

```

/*
    Classe qui permet à la fois de créer des objets correspondant à des vecteurs
    tridimensionnels et de faire des calculs usuels avec les vecteurs (par exemple
    la somme, le produit scalaire, le produit vectoriel et le produit mixte)

*/

package cms_tp7;

public class VTD
{
    //champs d'instance privés
    private double composante1, composante2, composante3;
    //champ statique privé
    private static int nbVecteurs=0;

// -----

    //constructeur sans arguments
    VTD()
    {
        composante1=0.0;
        composante2=0.0;
        composante3=0.0;
        nbVecteurs++;
    }

    //constructeur (surchargé) avec trois arguments
    VTD(double compo1, double compo2, double compo3)
    {
        composante1=compo1;
        composante2=compo2;
        composante3=compo3;
        nbVecteurs++;
    }

// -----

```

```
//méthodes d'accès aux champs privés
public void setComposante1(double composante1)
{
    this.composante1=composante1;
}

public void setComposante2(double composante2)
{
    this.composante2=composante2;
}

public void setComposante3(double composante3)
{
    this.composante3=composante3;
}

public double getComposante1()
{
    return composante1;
}

public double getComposante2()
{
    return composante2;
}

public double getComposante3()
{
    return composante3;
}

public static int getNbVecteurs()
{
    return nbVecteurs;
}
```

```
// -----
```

```
//méthode d'instance pour le calcul de la somme
```

```
public VTD calculerSomme(VTD deuxiemeVecteur)
{
    VTD somme = new VTD();
    somme.composante1 = composante1 + deuxiemeVecteur.composante1 ;
    somme.composante2 = composante2 + deuxiemeVecteur.composante2 ;
    somme.composante3 = composante3 + deuxiemeVecteur.composante3 ;
    return somme;
}
```

```
//méthode statique (surchargée) pour le calcul de la somme
```

```
public static VTD calculerSomme(VTD premierVecteur, VTD deuxiemeVecteur)
{
    VTD somme = new VTD();
    somme.composante1 = premierVecteur.composante1 + deuxiemeVecteur.composante1;
    somme.composante2 = premierVecteur.composante2 + deuxiemeVecteur.composante2;
    somme.composante3 = premierVecteur.composante3 + deuxiemeVecteur.composante3;
    return somme;
}
```

```
//
```

```
-----
//méthode d'instance pour le calcul du produit scalaire
```

```
public double calculerProduitScalaire(VTD deuxiemeVecteur)
{
    double produitScalaire;
    produitScalaire = composante1*deuxiemeVecteur.composante1+
                     composante2*deuxiemeVecteur.composante2+
                     composante3*deuxiemeVecteur.composante3;
    return produitScalaire;
}
```

```
//méthode statique (surchargée) pour le calcul du produit scalaire
```

```
public static double calculerProduitScalaire(VTD premierVecteur, VTD deuxiemeVecteur)
{
    double produitScalaire;
    produitScalaire = premierVecteur.composante1*deuxiemeVecteur.composante1+
                     premierVecteur.composante2*deuxiemeVecteur.composante2+
                     premierVecteur.composante3*deuxiemeVecteur.composante3;
    return produitScalaire;
}
```

```
// -----

//méthode d'instance pour le calcul du produit vectoriel
public VTD calculerProduitVectoriel(VTD deuxiemeVecteur)
{
    VTD produitVectoriel = new VTD();
    produitVectoriel.composante1 = composante2*deuxiemeVecteur.composante3-
                                composante3*deuxiemeVecteur.composante2;
    produitVectoriel.composante2 = composante3*deuxiemeVecteur.composante1-
                                composante1*deuxiemeVecteur.composante3;
    produitVectoriel.composante3 = composante1*deuxiemeVecteur.composante2-
                                composante2*deuxiemeVecteur.composante1;

    return produitVectoriel;
}

//méthode statique (surchargée) pour le calcul du produit vectoriel
public static VTD calculerProduitVectoriel(VTD premierVecteur, VTD deuxiemeVecteur)
{
    VTD produitVectoriel = new VTD();
    produitVectoriel.composante1 = premierVecteur.composante2*deuxiemeVecteur.composante3-
                                premierVecteur.composante3*deuxiemeVecteur.composante2;
    produitVectoriel.composante2 = premierVecteur.composante3*deuxiemeVecteur.composante1-
                                premierVecteur.composante1*deuxiemeVecteur.composante3;
    produitVectoriel.composante3 = premierVecteur.composante1*deuxiemeVecteur.composante2-
                                premierVecteur.composante2*deuxiemeVecteur.composante1;

    return produitVectoriel;
}

// -----

//méthode d'instance pour le calcul du produit mixte
//(on utilise l'expression explicite algébrique du produit mixte)
public double calculerProduitMixte(VTD deuxiemeVecteur, VTD troisiemeVecteur)
{
    double produitMixte;
    produitMixte = composante1*(deuxiemeVecteur.composante2*troisiemeVecteur.composante3-
                                deuxiemeVecteur.composante3*troisiemeVecteur.composante2)-
                composante2*(deuxiemeVecteur.composante1*troisiemeVecteur.composante3-
                                deuxiemeVecteur.composante3*troisiemeVecteur.composante1)+
                composante3*(deuxiemeVecteur.composante1*troisiemeVecteur.composante2-
                                deuxiemeVecteur.composante2*troisiemeVecteur.composante1);
}
```

```

        return produitMixte;
    }

    //méthode statique (surchargée) pour le calcul du produit mixte
    //(on calcule le produit mixte à l'aide des produits scalaire et vectoriel)
    public static double calculerProduitMixte(VTD premierVecteur, VTD deuxiemeVecteur,
                                              VTD troisiemeVecteur)
    {
        double produitMixte;
        produitMixte = calculerProduitScalaire(premierVecteur,
                                              calculerProduitVectoriel(deuxiemeVecteur, troisiemeVecteur));

        return produitMixte;
    }

    //-----

    //méthode d'instance ayant un effet similaire à l'opérateur +=
    public void calculerPlusEgal(VTD deuxiemeVecteur)
    {
        composante1=composante1+deuxiemeVecteur.composante1;
        composante2=composante2+deuxiemeVecteur.composante2;
        composante3=composante3+deuxiemeVecteur.composante3;

        //this.faireCopie(this.calculerSomme(deuxiemeVecteur));
        //faireCopie(calculerSomme(deuxiemeVecteur));
        //faireCopie(calculerSomme(this,deuxiemeVecteur));
        //faireCopie(calculerSomme(this,deuxiemeVecteur),this);
    }

    //-----

    //méthode d'instance pour vérifier l'égalité de deux vecteurs
    public boolean verifierEgalite(VTD deuxiemeVecteur)
    {
        if(composante1!=deuxiemeVecteur.composante1)
            return false;
        if(composante2!=deuxiemeVecteur.composante2)
            return false;
        if(composante3!=deuxiemeVecteur.composante3)
            return false;
    }

