Java : orienté_objet

- Objet : les trois piliers de la programmation orienté objet : encapsulation, héritage, polymorphisme
- Niveau requis :
  avisé
- Commentaires : *Créer une classe objet, contenant des objets*

Introduction : des outils

Outils de lecture et d'écriture

```
public class Lire
{
    // Par défaut, la bibliothèque ne lit que les premiers caractères de
chaque ligne
    private static boolean filtre = true;


    // Filtre :
    //  si leFiltre = vrai, seul le premier caractère de chaque ligne
    //  sera lu par la fonction Lire.c() (appel automatique de la fonction
Purge)
    // sinon tous les caractères sont lus, y compris les caractères de
contrôle
    public static void Filtre ( boolean leFiltre)
    {
       filtre = leFiltre;
    }

    // Purge : élimine tous les caractères jusqu'à la fin de la ligne
    public static void Purge()
    {
        try
        {
            char car ;    // car qui sert de  poubelle
            do
            {
                car = (char) System.in.read ();
            }
            while (car != '\n');
        }
        catch (java.io.IOException e)
        {
            System.out.println ("Erreur de saisie");
            System.exit (0);
        }
```

```
    }

    // Lecture d'une chaine terminée par un "RETURN" : saute la fin de ligne
    public static String Chaine()
    {
        char car ;
        String result = "";

        try
        {
            car = (char) System.in.read ();
            //lecture de la ligne jusqu'au retour charriot (13, 0xD)

            // pour unix/linux et windows
            while (car != '\r' && car != '\n')
            {
                result = result + car;
                car = (char) System.in.read ();
            }

            // Saut du fin de ligne (10, 0xA)
            if ( car != '\n')
            System.in.skip (1);// skip = sauter ;     //"System.in.skip (1)"
=> idem que la fonction "purge"
        }
        catch (java.io.IOException e)
        {
            System.out.println ("Erreur de saisie");
            System.exit (0);
        }
        return result;
    }

//------------------------- OVERLOAD n°1 de Chaine ------------------------
-----
    public static String Chaine(String message)
    {
        char car ;
        String result = "";

        try
        {
            System.out.print(message);
            car = (char) System.in.read ();
            //lecture de la ligne jusqu'au retour charriot (13, 0xD)
            while (car != '\r')
            {
                result = result + car;
                car = (char) System.in.read ();
            }
```

```
            // Saut du fin de ligne (10, 0xA)
            System.in.skip (1);
        }
        catch (java.io.IOException e)
        {
            System.out.println ("Erreur de saisie");
            // System.exit (0);
        }
        return result;
    }

    // Lecture d'un caractère : uniquement le premier caractère de la
nouvelle ligne
    //  si filtrage, n'importe quel caractère sinon
    public static char Caractere ()
    {
        char result = 0;

        try
        {
            result = (char) System.in.read ();
        }

        catch (java.io.IOException e)
        {
            System.out.println ("Erreur de saisie");
            System.exit (0);
        }

        if (filtre)
        {
            Purge ();
        }

        return result;
    }

    public static int Entier ()
    {
        int result = 0;

        try
        {
            result = Integer.parseInt ( Chaine () );
        }

        catch (NumberFormatException e)
        {
            System.out.println ("Format entier incorrect !");
            System.exit(0);
```

```
        }

        return result;
    }

    public static short EntierCourt ()
    {
        short  result = 0;

        try
        {
            result = Short.parseShort ( Chaine () );
        }

        catch (NumberFormatException e)
        {
            System.out.println ("Format entier incorrect !");
            System.exit(0);
        }

        return result;
    }

    public static long EntierLong ()
    {
        long result = 0;

        try
        {
            result = Long.parseLong ( Chaine () );
        }

        catch (NumberFormatException e)
        {
            System.out.println ("Format entier incorrect !");
            System.exit(0);
        }

        return result;
    }

    public static float Reel ()
    {
        float result = 0;

        try
        {
            result = Float.valueOf( Chaine() ).floatValue () ;
        }
```

```
        catch (NumberFormatException e)
        {
            System.out.println ("Format reel incorrect!");
            System.exit(0);
        }

        return result;
    }

    public static double ReelDouble ()
    {
        double result = 0;

        try
        {
            result = Double.valueOf( Chaine() ).doubleValue () ;
        }

        catch (NumberFormatException e)
        {
            System.out.println ("Format reel incorrect!");
            System.exit(0);
        }

        return result;
    }

    // Attente : permet de visualiser les résultats avant la sortie
    //    de l'application.
