xTP2\_Travail Pratique 2 – Employés et salaires

Table des matières

[1 Questions 2](#_Toc144991359)

[2 Mise en situation 2](#_Toc144991360)

[2.1 Informations générales 2](#_Toc144991361)

[2.2 Structure 2](#_Toc144991362)

[2.3 Données et actions 3](#_Toc144991363)

[2.3.1 Classe Employe 3](#_Toc144991364)

[3 Tâches à réaliser 4](#_Toc144991365)

[3.1 Import de l’interface utilisateur (GUI) 4](#_Toc144991366)

[3.2 Classe Employe et gestion des salaires 4](#_Toc144991367)

[3.3 Validations d’entrées de l’utilisateur dans le formulaire 5](#_Toc144991368)

[3.3.1 Structure 5](#_Toc144991369)

[3.3.2 Tâches 6](#_Toc144991370)

[4 Remise 6](#_Toc144991371)

**Consignes**

|  |
| --- |
| Toutes documentation permise.  Le code DOIT utiliser les notions vues en classe (avec exception pour les méthodes de validation, dont vous devez trouver la solution par vous-même). |

[]

# Mise en situation

## Informations générales

On désire modéliser une application qui permet de gérer la paie d’employés, avec le salaire brut et les différentes déductions produisant un salaire net.

Vous devez créer les package et les classes, choisir le type des données, les modificateurs d’accès et de non-accès et compléter la logique du code.

Identifiez les valeurs qui devraient être des constantes et traitez ces constantes selon les meilleures pratiques.

Une interface graphique est fournie avec le projet

## Structure

Pour ce projet, on travaille dans 3 packages avec 3 classes tel que définit ci-dessous.

Projet **C34\_TP2**

Package **gestionnaire\_paie**

* Classe **PaieGUI** La classe contient la méthode main(), est l’interface graphique, déjà créée pour vous.

Package **employe\_salaire**

* Classe **Employe** Classe pour l’entité Employe avec ses informations et méthodes de gestion de salaire

Package **validation**

* Classe **ValidationFormulaire** Classe d’outils pour valider les valeurs reçues de l’utilisateur

## Données et actions

### Classe Employe

**Champs**

* nom Nom de l’employe
* numero Un code alphanumérique unique à l’employé dans l’entreprise
* salaireHoraire Le salaire brut à l’heure de l’employé, nombre réel
* nbreHeuresSemaine Le nombre d’heures travaillées dans la semaine, nombre réel
* anciennete Le nombre d’années depuis que l’employé est à l’emploi de la compagnie, nombre entier

**Informations additionnelles**

Une interface graphique est fournie en tant que classe principale (main) pour tester votre code.

Comme cette interface fait appel à certaines de vos classes et de leurs méthodes, il est important de nommer les classes et méthodes **exactement** comme demandé et de respecter les types des champs.

# Tâches à réaliser

## Import de l’interface utilisateur (GUI)

1. Dans un nouveau projet **C34\_TP2**, créez le package **gestionnaire\_paie**.
2. Dans le gestionnaire de fichiers de Windows, dans le répertoire **C34\_TP2\scr\gestionnaire\_paie**, copiez les fichiers fournis : **PaieGUI.java** et **PaieGUI.form** qui vous sont fournis.

## Classe Employe et gestion des salaires

NOTE : toutes les méthodes de la classe **Employe** doivent être disponibles dans les classes des autres packages.

1. Dans le projet **C34\_TP2**, créez le package **employe\_salaire** et la classe **Employe** avec ses champs.
2. Créez une méthode **setEmploye()** qui reçoit un paramètre pour chacun des champs d’un objet **Employe** et qui permet d’initialiser tous ses champs.
3. Concevez une méthode **salaireBrut()** permettant de retourner le salaire brut d'un employé, soit le nombre d'heures travaillées \* le salaire horaire de cet employé.
4. Concevez une méthode **salaireNetAvantImpot()** retournant le salaire brut auquel on a enlevé les déductions suivantes :

|  |  |
| --- | --- |
| **déduction** | **montant** |
| assurance – emploi | 11,1 % du salaire brut |
| fonds de pension | 1,36% du salaire brut |
| cotisation syndicale | montant fixe de 20 dollars |

