

**INFORMATIQUE S5**

# **COMPTE RENDU TP-1 JAVA**

# TP

**Encadrer par :  
Prof M.MOUKHAFI**

**Réaliser par :  
Zakaria El Omari**

# 1

# TABLE DE CONTENUE

---

**01**

**Exercice 1: Correction des erreurs dans la classe**

**02**

**Exercice 2: Accesseurs et modificateurs**

**03**

**Exercice 3: Deux classes dans deux fichiers source pour une application**

**04**

**Exercice 4: Deux classes dans un seul fichier source**

**05**

**Exercice 5: Méthode toString()**

**06**

**Exercice 6: Les constructeurs**

**07**

**Exercice: Définition de la classe Individu**

**08**

**Conclusion**

# INTRODUCTION

Le TP abordé dans ce rapport se divise en deux parties distinctes, chacune visant à explorer et à mettre en pratique des concepts fondamentaux de la POO en Java.

La première partie se concentre sur la manipulation des classes et des objets, l'utilisation de méthodes statiques, la définition de constructeurs, et la création de méthodes d'affichage.

La seconde partie s'attache à développer une classe Individu en mettant en œuvre les notions de encapsulation, d'héritage, et de polymorphisme.

L'objectif de ce TP est de nous permettre de consolider nos compréhension des principes de base de la programmation orientée objet en Java, de renforcer nos compétences dans la création et la manipulation de classes, et d'approfondir nos compréhension des mécanismes clés de la POO.

# Exercice 1

## une première "vraie" classe

---

```
public class Livre {
// Variables
    private String titre, auteur;
    private int nbPages;

// Constructeur
    public Livre(String unAuteur, String unTitre) {
        auteur = unAuteur;
        titre = unTitre;
    }

// Accesseur
    public String getAuteur() {
        return auteur;
    }

// Modificateur
    public void setNbPages(int n) {
        if (n > 0) {
            nbPages = n;
        } else {
            System.out.println("Erreur : le nombre de pages doit être positif.");
        }
    }

// Méthode main
    public static void main(String[] args) {
        Livre livre1 = new Livre("Auteur1", "Titre1");
        Livre livre2 = new Livre("Auteur2", "Titre2");

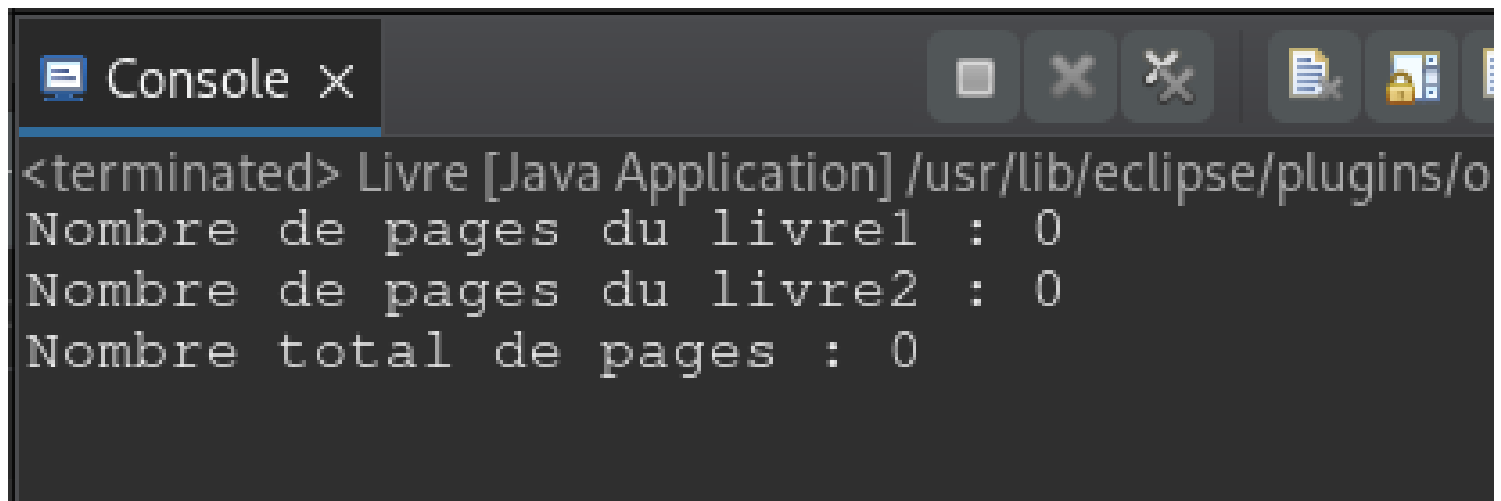
        System.out.println("Nombre de pages du livre1 : " + livre1.nbPages);
        System.out.println("Nombre de pages du livre2 : " + livre2.nbPages);

        System.out.println("Nombre total de pages : " + (livre1.nbPages + livre2.nbPages));
    }
}
```

