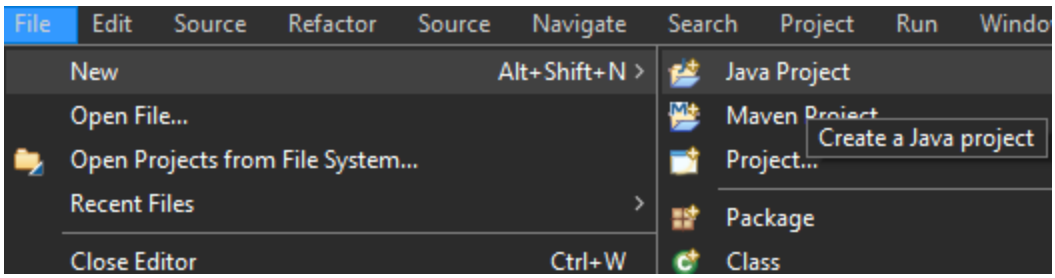
HAFAIDH MOHAMED
2BTS-SIO

Exercice : L'héritage appliqué aux employés d'une entreprise, polymorphisme

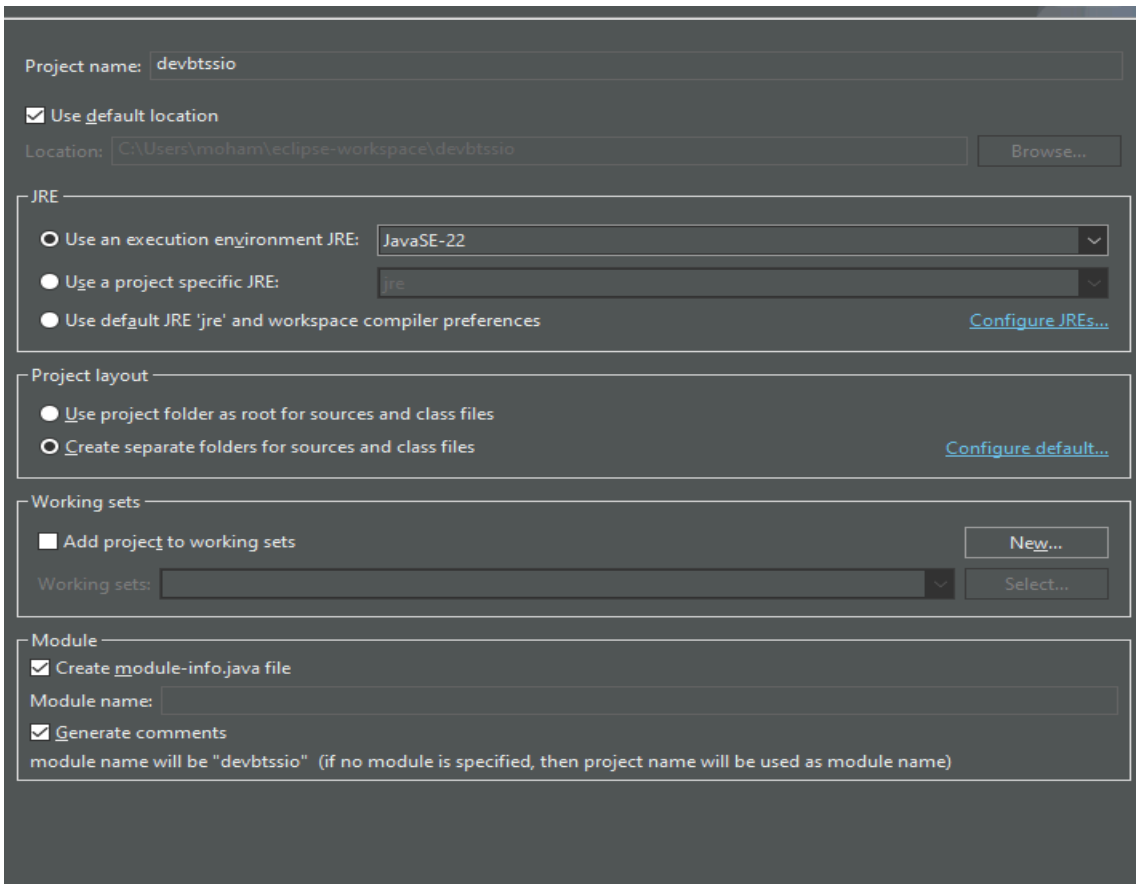
Architecture ULM :