```

```

        return true;
    }

    //méthode statique (surchargée) pour vérifier l'égalité de deux vecteurs
    public static boolean verifierEgalite(VTD premierVecteur,
                                         VTD deuxiemeVecteur)
    {
        if(premierVecteur.composante1 == deuxiemeVecteur.composante1
            && premierVecteur.composante2 == deuxiemeVecteur.composante2
            && premierVecteur.composante3 == deuxiemeVecteur.composante3)
            return true;
        return false;
    }

    //-----

    //méthode d'instance qui retourne le clone d'un vecteur
    public VTD creerClone()
    {
        VTD vecteurClone=new VTD(composante1,composante2,composante3);
        return vecteurClone;
    }

    //méthode statique (surchargée) qui retourne le clone d'un vecteur
    public static VTD creerClone(VTD vecteurSource)
    {
        VTD vecteurClone=new VTD();

        vecteurClone.composante1=vecteurSource.composante1;
        vecteurClone.composante2=vecteurSource.composante2;
        vecteurClone.composante3=vecteurSource.composante3;
        return vecteurClone;
    }

    //-----

    //méthode d'instance pour le calcul de la norme d'un vecteur
    public double calculerNorme()
    {
        return Math.sqrt(composante1*composante1+composante2*composante2+composante3*composante3);
    }

```

```
//méthode statique (surchargée) pour le calcul de la norme d'un vecteur
public static double calculerNorme(VTD vecteur)
{
    return Math.sqrt(vecteur.composante1*vecteur.composante1
        +vecteur.composante2*vecteur.composante2
        +vecteur.composante3*vecteur.composante3);
}
```

//-----

```
//méthode d'instance pour faire la copie d'un vecteur source vers un vecteur destination
//le vecteur source est le vecteur argument
//le vecteur destination est le vecteur appelant
public void faireCopie(VTD vecteurSource)
{
    composante1=vecteurSource.composante1;
    composante2=vecteurSource.composante2;
    composante3=vecteurSource.composante3;
}
```

```
//méthode statique (surchargée) pour faire la copie d'un vecteur source vers un vecteur destination
//le vecteur source est le premier vecteur argument
//le vecteur destination est le deuxième vecteur appelant
public static void faireCopie(VTD vecteurSource,
                               VTD vecteurDestination)
{
    vecteurDestination.composante1=vecteurSource.composante1;
    vecteurDestination.composante2=vecteurSource.composante2;
    vecteurDestination.composante3=vecteurSource.composante3;
}
```

//-----

```
//méthode statique pour comparer (les normes de) deux vecteurs
public static boolean comparerPlusGrand(VTD premierVecteur, VTD deuxiemeVecteur)
{
    if(calculerNorme(premierVecteur)>calculerNorme(deuxiemeVecteur))
        return true;
    return false;
}
```

```
//-----  
  
    //méthode pour afficher les composantes d'un vecteur  
    public void afficher()  
    {  
        System.out.print("(" + composante1 + ", " + composante2 + ", " + composante3 + ")");  
    }  
} //fin de la classe VTD
```