# Exercice 1

une première "vraie" classe

---



```
<terminated> Livre [Java Application] /usr/lib/eclipse/plugins/o
Nombre de pages du livre1 : 0
Nombre de pages du livre2 : 0
Nombre total de pages : 0
```

# Exercice 2

## accesseurs et modificateurs

---

```
public class Livre {
    // Variables
    private String titre, auteur;
    private int nbPages;

    // Constructeur
    public Livre(String unAuteur, String unTitre) {
        auteur = unAuteur;
        titre = unTitre;
    }

    // Accesseur
    public String getAuteur() {
        return auteur;
    }

    public String getTitre() {
        return titre;
    }

    public int getNbPages() {
        return nbPages;
    }

    // Modificateur
    public void setNbPages(int n) {
        if (n > 0) {
            nbPages = n;
        } else {
            System.out.println("Erreur : le nombre de pages doit être positif.");
        }
    }

    public void setAuteur(String nouvelAuteur) {
        auteur = nouvelAuteur;
    }

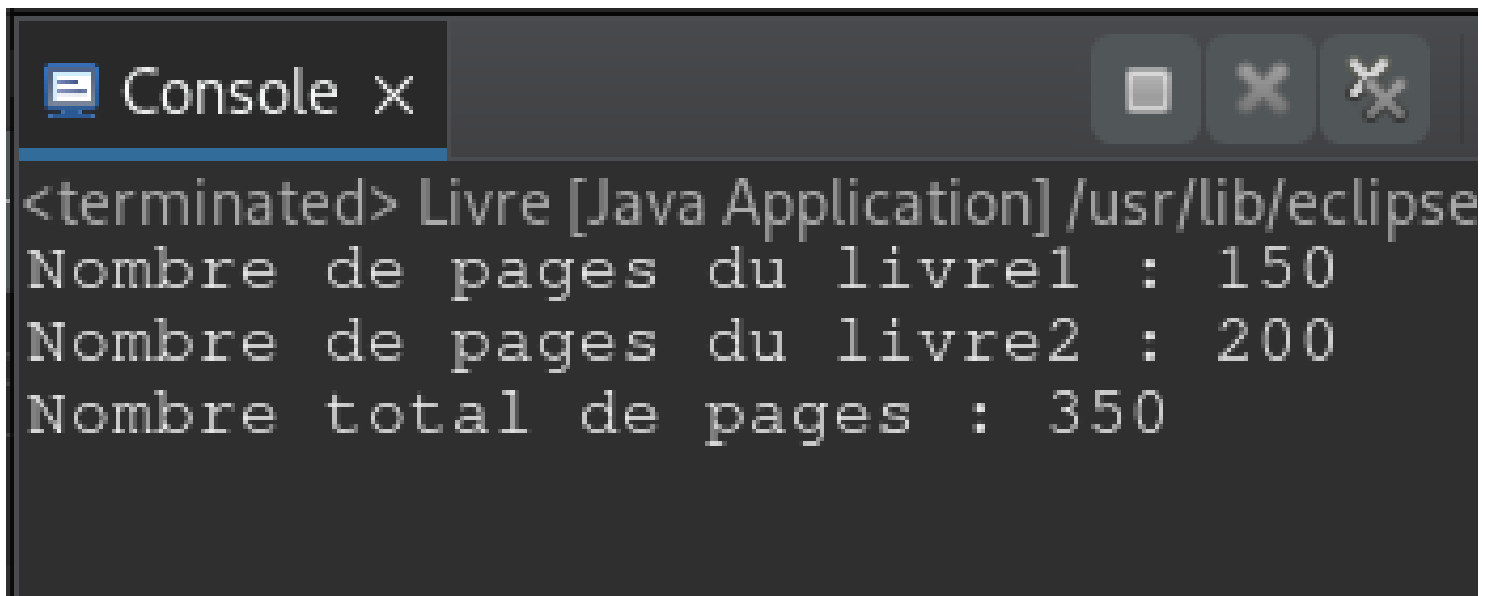
    public void setTitre(String nouveauTitre) {
        titre = nouveauTitre;
    }
}
```