    public static void Attente()
    {
        System.out.println ();
        System.out.println ("*** Tapez Entree pour Terminer ***");
        Lire.c();
    }

    // Attente : permet de visualiser les résultats avant la suite
    //    de l'application.
    public static void Suite()
    {
        System.out.println ();
        System.out.println ("*** Tapez Entree pour Continuer ***");
        Lire.c();
    }

    public static boolean Question(String msg)
    {
        char reponse ;

        do
        {
```

```
            System.out.print (msg + " (O/N ) ?" );
            reponse = Lire.c();
        }while
((reponse!='O')&&(reponse!='o')&&(reponse!='n')&&(reponse!='N'));
        // arrêt quand reponse est égal à O,o,N,n
        return (reponse == 'O') || (reponse == 'o') ;
    }

// Alias des fonctions


    public static String S ()
    {
        return Chaine();
    }

    public static short s ()
    {
        return EntierCourt();
    }

    public static long l ()
    {
        return EntierLong();
    }

    public static int i ()
    {
        return Entier();
    }

    public static char c ()
    {
        return Caractere();
    }

    public static float f ()
    {
        return Reel ();
    }

    public static double d ()
    {
        return ReelDouble ();
    }


// Fonction qui lit un tableau de char et renvoi ce tableau

       public static char[] remplirTableau(String invite, int motSize)
```

```
       {
           final int MOT_SIZE = motSize ;
           final char TERMINATEUR = '.';


           char mot[] ;
           char actualCar ;
           int count ;
           count = 0 ;

           mot = new char[MOT_SIZE] ;
           System.out.println(invite);

           Lire.Filtre(false);
            do
            {
                    actualCar = Lire.c();
                    mot[count] = actualCar ;
                    count ++ ;

            }
            while(actualCar != '\r' && actualCar != '\n' && actualCar !=
TERMINATEUR && count < mot.length) ;

            if(actualCar != TERMINATEUR)
            {

                mot[count - 1] = TERMINATEUR ;

            }
            if(actualCar != '\n')
            {

                Lire.Purge();

            }
            Lire.Filtre(true);

            return mot;


        /* Fonction qui lit un tableau de char sans terminateur et renvoi ce
tableau */
        public static char[] remplirMot(String invite)
        // invite est la chaine affichée à l'écran avant la saisie
        // le tableau retourné est le mot saisi, et sa longueur est la
longueur du mot.
       {
           char mot[] ;
           char actualCar ;
           int count ;
```

```
            count = 0 ;
            mot = new char[80];
            System.out.println(invite);

            Lire.Filtre(false);
            actualCar = Lire.c();
            while (actualCar != '\r' && actualCar != '\n' && count <
mot.length)
            {
                    mot[count] = actualCar ;
                    count ++ ;
                    actualCar = Lire.c();

             }

             if(actualCar != '\n')
             {

                 Lire.Purge();

             }
             Lire.Filtre(true);

             char [] newMot= new char[count];
             for (int i =0; i<count;i++)
             {
                 newMot[i] = mot[i];
             }
             return newMot;

        }
         public static void afficheMot(char[] mot)
         {
             for (int i = 0; i< mot.length; i++)
             {
                 System.out.print(mot[i]);
             }
         }
         public static void afficheTableau(char[] mot)
         {
             int i;
             for (i = 0; i< mot.length && mot[i] != '.'; i++)
             {
                 System.out.print(mot[i]);
             }
             System.out.print(mot[i]);
         }
}
```

Outils de vérification

```
public class BoiteAOutil
{
    /***
     * fonction booleanne test entree alphanumerique
     * @param entree chaine entree
     * @return vrai
     */
    public static boolean estAlpha(String entree)
    {


            if(entree != null && !entree.trim().isEmpty() &&
entree.matches("^[\\p{L} ]+$"))
                    {
                        return true;
                    }
            else {
                return false;
            }



    }

    public static boolean isInteger(String  entree)
    {


            if(entree != null && !entree.trim().isEmpty() &&
entree.matches("[0-9]"))
                    {
                        return true;
                    }
            else {
                return false;
            }



    }

    /***
     * fonction booleanne test entree numerique avec 3 chiffres
     * @param entree
     * @return vrai
     */
    public static boolean isEntierNumeroRue(String  entree)
    {
```

```
            if(entree != null && !entree.trim().isEmpty() &&
entree.matches("[0-9]{3}"))
                    {
                        return true;
                    }
            else {
                return false;
            }



    }

    /***
     * fonction booleanne test entree numerique avec 5 chiffres pour le code
postale
     * @param entree
     * @return vrai
     */

    public static boolean isEntierCodePostal(String  entree)
    {


            if(entree != null && !entree.trim().isEmpty() &&
entree.matches("[0-9]{5}"))
                    {
                        return true;
                    }
            else {
                return false;
            }



    }


    public static boolean isFloat(String entree)
    {


            if(entree != null && !entree.trim().isEmpty() &&
entree.matches("^([+-]?\\d*\\.?\\d*)$"))
                    {
                        return true;
                    }
            else
```

```
            {
                return false;
            }



    }

}
```

Un premier objet : l'individu

L'individu idéal ou primitif, tel que tout type d'individu instanciable en héritera les propriétés.

