Exercices #1

Soit la classe suivante :

```
// définition d'une classe pour un vaisseau
public class Vaisseau {
     private int nbMaxPassagers;   // Equipage + passager
     private String catégorie;     // Classe du vaisseau
     public double altitude;       // altitude effective de vol

     // Constructeurs -------------------------------------
     /**
      * Constructeur champ à champ
      */
     public Vaisseau( String cat, int nbPass, double alt) {
          this.setNbMaxPassagers( nbPass) ;
          this.setCatégorie( cat) ;
          this.altitude = alt ;
     }

     // Accesseur Getter/Setter  -------------------------
     /**
      * Fixe la catégorie du vaisseau
      * @param cat nom de la catégorie
      */
     public void setCatégorie( String cat){
          this.catégorie = cat;
     }

     /**
      * Retourne la catégorie du vaisseau
      * @return la catégorie du vaisseau
     */
     public String getCatégorie(){
          return this.catégorie;
     }

     /**
      * Fixe la capacité en nombre de passagers du vaisseau
      * @param n nombre maximal de passagers
      */
     public void setNbMaxPassagers( int n){
          this. nbMaxPassagers = n;
     }

     /**
      * Retourne la capacité en nombre de passagers du vaisseau
      * @return nombre maximal de passagers
     */
     public int getNbMaxPassagers(){
          return this.nbMaxPassagers;
     }
}
```

Question 1 : Identifiez et donnez la liste des variables d'instance, des constructeurs et des méthodes d'accès en lecture (getter) et en écriture (setter) de cette classe.

Les variables d'instance :
nbMaxPassagers, catégorie, altitude
Les constucteurs :
Ici il y en a un seul : **Vaisseau(String cat, int nbPass, double alt)**
Getter :
getNbMaxPassagers(),getCatégorie(),getAltitude()
Setter :
setNbMaxPassagers(int n), setCatégorie(String cat), setAltitude(double alt)

Question 2 : Donnez la vue publique et la vue privée de cette classe ?

Vue publique :
altitude, setNbMaxPassagers(int n), setCatégorie(String cat), setAltitude(double alt),
getNbMaxPassagers(),getCatégorie(),getAltitude()
,Vaisseau(String cat, int nbPass, double alt)
Privée:
Les 2 autres variables d'instance

Question 3 : Ajoutez le constructeur par défaut qui initialise un vaisseau comme étant de la catégorie « Vaisseau Léger », admettant un maximum de 7 personnes à son bord et positionné au sol (altitude 0).

```
public Vaisseau()
{
     this( "Vaisseau léger", 7, 0.f);
}
```

On ajoute la méthode suivante à la classe **Vaisseau** :

```
1    public static void main(String [] args)
2    {
3         Vaisseau xwingT65 ;
4         xwingT65 = new Vaisseau("Chasseur Léger", 2, 10000) ;
5         Vaisseau stalker;
6         stalker = new Vaisseau("Vaisseau Lourd", 46785, 1800000) ;
7         Vaisseau surprise = xwingT65 ;
8
9         System.out.println(stalker.getAltitude()) ;
10        System.out.println(stalker.nbMaxPassagers) ;
11        System.out.println(surpise.getAltitude ()) ;
12
13        xwingT65.setAltitude(10) ;
14        xwingT65.nbMaxPassagers = 1 ;
15        System.out.println(xwingT65.getNbMaxPassagers()) ;
```

```
16          xwingT65= stalker ;
17    }
```

Question 4 : Dessinez les classes et les instances des instructions des lignes 3 à 7.

Question 5 : Quels sont les affichages produits par ce programme ?

```
Ligne  9 : 1800000
Ligne 10 : 10000
Ligne 11 : 2
Ligne 15 : 1
```

Question 6 : Redessinez les instances à la fin de l'exécution du programme.

Question 7 : Cette classe respecte-t-elle le principe d'encapsulation ? Si ce n'est pas le cas proposez les modifications nécessaires.

Non, la classe ne respecte pas le principe d'encapsulation puisque la variable d'instance est déclarée comme **public**
Pour respecter le principe d'encapsulation il faut interdire l'accès direct à la variable d'instance **altitude**

```
    private double altitude;            // altitude effective de vol
```

et ajouter des accesseurs **public** *sur cette variable :*

```
    /**
     * Fixe l'altitude de vol du vaisseau
     * @param alt altitude de vol
     */
    public void setAltitude( double alt){
        this.altitude = alt;
    }

    /**
     * Retourne le nombre de passagers du vaisseau
     * @return l'altitude du vaisseau
     */
    public double getAltitude(){
        return this.altitude;
    }
```

et modifier le constructeur en conséquence :

```
    public Vaisseau( String cat, int nbPass, double alt) {
        this.setNbMaxPassagers( nbPass) ;
        this.setCatégorie( cat) ;
        this.setAltitude( alt) ;      // utilisation de la méthode
    }
```

Placez la méthode **`main`** précédemment définie dans une classe **`TestVaisseau`** comme présenté ci-dessous :

```
1 // définition d'une classe de Test pour un vaisseau
2 public class TestVaisseau
3 {
4    public static void main(String [] args)
5    {
6         [...]

20    }
21 }
```

Question 8 : Identifiez les instructions rejetées par le compilateur, expliquez pourquoi certaines d'entre elles fonctionnaient lorsque la méthode **`main`** était dans la classe **`Vaisseau`**.

Ligne 13 et 17 : Il faut faire appel aux méthodes **getNbMaxPassagers()** *et* **setNbMaxPassagers()** .
Le respect de l'encapsulation rend
private nbMaxPassagers.
nbMaxPassagers *est donc inaccessible depuis une autre classe mais le reste dans la classe de définition* **Vaisseau**

Question 9 : Pourquoi est-il important de confier le test d'une classe à une autre classe (une classe de test dédiée) ?

Pour se rendre compte du caractère privé de certaines (toutes) variables et méthode (ex. **`xwing.nbMaxPassagers`***). Si on reste dans la classe de développement, alors on a accès sans restriction y compris aux variables et méthodes privées.*

Question 10 : Proposez une adaptation valide du code de la méthode **`main`** de la classe **`TestVaisseau`**.

```
1 // définition d'une classe de Test pour un vaisseau
2 public class TestVaisseau
3 {
4    public static void main(String [] args)
5    {
6         Vaisseau xwingT65 ;
7         xwingT65 = new Vaisseau("Chasseur Léger", 2, 10000) ;
8         Vaisseau stalker;
9         stalker = new Vaisseau("Vaisseau Lourd", 46785, 1800000) ;
10        Vaisseau surprise = xwingT65 ;
11
12        System.out.println(stalker.getAltitude()) ;
```

```
13          System.out.println(stalker.getNbMaxPassagers()) ; // SETTER
14          System.out.println(surprise.getAltitude()) ;
15
16          xwingT65.setAltitude(10) ;
17          xwingT65.setNbMaxPassagers(1) ;                    // GETTER
18          System.out.println(xwingT65.getNbMaxPassagers()) ;
19          xwingT65= stalker ;
20      }
21  }
```

Exercices complémentaires

Question 11 : Définissez une classe **`Droid`** qui sera caractérisée par un modèle et une taille. Cette classe proposera un constructeur par défaut et un constructeur champ à champ et respectera les règles d'encapsulation.
Question 12 : Définissez une classe **`TestDroid`** proposant un ensemble d'instructions permettant de vérifier l'encapsulation de tester la vue publique de la classe. Vous pourrez par exemple instancier un de modèle « série R2 » de 96cm de haut et un autre de modèle « 3PO »