# Exercice 2

## accesseurs et modificateurs Suite du code

// Méthode main

```
public static void main(String[] args) {  
    Livre livre1 = new Livre("Auteur1", "Titre1");  
    Livre livre2 = new Livre("Auteur2", "Titre2");  
  
    livre1.setNbPages(150);  
    livre2.setNbPages(200);  
  
    System.out.println("Nombre de pages du livre1 : " + livre1.getNbPages());  
    System.out.println("Nombre de pages du livre2 : " + livre2.getNbPages());  
  
    System.out.println("Nombre total de pages : " + (livre1.getNbPages() +  
    livre2.getNbPages()));  
}
```



The screenshot shows a console window titled "Console x" with standard window controls (minimize, maximize, close). The output text is as follows:

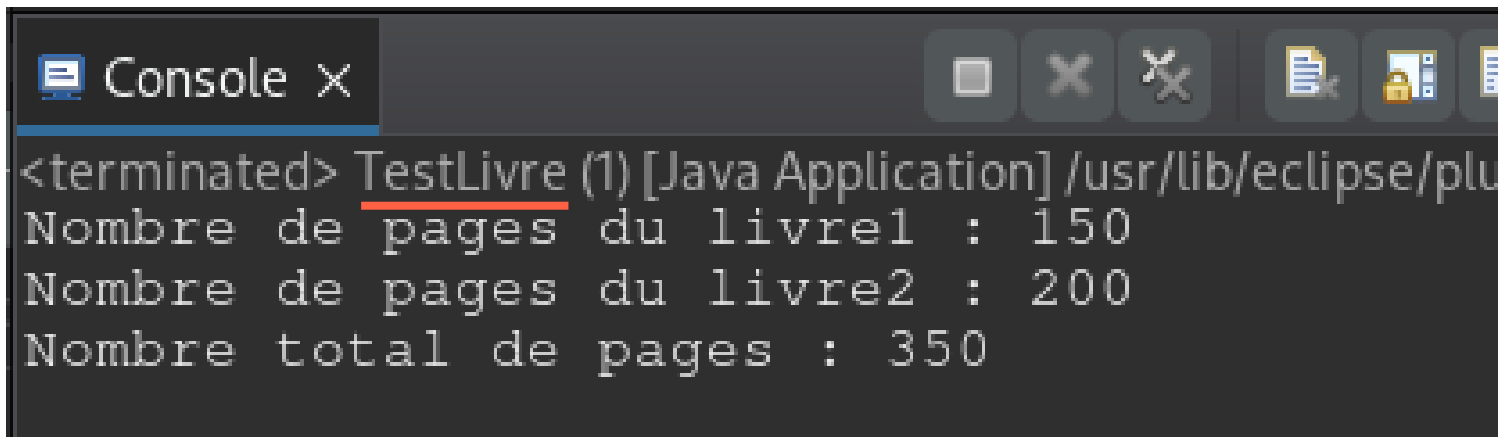
```
<terminated> Livre [Java Application] /usr/lib/eclipse  
Nombre de pages du livre1 : 150  
Nombre de pages du livre2 : 200  
Nombre total de pages : 350
```