```
public abstract class Individu
{
    private  String m_strNom;
    private  String m_strPrenom;
    private  Adresse m_adresse = new Adresse();


    //------------------------------------ CONSTRUCTEUR PAR DEFAUT -------
----------------------------------


    public Individu ()
    {
        this("mon nom est personne", "mon prénom est personne,", new
Adresse());
    }


    //---------------------------------------  CONSTRUCTEUR  --------------
----------------------------------


    /**
     *
     * @param strNom
     * @param strPrenom
     * @param adresse
     */
    public Individu (String strNom, String strPrenom, Adresse adresse)
    {
        // si les valeurs données ne satisfont pas les "set", alors valeur
par défaut :
        if (!setNom( strNom))         setNom("mon nom est personne");
        if (!setPrenom( strPrenom)) setPrenom("mon prénom est personne");
```

```
        if (adresse == null)
        {
            m_adresse = new Adresse();
        }
        else
        {
            m_adresse = adresse;
        }
    }



    //-------------------------------------------------------------
    /**    Affecte (si possible) la chaîne donnée à la propriété m_strNom.
     *
     * @param nomSaisi : Chaîne à affecter (doit être alphabétique, plus :
espace, tiret, apostrophe).
     * @return    VRAI si affectation possible.
     */


    public boolean setNom(String nomSaisi)
    {
        boolean oK = false;

        if (OutilsTests.estAlphaPlus(nomSaisi))
        {
            oK = true;
            m_strNom = nomSaisi.trim();
        }

        return oK;
    }
    //-------------------------------------------------------------
    /** Affecte (si possible) la chaîne donnée à la propriété m_strPrenom.
     *
     * @param prenomSaisi : Chaîne à affecter (doit être alphabétique, plus
: espace, tiret, apostrophe).
     * @return VRAI si affectation possible.
     */
    public boolean setPrenom(String prenomSaisi)
    {
        boolean oK = false;

        if (OutilsTests.estAlphaPlus(prenomSaisi))
        {
            oK = true;
            m_strPrenom = prenomSaisi.trim();
        }
```

```
        return oK;

    }
    //-------------------------------------------------------------
    /** Affecte (si possible) la chaîne donnée à la propriété m_adresse.
     *
     * @param adresseSaisie : Chaîne à affecter (voir contrainte de la class
Adresse).
     * @return VRAI si affectation possible.
     */
    public boolean setAdresse(String adresseSaisie)
    {
        return (m_adresse.setStructAdresse(adresseSaisie));
    }
    //-------------------------------------------------------------
    /**    Récupère la propriété m_strNom.
     *
     * @return     Contenu de m_strNom.
     */
    public String getNom()
    {
        return m_strNom;
    }
    //-------------------------------------------------------------
    /**    Récupère la propriété m_strPrenom.
     *
     * @return     Contenu de m_strPrenom.
     */
    public String getPrenom()
    {
        return m_strPrenom;
    }
    //-------------------------------------------------------------
    /**    Récupère la propriété m_adresse.
     *
     * @return     Contenu de m_adresse.
     */
    public String getAdresse()
    {
        return m_adresse.toString();
    }
    //-------------------------------------------------------------
    /**    Acquisition des propriétés de l'individu.
