PROJET N°1 *Coder en Java avec méthode : documenter et tester*

Objectifs : coder avec méthode

- Documenter son code et générer la javadoc
- Écrire des tests (méthode artisanale pour le moment)

Exercice 1 *Documentation de code*

On vous propose le code java de deux classes :

```java
public class Chat {
    /** Nom du chat */
    private String nom;
    /** Cet attribut modélise à quel point le chat est bavard */
    private int bavard;
    /** permet de construire un Chat plus ou moins bavard */
    public Chat(String nomDuChat, int bavard) {
        this.nom = nomDuChat;
        this.bavard = bavard;
    }
    /** permet de construire un Chat "normal" */
    public Chat(String nomDuChat) {
        this(nomDuChat, 1);
    }
    public String getNom() {
        return this.nom;
    }
    public void setNom(String nouveauNom) {
        this.nom=nouveauNom;
    }
    public void devientMuet() {
        this.bavard = 0;
    }
    /**
     * Cette méthode fait miauler le chat en affichant un message
     * sur la sortie standard
     * Plus le chat est bavard, et plus il miaule
     */
    public void miaule() {
        System.out.print(this.nom);
        for(int i = 0; i < this.bavard; ++i) {
            System.out.print(" Miaou! ");
        }
        System.out.println(" ...");
    }
    /**
     * Cette méthode détermine si le chat est endormi
     * @param heure un double compris entre 0 et 24
     * @return true si le chat dort (presque tout le temps)
     * et false sinon (c'est à dire entre 3 et 4 heures du matin)
     */
    public boolean estEndormi(double heure) {
        return (heure<=3 || heure>4);
    }
}
```

```java
public class Dialogue{
    public static void main(String [] args){
        Chat felix = new Chat("Felix");
        Chat chloe = new Chat("Chloé", 5);
        Chat ozone = new Chat("Ozone", 1);
        System.out.println(chloe.getNom());
        chloe.miaule();
        System.out.println(felix.getNom());
        felix.miaule();
        System.out.println(ozone.getNom());
        ozone.miaule();
  }
}
```

1.1 Sans ordinateur

a) Identifiez la/les classe(s) exécutable(s) ? Comment les reconnaître ?

b) Donnez la représentation UML de la classe `Chat`.

c) Quels sont les fichiers à créer pour copier ces codes ?

d) Proposez deux commandes bash qui permettent de compiler ces codes.

e) Quelle commande bash permet d'exécuter la classe exécutable ?

f) Anticipez ce que l'exécutable va faire.

1.2 Avec ordinateur

Créez les fichiers nécessaires dans un dossier `tp1/chat/` pour y copier le code donné.
Vérifiez que le code compile et s'exécute correctement. Vérifiez votre réponse à la question **1.2.c**.

1.3 Générer la documentation

Vous avez remarqué que le code proposé est *commenté* avec une syntaxe particulière. En réalité, il ne s'agit pas de *commentaires* mais de *documentation* qui permet de générer la *javadoc*.

Générer la javadoc

Pour générer la java doc de plusieurs classes java dans un dossier `doc`, il faut taper la commande :
`javadoc [options] chemin/vers/MaClasse1.java chemin/vers/MaClasse2.java`

Les options les plus utilisées sont les suivantes :

- `-d chemin/vers/doc` permet de placer les fichiers générés dans un dossier |doc| dont il faut spécifier le chemin. Si le dossier spécifié n'existe pas, il est créé.
- `-charset utf8` permet de préciser l'encodage utf8
- `-private` pour faire figurer les attributs et méthodes privées (private)
- `-noqualifier java.lang` pour ne pas faire figurer le chemin java.lang devant String par exemple

Il suffit ensuite d'ouvrir le fichier `index.html` qui se trouve dans le dossier `doc` avec un navigateur pour pouvoir explorer la documentation du projet.

Génèrez la documentation de la classe `Chat` et vérifiez son contenu.

Exercice 2 *La classe Vaisseau*

On donne la représentation UML d'une classe `Vaisseau` et le code d'un exécutable.

