

## Exercice 1 : Modélisation des employés et calcul des salaires

### Objectif :

Calculer les salaires des employés d'une entreprise selon leur type de rémunération, tout en utilisant le polymorphisme pour gérer les différentes règles de calcul.

### Pour cela, nous allons tout d'abord

**Créer du package fr.DEMIR.employe**, qui va contenir toutes les classes relatives aux employés et à la gestion de leur salaire.

Ensuite, nous allons créer une classe dans ce package nommé **"Employe"**, contenant le code :

```
package fr.DEMIR.employe;
public abstract class Employe {
    private final String nom;
    public Employe(String nom) {
        this.nom = nom;
    }
    public String getNom() {
        return nom;
    }
    public abstract double getSalaire();
}
```

Puis les classes:

- **EmployeHoraire** : Employés payés à l'heure, avec une augmentation de 30 % pour les heures supplémentaires, au-delà de 39h, CODE:

```
package fr.DEMIR.employe;
public class EmployeHoraire extends Employe {
    private double heuresTravaillées;
    private double tarifHoraire;
    public EmployeHoraire(String nom, double heuresTravaillées, double tarifHoraire) {
        super(nom);
        this.heuresTravaillées = heuresTravaillées;
        this.tarifHoraire = tarifHoraire;
    }
}
```

```

public void setInfosSalaire(double heuresTravaillées, double tarifHoraire) {
    this.heuresTravaillées = heuresTravaillées;
    this.tarifHoraire = tarifHoraire;
}
@Override
public double getSalaire() {
    if (heuresTravaillées > 39) {
        double heuresSup = heuresTravaillées - 39;
        return (39 * tarifHoraire) + (heuresSup * tarifHoraire * 1.30);
    } else {
        return heuresTravaillées * tarifHoraire;
    }
}
}

```

- **EmployeHoraire50** : les heures supplémentaires sont augmentées de 50 %, CODE:

```

package fr.DEMIR.employe;
public class EmployeHoraire50 extends Employe {
    private double heuresTravaillées;
    private double tarifHoraire;
    public EmployeHoraire50(String nom, double heuresTravaillées, double
tarifHoraire) {
        super(nom);
        this.heuresTravaillées = heuresTravaillées;
        this.tarifHoraire = tarifHoraire;
    }
    public void setInfosSalaire(double heuresTravaillées, double tarifHoraire) {
        this.heuresTravaillées = heuresTravaillées;
        this.tarifHoraire = tarifHoraire;
    }
    @Override
    public double getSalaire() {
        if (heuresTravaillées > 39) {
            double heuresSup = heuresTravaillées - 39;
            return (39 * tarifHoraire) + (heuresSup * tarifHoraire * 1.50);
        } else {
            return heuresTravaillées * tarifHoraire;
        }
    }
}
}

```

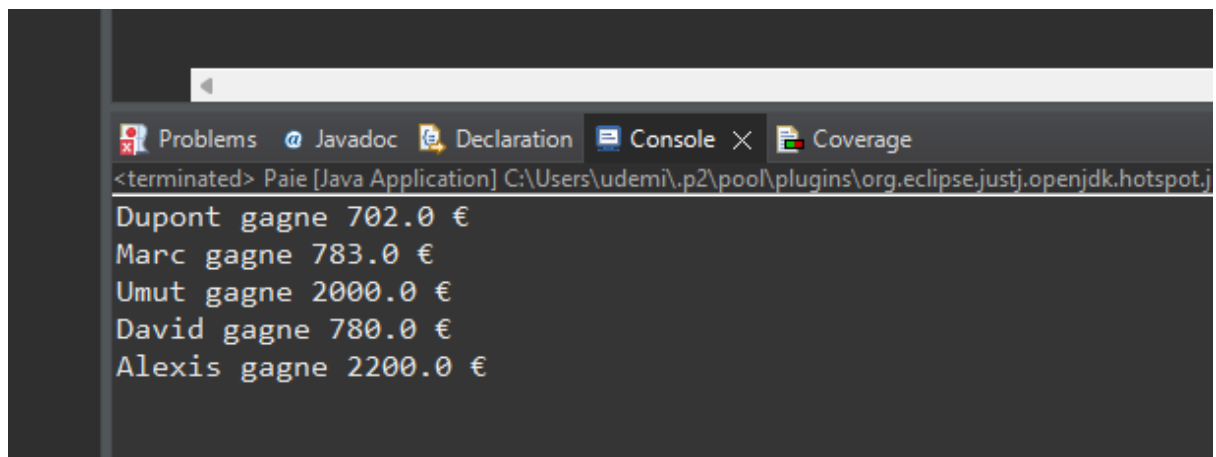
- **Commercial** : Employé avec un salaire fixe plus un pourcentage (1 %) du chiffre d'affaires réalisé, CODE :

```
package fr.DEMIR.employe;
public class Commercial extends Employe {
    private double chiffreAffaire;
    private double salaireFixe;
    public Commercial(String nom, double chiffreAffaire, double salaireFixe) {
        super(nom);
        this.chiffreAffaire = chiffreAffaire;
        this.salaireFixe = salaireFixe;
    }
    public void setInfosSalaire(double chiffreAffaire, double salaireFixe) {
        this.chiffreAffaire = chiffreAffaire;
        this.salaireFixe = salaireFixe;
    }
    @Override
    public double getSalaire() {
        return salaireFixe + (chiffreAffaire * 0.01);
    }
}
```

