



06

Chapitre

Code propre

1I2AC1 : Génie logiciel et Conception orientée objet

Régis Clouard, ENSICAEN - GREYC

« Codez toujours comme si le type qui sera chargé de maintenir votre code est un violent psychopathe qui sait où vous habitez. »

John Woods

Plan du chapitre

1

Pourquoi faire
propre ?

Question

- Qu'est ce qu'un code propre ?
 - Que vous a t-on enseigné pour faire du code propre ?

Code propre

- Exemple
 - Calcul des 100 premiers nombres premiers

Plan du chapitre

1

Pourquoi faire
propre ?

2

Pourquoi
la pratique
classique
est fausse ?

**Les commentaires sont néfastes au
code**

Les commentaires mentent

- Le code évolue toujours plus vite que les commentaires

```
// returns null if register doesn't exist
public void registerItem( Item item ) throws NoRegistryException {
    /* ... */
}
```

Les commentaires n'apportent aucune information supplémentaire

- Les commentaires parodient le code : inutiles et lourds

```
// Default constructor
protected AnnualDateRule() {
}

/** The day of the month */
private int dayOfMonth;

/** @return the day of the month */
public int getDayOfMonth() {
    return dayOfMonth;
}
```


Les commentaires conduisent à une incompréhension

- Les commentaires sont bâclés : incomplets, confus, ambigus

```
/*  
 * Returns whether the light is on.  
 */  
bool getLightStatus( Light light) {  
    if (!light.isOn()) {  
        resetLight(light);  
    }  
    return light.isOn();  
}
```

**Les commentaires rendent le code
illisible**

Les commentaires brulent le code

- Les commentaires de fin de bloc sont inutiles et encombrants

```
public static int main( String args[] ) {
    try {
        while ((line = in.readLine()) != null) {
            lineCount++;
            charCount += line.length();
            String words[] = line.split("\\W");
            wordCount += words.length;
        } // while
        System.out.println("wordCount = " + wordCount);
        System.out.println("lineCount = " + lineCount);
        System.out.println("charCount = " + charCount);
    } // try
    catch (IOException e) {
        System.err.println("Error:" + e.getMessage());
    } //catch
} //main
```


**Les commentaires décrédibilisent
le développeur**

Les fautes d'orthographe

```
int i; /* Conter variable for "for" loop. */
int t; /* Total of additions for calculaton */
int d; /* Individual number for calclatuion */
/* "for" loop */
for (i=0; i<100; i++) { /* increment i by one until hunderd */
    d = f(); /* get the calue for d */
    t = t + d; /* ad it to t */
}
```

Les copier / coller malheureux

- Les copier/coller malheureux

```
/* The version. */  
private String version;  
/* The licenceName. */  
private String licenceName;  
/* The version. */  
private String info;
```

Bilan

Bilan

■ Constat

- Les commentaires introduisent de la confusion ou pire des mensonges
- Les commentaires créent du bruit dans le code
- Les commentaires nuisent à la lisibilité du code

■ Conclusion

- Il faudrait supprimer les commentaires comme la documentation

Attention : Cas particulier des API

18

- Dans ce cas, la documentation (type Doxygen / Javadoc) est **obligatoire**
 - L'utilisateur ne doit pas avoir à se prolonger dans le code pour comprendre la fonction et son utilisation

```
/**
 * Computes the matching between the reference regions
 * and the segmentation output regions.
 * @param segmentation the output region map.
 * @param reference the reference region map.
 * @result the region map with the best matching.
 */
RegionMap *matching( RegionMap *segmentation, RegionMap *reference ) {
    ...
}
```

- Mais, contrairement à du code de logiciel, les codes des API sont stables par essence (utilisation @deprecated pour les choses qui changent)