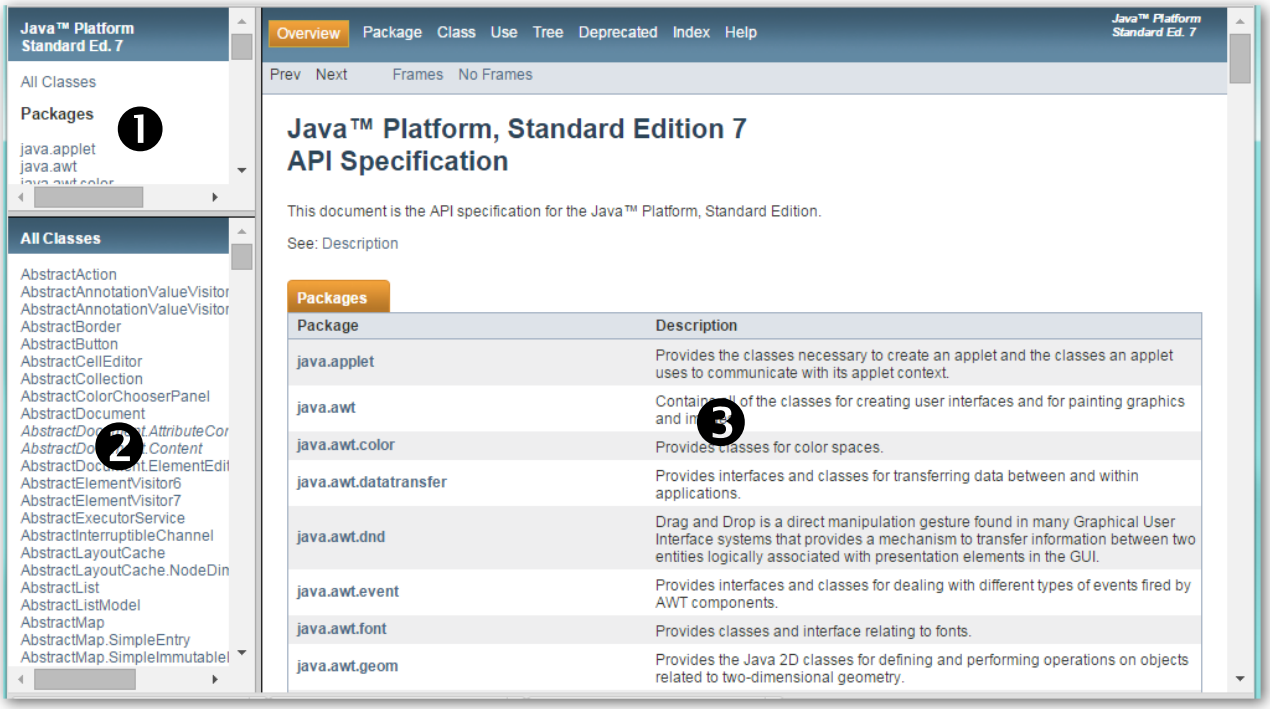
R2.01 – Développement orienté objets

TP n° 5 – Javadoc
Classe *Monôme* – tests et utilisation

I- L'utilitaire Javadoc

1) Consultation de la documentation des classes standards Java

Lorsqu'on développe en Java, il est souhaitable d'avoir à tout instant à portée de main la documentation des classes standards Java. Celle-ci a été générée au format *javadoc* et est disponible sur internet https://docs.oracle.com/javase/8/docs/api/ ou en cliquant sur « *Javadoc JDK 8*» situé dans la partie cours de Moodle.



La fenêtre qui s'affiche est découpée en 3 sous-fenêtres :

- la sous-fenêtre ❶ contient la liste des paquetages,
- la sous-fenêtre ❷ contient la liste des classes et interfaces,
- la sous-fenêtre ❸ permet l'accès au détail d'un paquetage ou d'une classe. Par défaut, elle est positionnée sur l'onglet « *overview* » et visualise les paquetages Java.

Un exemple de recherche : La classe *Math*

- Taper *<ctrl> F* et dans la fenêtre du navigateur qui s'affiche, taper « *Math* ».
- Rechercher dans ❷ la classe « *Math »* surlignée en jaune.

On peut constater que la recherche peut s'avérer longue car il y a beaucoup de classes et d'interfaces. Elle aurait pu être améliorée en sachant que la classe « *Math* » appartient au paquetage « *java.lang* » :

- Dans ❶, rechercher et sélectionner le paquetage « *java.lang* ».
- Rechercher et sélectionner dans ❷ la classe « *Math »*. Cette fenêtre ne contient à présent que les classes et interfaces du paquetage « *java.lang* ».
- Sélectionner la classe « *Math* » dans ❷. Apparaît alors dans ❸ le détail de la classe sélectionnée.
- Rechercher dans la partie « *Method Summary* » de la fenêtre ❸ la méthode « *sqrt* » utilisée dans le TP n°1 pour calculer une racine carrée.
- Cliquer sur le lien hypertexte « *sqrt* » pour voir le détail de la méthode.