# Exercice 3

## Deux Classes dans deux Fichier

**En gardant le même code et en mettant notre fonction main() dans une autre classe TestLivre {...}**

```
public class TestLivre {  
    public static void main(String[] args) {  
        Livre livre1 = new Livre("Auteur1", "Titre1");  
        Livre livre2 = new Livre("Auteur2", "Titre2");  
  
        livre1.setNbPages(150);  
        livre2.setNbPages(200);  
  
        System.out.println("Nombre de pages du livre1 : " + livre1.getNbPages());  
        System.out.println("Nombre de pages du livre2 : " + livre2.getNbPages());  
  
        System.out.println("Nombre total de pages : " + (livre1.getNbPages() +  
        livre2.getNbPages()));  
    }  
}
```



The screenshot shows the Eclipse IDE's console window. The title bar reads "Console x". The output text is as follows:

```
<terminated> TestLivre (1) [Java Application] /usr/lib/eclipse/plu  
Nombre de pages du livre1 : 150  
Nombre de pages du livre2 : 200  
Nombre total de pages : 350
```



# Exercice 4

## Deux Classes dans un Seul Fichier

---

L'exercice 4 offre une exploration détaillée de la manière dont Java gère la compilation et l'exécution lorsque deux classes distinctes sont définies dans un seul fichier source.

Bien que les deux classes aient été définies dans un seul fichier source `Livre.java`, le compilateur Java a créé deux fichiers `".class"` distincts lors de la compilation : `Livre.class` et `TestLivre.class`.

Bien que la bonne pratique soit d'avoir un fichier source `(.java)` par classe, Java offre une certaine latitude dans l'organisation des fichiers. Cependant, il est essentiel de noter que la préservation de la clarté et de la structure du code reste primordiale. L'exercice 4 souligne également l'importance de comprendre comment le compilateur Java traite les fichiers sources pour garantir des résultats d'exécution cohérents.

# Exercice 5

## méthode toString()

---

```
public class Livre {
// Variables
    private String titre, auteur;
    private int nbPages;

// Constructeur
    public Livre(String unAuteur, String unTitre) {
        auteur = unAuteur;
        titre = unTitre;
    }

// Accesseur
    public String getAuteur() {
        return auteur;
    }

    public String getTitre() {
        return titre;
    }

    public int getNbPages() {
        return nbPages;
    }

// Modificateur
    public void setNbPages(int n) {
        if (n > 0) {
            nbPages = n;
        } else {
            System.out.println("Erreur : le nombre de pages doit être positif.");
        }
    }

    public void setAuteur(String nouvelAuteur) {
        auteur = nouvelAuteur;
    }

    public void setTitre(String nouveauTitre) {
        titre = nouveauTitre;
    }
}
```

# Exercice 5

## méthode toString() Suite au code

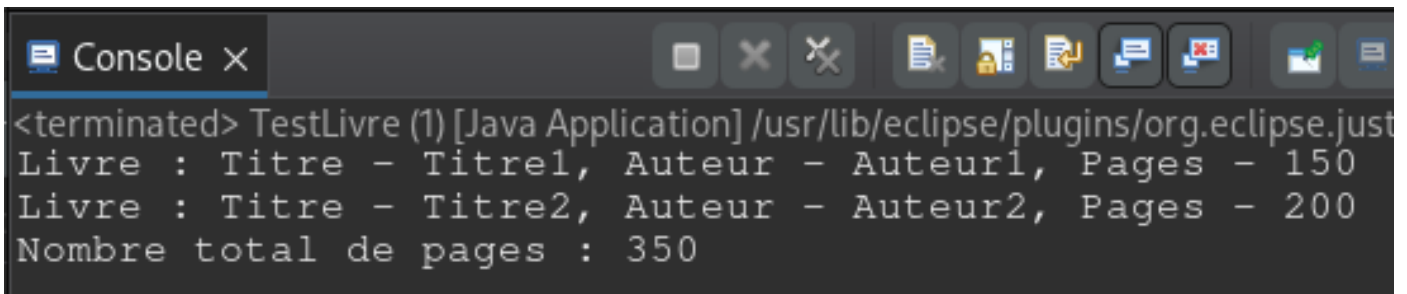
@Override

```
public String toString() {  
    return "Livre : Titre - " + titre + ", Auteur - " + auteur + ", Pages - " + nbPages;  
}
```

```
public void afficheToi() {  
    System.out.println(this.toString());  
}  
}
```