     *
     */
    public void lire()
    {
        String  saisie, invite;

        // Acquisition du nom :
            invite = "Donnez un nom  : ";
```

```
            do
            {
                saisie = Lire.Chaine(invite);


            } while (!(this.setNom(saisie)));    // Arrêt quand la saisie
est correcte

        // Acquisition du prénom :
            invite = "Donnez un prénom : ";
            do
            {
                saisie = Lire.Chaine(invite);


            } while (!(this.setPrenom(saisie)));    // Arrêt quand la saisie
est correcte


        // Acquisition de l'adresse : remarque ! L'adresse se lit elle-même
tout comme l'individu se lit lui-même
            m_adresse.lire();

    }


    //-----------------------------------------------------------------
    /**    Afficher les propriétés de l'individu.
     *
     */

    public void afficher()
    {
        System.out.println(this.getPrenom() + " " + this.getNom());
        //System.out.println(this.getAdresse());
        m_adresse.affiche();
    }

}
```

La classe Adresse

```
public class Adresse
{
    private int m_numRue;
    private String m_strNomRue;
    private int m_nCodePost;
    private String m_strVille;
```

```
    //-------------------------------------------------------------
        public static final String SEP_ADR = ",";        // Séparateur des
différents champs d'une adresse de type String
    //-------------------------------------------------------------


        //------------------------------- CONSTRUCTEUR   -------------------
--------------

        /**
         *
         * @param strNom
         * @param strPrenom
         * @param adresse
         */
        public Adresse (int numRue, String strNomRue, int nCodePost, String
strVille)
        {
            if(!setNumRue( numRue))  {setNumRue ("pas de nom");};
            if(!setNomRue(strNomRue)) {setNomRue ("pas de prénom");};
            if(!setCP( nCodePost)) {setCP("pas d'adresse");};
            if(!setVille( strVille)) {setVille("pas d'adresse");};

        }


        /*------------------------------ CONSTRUCTEUR PAR DEFAUT -----------
-------------------------------*/

        public Adresse ()
        {
            this(00, "RueInconnue", 99999, "VilleInconnue");
        }


        /* ****************************  GETTER LES PROPRIETES
**************************************** */

    /**    Récupère la propriété m_numRue.
     *
     * @return     Contenu de m_numRue.
     */

    public int getNumRue()
    {
        return m_numRue;
    }

    //-------------------------------------------------------------

    /**    Récupère la propriété m_strNomRue.
```

```
     *
     * @return     Contenu de m_strNomRue.
     */
    public String getNomRue()
    {
        return m_strNomRue;
    }

    //-------------------------------------------------------------

    /**    Récupère la propriété m_nCodePost.
     *
     * @return     Contenu de m_nCodePost.
     */
    public int getCP()
    {
        return m_nCodePost;
    }

    //-------------------------------------------------------------

    /**    Récupère la propriété m_strVille.
     *
     * @return     Contenu de m_strVille.
     */
    public String getVille()
    {
        return m_strVille;
    }


    // ****************************  SETTER LES PROPRIETES
****************************************

    /** Affecte (si possible) la chaîne donnée à la propriété m_strNomRue.
     *
     * @param strSaisi : Chaîne à affecter (doit être alphabétique, plus :
noombres, espace, tiret, apostrophe).
     * @return VRAI si affectation possible.
     */

    public boolean setNomRue(String strSaisi)
    {
        boolean oK = false;

        if (OutilsTests.estAlphaPlus(strSaisi))
        {
            oK = true;
            m_strNomRue = strSaisi.trim();
        }
```

```
        return oK;
    }
    //-------------------------------------------------------------------------
------------------------------

    /** Affecte (si possible) la chaîne donnée à la propriété m_strVille.
     *
     * @param strSaisi : Chaîne à affecter (doit être alphabétique, plus :
espace, tiret, apostrophe, slash).
     * @return VRAI si affectation possible.
     */
    public boolean setVille(String strSaisi)
    {
        boolean oK = false;

        if (OutilsTests.estAlphaPlus2(strSaisi))
        {
            oK = true;
            m_strVille = strSaisi.trim();
        }

        return oK;
    }
    //-------------------------------------------------------------------------
----------------------------

    /** Affecte (si possible) la convertion de la chaîne donnée à la
propriété m_numRue.
     *
     * @param numSaisi : Chaîne à affecter (doit représenter un entier).
     * @return VRAI si affectation possible.