2) Utilisation de l'utilitaire *javadoc* pour créer la documentation de la classe *Monôme*

Un monôme est représenté en mathématique sous la forme : $a_i x^i$ où x représente une variable, a_i son coefficient (réel) et i son exposant (entier).
Voici la spécification fonctionnelle de ce TAD :

```
Type Monôme

Opérations
      unMonôme :          Réel x Entier → Monôme
      coefficient :       Monôme → Réel
      exposant :          Monôme → Entier
      somme :             Monôme x Monôme → Monôme
      produit :           Monôme x Monôme → Monôme
      dérivée :           Monôme → Monôme
      estNul :            Monôme → Booléen

Préconditions
      unMonôme(c, e) ssi e ≥ 0
      somme (m1, m2) ssi exposant (m1) = exposant (m2)

Propriétés
      (P1)  coefficient (unMonôme (c, e)) = c
      (P2)  exposant (unMonôme (c, e)) = e
      (P3)  somme (m1, m2) =
                  unMonôme (coefficient (m1) + coefficient (m2), exposant (m1))
      (P4)  produit (m1, m2) =
                  unMônome(coefficient (m1) * coefficient (m2),exposant (m1) +
                        exposant (m2))
      (P5)  dérivée (m) =
                  si exposant (m) = 0 alors unMonôme (0.0, 0)
                  sinon unMônome (coefficient (m) * exposant (m), exposant (m) - 1)
      (P6)  estNul (unMonôme (c, e)) = (c = 0)
```

Consulter le document moodle intitulé « *Utilitaire javadoc* » où est détaillé comment commenter une classe au format *javadoc* pour pouvoir générer sa documentation.

a) Créer un nouveau projet sous eclipse et importer dans le répertoire *src* le fichier *R2-01_TP5Etudiants.ZIP*. Ce dernier contient la classe *Monôme* mettant en œuvre le TAD *Monôme* accompagnée de sa classe de tests unitaires *MonômeTest*.

b) Produire les commentaires au format javadoc de la classe *Monôme*.

Remarque : Eclipse peut aider à produire les commentaires au format javadoc.

Pour cela :

- positionner le curseur sur la ligne au-dessus de la méthode à commenter, taper /** suivi de l'appui sur la touche *entrée*

```
1  public class Monôme {
2
3      private float coefficient;
4      private int exposant;
5
6      /**
7      public Monôme(float coefficient, int exposant)
8              throws IllegalArgumentException {
9          if (exposant < 0) {
10             throw new IllegalArgumentException("exposant négatif");
11         }
12         this.coefficient = coefficient;
13         this.exposant = exposant;
14     }
15
```

- un squelette du commentaire est généré avec les balises javadoc,

```
1  public class Monôme {
2
3      private float coefficient;
4      private int exposant;
5
6      /**
7       * 
8       * @param coefficient
9       * @param exposant
10      * @throws IllegalArgumentException
11      */
12     public Monôme(float coefficient, int exposant)
13             throws IllegalArgumentException {
14         if (exposant < 0) {
15             throw new IllegalArgumentException("exposant négatif");
16         }
17         this.coefficient = coefficient;
18         this.exposant = exposant;
19     }
```

- il suffira de le compléter.

```
1  public class Monôme {
2
3      private float coefficient;
4      private int exposant;
5
6      /**
7       * crée un monôme
8       * @param coefficient coefficient du monôme
9       * @param exposant exposant du monôme
10      * @throws IllegalArgumentException si l'exposant est négatif
11      */
12     public Monôme(float coefficient, int exposant)
13             throws IllegalArgumentException {
14         if (exposant < 0) {
15             throw new IllegalArgumentException("exposant négatif");
16         }
17         this.coefficient = coefficient;
18         this.exposant = exposant;
19     }
```

c) Générer la documentation au format Javadoc

Pour cela, après avoir commenté toutes les méthodes au format javadoc, sélectionner :

Menu « Project » → *» Generate Javadoc »*

Si le champ *Java Command* n'est pas renseigné, rechercher la commande par *configure* :