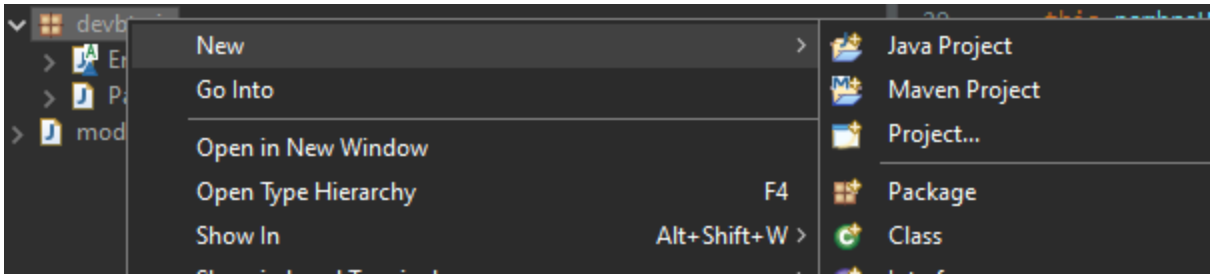
Tout d'abord nous allons créer un projet java sur eclipse pour cela nous devons créer nous devons nous rendre dans file puis new et java project :

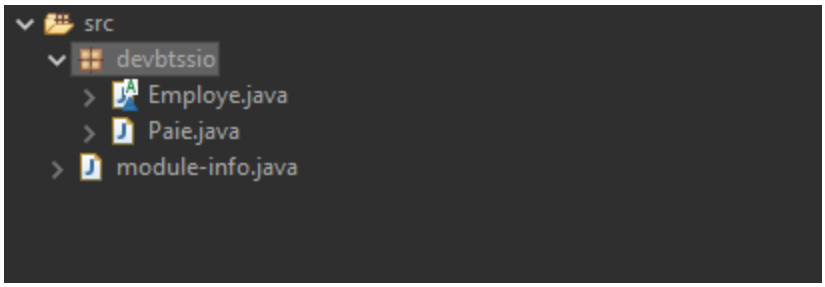


Une fois cela fait nous devons donner un nom à notre projet vous l'appelez comme vous voulez bien évidemment, pour mon cas je l'ai appelé devbtssio. Puis vous laisser les paramètre par défaut :



Une fois le projet créé vous devez créer 2 classes dont une classe qui comporte le nom Employe et une autre dont le nom Paie.





Dans la classe Employe nous devons déclarer nom :

```java
package devbtssio;

//Classe de base Employe
abstract class Employe {
 protected String nom;

 public Employe(String nom) {
     this.nom = nom;
 }

```

Attention nous devons déclarer le nom en protected. Le but principal du mot-clé protected est de faire hériter la méthode ou la variable au classe.

Nous allons déclarer une Méthode pour définir les informations sur le salaire

```
// Méthode pour définir les informations sur le salaire
public void setInfosSalaire(double... infos) {}
```

J'ai utilisé le double... infos le symbole (...) après le double indique que cette méthode peut accepter un nombre variable d'arguments de type double c'est à dire que lorsque nous allons appeler cette méthode on peut passer zéro, un ou plusieurs arguments de type double

Exemple : Seul argument :

```
setInfosSalaire(1200.0);
```

Exemple : Plusieurs argument :

```
setInfosSalaire(2500.0, 3000.0, 3500.0);
```

Maintenant nous allons passer au getSalaire :

```
 // Méthode abstraite pour calculer le salaire
 public abstract double getSalaire();
}
```

Attention utilisez bien abstract car ce mot-clé dit que la méthode est abstraite ça veut dire qu'elle ne contient pas de corps autrement dit pas d'implémentation.

```
//Classe EmployeHoraire avec heures supplémentaires payées 30% de plus
class EmployeHoraire extends Employe {
 protected double nombreHeuresTravail;
 protected double tarifHoraire;
```

La classe `EmployeHoraire` hérite de la classe `Employe` ça veut dire qu'elle doit implémenter toutes les méthodes abstraites utilisées dans `Employe` exemple : Comme la méthode dans `getSalaire()`.

Les attributs protégés sont nombreHeuresTravail puis tarifHoraire :

```java
public EmployeHoraire(String nom) {
    super(nom);
}

public EmployeHoraire(String nom, double nombreHeuresTravail, double tarifHoraire) {
    super(nom);
    this.nombreHeuresTravail = nombreHeuresTravail;
    this.tarifHoraire = tarifHoraire;
}
```

Constructeur : EmployeHoraire(String nom) :

EmployeHoraire : Un constructeur qui prend un paramètre nom puis il fait appelle constructeur de la classe Employe.

