جامعة هواري بومدين
للعلوم والتكنولوجيا
U S T H B

# Corrigé d'exercices de TD POO

## Série n°4 – Héritage et Polymorphisme

**S. BOUKHEDOUMA**

**USTHB – FEI – département d'Informatique**
**Laboratoire des Systèmes Informatiques -LSI**
sboukhedouma@usthb.dz

# Exercice 1 –Série n°4

## Exercice 1

On s'intéresse à l'implémentation d'une entité Employé. Un employé est décrit par un numéro (NSS), un nom, un prénom, une date de naissance et exerce une fonction dans l'entreprise. Les employés de l'entreprise peuvent être de trois catégories « permanent », « stagiaire » ou « contractuel ». Un employé permanent possède une expérience (en nombre d'années), il a un salaire (égal au salaire de base + primes – retenues, les primes et les retenues sont un pourcentage du salaire de base avec des taux définis). Un contractuel possède aussi une expérience, une date de début de contrat et une durée de contrat ; le contrat peut être renouvelé pour une durée donnée (en nombre de mois), un contractuel possède un salaire calculé de la même façon qu'un employé permanent. Un stagiaire possède une date de début de stage, une durée de stage et un présalaire fixe.

On voudrait pouvoir obtenir des informations sur les entités manipulées (ou modifier des données) grâce aux méthodes définies dans les classes.

# Exercice 1 –Série n°4

## Exercice 1

1. Proposez une implémentation de l'entité Employé décrite ci-dessus avec les différents types comportant un constructeur, une méthode de saisie, d'affichage et les accesseurs. (on crée d'abord un type Date avec les méthodes nécessaires qu'on utilisera au besoin).

NB : respecter le principe d'encapsulation.

# Exercice 1 –Série n°4

```
Import java.util.Scanner;
public class Employé {
    private String NSS, nom, prénom;
    private Date dat_naiss;  String fonction;
                                        // constructeurs
public Employé  (String NSS, String nom, String prénom, Date d, String fonction )
{  this.NSS = NSS; this.nom = nom; this.prénom = prénom;
   dat_naiss = new Date (d.jour, d.mois, d.année); this.fonction = fonction; }

public Employé () {}


public void saisir()
{ System.out.print ("introduire la description de l'employé"); }
   Scanner e = new Sacnner (System.in);
   NSS = e.nextline(); nom = e.nextLine(); prénom = e.nextLine();
   dat-naiss.lire();     //on suppose la méthode définie dans la classe Date
   fonction = e.nextLine(); }
```

# Exercice 1 –Série n°4

```
                                    //suite de la classe Employé
public String toString()
            { return  (NSS + "\t" +nom+ "\t" + prénom+ "\t" +
               dat-naiss.toString()+ "\t" + fonction); }

Public void afficher ()
   { System.out.print (this.toString()); }

                                                  // accesseurs
public String getNSS()  { return NSS; }
public String getNom()  { return nom; }
public String getPrénom()  { return prénom; }
public Date getDat_nais()  { return dat_nais; }
public String getFonction()  { return fonction; }
public void setFonction(String fct)  { fonction = fct; }
} //fin de la classe Employé
```

# Exercice 1 –Série n°4

```
Import java.util.Scanner;
public class Permanent extends  Employé {
    private int experience;
    private float salaire_Base ;
    private static float retenue = 0,20;    // pourcentage fixe pour tous les employés
    private static float prime = 0, 16;      // attribut de classe

                                                    // constructeurs
public Permanent (String NSS, String nom, String prénom, Date d, String fonction,
Int experience, float sbase )
{  super (NSS, nom, prénom, d, fonction);
                                                    // appel au constructeur de la superclasse
    this.experience = experience;
     this.salaire_base = sbase; }

public  Permanent () {super(); }
```

# Exercice 1 –Série n°4

```
// suite de la classe Permanent

public void saisir()
{ System.out.print ("Employé permanent");
  Scanner e = new Sacnner (System.in);
  super.saisir();
  experience = e.nextInt();   salaire_base = e.nextFloat();
}
public String toString()  { return (super.toString() + "\t" +  experience + "\t" +
                                    salaire_base );}

public void afficher()
{ System.out.print ("Employé permanent");
  System.out.print  (this.toString());}
```