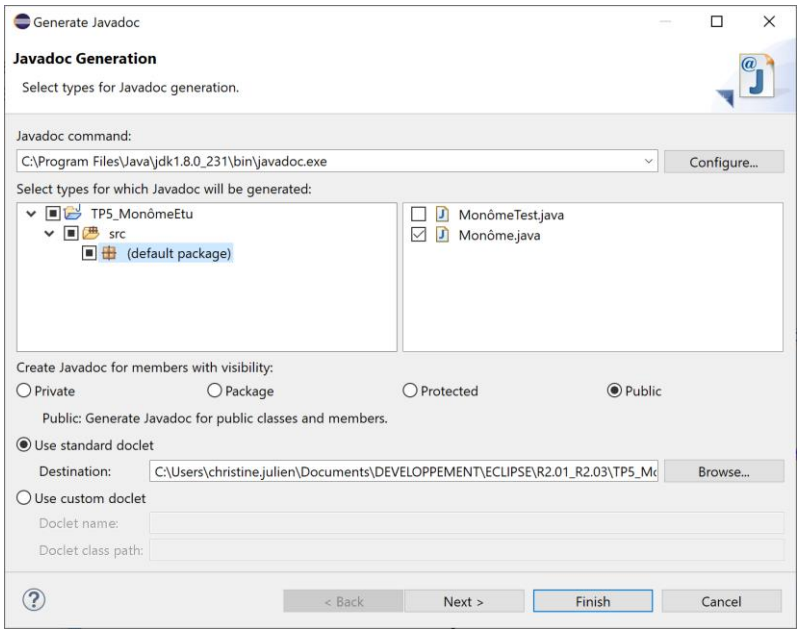
C:Program Files\Java\jdkXXX\bin\javadoc.exe

Sélectionner le fichier *Monome.java* dont on souhaite générer la documentation.

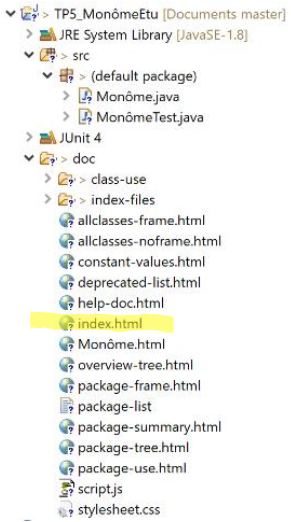
Choisir ensuite la visibilité (si *Public* : la javadoc sera générée uniquement pour les attributs et méthodes publiques, si *Private* : la javadoc sera générée pour tous les attributs et méthodes).

Cocher *Use Standard Dotlet*.

Cliquer sur *Finish*. Dans le projet sera généré un répertoire *doc* contenant la javadoc. Si nécessaire faire un *refresh* sur le projet pour le rendre visible.



Double cliquer sur *doc* → *ressources* → *index.htm* pour visualiser la documentation générée.



II-La classe *Monôme*

1) La classe Monôme et les tests

Remarque : En examinant le code de la classe Monôme, nous pouvons constater que la méthode *somme()* déclenche une exception de type *ArithmeticException* (classe standard Java) si les exposants des 2 monômes ne sont pas les mêmes. Dans ce contexte, ce type est plus approprié que *IllegalArgumentException*.

a) Compléter dans la classe *MonômeTest* les tests *testEstMonômeNul()* et *testEstMonômeNonNul()* vérifiant les 2 cas de figures de la propriété (P6).

Remarque : tester un résultat booléen se fait par l'utilisation de l'assertion *assertTrue* ou *assertFalse* de JUnit.

b) Exécuter les tests.

2) La méthode *toString()*

a) Rajouter dans la classe *Monôme* une méthode `toString()` produisant une version unicode d'un monôme. Voici des exemples de chaînes à produire :

monôme	version unicode
$\mathbf{0x^5}$	`"0.0"`
$\mathbf{-10x^5}$	`" - 10.0xe5"`
$\mathbf{-1x^5}$	`" - xe5"`
$\mathbf{-1x^0}$	`" - 1.0"`
$\mathbf{-10x^1}$	`" - 10.0x"`
$\mathbf{-1x^1}$	`" - x"`
$\mathbf{-10x^0}$	`" - 10.0"`
$\mathbf{10x^5}$	`" + 10.0xe5"`
$\mathbf{1x^5}$	`" + xe5"`
$\mathbf{1x^0}$	`" + 1.0"`
$\mathbf{10x^1}$	`" + 10.0x"`
$\mathbf{1x^1}$	`" + x"`
$\mathbf{10x^0}$	`" + 10.0"`

b) Dans les tests unitaires de la classe Monôme, rajouter ceux de la méthode `toString()`. Il faudra tester tous les cas recensés ci-dessus.

III- Application *Monôme*

Ecrire une application Java cliente de *Monôme* qui crée 2 monômes nuls et qui propose le menu ci-dessous. Celui-ci orientera sur le traitement correspondant en fonction du choix de l'utilisateur (utilisation d'une structure de contrôle ***switch***). Un choix de menu erroné conduira à l'affichage d'un message d'erreur (utilisation de la clause ***default***).

Le menu et le traitement associé au choix de l'utilisateur seront itérés jusqu'à ce que le choix soit égal à 9 (utilisation d'une structure de contrôle ***do … while***).

```
Quel est votre choix :
    1- modifier le premier monôme
    2- modifier le deuxième monôme
    3- afficher le premier monôme
    4- afficher le deuxième monôme
    5- calculer la somme des 2 monômes
    6- calculer le produit des 2 monômes
    7- calculer la dérivée du premier monôme
    8- calculer la dérivée du deuxième monôme
    9- quitter l'application
```