Constructeur : EmployeHoraire(String nom, double nombreHeuresTravail, double tarifHoraire) :

```java
33  @Override
34  public void setInfosSalaire(double... infos) {
35      this.nombreHeuresTravail = infos[0];
36      this.tarifHoraire = infos[1];
37  }
38
39  @Override
40  public double getSalaire() {
41      double heuresNormales = Math.min(nombreHeuresTravail, 35);
42      double heuresSup = Math.max(nombreHeuresTravail - 35, 0);
43      return heuresNormales * tarifHoraire + heuresSup * tarifHoraire * 1.3;
44  }
45 }
```

setInfosSalaire(double... infos) :

Dans cette méthode j'ai fait comme un tableau :

`infos[0]` : nombre d'heures travaillées,

`infos[1]` : tarif horaire.

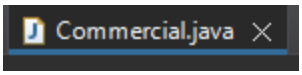
La ligne 41 sert à calculer des heures normales, c'est pour cela que j'ai utilisé la fonction `Math.min()` cela me permet d'obtenir le plus petit des deux nombres entre `nombreHeuresTravail` et `35` heures. Ceci nous aide car si l'employé a travaillé plus de 35h, seules 35 heures sont considérées comme ses horaires normales.

La ligne 42 sert à calculer les heures supplémentaires, pour cette ligne de code j'utilise la fonction `Math.max()` pour calculer les heures supplémentaires. Donc si l'employé a travaillé plus de `35 heures`, donc `nombreHeuresTravail - 35` donnera les heures supplémentaires. Sinon si les `nombreHeuresTravail` est inférieur ou égal à `35` il n'y a pas d'heures supplémentaires donc le résultat sera automatiquement `0`.

La ligne 43 sert à calculer le salaire cette ligne calcule le salaire total donc

Heures normales : heuresNormales * tarifHoraire

Heures supplémentaires : heureSup * tarifHoraire * 1.3 pourquoi 1.3 car les heures supplémentaires sont payées à 130% du tarif horaire normal donc un coefficient de 1,3.



Nous allons créer une class Commercial :

```java
package devbtssio;

//Classe Commercial avec salaire fixe + 1% du chiffre d'affaires
class Commercial extends Employe {
private double salaireFixe;
private double chiffreAffaires;

public Commercial(String nom) {
    super(nom);
}
```

SalaireFixe = c'est le salaire fixe de l'employé commercial

chiffreAffaires = le chiffre d'affaire réalisé par un employé commercial

quand un objet commercial est créé le nom de l'employé est transmis à la classe Employe pour initialiser la partie héritée de l'objet.

```java
public Commercial(String nom, double salaireFixe, double chiffreAffaires) {
    super(nom);
    this.salaireFixe = salaireFixe;
    this.chiffreAffaires = chiffreAffaires;
}

@Override
public void setInfosSalaire(double... infos) {
    this.salaireFixe = infos[0];
    this.chiffreAffaires = infos[1];
}

@Override
public double getSalaire() {
    return salaireFixe + chiffreAffaires * 0.01;
}
}
```

Puis ce code ressemble exactement que le code de la classe Employe sauf pour le chiffre d'affaires et salaire fixe j'ai changer le pourcentage qui est de 1% car le commercial gagne 1% sur le chiffre d'affaires.

EmployeHoraire50.java

Vous devez créer une autre classe EmployeHoraire50 :

```java
package devbtssio;

//Classe EmployeHoraire50 avec heures supplémentaires payées 50% de plus
class EmployeHoraire50 extends EmployeHoraire {

public EmployeHoraire50(String nom) {
    super(nom);
}

public EmployeHoraire50(String nom, double nombreHeuresTravail, double tarifHoraire) {
    super(nom, nombreHeuresTravail, tarifHoraire);
}

@Override
public double getSalaire() {
    double heuresNormales = Math.min(nombreHeuresTravail, 35);
    double heuresSup = Math.max(nombreHeuresTravail - 35, 0);
    return heuresNormales * tarifHoraire + heuresSup * tarifHoraire * 1.5;
}
}
```

Cette classe EmployeHoraire50. Un employé est payé normalement puis avec un tarif majoré de 50% pour ses heures supplémentaires bien sur effectuées au-delà de 35h. Cette classe utilise un héritage de la classe EmployeHoraire puis la méthode getSalaire pour calculer le salaire de 50% .