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

```java
public class ExecutableVaisseau{
  public static void main(String [] args){
    Vaisseau faucon = new Vaisseau("Faucon Millenium", 4, 6);
    System.out.println(faucon.getNom());
    System.out.println(faucon.getNombrePassagers());
    System.out.println(faucon.getPuissance());
    System.out.println(faucon.transportePassagers());
    assert "Faucon Millenium".equals(faucon.getNom());
    assert 6 == faucon.getNombrePassagers();
    assert 4 == faucon.getPuissance();
    assert faucon.transportePassagers() : "Le Faucon Millenium transporte des
       passagers";
    Vaisseau chasseur = new Vaisseau("Chasseur Tie", 8);
    assert "Chasseur Tie".equals(chasseur.getNom());
    assert 0 == chasseur.getNombrePassagers();
    assert 8 == chasseur.getPuissance();
    assert !chasseur.transportePassagers() : "Le Chasseur Tie ne transporte pas
        de passagers";
  }
}
```

2.1 Dans un dossier `tp1/vaisseau`, créez un fichier `ExecutableVaisseau.java` dans lequel vous copierez le code donné, puis un fichier `Vaisseau.java` vide.

2.2 Écrivez le code MINIMAL de la classe `Vaisseau` pour que le projet compile. Pour le moment, le code ne fait RIEN. Vérifiez que votre projet compile.

2.3 Vérifiez que les tests de l'exécutable ÉCHOUENT.

2.4 Complètez petit à petit le code des constructeurs et méthodes de la classe `Vaisseau`. Vérifiez au fur et à mesure que votre code est correct en activant les tests de l'exécutable. Quelques précisions :

- L'attribut `nombreDePassagers` indique le nombre maximum de passagers que le vaisseau peut transporter.
- La méthode `transportePassagers()` renvoie un booléen indiquant si le vaisseau est capable de transporter des passagers.

Exercice 3 *Modélisation (simpliste) d'un jeu d'affrontement*

On donne le diagramme de classe d'une classe `Champion`. Cette classe permet de modéliser un Champion dans un jeu d'affrontement :

- Quand deux Champions s'affrontent, ils perdent chacun un nombre de points de vie égal à la valeur d'attaque de leur opposant.
- Quand un Champion boit une potion de soin, il récupère 5 points de Vie.
- Quand un Champion soigne un Champion blessé, il lui redonne un nombre de points de vies égal à sa capacité de soin.

Champion
- nom:String - pointsDeVie:int - attaque:int - soin:int
+ Champion(nom:String, pointsDeVie:int, attaque:int, soin:int) + combat(adversaire:Champion):void + boitUnePotionDeSoin():void + soigne(championBlesse:Champion):void + estEnVie():boolean + toString():String

Voici le code de d'une classe exécutable :

```
public class ExecutableChampion {
    public static void main(String [] args) {
        /* Création des instances */
        Champion teemo = new Champion("Teemo", 37, 7, 0);
        Champion darius = new Champion("Darius", 38, 11, 0);
        Champion sona =  ...
        System.out.println(sona.toString()); // Sona pv=27, att=3, soin=5
        teemo.combat(darius);
        System.out.println(teemo.toString()); // Teemo pv=26, att=7, soin=0
        teemo.boitUnePotionDeSoin();
        System.out.println("premier combat");
        System.out.println(darius.toString()); // Darius pv=31, att=11, soin=0
        System.out.println(teemo.toString()); // Teemo pv=31, att=7, soin=0
        while (teemo.estEnVie() && darius.estEnVie()) {
            sona.soigne(teemo);
            teemo.combat(darius);
        }
        System.out.println("fin du combat");
        System.out.println(darius.toString());
        System.out.println(teemo.toString());
    }
}
```

3.1 Quelles sont les caractéristiques du champion nommé Teemo ? Complètez l'exécutable de façon à créer une instance d'un Champion nommé Sona avec une valeur d'attaque de 3, une capacité de soin de 5 et 27 points de vie.

3.2 Dans un dossier `tp1/lol`, créez un fichier `ExecutableChampion.java` dans lequel vous copierez le code donné, puis un fichier `Champion.java` vide.

3.3 Écrivez le code MINIMAL de la classe `Champion` pour que le projet compile. Pour le moment, le code ne fait RIEN. Vérifiez que votre projet compile.

3.4 Complètez petit à petit le code des constructeurs et méthodes de la classe `Champion`. Vérifiez au fur et à mesure que votre code est correct en ajoutant des tests dans l'exécutable.