Exercices #2

On suppose que vous disposez de la classe **`Vaisseau`** précédemment définie et de la classe développée dans le cours.

Question 1 : Définissez une classe **`VolEnFormation`** composée d'un tableau d'objets **`Vaisseau`**, d'une position définie par une abscisse et une ordonnée et une altitude de vol souhaitée. Vous définirez les variables d'instance.

- La responsabilité des **`Vaisseau`** revient à la classe qui instancie le **`VolEnFormation`**

Vous pourrez choisir d'implémenter la position au moyen d'un **`PointPlan`**.

Correction :

```
/**
 * Décrit un Vol en formation composé de plusieurs Vaisseau
 * @version 1.1
 * @author Guillaume Santini
 */

public class VolEnFormation {

     // Variables d'instance
     private Vaisseau [] formation ;
     private PointPlan position ;
     private double altitudeFixée ;
}
```

Question 2 : Définissez des accesseurs publiques aux variables d'instance dont la classe **`VolEnFormation`** à la responsabilité. Il devra en outre être possible d'accéder directement à n'importe quel **`Vaisseau`** du **`VolEnFormation`**.

Correction :

```
     // Accesseur Getter/Setter  -------------------------
     /**
      * Retourne le tableau contenant les Vaisseau de la formation
      * @return le tableau de Vaisseau
      */
     public Vaisseau [] getFormation() {
          return this.formation ;
     }

     /**
      * Fixe le tableau contenant les Vaisseau de la formation
      * @param formation le tableau de Vaisseau
      */
     public void setFormation( Vaisseau [] formation ) {
          this.formation = formation ;
     }
```

```
    /**
     * Retourne le ième Vaisseau de la formation et null si i dépasse
la taille du tableau
     * @param i le numero du Vaisseau
     * @return le Vaisseau numéro i
     */
    public Vaisseau getVaisseaux(int i) {
        if ( i < this.getFormation().length )
            return this.formation[i] ;
        else
            return null;
    }

    /**
     * Fixe le ième Vaisseau de la formation
     * @param i le numero du Vaisseau que l'on souhaite fixer
     * @param newV le nouveau Vaisseau
     */
    public void setVaisseau( int i, Vaisseau newV ) {
        this.formation[i] = newV ;
    }

    /**
     * Retourne l'abscisse de la position de la formation
     * @return l'abscisse de la position de la formation
     */
    public double getAbscisse() {
        return this.position.getAbscisse();
    }

    /**
     * Retourne l'ordonnée de la position de la formation
     * @return l'ordonnée de la position de la formation
     */
    public double getOrdonnée() {
        return this.position.getOrdonnée();
    }

    /**
     * Fixe l'abscisse de la position de la formation
     * @param newX l'abscisse de la position de la formaiton
     */
    public void setAbscisse( double newX) {
        this.position.setAbscisse( newX );
    }

    /**
     * Fixe l'ordonnée de la position de la formation
     * @param newY l'ordonnée de la position de la formaiton
     */
    public void setOrdonnée( double newY) {
        this.position.setOrdonnée( newY );
    }
```

```
**
      * Retourne l'altitude fixée pour la formation
      *  @return l'altitude fixée pour la formation
      */
     public double getAltitudeFixée() {
          return this.altitudeFixée;
     }

     /**
      * Fixe l'altitude de la formation
      * @param newAlt l'altitude fixée
      */
     public void setAltitudeFixée( double newAlt ) {
          this.altitudeFixée = newAlt ;
}
```

Question 3 : Définissez le constructeur champ à champ de la classe **`VolEnFormation`**.

Correction :

```
// Constructeurs -----------------------------------
**
 Constructeur champ à champ
 * @param formation le tableau des vaisseaux de la formation
* @param posX l'abscisse de la position de la formation
* @param posY l'ordonnée de la position de la formation
* @param altitude l'altitude fixée pour la formation
*/
public VolEnFormation( Vaisseau [] formation, double posX, double
posY, double altitude) {
     this.formation = formation ;
     this.position = new PointPlan( posX, posY, "Position") ;
     this.altitudeFixée = altitude ;
}
```

Question 4 : Définissez une méthode publique **`nbDeVaisseaux()`** qui retourne le nombre de constitutifs de la formation.