Pour finir, nous allons créer la classe "Paie" :

```
package fr.DEMIR.employe;
public class Paie {
    public static void main(String[] args) {
        Employee[] employees = new Employee[5];
        employees[0] = new EmployeHoraire("Dupont", 45, 15);
        employees[1] = new EmployeHoraire50("Marc", 42, 18);
        employees[2] = new Commercial("Umut", 50000, 1500);
        employees[3] = new EmployeHoraire("David", 39, 20);
        employees[4] = new Commercial("Alexis", 20000, 2000);
        for (Employee employee : employees) {
            System.out.println(employee.getNom() + " gagne " + employee.getSalaire()
+ " €");
        }
    }
}
```

Voici le résultat de ce code obtenu sur la console:



```
<terminated> Paie [Java Application] C:\Users\udemil.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.j
Dupont gagne 702.0 €
Marc gagne 783.0 €
Umut gagne 2000.0 €
David gagne 780.0 €
Alexis gagne 2200.0 €
```

## Exercice 2 (Création des Exceptions) :

Nous allons créer une classe nommé **“Entreprise”**, contenant le code :

```
package fr.DEMIR.employe;
class SecretMissionException extends Exception {
    private static final long serialVersionUID = 1L;
    public SecretMissionException(String message) {
        super(message);
    }
}
class NonProfitException extends Exception {
    private static final long serialVersionUID = 1L;
    public NonProfitException(String message) {
        super(message);
    }
}
class Entreprise {
    private String nom;
    private String mission;
    private int capital;
    public Entreprise(String nom, String mission, int capital) {
        this.setNom(nom);
        this.mission = mission;
        this.capital = capital;
    }
    public String mission() throws SecretMissionException {
        if (this.mission.equals("Secret")) {
            throw new SecretMissionException("Mission secrète pour l'entreprise " +
getNom());
        }
    }
}
```

```

        return this.mission;
    }
    public int capital() throws NonProfitException {
        if (this.capital <= 0) {
            throw new NonProfitException("Entreprise sans profit : " + getNom());
        }
        return this.capital;
    }

    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
}

```

Puis, nous allons les classes, suivie du code ci-dessous :

- **EntrepriseSecrete :**

```

package fr.DEMIR.employe;
class EntrepriseSecrete extends Entreprise {
    public EntrepriseSecrete(String nom, String mission, int capital) {
        super(nom, mission, capital);
    }
    public String mission() throws SecretMissionException {
        throw new SecretMissionException("Mission secrète pour l'entreprise secrète.");
    }
}

