Commentaires et documentation

Alexis Nasr (d'après les slides de Arnaud Labourel)

Commentaires inutiles

```
String get(String[] source, int index) {
    // Teste si l'index est dans les limites du tableau.
    if (index < 0 || index >= source.length)
        return null;
    return source[index];
}
```

Si un commentaire semble nécessaire, le remplacer par une méthode :

```
boolean indexIsInBounds(String[] source, int index) {
    return index >= 0 && index < source.length;
}
String get(String[] source, int index) {
    if (!indexIsInBounds(source, index)) return null;
    return source[index];
}
```

Commentaires inutiles

Les commentaires se désynchronisent souvent du code :

```
/* doit toujours retourner true. */
boolean isAvailable() {
    return true;
}
```

risque de devenir un jour :

```
/* doit toujours retourner true. */
boolean isAvailable() {
    return false;
}
```

Commentaires inutiles = répétition

Commentaires inutiles

Des commentaires qui peuvent sembler utiles :

```
/* une méthode qui retourne que les carrés : */
List<Rectangle> get(List<Rectangle> list) {
    /* le résultat sera stocké dans cette liste : */
    List<Rectangle> list2 = new ArrayList<Rectangle>();
    for (Rectangle x : list)
        if (x.w == x.h /* un carré ? */)
            list2.add(x);
    return list2;
}
class Rectangle {
    public int w; /* largeur */
    public int h; /* hauteur */
}
```

Commentaires inutiles

On peut se passer de commentaire en rajoutant une méthode et en nommant correctement les éléments du code.

```
List<Rectangle> findSquares(List<Rectangle> rectangles) {
    List<Rectangle> squares = new ArrayList<Rectangle>();
    for (Rectangle rectangle : rectangles)
        if (rectangle.isSquare())
            squares.add(rectangle);
    return squares;
}
class Rectangle {
    private int width, height;
    boolean isSquare() {
        return width == height;
    }
}
```

Commentaires **utiles**

Des commentaires pour décrire les tâches à réaliser peuvent être utiles

```
void processElement(Stack<Formula> stack,
                    String element) {
    // TODO : prendre en compte les signes '-' et '/'
    switch (element) {
        case "+": processSum(stack); break;
        case "*": processProduct(stack); break;
        default : processInteger(stack, element);
        break;
    }
}
```

Commentaires **utiles**

Documentation ou spécification du comportement d'une méthode :

```
/**
 * Returns <tt>true</tt> if this list contains the
 * specified element. More formally, returns
 * <tt>true</tt> if and only if this list contains
 * at least one element <tt>e</tt> such that
 * <tt>(o==null ? e==null : o.equals(e))</tt>.
 *
 * @param o element whose presence in this list is
 * to be tested
 * @return <tt>true</tt> if this list
 * contains the specified element
 */
public boolean contains(Object o) {
    return indexOf(o) >= 0;
}
```

JavaDoc

JavaDoc permet de générer automatiquement une documentation du code à partir de commentaires associés aux classes, méthodes, propriétés, ...

La documentation contient :

- Une description des membres : attributs et méthodes (publics et protégés) des classes
- Une description des classes, interfaces, ...
- Des liens permettant de naviguer entre les classes
- Des informations sur les implémentations et extensions

Un bloc de commentaire Java commençant par /** deviendra un bloc de commentaire Javadoc qui sera inclus dans la documentation du code source.

JavaDoc

Tag	Description
@author	pour préciser l'auteur de la fonctionnalité
@deprecated	indique que l'attribut, la méthode ou la classe est dépréciée
@return	pour décrire la valeur de retour
{@code `literal`}	Formate `literal` en code
{@link `reference`}	permet de créer un lien

Pour générer la javadoc

- avec IntelliJ : Tools → generate Javadoc
- à partir du terminal : javadoc MaClasse.java

JavaDoc

```
/**
 * The {@code Byte} class wraps a value of primitive
 * type {@code byte} in an object.  An object of type
 * {@code Byte} contains a single field whose type is
 * {@code byte}.
 *
 * <p>In addition, this class provides several methods
 * for converting a {@code byte} to a {@code String}
 * and a {@code String} to a {@code byte}, as well as
 * other constants and methods useful when dealing
 * with a {@code byte}.
 *
 * @author  Nakul Saraiya
 * @author  Joseph D. Darcy
 */
```