PROJET N°2 *Coder en Java avec méthode : ranger*

Objectifs : coder avec méthode

- Ranger ses fichiers
- Compiler, exécuter du code, générer la javadoc en ligne de commande
- Écrire et lancer des tests (méthode artisanale pour le moment)

Exercice 1 *Range ton projet !*
Il est temps pour toi d'apprendre à ranger les fichiers de tes projets java.

Méthodologie

On crée un dossier par projet. Dans ce dossier, on crée plusieurs sous-dossiers :

- un dossier `src` pour les fichiers `.java`
- un dossier `bin` ou `out` pour les fichiers `.class`
- un dossier `doc` dans lequel on génèrera la javadoc
- un dossier `images` pour les images
- ... etc

Pour **compiler** les fichiers sources (les fichiers `*.java`) qui sont dans le dossier `src/` et mettre les fichiers `*.class` dans le dossier `bin/`

```
javac -d chemin/vers/bin chemin/vers/src/*.java
```

Pour **exécuter** la classe `Executable` (les fichiers `.class` étant dans le dossier `bin`)

```
java -cp chemin/vers/bin Executable
```

ou encore

```
java -classpath chemin/vers/bin Executable
```

Pour **générer la javadoc** dans le dossier `doc/`

```
javadoc [options] -d doc chemin/vers/src/*.java
```

Commandes à connaître

Si on se trouve à la racine de son projet :

- **Compiler** les fichiers source : `javac -d bin src/*.java`
- **Exécuter** la classe `Executable` : `java -cp bin Executable`
- **Générer la javadoc** dans le dossier `doc/`
  `javadoc -charset utf8 -noqualifier java.lang -d doc src/*.java`

1.1 Range dans un dossier `src` les fichiers sources du projet `Vaisseau` de la feuille Projet N° 1 et supprime tous les autres fichiers.

1.2 Compile les fichiers sources et lance l'exécutable en respectant la méthodologie proposée.

1.3 Génère la javadoc.

Exercice 2 *La classe Vaisseau (suite)*

On reprend le projet de l'exercice 2 de la feuille Projet N° 1. On veut compléter le projet de façon à ce que le code suivant s'exécute sans erreur :

```
class ExecutableVaisseau{
  public static void main(String [] args){
    ... // reprendre le code de la feuille de projet précédente
    Vaisseau executor = new Vaisseau("Super Star Destroyer", 250, 38000);
    Vaisseau corvette = new Vaisseau("Corvette", 2, 80);

    Flotte empire = new Flotte();
    empire.ajoute(chasseur);
    empire.ajoute(executor);

    assert "Nouvelle Flotte".equals(empire.getNom());
    assert 2 == empire.nombreVaisseaux();
    assert 250 + 8 == empire.totalPuissance();

    Flotte alliance = new Flotte("Alliance rebelle");
    alliance.ajoute(faucon);
    alliance.ajoute("A-Wing", 11);
    alliance.ajoute("Nautilian", 175, 10000);
    alliance.ajoute(corvette);
    alliance.ajoute(new Vaisseau("B-Wing", 7, 0));

    assert "Alliance rebelle".equals(alliance.getNom());
    assert 5 == alliance.nombreVaisseaux();
    assert 4 + 11 + 175 + 2 + 7 == alliance.totalPuissance();
  }
}
```

2.1 Quelle classe doit-on ajouter au projet ? Complète le diagramme de classes donné en annexe, sans oublier les associations.

2.2 Écris le code MINIMAL pour que le projet compile.

2.3 Vérifie que les tests de l'exécutable ÉCHOUENT.

2.4 Complète petit à petit le code et vérifie au fur et à mesure que ton code est correct en activant les tests de l'exécutable.

Exercice 3 *Manipuler des listes*

3.1 Identifie les méthodes nécessaires à la bonne exécution du code ci-dessous en précisant pour chacune la classe à laquelle elle appartient.

3.2 Complète le code de façon à ce que les tests passent. Ajoute quelques tests pour assurer la robustesse de ton code.

```
public class ExecutableVaisseau{
  public static void main(String [] args){
    ... // reprendre le code précédent
    assert 1 == empire.nombreDeVaisseauxSansPassagers();
    assert 2 == alliance.nombreDeVaisseauxSansPassagers();
    assert 250 == empire.puissanceDeFeuMax();
    assert 175 == alliance.puissanceDeFeuMax();
    assert "Chasseur Tie".equals(empire.nomDuVaisseauLeMoinsPuissant());
    assert "Corvette".equals(alliance.nomDuVaisseauLeMoinsPuissant());

  }
}
```

Prénom Nom : ..... Groupe : .....

Annexe pour l'exercice 2

Vaisseau
- nom : String - nombreDePassagers:int = 0 - puissanceDeFeu : int
+ Vaisseau(nom:String, puissance:int) + Vaisseau(nom:String, puissance:int, passagers:int) + getNom():String + nombrePassagers():int + getPuissance():int + transportePassagers():boolean