# Exercice 1 –Série n°4

```
// suite de la classe Permanent

// accesseurs
public int getExperience()  { return experience; }
public void setExperience(int exper)  { experience= exper; }
public float getSalaireBase()  { return salaire_base; }
public void setSalaireBase(float sbase)  { salaire_base= sbase; }

public float salaireNet()
        { return (salaire_base + (prime – retenue)*salaire_base); }
}  // fin class Permanent
```

# Exercice 1 –Série n°4

```
Import java.util.Scanner;
public class Contractuel extends  Employé {
   private int experience;
   private float salaire_Base ;
   private static float retenue = 0,18;    // pourcentage fixe pour tous les contractuels
   private static float prime = 0, 11;     // attribut de classe, peut changer
   private Date debutCont;
   private int durée;
                                                  // constructeur
public Contractuel (String NSS, String nom, String prénom, Date d, String fonction,
Int experience, float sbase, Date deb, int durée )
{  super (NSS, nom, prénom, d, fonction);
                                                  // appel au constructeur de la superclasse
   this.experience = experience;  debutCont = new Date  ( deb.jour, deb.mois, deb.année);
    this.salaire_base = sbase;    this.durée = durée; }

public  Contractuel() {super(); }
```

# Exercice 1 –Série n°4

```
// suite de la classe Contractuel

public void saisir()
{ System.out.print ("Employé contractuel");
  Scanner e = new Sacnner (System.in);
  super.saisir();
  experience = e.nextInt();   salaire_base = e.nextFloat();
  debutContr.lire();
  durée = e.nextInt();
}
public String toString()
   { return (super.toString() + "\t" + experience + "\t" + salaire_base + "\t"
              + debutContr.toString() +  "\t" + durée; );}
```

# Exercice 1 –Série n°4

```
// suite de la classe Contractuel
public void afficher()
{ System.out.print ("Employé contractuel");
  System.out.print  (this.toString());}

// accesseurs
public int getExperience()  { return experience; }
public void setExperience(int exper)  { experience= exper; }
public float getSalaireBase()  { return salaire_base; }
public void setSalaireBase(float sbase)  { salaire_base= sbase; }
public Date getDebutContr()  { return debutCont; }
public void setDebutContr(Date d)
    { debutContr.jour = d.jour; debutContr.mois = d.mois;
      debutContr.année = d.année;}

}
```

# Exercice 1 –Série n°4

```
                    // suite de la classe Contractuel
public int getDurée{ return durée; }
public void setDurée (int d)
     { durée = d;}

public void renouvelerContr (Date deb, int d)
          { this.setDebutContr(deb);
            this.setDurée (d); }

public float salaireNet()
          { return (salaire_base + (prime – retenue)*salaire_base); }

} // fin class Contractuel
```

# Exercice 1 –Série n°4

```
Import java.util.Scanner;

public class Stagiaire extends  Employé {
   private final static float présalaire = 18000;     // constante de classe
   private Date debutStage;
   private int durée;
                                                  // constructeur
public Stagiaire (String NSS, String nom, String prénom, Date d, String fonction,
Date deb, int durée )
{  super (NSS, nom, prénom, d, fonction);
                                                  // appel au constructeur de la superclasse
    debutStage = new Date  ( deb.jour, deb.mois, deb.année);
    this.durée = durée; }

public  Stagiaire () {super(); }
```

# Exercice 1 –Série n°4

```
// suite de la classe Stagiaire

public void saisir()
{ System.out.print ("Employé stagiaire");
  Scanner e = new Scanner (System.in);
  super.saisir();
  debutStage.lire();
  durée = e.nextInt();
}
public String toString()
   { return (super.toString() + "\t" + debutStage.toString() + "\t" + durée +
                "\t" +  présalaire );
   }
```

# Exercice 1 –Série n°4

```
                                    // suite de la classe Stagiaire
public void afficher()
{ System.out.print ("Employé stagiaire");
   System.out.print  (this.toString());}

                                                  // accesseurs
public float getPrésalaire()  { return présalaire; }
public Date getDebutStage()  { return debutStage; }
public int getDurée { return durée; }

}  // fin class Stagiaire
```

# Exercice 1 –Série n°4

**Exercice 1 (suite)**

2. Ecrire un programme qui crée une entité de type employé « contractuel », renouvelle son contrat pour une durée d donnée et affiche la description complète de l'objet.