     */
    private boolean setNumRue(String numSaisi)
    {
        boolean oK = false;
        numSaisi = numSaisi.trim();

        if (OutilsTests.estEntier(numSaisi))
        {
            oK = setNumRue(Integer.parseInt(numSaisi));

        }

        return oK;
    }

    //-------------------------------------------------------------------------
----------------------------

    public boolean setNumRue(int numSaisi)
```

```
    {
        boolean oK = false;


        if (numSaisi>0)
        {
            oK = true;
            m_numRue =numSaisi ;
        }

        return oK;
    }

    //-------------------------------------------------------------------------
------------------------------

    /** Affecte (si possible) la convertion de la chaîne donnée à la
propriété m_nCodePost.
     *
     * @param numSaisi : Chaîne à affecter (doit représenter un entier de 5
chiffres).
     * @return VRAI si affectation possible.
     */
    private boolean setCP(String numSaisi)
    {
        boolean oK = false;
        numSaisi = numSaisi.trim();

        if (OutilsTests.estCP(numSaisi))
        {
            oK = true;
            m_nCodePost = Integer.parseInt(numSaisi) ;
        }

        return oK;
    }

    //-------------------------------------------------------------------------
------------------------------

    public boolean setCP(int numSaisi)
    {
        boolean oK = false;

        if(numSaisi>0)
        {
            oK = true;
            m_nCodePost = numSaisi;
        }
```

```
    return oK;
    }

    //-------------------------------------------------------------------------
------------------------------

    /** Découpe et affecte (si possible) la chaîne donnée aux différentes
propriétés de Adresse.
     *
     * @param strSaisie : Chaîne à traiter/affecter selon la structure :
"entier, nom rue, code postal, nom ville".
     * @return VRAI si affectation possible.
     */
    public boolean setStructAdresse(String strSaisie)
    {
        boolean oK = false ;    // A priori, c'est pas bon.

        if (strSaisie != null) {
            // la chaîne saisie doit se composer de 4 parties séparées par
des SEP_ADR :
            String[] strDecoupe = strSaisie.split(SEP_ADR);
            if ((strDecoupe.length == 4) && (this.setNumRue(strDecoupe[0]))
                    && (this.setNomRue(strDecoupe[1]))
                    && (this.setCP(strDecoupe[2]))
                    && (this.setVille(strDecoupe[3]))) {
                oK = true;
            }


        }
        return oK;
    }


    //---------------------- remarque sur l'override du string  -------------
----------------------------
    // Si on ne crée pas d'override, objet.toString() => renvoie le nom de
la classe de l'objet
    // l'override ci-dessous permet d'afficher les propriétés de l'objet


    @Override
    public String toString()
    {
        return  Integer.toString(this.getNumRue())
                + SEP_ADR+ this.getNomRue()
                + SEP_ADR+ Integer.toString(this.getCP())
                + SEP_ADR+ this.getVille();
    }
```

```
    //------------------- permet à l'adresse de se lire elle-même ----------
------------------------------

    public void lire()
    {
        // lire le n° de rue :
            String saisie;
            String invite = "Donnez un numéro de rue  : ";
            do
            {
                saisie = Lire.Chaine(invite);


            } while (!(this.setNumRue(saisie)));    // Arrêt quand la saisie
est correcte


        // lire la rue :
            invite = "Donnez le nom de la rue  : ";
            do
            {
                saisie = Lire.Chaine(invite);


            } while (!(this.setNomRue(saisie)));    // Arrêt quand la saisie
est correcte


        // lire la ville :
            invite = "Donnez la Ville  : ";
            do
            {
                saisie = Lire.Chaine(invite);


            } while (!(this.setVille(saisie)));    // Arrêt quand la saisie
est correcte


        // lire le code postal :
            invite = "Donnez le code postal  : ";
            do
            {
                saisie = Lire.Chaine(invite);


            } while (!(this.setCP(saisie)));    // Arrêt quand la saisie est
correcte
    }
```

```
    // permet d'afficher une adresse :

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

}
```

Une classe tableau d'objets

```
public class Groupe
{

    /***
     * Attribut de la classe groupe MAx,m_nCpt,m_tableauindividu[]
     */
    private static final int MAX= 20;
    private int m_nCpt=0;
    private static int m_nMax;
    private Individu m_tableauindividu[];


    /***
     * constructeur par defaut
     */
    public Groupe()
    {
        this(MAX);
        setMax(MAX);

    }
    /***
     * Constructeur d'initialisation
     * @param n : c'est le nombre d'individu
     */

    public Groupe(int n)
    {
        if (n>0)
        {

        m_tableauindividu = new Individu[n];
        setMax(n);
        }
```

```
        else
        {
            m_tableauindividu = new Individu[MAX];
            setMax(MAX);
        }
    }



    /***
     * fonction get m_nCpt qui retourne le nombre effectif dans le groupe
     * @return m_nCpt le retourne le nombre d'individus effectifs
     */
    public int getNb()
    {
        return m_nCpt;
    }


/***
 * fonction set du m_nCpt met à jour le nombre effectif d'individus
 * @param m_nCpt
 */
    private void setM_nCpt(int m_nCpt)
    {
        this.m_nCpt = m_nCpt;
    }







    /***
     * Fonction de la saisie lire() permet de saisir les donnees de chaque
personne
     */
    public void lire()
    {




        //int i =0;
        int i = getNb();
        if (isfull(i))
        {

        }
        else
```

```
        {

            do
            {


                System.out.println("Tapper votre choix");
                System.out.println();
                System.out.println("Client '0' ... Salairé '1' ...