Correction :

```
/**
 * Retourne le nombre de Vaisseau constituant la formation
 * @return le nombre de Vaisseau de la formation
 **/
public int nbDeVaisseaux() {
     return this.getFormation().length;
}
```

Question 5 : Augmentez chacune des classes **`VolEnFormation`** , **`Vaisseau`** et **`PointPlan`** d'une méthode **`toString()`**. Celles-ci vous serviront désormais lors des tests de vos classes. La méthode **`toString()`** de la classe **`VolEnFormation`** appellera évidement la méthode **`toString()`** des instances de **`Vaisseau`** qui la compose.

Correction :

Pour la classe **`PointPlan`** :

```
     /**
      * Renvoie une chaine formatée décrivant l'état du point
      * @return la chaine décrivant le point
      */
     public String toString()
     {
          return(   this.getNom() +"(x = " +.getAbscisse() +
                    ", y = " + this.getOrdonnée() +")" );
     }
```

Pour la classe **`Vaisseau`**:

```
public String toString()
{
     String desc = "\n\n" ;
     desc += "                    /--------- Vaisseau\n" ;
     desc += "     -------------\n" ;
     desc += "  /  Capacité  = " + this.getNbMaxPassagers() + "
passagers\n" ;
     desc += "<    Altitude  = " + this.getAltitude() + " mètres\n" ;
     desc += "  \\  Catégorie = " + this.getCatégorie() + "\n" ;
     desc += "     -------------\n" ;
     desc += "                    \\--------- Vaisseau\n" ;
     return desc ;
}
```

Pour la classe **`VolEnFormation`**:

```
public String toString() {
     String desc = "";
     desc += "-------------------------------------\n";
     desc += "+- Vol En Formation\n";
     desc += "   |- Altitude : " + this.getAltitudeFixée() + "\n" ;
     desc += "   |- Position : (" + this.getAbscisse() + "," +
this.getOrdonnée() + ")\n" ;
     for (int i = 0; i < this.nbDeVaisseaux(); i++) {
          desc += "   |- Vaisseau " + i + " : " + "\n" ;
          desc += this.getVaisseau(i) + "\n";
     }
     desc += "+- Vol En Formation\n";
     desc += "-------------------------------------\n";
     return desc ;
}
```

Question 6 : Définissez dans la classe **`VolEnFormation`** une méthode privée **`altitudeValide()`**, qui renvoie un booléen : **`True`** si l'altitude fixée est strictement

supérieur à 0m et **`False`** sinon. Cette méthode sera appelée dans le corps de la méthode d'accès en écriture (setter) de la variable définissant l'altitude. Si l'altitude transmise au setter n'est pas valide, alors elle est fixée arbitrairement à une valeur seuil de 1000m.

Correction :

```
/**
 * Méthode publique de test de l'altitude. Permet de
 * savoir si l'altitude de vol prévue pour la formation
 * est au dessus du sol!
 * @return true si l'altitde de la formation est au
 *         dessus de l'altitude 0.
 */
public boolean altitudeValide()
{
     return ( 0. < this.getAltitude());
}

/**
 * Fixe l'altitude de la formation
 * @param newAlt l'altitude fixée
 */
public void setAltitudeFixée( double newAlt ) {
     this.altitudeFixée = newAlt ;
     if ( ! this.altitudeValide() )
          this.setAltitudeFixée( 1000. );
}
```

Question 7 : Définissez dans la classe **`VolEnFormation`** une méthode publique **`appliqueAltitudeDeVol()`** qui lorsqu'elle est appelée fixe l'altitude effective de vol des vaisseaux de la formation à l'altitude souhaitée pour le vol en formation.

Correction :

```
/**
 * Répercute l'altitude de vol (transmet) aux
 * vaisseaux qui composent la formation de vol
 **/
public void appliqueAltitudeDeVol() {
     for( int i = 0 ; i < this.nbDeVaisseaux(); i++)
          this.getVaisseau( i ).setAltitude(this.getAltitudeFixée()) ;
}
```

Question 8 : Définissez dans la classe **`VolEnFormation`** une méthode **`altitudesDesVaisseaux()`**qui retourne la liste des altitudes des vaisseaux qui composent la formation.

