Exercices #4

Ces exercices s'appuient sur les classes **`Vaisseau`** et **`VolEnFormation`**. Vous pouvez télécharger ces classes disponibles dans l'archive seance4_debut.tgz à l'adresse suivante :

https://www.lipn.univ-paris13.fr/~santini/RN3/

Question 0 :
Afin de vous aider dans votre développement, il vous est conseillé de compiler la JavaDoc. Ainsi, vous noterez entre autres choses l'apparition d'une méthode **`initRandom()`** dans la classe **`Vaisseau`**.

Question 1 :
Dans une classe de test, créez un vol en formation de 10 Vaisseaux aléatoires, puis affichez la formation.

Correction :

```
public class TestVolEnFormation
{
     public static void main( String [] args)
     {
          // déclaration du tableau et instanciation
          Vaisseau [] groupe = new Vaisseau [5] ;

          // Instanciation des Vaisseaux et initialisation
          // aléatoire
          for ( int i = 0; i < groupe.length; i++)
          {
               groupe[i] = new Vaisseau();
               groupe[i].initRandom() ;
          }

          // déclaration de la formation
          VolEnFormation formation = new VolEnFormation(groupe, new
PointPlan( 10., 10.), 12000.);

          // Affichage
          System.out.println(formation);
     }
}
```

Question 2 :
Toujours dans la même classe, créez un autre vaisseau aléatoire et insérez le dans la formation à une position déterminée par un indice saisi au clavier par un utilisateur. Que se passe-t-il si vous saisissez 10 au clavier. Expliquez pourquoi ?

Correction :

```
{
        // Lecture de l'indice de position depuis le clavier
        System.out.println( "Saisir la position du nouveau venu");
        Scanner sc = new Scanner(System.in);
        int indice = sc.nextInt();

        // insertion dans la formation en vol
        Vaisseau enPlus = new Vaisseau();
        enPlus.initRandom() ;
        formation.setVaisseau( indice, enPlus);

        // Affichage
        System.out.println(formation);
}
```

Si on saisi 10 un message s'affiche indiquant que cet indice de position n'est pas correct. Par ailleurs aucun Vaisseau n'est inséré.

```
Position d'insertion dans le tableau invalide
```

Plutôt que de faire appel à des messages d'erreur, la bonne façon de gérer ce type de situation en Java est d'utiliser le système de levée d''exception.

Question 3 :

Définir une classe **`FormationInsertionException`** héritant de la classe **`Exception`** prédéfinie en Java (voir la documentation de l'API Java). En plus de la variable d'instance de type `String` héritée d'**`Exception`**, ajouter deux attributs **`rangSouhait`** et **`rangMax`** de type **`int`**:

- créer un constructeur membre à membre en conséquence
- redéfinir la méthode **`toString()`** pour qu'elle retourne la chaîne de caractères prévue par **`Exception`**, à laquelle on ajoute par exemple **`" : 10>5"`**. Dans cet exemple, on a souhaité ajouter un **`Vaisseau`** directement au **`rangSouhait`** 10 alors qu'il n'y a que 5 **`Vaisseau`** pour l'instant et qu'en conséquence, on ne peut ajouter directement de **`Vaisseau`** au delà du **`rangMax`** 4.

Correction :

```
public class FormationInsertionException extends Exception
{

     private final int rangSouhait;
     private final int rangMax;

     /**
      * Exception levée lorsqu'on cherche à insérer un Vaisseau dans
      * une formation à un indice dépassant la taille de la formation
      * @param message une message informatif
      * @param posS la position d'insertion souhaitée
```

```
      * @param posM le position d'insertion la plus grande possible
dans la formation
      */
     public FormationInsertionException( String message, int posS, int
posM)
     {
          super( message) ;
          this.rangSouhait = posS;
          this.rangMax = posM;
     }

     /**
      * Redéfinition du format d'affichage de l'exception pour tenir
      * compte des paramètre additionnels de position possible et
      * poisition souhaitée ayant produit la levée d'exception
      * @return le message
      */
     public String toString()
     {
          return super.toString() + ": " + this.rangSouhait + " > " +
this.rangMax;
     }
}
```

Question 4 :
Modifiez la méthode **`setVaisseau(...)`** de la classe **`VolEnFormation`** de manière à ce qu'elle lève une exception **`FormationInsertionException`** en cas d'erreur, au lieu de simplement procéder à un **`println().`**

Correction :

```
public void setVaisseau( Vaisseau v, int i) throws
FormationInsertionException
{
     if ( i >= this.TAILLE )
          throw new FormationInsertionException("Insertion dans la
formation impossible",i, this.TAILLE);
     this.formation[i] = v ;
}
```

Question 5 :
Ajoutez **`throws Exception`** à la signature de la méthode **`main()`** du programme de Test. De cette manière, on peut se passer de gérer n'importe quelle exception dans la méthode **`main()`**, en déléguant leur gestion à la machine virtuelle java elle-même. Lancez le programme de test pour constater la manière dont la machine virtuelle gère une exception explicite jamais traitée. *ATTENTION*, procéder ainsi, c'est *MAL* : cette question est uniquement posée pour montrer le fonctionnement des exceptions.

Correction :
Exemple d'affichage

```
Saisir la position du nouveau venu
10
Exception in thread "main" FormationInsertionException: Insertion dans
la formation impossible: 10 > 5
     at VolEnFormation.setVaisseau(VolEnFormation.java:89)
     at TestVolEnFormation.main(TestVolEnFormation.java:32)
```

Question 6 :
Reproduisez exactement le même affichage en supprimant le **`throws Exception`** de la classe de test mais en ajoutant un **`try-catch`** autour de l'insertion et de l'affichage immédiatement après. Pour que la pile d'exécution soit affichée, utilisez dans le bloc **`catch`** la méthode **`printStackTrace()`** définie pour toutes les exceptions.

Correction :

```
try
{
     formation.setVaisseau( indice, enPlus);

     // Affichage
     System.out.println(formation);
}
catch (ErreurInsertion e)
{
     e.printStackTrace(););
}
```

En plus de permettre de constater des erreurs, les exceptions offrent un mécanisme pour pouvoir les récupérer et les corriger sans empêcher le programme de continuer à fonctionner.

Question 7 :
Modifiez votre programme pour que si le rang d'insertion saisi est trop grand ce dernier soit alors redemandé à l'utilisateur jusqu'à ce que l'indice soit valide (inférieur à la taille du tableau). Ainsi, l'erreur est récupérée et le **`Vaisseau`** finir toujours par être ajouté à une position valide. Vous pouvez d'ailleurs sortir l'affichage du **`VolEnFormation`** du **`try-catch`** ou bien le mettre dans un bloc **`finally`**.

Correction :

```
// insertion dans la formation en vol
Vaisseau enPlus = new Vaisseau();
```

```
enPlus.initRandom() ;
boolean raised ;
do {
     raised = false;
     try
     {
          // Lecturede l'indice de position depuis le clavier
          System.out.println( "Saisir la position du nouveau venu");
          Scanner sc = new Scanner(System.in);

          int indice = sc.nextInt();
          formation.setVaisseau( indice, enPlus);

          // Affichage
          System.out.println(formation);
     }
     catch (FormationInsertionException e)
     {
          raised = true ;
          System.err.println( e);
          //e.printStackTrace();
     }
} while ( raised ) ;
```