Commercial  '2' ... Individu '3'");
                char d =Lire.c();


                switch(d)
                {

                    case '0':


                    System.out.println("Saisie d'un client");
                    m_tableauindividu[i] = new Client();
                    m_tableauindividu[i].lire();

                        break;

                    case '1':

                    System.out.println("Saisie d'un salarié");
                    m_tableauindividu[i] = new Employé();
                    m_tableauindividu[i].lire();

                    break;

                    case '2':

                    System.out.println("Saisie d'un Commercial");
                    m_tableauindividu[i] = new Commercial();
                    m_tableauindividu[i].lire();


                    break;

                    case '3':

                    System.out.println("Saisie d'un individu");
                    m_tableauindividu[i] = new Personne();
                    m_tableauindividu[i].lire();


                    break;
```

```
                }

                i = i + 1;



            }while(!isfull(i) && Lire.Question("Voulez enregistrer d'autres
informations ? [o/n] : "));

            this.setM_nCpt(i);

        }
    }




    /***
     * fonction afficher() qui permet d'afficher chaque personne du groupe
     */
    public  void afficher()
    {


        int i =0;

        if(getNb()==0)
        {

            System.out.println("le tableau est vide");
        }
        else
        {
        do
        {
            System.out.println("Individu N° " + i + " : ");
            System.out.println();
            m_tableauindividu[i].afficher();
            System.out.println();
            i = i+1;

        }while(i < getNb());
        System.out.println();
        System.out.println("Nous avons " + getNb()+ " personne(s) de
rentrée(s)");
        }



    }
```

```
/***
 *
 * @return max
 */

    public int getMax()
    {
        return m_nMax;
    }



/***
 * renvoie une reference sur l'individu dans la case visée
 * @param i
 * @return
 */
    public Individu individuAt(int i)
    {
    if( i < 0 && i > getNb())
    {
        return null;
    }
    else {

        return m_tableauindividu[i];
    }



    }
    /***
     * permet de voir si le groupe est rempli
     * @param i
     * @return
     */
   public boolean isfull(int i)
   {

    if (i==getMax())
    {
        System.out.println("le tableau est plein");
        return true;
    }
    else
    {
        return false;
    }
   }
```

```
   private void setMax(int m_nMax)
   {
    Groupe.m_nMax = m_nMax;
   }


}
```

Des classes héritiaires "d'Individu"

Personne : une classe héritaire d'individu

```
public class Personne extends Individu
{

    public Personne()
    {

    }

    public Personne(String nom, String prenom, Adresse adresse)
    {
        super(nom, prenom, adresse);

    }


}
```

Salarié: une classe abstraite et héritiaire d'individu

```
abstract class Salarié extends Individu
{
    private  final float SALAIREDEFAUT =2000;
    private float m_salaire;
    private int m_statut;


    /***
     * Constructeur par defaut
     */
    public Salarié()
    {
        this.setSalaire(SALAIREDEFAUT);
        this.setStatut("1");
    }
```

```
    /***
     * Constructeur d'initialisation
     * @param nom
     * @param prenom
     * @param adresse
     * @param salaire
     * @param statut
     */
    public Salarié(String nom,String prenom,Adresse adresse,float
salaire,int statut)
    {
        super( nom, prenom, adresse);
        if(!setSalaire(salaire))this.setSalaire(SALAIREDEFAUT);
        if(!setStatut(statut))this.setStatut("1");


    }



@Override
    public String toString()
    {

        return super.toString()+ " votre salaire est de : "+  getSalaire()+
" votre satatut est : "+ getStatut();
    }


/***
 * cette methode recupere le salaire
 * @return le salsaire
 */
    public float getSalaire()
    {

        return m_salaire;
    }


    /***
     * cette methode met a jour le salaire l'entree est en float
     * @param m_salaire