### // Dans la classe TestLivre

```
public class TestLivre {  
    public static void main(String[] args) {  
        Livre livre1 = new Livre("Auteur1", "Titre1");  
        Livre livre2 = new Livre("Auteur2", "Titre2");  
  
        livre1.setNbPages(150);  
        livre2.setNbPages(200);  
  
        livre1.afficheToi();  
        livre2.afficheToi();  
  
        System.out.println("Nombre total de pages : " + (livre1.getNbPages() +  
livre2.getNbPages()));  
    }  
}
```



The screenshot shows the Eclipse IDE's console window. The title bar reads "Console ×". The output text is as follows:

```
<terminated> TestLivre (1) [Java Application] /usr/lib/eclipse/plugins/org.eclipse.just  
Livre : Titre - Titre1, Auteur - Auteur1, Pages - 150  
Livre : Titre - Titre2, Auteur - Auteur2, Pages - 200  
Nombre total de pages : 350
```

# Exercice 6

## les constructeurs

---

```
public class Livre {
// Variables
    private String titre, auteur;
    private int nbPages;

////////////////////////////////////////// Constructeur
    public Livre() {

    }

    public Livre(String unAuteur, String unTitre) {
        auteur = unAuteur;
        titre = unTitre;
    }

    public Livre(String unAuteur, String unTitre, int nombreDePages) {
        auteur = unAuteur;
        titre = unTitre;
        nbPages = nombreDePages;
    }
// Accesseur
    public String getAuteur() {
        return auteur;
    }

    public String getTitre() {
        return titre;
    }

    public int getNbPages() {
        return nbPages;
    }
}
```

# Exercice 6

## les constructeurs Suite au code

---

// Modificateur

```
public void setNbPages(int n) {  
    if (n > 0) {  
        nbPages = n;  
    } else {  
        System.out.println("Erreur : le nombre de pages doit être positif.");  
    }  
}
```

```
public void setAuteur(String nouvelAuteur) {  
    auteur = nouvelAuteur;  
}
```

```
public void setTitre(String nouveauTitre) {  
    titre = nouveauTitre;  
}
```

@Override

```
public String toString() {  
    return "Livre : Titre - " + titre + ", Auteur - " + auteur + ", Pages - " + nbPages;  
}
```

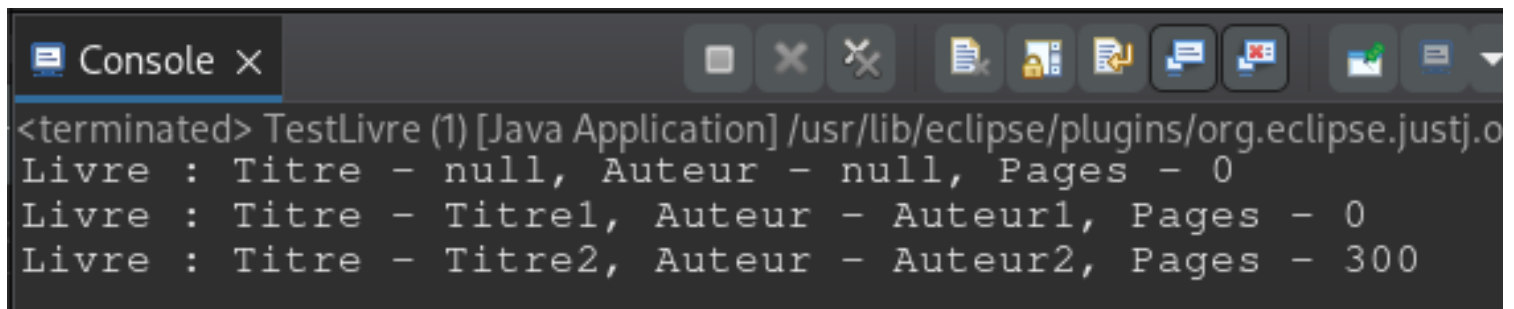
```
public void afficheToi() {  
    System.out.println(this.toString());  
}  
}
```