Pour fini on doit cree une classe Paie.java :

Dans cette classe nous devons créer une méthode principale qui est exécutée lorsque le programme démarre.

```java
package devbtssio;

//Classe principale pour gérer la paie
public class Paie {

public static void main(String[] args) {
    // Création d'un tableau d'employés
    Employe[] employes = new Employe[5];

    // Employé horaire avec heures supplémentaires à 30%
    employes[0] = new EmployeHoraire("Mohamed", 40, 20);

    // Employé horaire avec heures supplémentaires à 50%
    employes[1] = new EmployeHoraire50("Nilda", 45, 25);

    // Commercial avec un salaire fixe et un chiffre d'affaires
    employes[2] = new Commercial("Tamdam", 1000, 50000);

    // Employé horaire créé avec uniquement le nom, informations ajoutées après
    employes[3] = new EmployeHoraire("Nico");
    employes[3].setInfosSalaire(38, 22);

    // Un autre commercial créé avec des informations
    employes[4] = new Commercial("Charle", 1200, 60000);

    // Boucle pour afficher les salaires
    for (Employe employe : employes) {
        System.out.println(employe.nom + " gagne " + employe.getSalaire() + " €");
    }
}
}
```

Nous avons besoin d'un tableau d'employés un tableau avec 5 éléments pour stocker des objet de type Employe

Puis on ajoute des employés avec heure supplémentaire à 30% comme la ligne 11 :

```
employes[0] = new EmployeHoraire("Mohamed", 40, 20);
```

Monsieur Mohamed travaille 40 heures par semaine avec un tarif horaire de 20 € puis les heures sont payées à 30% de plus.

```
employes[1] = new EmployeHoraire50("Nilda", 45, 25);
```

Monsieur Nilda Il travaille 45 heures par semaine avec un tarif horaire de 25 €. Les heures supplémentaires sont payées à 50 % de plus, comme défini dans la classe EmployeHoraire50.

```
employes[2] = new Commercial("Tamdam", 1000, 50000);
```

Monsieur Tandam Il a un salaire fixe de 1000 € et réalise un chiffre d'affaires de 50 000 €. Sa rémunération inclut une commission basée sur ce chiffre d'affaires.

```
employes[3] = new EmployeHoraire("Nico");
employes[3].setInfosSalaire(38, 22);
```

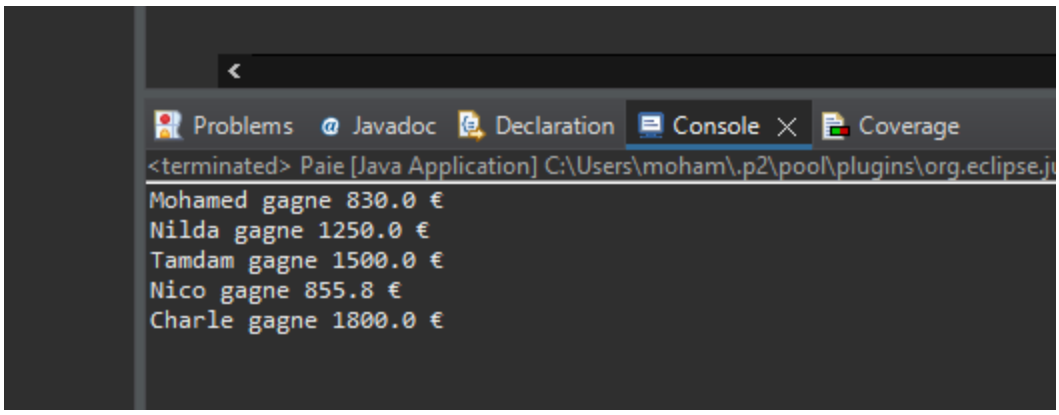
Les heures travaillées (38) et le tarif horaire (22 €) sont ajoutés ensuite avec la méthode `setInfosSalaire`

```
employes[4] = new Commercial("Charle", 1200, 60000);
```

Charle Il a un salaire fixe de 1200 € et un chiffre d'affaires de 60 000 €.

Puis j'ai fait une boucle pour afficher les salaire comme la ligne 27 donc c'est une boucle for cette boucle parcours chaque élément du tableau employes, pour chaque employé elle affiche son nom et son salaire en appelant la méthode getSalaire() de chaque objet Employe.

Résultat finale :




FIN-TASTIC!