     * @return retourne ok
     */
    public boolean setSalaire(float m_salaire)
    {

        boolean ok = false;
```

```
        if(m_salaire>0.0)
                {

                        this.m_salaire = (m_salaire);
                        ok =true;

                }
        else
        {
            ok=false;
        }

    return ok;
    }

/***
 * cette methode (polymorphe) de la methode setSalaire (float)met à jour le
salaire l'entree en string
 * @param m_salaire
 * @return ok
 */
    private boolean setSalaire(String m_salaire)
    {

        boolean ok = false;

        if(BoiteAOutil.isFloat(m_salaire))
                {


                    ok=setSalaire(Float.parseFloat(m_salaire));

                }
        else
        {
            ok=false;
        }

    return ok;
    }


/***
 * cette methode recupere le stat
 * @return le statut
 */
    public int getStatut()
    {
        return m_statut;
```

```
    }



/***
 * cette  methode met à jour le stat
 * @param m_statut
 * @return bollean ok
 */
    public boolean setStatut(int m_statut)
    {
    boolean ok = false;

        if(m_statut> 0)
                {
                    this.m_statut = (m_statut);
                    ok =true;
                }
        else {
            ok=false;
        }

    return ok;

    }



    /***
     * cette methode (polymorphe) de la methode setstat (float)met à jour le
statut l'entree en string
     * @param m_statut
     * @return boolean ok
     */

    private boolean setStatut(String m_statut)
    {
        boolean ok = false;

            if(BoiteAOutil.isInteger(m_statut))
                    {

                        ok = setStatut(Integer.parseInt(m_statut));
                    }
            else
            {
                ok=false;
            }

        return ok;
```

```
    }




    /***
     * Methode lire qui permet de récuperer les donnees saisie par
l'opérateur
     */

    @Override
    public void lire()
    {
        super.lire();
        boolean oksalaire = false;
        boolean okstatut = false;
        System.out.println();

        while (!oksalaire)
        {
        System.out.println("Combien gagnez-vous? ");
        oksalaire = this.setSalaire(Lire.S());
        }
        System.out.println();

        while (!okstatut)
        {
        System.out.println("Quel est votre statut? ");

        okstatut= this.setStatut(Lire.S());

        }
    }

}
```

Employé : une classe dérivée de Salarié

```
public class Employé extends Salarié
{
    public Employé()
    {



    }

    public Employé(String nom, String prenom, Adresse adresse, float
salaire, int statut)
    {
```

```
        super(nom, prenom, adresse, salaire, statut);

    }

}
```

Commercial: Une classe dérivée de Salarié

```
public class Commercial extends Salarié
{
    private float m_chiffreaffaire;
    private float m_ratio;
    private final float CADEFAUT=0;
    private final float RATIO=0.1f;  // f : pour lui dire qu'il s'agit d'un
float ; pareil que convertTo float sur un double

    /***
     * constructeur par defaut
     */
    public Commercial()
    {
        this.setChiffreAffaire(CADEFAUT);
        this.setRatio(RATIO);

    }

    /***
     * constructeur d'initialisation
     * @param nom
     * @param prenom
     * @param adresse
     * @param salaire
     * @param statut
     * @param chiffreaffaire
     * @param ratio
     */
    public Commercial(String nom,String prenom,Adresse adresse,float
salaire,int statut,float chiffreaffaire,float ratio)
    {
        super(nom, prenom, adresse, salaire, statut);
if(!setChiffreAffaire(chiffreaffaire))this.setChiffreAffaire(CADEFAUT);
        if(!setRatio(ratio))this.setRatio(RATIO);

    }




    /***
```

```
     * methode getsalaire du commercial qui en plus rajoutte le chiffre d
affaire et le ratio
     */
    @Override
    public float getSalaire()
    {

        return super.getSalaire()+getChiffreAffaire()*getRatio();

    }

    /***
     * cette methode recupere le chiffre d affaire
     * @return chiffreaffaire
     */

    private float getChiffreAffaire()
    {
