Annexe 9 : Documenter son code avec javadoc

Pour éviter ce qui a trop tendance à se produire, à savoir que les programmeurs ne documentent pas leurs codes, Java intègre un système de documentation automatique (enfin presque) des classes. Ainsi si vous utilisez les commentaires de documentation `/** ... */` et que vous lancez `javadoc` sur vos fichiers java, vous obtiendrez des pages renseignant votre code au format HTML. Voici à quoi cela ressemble :

```
// Property.java

import java.util.*;

/** Un fichier exemple pour l'utilisation de javadoc.
*   Le programme ci-dessous permet d'afficher quelques informations
*   systèmes concernant la machine sur laquelle il tourne. Ce code est
*   extrait de « think in Java » de Bruce Eckel.
* @author Bruce Eckel : http://www.BruceEckel.com
* @version 1.0
*/

public class Property {

  /** Sole entry point to class & application
  * @param args array of string arguments
  * @return No return value
  * @exception exceptions No exceptions thrown
  */
  public static void main(String[] args) {
          System.out.println(new Date());
          Properties p = System.getProperties();
          p.list(System.out);
          System.out.println("--- Memory Usage:");
          Runtime rt = Runtime.getRuntime();
          System.out.println("Total Memory = " + rt.totalMemory() +
                  " Free Memory = " + rt.freeMemory());
  }
}
```

Le commentaire de documentation commence par `/**` et se termine par `*/`. Les premières phrases du commentaire doivent être des phrases de description concises mais précises sur la classe, la variable ou la méthode documentée. Ensuite on peut utiliser différents "tags" commençant par @.

Pour lancer la génération des pages HTML il suffit de taper la ligne de commande :

```
javadoc MonFichier.java
```

Une fois ceci fait vous n'avez plus qu'à lire la page HTML correspondante : `MonFichier.html`

@see

Le tag `@see` s'utilise pour renseigner une classe, une interface, une variable ou une méthode. Il indique une référence croisée à une autre classe, interface, méthode, à un constructeur, un champ, un URL, etc.

```
@see classname
@see fully-qualified-classname
@see fully-qualified-classname#method-name
@see java.lang.String
@see java.lang.Object#wait(int)
@see Character#MAW_RADIX
```

Le caractère # sépare le nom d'une classe d'un de ses champs, méthodes ou constructeurs. Un commentaire de documentation peut avoir plusieurs `see` tag.

@author

Ce tag est utilisé lors de la documentation des classes ou des interfaces. Il ne doit y avoir qu'un seul tag `@author` par commentaire de documentation.

```
@author Moi Monnom, monmail@mon.adresse.fr
@author Jack Kent, Henry Fellucci, Maldred Felarit,
```

@version

Ce tag est utilisé lors de la documentation des classes ou des interfaces. Il ne doit y avoir qu'un seul tag `@version` par commentaire de documentation.

```
@version info sur la version
@version 493.0.1 beta
```

@param

Ce tag est utilisé lors de la documentation des méthodes. Il doit y avoir autant de tag `@param` qu'il y a de paramètres dans la méthode documentée.

```
@param nom_du_paramètre description
@param file the file to be searched
```

@return

Ce tag est utilisé lors de la documentation des méthodes. Il doit y avoir un seul tag `@return` par méthode documentée.

```
@return description
@return No return value
```

@exception

Ce tag est utilisé lors de la documentation des méthodes. Il doit y avoir autant de tag `@exception` que d'exceptions levées par la méthode documentée.

```
@exception NomClasseException conditions de levée d'exception
@exception java.io.FileNotFoundException the file does not exist.
```

@deprecated

Ce tag est utilisé lors de la documentation des méthodes. Il est apparu avec la version 1.1 de Java. Il signale qu'une méthode n'est plus d'actualité et qu'elle risque de ne plus être supportée par les prochaines versions. Lorsqu'il tombe sur un tel tag le compilateur affiche un warning pour prévenir l'utilisateur.