1. Concevez une méthode **impotFederal()** retournant le montant correspondant à l'impôt fédéral retenu. Il correspond à **19%** du salaire net avant impôt.
2. Concevez une méthode **impotProvincial()** retournant le montant correspondant à l'impôt provincial retenu. Il correspond à **20%** du salaire net avant impôt.
3. La méthode **salaireNetApresImpot()** correspond donc au salaire net avant impôt auquel on soustrait les montants correspondants à l'impôt fédéral et à l'impôt provincial.
4. La méthode **joursVacances()** sert à retourner le nombre de jours de vacances auquel un employé a droit. Ce nombre dépend de l'ancienneté de l'employé et de sa classe d'employé reflétée par son numéro d'employé. Le tableau suivant représente la situation :

|  |  |  |
| --- | --- | --- |
| **classe d'employé** | **caractéristique** | **nombre de jours de vacances** |
| A | le premier chiffre du numéro d'employé est un '1' | 5 jours + 1 jour par année d'ancienneté |
| B | le premier chiffre du numéro d'employé est un '2' | 10 jours + 1 jour par année d'ancienneté |
| C | le premier chiffre du numéro d'employé est un '3' | 15 jours + 1 jour par année d'ancienneté |
| D | le premier chiffre du numéro d'employé est un '4' | 20 jours + 1 jour par année d'ancienneté |

1. Une dernière méthode, appelée **heuresSup()**, permet d'ajouter un nombre d'heures supplémentaires passé en paramètre à la donnée **nbreHeuresSemaine**. Cette méthode ajuste donc le nombre d'heures travaillées et ne retourne rien.

## Validations d’entrées de l’utilisateur dans le formulaire

La validation de champs est une tâche typique lorsqu’une interface utilisateur (GUI) est présente.

On met en place un système de validation simplifié pour les champs du formulaire de gestion de paie.

Tous ces outils seront regroupés dans une classe unique dans un package séparé et seront accessibles sans avoir besoin de créer d’objet, avec la notation *NomDeClasse.NomDeMéthode()*.

### Structure

Package **validation**

* Classe **ValidationFormulaire** Classe d’outils pour valider les valeurs reçues de l’utilisateur

Cette classe d’outils de validation n’a aucun champ.

Il y a une méthode de validation pour chaque champ d’un employé, ces méthodes ne seront disponibles que dans la classe **ValidationFormulaire**.

On a une méthode pour valider tous les champs d’un employe, qui utilise les méthodes précédentes, et qui sera disponible dans d’autres package (utilisée dans la classe **Employe**).

### Tâches

1. Créez le package **validation** et dans celui-ci une classe **ValidationFormulaire**. Toutes les méthodes qui suivent sont ajoutées à cette classe
2. Concevez une méthode **validerNom()** acceptant un nom en paramètre et retournant **vrai** si le nom ne contient que des lettres ou le caractère tiret "-" et **faux** sinon.
3. Concevez une méthode **validerNumero()** acceptant un numéro en paramètre et retournant **vrai** si le nom ne contient que des lettres ou des chiffres et **faux** sinon.
4. Concevez une méthode **validerSalaireHoraire()** acceptant un salaire horaire en paramètre et retournant **vrai** si le salaire est de $15.25 ou plus et **faux** sinon.
5. Concevez une méthode **validerNbreHeuresSemaine()** acceptant un nombre d’heures travaillées dans la semaine retournant **vrai** si le nom nombre d’heures n’est pas négatif et ne dépasse pas 80 et **faux** sinon.
6. Concevez une méthode **validerAnciennete()** acceptant un nombre d’années d’ancienneté et retournant **vrai** si le nombre n’est pas négatif et **faux** sinon.
7. Finalement, concevez une méthode **validerEmploye()** qui accepte 5 paramètres, soit chacun des champs d’un employé, et qui retourne vrai si chacun des champs est valide (utilisez les 5 méthodes précédentes) et faux sinon.

Toutes les méthodes ci-haut doivent être accessible sans qu’on ait besoin de créer un objet.

La classe **validerEmploye()** doit être disponible dans les autres classes des autres package, les 5 autres méthodes doivent n’être disponibles que dans cette classe.

1. Finalement, modifiez la méthode **setEmploye()** de la classe **Employe** pour que les champs soient initialisés seulement si tous les champs sont valides (appelez la méthode validerEmploye().

Si les champs ne sont pas tous valides, ils devraient tous être initialisés à 0 ou chaine vide “”.

# Remise

Remettez le répertoire de projet **C34\_TP2** et tout son contenu ET ce document avec réponses aux questions sous format compressé (**zip**), sur **Léa** dans la section **travaux** sous **Travail Pratique 2**.