Correction :

```
/**
 * Retourne le tableau des altitudes de chaque vaisseau
 * composant la formation
 * @return un tableau d''altitudes
```

```
 */
public double [] altitudesDesVaisseaux() {
     double [] tab = new double [ this.nbDeVaisseaux() ] ;
     for( int i = 0 ; i < tab.length ; i++ )
          tab[i] = this.getVaisseau(i).getAltitude();
     return tab ;
}
```

Question 9 : Définissez une classe **`TestVolEnFormation`** dans laquelle vous définirez complètement une formation en vol composées de 3 **`Vaisseau`** ayant des altitudes de vol différentes. Vous fixerez ensuite l'altitude souhaitée de vol pour la formation à 167000m au moyen de l'accesseur approprié. Le programme récuperera les altitudes des **`Vaisseau`** au moyen de la méthode **`altitudesDesVaisseaux`** et les comparera à l'altitude fixée pour la formation. Si l'une des altitudes des **`Vaisseau`** est différentes de l'altitude fixée pour le vol en formation alors cette dernière sera transmise et appliquée à chaque **`Vaisseau`**.
Au moyen d'affichages, vous procéderez à tous les contrôles nécessaires pour vérifier le bon comportement de votre programme. Notamment, vous modifierez la catégorie d'un des vaisseaux depuis le programme principal et vous vérifierez qu'elle est aussi modifiée dans le vol en formation.

Correction :

```
public class TestVolEnFormation
{
     public static void main(String [] args)
     {
          Vaisseau [] vaisseaux = new Vaisseau [3] ;
          vaisseaux[0] = new Vaisseau("Chasseur Léger", 2,10000);
          vaisseaux[1] = new Vaisseau("Vaisseau Lourd", 2, 10) ;
          vaisseaux[1] = new Vaisseau("Vaisseau Lourd", 46785,
1800000) ;
          VolEnFormation liberty = new VolEnFormation( vaisseaux, 10,
13, 0.);
          System.out.println( liberty);
          liberty.setAltitudeFixée( 167000.);
          liberty.appliqueAltitudeDeVol();
          System.out.println( liberty);
     }
}
```

Question 10 : Définissez les constructeurs par copie des classes **`Vaisseau`** et **`VolEnFormation`**.

Correction :

Pour la classe **`Vaisseau`**:

```
/**
 * Constreur par copie
 * @param v le modèle de Vaisseau
 */
```

```
    public Vaisseau( Vaisseau v ) {
        this.catégorie = v.catégorie
        this.nbMaxPassagers = v.nbMaxPassagers ;
        this.altitude = v.altitude ;
    }
```

Pour la classe **VolEnFormation**:

```
/**
 * Constructeur par copie
 * @param vol le modèle de VolEnFormation
 */
public VolEnFormation( VolEnFormation vol) {
    this.formation = new Vaisseau [ vol.formation.length ];
    for( int i = 0 ; i < this.formation.length ; i++ )
        this.formation[i] = new Vaisseau( vol.formation[i] ) ;
    this.position = new PointPlan( vol.position ) ;
    this.altitudeFixée = vol.altitudeFixée ;
}
```

Question 11 : Dans la classe **TestVolEnFormation** proposez un jeu d'instruction permettant de tester les constructeurs par copie

Correction :

```
VolEnFormation phantom = new VolEnFormation( liberty ) ;
phantom.setVaisseau( 0 , new Vaisseau( phantom.getVaisseau( 0 ) ) ) ;
phantom.setAltitudeFixée( 0.) ;
phantom.appliqueAltitudeDeVol();

System.out.println( " Comparaison des vols en formation (test des
constructeurs par copiei\n" ) ;
System.out.println( "Vol en formation liberty: \n\n" + liberty );
System.out.println( "Vol en formation phantom: \n\n" + phantom );
```

Exercices complémentaires

Question 12 : Dessinez les relations entre les objets en présence et leur évolution.

Dans la classe **VolEnFormation** :

Question 13 : Définissez une méthode **déplacement()** qui permet de modifier la position du vol en formation en passant en paramètre de combien il faut modifier l'abscisse et l'ordonnée de la position.

Question 14 : Définissez une méthode **destruction()** qui prend en paramètre un entier. Si l'entier saisie est 1 (resp. 2), le premier (resp. second) **Vaisseau** est détruit (la référence dans le vol en formation doit être la référence ***null***).

Question 15 : Définissez la Javadoc de toutes ces classes