        return m_chiffreaffaire;
    }
/***
 * cette methode met à jour le ca l'entree est en float
 * @param m_chiffreaffaire
 * @return ok
 */
    private boolean setChiffreAffaire(float m_chiffreaffaire)
    {

            boolean ok = false;

                if(m_chiffreaffaire>0.0)
                        {

                                this.m_chiffreaffaire = (m_chiffreaffaire);
                                ok =true;

                        }
                else
                {
                    ok=false;
                }

                return ok;

    }
/***
 * cette methode (polymorphe) de la methode setChiffreAffaire (float)met à
jour le ca l'entree en string
 * @param m_chiffreaffaire
 * @return
```

```
 */
    private boolean setChiffreAffaire(String m_chiffreaffaire)
    {

    boolean ok = false;

        if(BoiteAOutil.isFloat(m_chiffreaffaire))
                {


ok=setChiffreAffaire(Float.parseFloat(m_chiffreaffaire));

                }
        else
        {
            ok=false;
        }

    return ok;
    }




    /***
     * cette methode récupère le ratio
     * @return ratio
     */
    private float getRatio()
    {
        return m_ratio;
    }

    /***
     * cette methode met a jour le ratio l'entree est en float
     * @param m_ratio
     * @return ok
     */

    private boolean setRatio(float ratio)
    {
        boolean ok = false;

        if(ratio>=0.02f  && ratio <=0.1f)
                {

                        this.m_ratio = (ratio);
                        ok =true;

                }
        else
```

```
        {
            ok=false;
        }

        return ok;


}
    private boolean setRatio(String m_ratio)
    {
            boolean ok = false;

        if(BoiteAOutil.isFloat(m_ratio))
                {


                    ok=setRatio(Float.parseFloat(m_ratio));

                }
        else
        {
            ok=false;
        }

    return ok;
    }



    /***
     * methode to string de la classe commercial
     */
    @Override
    public String toString()
    {
        return super.toString()+ " votre chiffre d'affaire est de : " +
getChiffreAffaire()+ " votre ratio est de : " + getRatio();
    }
    /***
     * methode lire de la classe commercial
     */
    @Override
    public void lire()
    {
        super.lire();
        boolean okchiffreaffaire = false;
        boolean okratio = false;
        System.out.println();

        while (!okchiffreaffaire)
```

```
        {
        System.out.println("Quel est votre chiffre d'affaire? ");
        okchiffreaffaire = this.setChiffreAffaire(Lire.S());
        }
        System.out.println();

        while (!okratio)
        {
        System.out.println("Quel est votre ratio? ");

        okratio= this.setRatio(Lire.S());

        }
    }



}
```

Programme de test

```
public class Program {

    public static void main(String[] args)
    {
        Groupe monGroupe = new Groupe();    // Groupe des individus saisis

        //  Affichage du nom et de la version du programme
        System.out.println("**** Saisie et affichage d'un tableau (groupe)
d'individus (V1.0, 25/03/2015) ****" );
        System.out.println("");


        // Acquisition + affichage d'un certain nombre d'individus :

        do
        {
            System.out.println("--- Saisie d'au plus "+ monGroupe.getMax()
+" personnes ---" );
            System.out.println("");

            // Acquisition d'un certain nombre d'individus :
            int i = 0;          // indice de parcours du tableau d'individus
            do
            {
                   monGroupe.lire();
                   i = i + 1;
            }
            while (Lire.Question("Voulez-vous saisir un autre individu ?")
```

```
// Arrêt quand  "non"
                    && (i < monGroupe.getMax()))    ;
//          ou si tableau plein

            // Affichage des individus saisis :
            System.out.println("");
                monGroupe.afficher();


        } while (Lire.Question("Voulez-vous saisir un autre groupe
d'individus ?") ) ;  // Arrêt quand "non"


    }

}
```