Présentation de la Javadoc

Cours IFT 1025

Rafik Gouiaa

Email: rafik.gouiaa@umontreal.ca

Bureau: 2317

Pourquoi commenter son code

- Pour une utilisation personnelle:
  - ✓ Commenter pour comprendre ce qu'on a fait.
  - ✓ Revenir sur le code sans le refaire de nouveau.
  - ✓ Anticiper le codage d'une fonction ou une classe en rédigeant le commentaire.
- Pour une utilisation professionnelle :
  - ✓ Notre code sera utilisé par d'autre personne.
  - ✓ Des commentaires utiles rendre la communication facile sur le travail.
  - ✓ C'est toujours mieux vu de fournir un code commenté.

Problèmes des commentaires internes

- Pour avoir juste une vue globale sur le code, il faut lire intégralement le code.
- Si on développe une méthode quelconque sans le commenter, il ne peut comprendre son principe sans retour sur google, etc.
- Perte de temps.

L'outil JavaDoc

- Javadoc est un outil développé par Sun Microsystems.
- Permet de produire une documentaion très complète de notre code.
- L'outil Javadoc est inclus dans tous les JDK (ou SDK) de java.
- L'outil Javadoc se trouve dans le répertoire **bin du dossier JDK.**

L’outil JavaDoc

- L'outil génère des pages HTML contenant au minimum la liste des classes, la liste des méthodes et la liste des variables.
- L’avantage de générer des pages HTML tient dans la présence de liens hypertextes.
- Il devient très facile de naviguer dans la documentation.
- Exemple: http://docs.oracle.com/javase/7/docs/api/

Les tags JavaDoc

- Il est possible d'ajouter une grande quantité d'information à un code, en utilisant les commentaires en lignes ou en plusieurs lignes.
- Mais, il ne sont pas pris en compte par l'outil Javadoc.
- Sun a proposé des tags précis qui permettent de détailler des informations sur chaque classe, chaque méthode, chaque attribut, etc.

Les tags JavaDoc

- Il existe plusieurs tags:
  - @param: Les paramètres d'une fonction.
  - @return: L'objet retourné par une méthode.
  - @throws: indique la présence d'une exception.
  - @author: l'auteur du programme.
  - @version: La version du programme.
  - @see: réfère à une méthode ou à une classe.
  - @since: permet de dater la présence d'une méthode, d'un paramètre, d'une classe.

Les tags JavaDoc

- @**deprecated:** décrire la version depuis laquelle cette méthode ou classe est dépréciée, et avec quelle méthode ou classe et remplacée.
- {@**code}** traiter une partie de la documentation sous la forme d'un code java.

Exemples: @param

```
/**
 *
 * @param titre
 * @param nomAuteur
 * @param nbPages
 * @param prix
 */
public Livre(String titre, String nomAuteur, int nbPages,
double prix)
{
    this(titre, nomAuteur, nbPages, prix, false, 0.0);
    nbLivres++;
}
```

Exemples: @return

```
/**
 *
 * @return
 */
public String getNomAuteur(){

    return nomAuteur;

}
```

Exemples: @see

```
/**
 *
 * @param titre
 * @param nomAuteur
 * @param nbPages
 * @param prix
 * @see #Livre(String, String, int, double, boolean, double)
 */
public Livre(String titre, String nomAuteur, int nbPages,
double prix)
{
    this(titre, nomAuteur, nbPages, prix, false, 0.0);
    nbLivres++;
}
```

Exemples: @see

```
/**
 *
 * @param titre
 * @param nomAuteur
 * @param nbPages
 * @param prix
 * @see #Livre(String, String, int, double, boolean, double)
 */
public Livre(String titre, String nomAuteur, int nbPages,
double prix)
{
    this(titre, nomAuteur, nbPages, prix, false, 0.0);
    nbLivres++;
}
```