```

- **EntrepriseSansProfit :**

```

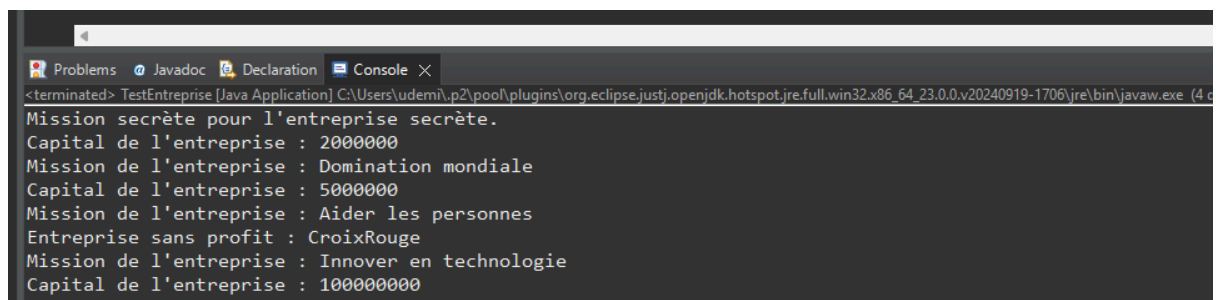
package fr.DEMIR.employe;
class EntrepriseSansProfit extends Entreprise {
    public EntrepriseSansProfit(String nom, String mission, int capital) {
        super(nom, mission, capital);
    }
    @Override
    public int capital() throws NonProfitException {
        throw new NonProfitException("Entreprise sans profit : " + getNom());
    }
}

```

Pour finir, pour tester les exceptions, je les ai mis dans un fichier nommé "TestEntreprise", voici le code :

```
package fr.DEMIR.employe;
public class TestEntreprise {
    public static void afficherEntreprise(Entreprise[] entreprises) {
        for (Entreprise e : entreprises) {
            try {
                System.out.println("Mission de l'entreprise : " + e.mission());
            } catch (SecretMissionException ex) {
                System.out.println(ex.getMessage());
            }
            try {
                System.out.println("Capital de l'entreprise : " + e.capital());
            } catch (NonProfitException ex) {
                System.out.println(ex.getMessage());
            }
        }
    }
    public static void main(String[] args) {
        Entreprise[] entreprises = {
            new Entreprise("Ford", "Construire des voitures", 1000000),
            new EntrepriseSecrete("CIA", "Secret", 2000000),
            new Entreprise("Spectre", "Domination mondiale", 5000000),
            new EntrepriseSansProfit("CroixRouge", "Aider les personnes", 0),
            new Entreprise("Microsoft", "Innover en technologie", 100000000)
        };
        afficherEntreprise(entreprises);
    }
}
```

Voici le résultat obtenu sur la console:



```
<terminated> TestEntreprise [Java Application] C:\Users\udemil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-1706\jre\bin\javaw.exe (4 c
Mission secrète pour l'entreprise secrète.
Capital de l'entreprise : 2000000
Mission de l'entreprise : Domination mondiale
Capital de l'entreprise : 5000000
Mission de l'entreprise : Aider les personnes
Entreprise sans profit : CroixRouge
Mission de l'entreprise : Innover en technologie
Capital de l'entreprise : 100000000
```

### Exercice 3 : Gestion des exceptions dans un cinéma

L'objectif est de lever une exception lorsque la salle est pleine et d'ouvrir une nouvelle salle pour accueillir un spectateur supplémentaire.

**Pour commencer, je vais créer la classe "SallePleineException",** Cette exception est conçue pour être activé lorsque le cinéma tente de placer un spectateur dans une salle pleine. :

```
package fr.DEMIR.employe;

public class SallePleineException extends Exception {
    private static final long serialVersionUID = 1L;
    private Spectateur spectateur;
    public SallePleineException(Spectateur spectateur) {
        super("La salle est pleine pour le spectateur: " + spectateur.getNom());
        this.spectateur = spectateur;
    }
    public Spectateur getSpectateur() {
        return spectateur;
    }
}
```

Puis la classe "Salle", représente une Salle, qui a une capacité maximale, si cette salle est pleine lorsqu'on utilise la méthode "AjouterSpectateur()", l'exception "Salle Pleine Exception" va être activé pour indiquer que la salle est pleine :

```
package fr.DEMIR.employe;
import java.util.ArrayList;
import java.util.List;
public class Salle {
    private int capacite;
    private List<Spectateur> spectateurs;
    public Salle(int capacite) {
        this.capacite = capacite;
        this.spectateurs = new ArrayList<>();
    }
    public void ajouterSpectateur(Spectateur spectateur) throws SallePleineException
    {
        if (spectateurs.size() >= capacite) {
            throw new SallePleineException(spectateur);
        }
        spectateurs.add(spectateur);
    }
    public boolean estPleine() {
        return spectateurs.size() >= capacite;
    }
}
```