# Exercice 1 –Série n°4

```
import java.util.Scanner;
public class ProgEmployé
  {  public static void main ( String args[])
     {  Scanner e = new Scanner (System.in);

        // création de l'objet Employé Contractuel
         Contractuel  E = new Contractuel ();     E. saisir();
              // renouveler contrat
        Date d = new Date (07, 05, 2020);
         System.out.println ("donner une durée");
         int durée = e.nextInt();
         E.renouvelerContr (d, durée);
         System.out.println ("voici la description de l'objet E");
         System.out.println (E.toString());
                                  //ou  System.out.println (E);  ou E.afficher();
} }
```

# Exercice 1 –Série n°4

**Exercice 1 (suite)**

3. Créer une structure de type Vector (classe prédéfinie dans java.util) pour stocker les informations d'un nombre n d'employés de différents types (permanent, contractuel, stagiaire). Afficher les noms et prénoms de tous les *stagiaires* ayant une durée de stage égale à 6 mois.

La classe **Vector** permet de créer des vecteurs de tailles variables (dynamique) et offre les méthodes d'ajout (add), suppression (remove), consultation (elementAt, get), etc…

# Exercice 1 –Série n°4

```
import java.util.Scanner;
Import java.util.Vector;

public class ProgEmployé
  {  public static void main ( String args[])
    {  Scanner e = new Scanner (System.in);
       Vector <Employé>  V = new Vector <Employé> ();
       System.out.println ("donner le nombre d'employés");
        int n = e.nextInt();          Employé E;   // référence d'objet

    // création des objets Employé
     for (int i = 0, i< n; i++)
      { System.out.println ("Donner le type de l'employé: Permanent: P,
        Contractuel: C, Satagiaire: S");
        char type = e.next().charAt(0);    // lire le type de l'employé
```

# Exercice 1 –Série n°4

```
switch (type)
{ case 'P': {E = new Permanent (); E.saisir(); break;}
  case 'C': {E = new Contractuel (); E.saisir(); break;}
  case 'S': {E = new Stagiaire (); E.saisir(); break;}
  }
  // ajouter l'élément au vecteur V

V.add (E);   // méthode prédéfinie de la classe Vector


// afficher les informations des stagiaires

System.out .println ("les stagiaires de 6 mois");
for (int i = 0, i< V.size(); i++)
                // la méthode size() donne le nombre d'éléments dans le vecteur
        { E = V.elementAt(i))
        // elementAt(i) méthode prédéfinie de la classe Vector,
                        on peut utiliser get(i) aussi
```

# Exercice 1 –Série n°4

```
System.out .println ("les stagiaires de 6 mois");

for (int i = 0, i< V.size(); i++)
                // la méthode size() donne le nombre d'éléments dans le vecteur

        { E = V.elementAt(i))
          // elementAt(i) méthode prédéfinie de la classe Vector,
                        on peut utiliser get(i) aussi

   if ( E instanceof  Stagiaire && E.getDurée() = = 6)
          System.out .println ( E.getNom() + "\t" + E.getPrénom() );  }

} }
```

# L'opérateur instanceof

instanceof appliqué à une référence d'objet permet de dire si l'objet est du type (nom de classe) spécifié ou non.

***En règle générale, tout objet est instance de sa propre classe et de sa superclasse***

Syntaxe if (référence d'objet **instanceof** nom de classe = =true) …

Par exemple (exemple du cours)

if (C1 **instanceof** Cercle) → true
if (CA1 **instanceof** Forme) → true
if (C1 **instanceof** Forme) → true
if (CA1 **instanceof** Carré) → true
if (CA1 **instanceof** Cercle) → **false**
if (C1 **instanceof** Carré) → **false**