# Exercice 6

## les constructeurs Suite au code

### // Dans la classe TestLivre

```
public class TestLivre {  
    public static void main(String[] args) {  
        Livre livre1 = new Livre();  
        Livre livre2 = new Livre("Auteur1", "Titre1");  
        Livre livre3 = new Livre("Auteur2", "Titre2", 300);  
  
        // Affichage des informations  
        livre1.afficheToi();  
        livre2.afficheToi();  
        livre3.afficheToi();  
    }  
}
```



The screenshot shows the Eclipse IDE's console window. The title bar reads "Console x". The console output is as follows:

```
<terminated> TestLivre (1) [Java Application] /usr/lib/eclipse/plugins/org.eclipse.justj.o  
Livre : Titre - null, Auteur - null, Pages - 0  
Livre : Titre - Titre1, Auteur - Auteur1, Pages - 0  
Livre : Titre - Titre2, Auteur - Auteur2, Pages - 300
```

# Partie - 2

## Définition de la classe Individu

---

```
public class Individu {
    private String nom, adresse, numeroTelephone;
    private static int nombreDIndividus = 0;
    private static Individu[] tableauIndividus = new Individu[255];

    // Constructeurs
    public Individu(String nom, String adresse, String numeroTelephone) {
        this.nom = nom;
        this.adresse = adresse;
        this.numeroTelephone = numeroTelephone;
        nombreDIndividus++;
        addIndividu(this);
    }

    // Méthode pour ajouter un individu au tableau
    private static void addIndividu(Individu individu) {
        tableauIndividus[nombreDIndividus-1] = individu;
    }

    // Accesseurs
    public String getNom() {
        return nom;
    }

    public String getAdresse() {
        return adresse;
    }

    public String getNumeroTelephone() {
        return numeroTelephone;
    }
}
```

# Partie - 2

## Définition de la classe Individu Suite du code

---

// Modificateurs

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

```
public void setAdresse(String adresse) {  
    this.adresse = adresse;  
}
```

```
public void setNumeroTelephone(String numeroTelephone) {  
    this.numeroTelephone = numeroTelephone;  
}
```

// Méthode toString pour afficher les détails de l'individu

@Override

```
public String toString() {  
    return "Nom: " + nom + ", Adresse: " + adresse + ", Téléphone: " + numeroTelephone;  
}
```

// Méthode d'affichage

```
public void afficher() {  
    System.out.println(this.toString());  
}
```

// Accesseur pour le nombre d'objets créés

```
public static int nombreIndividu() {  
    return nombreDIndividus;  
}
```

// Affichez la liste des individus créés

```
public static void afficherTableauIndividus() {  
    for (int i = 0; i < nombreDIndividus; i++) {  
        tableauIndividus[i].afficher();  
    }  
}
```

