Programmation orientée objet
Nommage et documentation

BTS SIO 1 – SLAM2

Programme

- Programmation Orientée Objet (POO)
  - Conventions de nommage
  - Documentation

Conventions de nommage
Les packages

- Le nom d'un package doit respecter les conventions suivantes:
  - Tout en minuscule.
  - Utiliser seulement [a-z], [0-9] et le point '.': Ne pas utiliser de tiret '-', d'underscore '_', d'espace, ou d'autres caractères ($, *, accents, ...).
  - La convention de Sun indique que tout package doit avoir comme root un des packages suivant: com, edu, gov, mil, net, org ou les deux lettres identifiants un pays (ISO Standard 3166, 1981).

Exemples

- `com.monProjet`
- `com.apple.quicktime.v2`
- `edu.cmu.cs.bovik.cheese`

Exemple de packages java

- `java.lang`
  Types fondamentaux et fonctionnalités basiques du langage
- `java.util`
  Gestion des collections et opérations associées
- `java.io`
  Gestion des flux (ex : fichiers) en lecture et écriture
- `java.math`
  Bibliothèque mathématique
- `java.net`
  Gestion des communications via les sockets
- `java.security`
  Génération des clés, chiffrement et déchiffrement
- `java.sql`
  Gestion des accès à la base de données (JDBC)

Conventions de nommage
Les classes

- Les noms des classes doivent respecter les conventions suivantes (d'après SUN):
  - 1ère lettre en majuscule
  - Avec la première lettre de chaque mot en majuscule
  - Donner des noms simples et descriptifs
  - Eviter les acronymes : hormis ceux commun (XML, URL, HTML, ...)
  - N'utiliser que les lettres [a-z] et [A-Z] et [0-9] : Ne pas utiliser de tiret '-', d'underscore '_', ou d'autres caractères ($, *, accents, ...).

Conventions de nommage
Les variables

- Les noms des variables doivent respecter les conventions suivantes (d'après SUN):
  - 1ère lettre en minuscule
  - Avec la première lettre de chaque mot en majuscule ensuite
  - Donner des noms simples et descriptifs
  - Ne pas commencer les noms avec '$' ou '_' bien que ce soit possible.
  - N'utiliser que les lettres [a-z] et [A-Z] et [0-9] : Ne pas utiliser de tiret '-', d'underscore '_', ou d'autres caractères ($,*, accents, ...).

Conventions de nommage
Les constantes

- Les noms des variables doivent respecter les conventions suivantes (d'après SUN):
  - Tout en majuscule
  - Séparer les mots par underscore '_'
  - Donner des noms simples et descriptifs
  - N'utiliser que les lettres [A-Z], [0-9] et '_' : Ne pas utiliser de tiret '-' ou d'autres caractères ($, *, accents, ...).

Conventions de nommage
Les bibliothèques (*.jar)

- Les noms des fichiers jars doivent respecter les conventions suivantes :
  - Tout en minuscule
  - Séparer les mots par un tiret '-'
  - Utiliser seulement les lettres [a-z], [0-9] et '-' : Ne pas utiliser d'underscore '_' ou d'autres caractères ($, accents, ...).
- Exemples
  - `gb-fwk-1.0.jar`
  - `gb-tools-1.0-beta-7.jar`
  - `js-1.5R4-RC3.jar`

Documentation java

- On peut générer automatiquement des pages HTML correspondants à la documentation de notre code. Que ce soit en Java ou en PHP, il suffit d'utiliser les annotations.
- En renseignant ces annotations, l'utilitaire Javadoc va réaliser des pages HTML. Il est nécessaire de mettre ces annotations dans des balises de commentaires un peu particulière :

```
/**
 * mon commentaire
 */
```

Attention : Il faut respecter les deux `**`.

Exemple

```
/** Classe A
* @author XXXX
* @version 1.0
*/
public class A {
    /**
    * Un constructeur avec un entier en paramètre
    * @param n un entier
    */
    public A(int n) {
     …
    }

    /** Pour faire un essai
    * @param c un caractère
    * @return la valeur entière de c
    */
    public int faire(char c) {
       return (int)c;
    }
}
```

Liste des balises javadoc (1/2)

- `@author`
  Nom du développeur
- `@param`
  Définit un paramètre de méthode. Requis pour chaque paramètre.
- `@return`
  Documente la valeur de retour. Ce tag ne devrait pas être employé pour des constructeurs ou des méthodes définis avec un type de retour void.
- `@version`
  Donne la version d'une classe ou d'une méthode.

Liste des balises javadoc (2/2)

- `@deprecated`
  Marque la méthode comme dépréciée. Certains IDEs créent un avertissement à la compilation si la méthode est appelée.
- `@exception`
  Documente une exception lancée par une méthode — voir aussi @throws
- `@see`
  Documente une association à une autre méthode ou classe.
- `@since`
  Précise à quelle version de la SDK/JDK une méthode a été ajoutée à la classe.
- `@throws`
  Documente une exception lancée par une méthode. Un synonyme pour @exception disponible depuis Javadoc 1.2.