Ensuite, nous avons la classe “**Cinema**”, elle contient les listes des salles et gère les ajouts des spectateurs, pour compenser à ce manque de place, nous allons utiliser la méthode “**placer()**” qui va essayer d’ajouter des spectateurs à une salle disponible. Cependant, si cette salle est pleine, une exception “**SallePleineException**” est lancée, et une nouvelle salle est créée pour accueillir les spectateurs, voici le code :

```
package fr.DEMIR.employe;
import java.util.ArrayList;
import java.util.List;
public class Cinema {
    List<Salle> salles;
    public Cinema() {
        this.salles = new ArrayList<>();
    }
    public void placer(Spectateur spectateur) {
        try {
            if (!salles.isEmpty() && !salles.get(salles.size() - 1).estPleine()) {
                salles.get(salles.size() - 1).ajouterSpectateur(spectateur);
                System.out.println(spectateur.getNom() + " a été placé dans la salle " +
salles.size());
            } else {
                Salle nouvelleSalle = new Salle(5);
                nouvelleSalle.ajouterSpectateur(spectateur);
                salles.add(nouvelleSalle);
                System.out.println(spectateur.getNom() + " a été placé dans une nouvelle
salle " + salles.size());
            }
        } catch (SallePleineException e) {
            System.out.println("Erreur : " + e.getMessage());
        }
    }
}
```

Ensuite, nous allons coder l’application permettant de tester le code :

```
package fr.DEMIR.employe;
public class TestCinema {
    public static void main(String[] args) {
        Cinema cinema = new Cinema();
        Spectateur spectateur1 = new Spectateur("Amine");
```



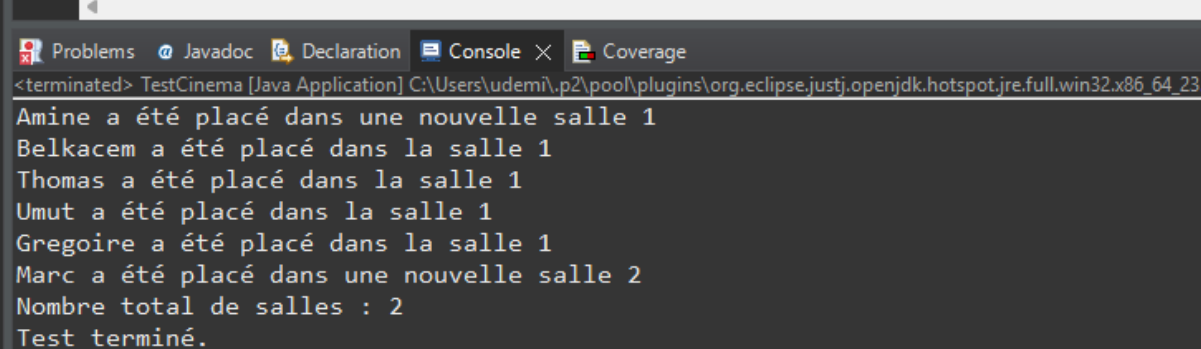
```

Spectateur spectateur2 = new Spectateur("Belkacem");
Spectateur spectateur3 = new Spectateur("Thomas");
Spectateur spectateur4 = new Spectateur("Umut");
Spectateur spectateur5 = new Spectateur("Gregoire");
Spectateur spectateur6 = new Spectateur("Marc");

cinema.placer(spectateur1);
cinema.placer(spectateur2);
cinema.placer(spectateur3);
cinema.placer(spectateur4);
cinema.placer(spectateur5);
cinema.placer(spectateur6);
System.out.println("Nombre total de salles : " + cinema.salles.size());
System.out.println("Test terminé.");
}
}

```

Voici le résultat obtenu sur la console:



```

Problems Javadoc Declaration Console Coverage
<terminated> TestCinema [Java Application] C:\Users\udemil\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.
Amine a été placé dans une nouvelle salle 1
Belkacem a été placé dans la salle 1
Thomas a été placé dans la salle 1
Umut a été placé dans la salle 1
Gregoire a été placé dans la salle 1
Marc a été placé dans une nouvelle salle 2
Nombre total de salles : 2
Test terminé.

```