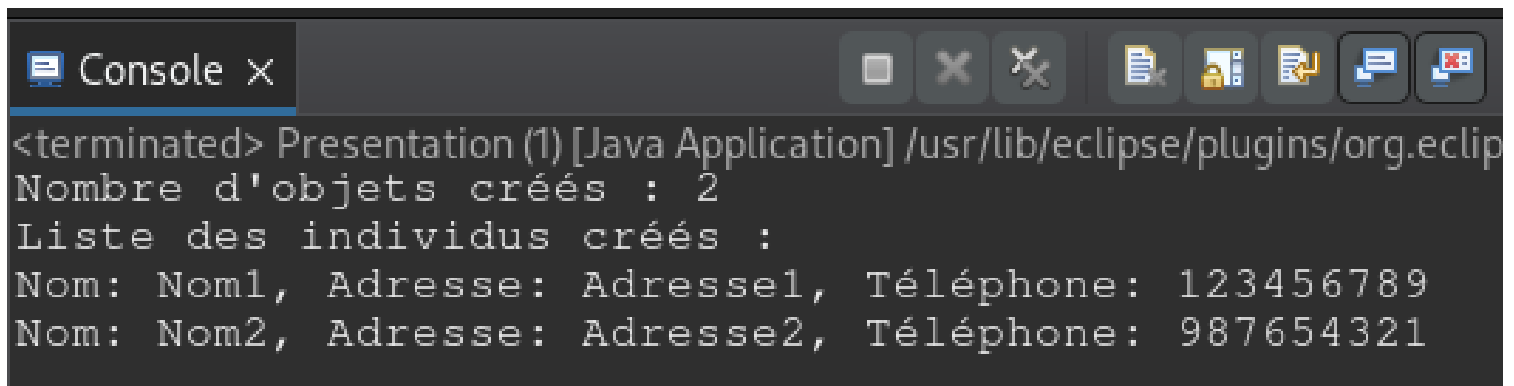


# Partie - 2

## Définition de la classe Individu Suite du code

### // Dans la classe MaClassPresentation

```
public class MaClassPresentation {  
    public static void main(String[] args) {  
        Individu individu1 = new Individu("Nom1", "Adresse1", "123456789");  
        Individu individu2 = new Individu("Nom2", "Adresse2", "987654321");  
  
        System.out.println("Nombre d'objets créés : " + Individu.nombreIndividu());  
        System.out.println("Liste des individus créés :");  
  
        // Affichez la liste des individus créés  
        Individu.afficherTableauIndividus();  
    }  
}
```



The screenshot shows the Eclipse IDE's console window. The title bar reads "Console x". The output text is as follows:

```
<terminated> Presentation (1) [Java Application] /usr/lib/eclipse/plugins/org.eclip  
Nombre d'objets créés : 2  
Liste des individus créés :  
Nom: Nom1, Adresse: Adresse1, Téléphone: 123456789  
Nom: Nom2, Adresse: Adresse2, Téléphone: 987654321
```

# CONCLUSION

---

Ce TP sur la programmation orientée objet en Java a fourni une introduction pratique aux concepts fondamentaux de la POO. En résumé, voici ce que nous avons couvert :

1. **Création d'une classe (Livre) :** Nous avons créé une classe simple représentant un livre avec des variables membres privées, un constructeur, des accesseurs, des modificateurs, et des méthodes pour afficher les informations du livre.
2. **Utilisation de tableaux simples :** Nous avons illustré la gestion d'une liste d'objets à l'aide d'un tableau simple dans la classe Individu. Chaque objet individu est ajouté au tableau lors de sa création.
3. **Méthode toString() :** Nous avons ajouté une méthode toString() à nos classes pour fournir une représentation textuelle significative des objets, améliorant ainsi leur affichage dans la console.
4. **Constructeurs multiples :** La classe Livre a été enrichie avec plusieurs constructeurs pour montrer comment offrir une flexibilité dans la création d'objets en acceptant différents ensembles de paramètres.
5. **Utilisation d'une classe distincte pour la présentation (MaClassPresentation) :** Pour séparer la logique de création d'objets et de manipulation des données, nous avons créé une classe distincte (MaClassPresentation) pour contenir la méthode main et présenter les résultats.
6. **Utilisation de boucles pour l'affichage :** La classe MaClassPresentation utilise une boucle pour afficher la liste des individus, rendant le code plus lisible et concis.