

Optimus Code Commenting Guidelines

Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No. | Release Date | Review By | Description |
| 1 | May 17, 12 | Vipul | First Draft |
|  |  |  |  |

Table of Contents

[Objective 3](#_Toc325025071)

[Header Comments 3](#_Toc325025072)

[File Header Comment Block 3](#_Toc325025073)

[Class Header Comment Block 3](#_Toc325025074)

[Method Header Comment Block 3](#_Toc325025075)

[Inline Comments 4](#_Toc325025076)

[Variables 4](#_Toc325025077)

[Body of Method 4](#_Toc325025078)

[Other Resources 5](#_Toc325025079)

[Metrics 5](#_Toc325025080)

[Spell Check Helps 5](#_Toc325025081)

# Objective

This guide aims to provide the information required for different comment types. This is not to illustrate the comment styles because the styles vary between programming languages.

# Header Comments

## File Header Comment Block

A file header comment block should be placed in the beginning of the file. It contains the following information:

* Project
* File name
* File description
* Copyright (only if the project is for Optimus)
* Defect IDs?
* Name of the programmer who first creates the file?

Example:

//////////////////////////////////////////////////

// OptimusMobile Android Project

// File: JSONParser.java

//

// Parser to parse JSON response to String

//////////////////////////////////////////////////

## Class Header Comment Block

A method comment block should be placed before the class declaration. It contains the following information:

* Class description (Within 5 sentences)
  + What the class is for
  + How it is used

Example:

/\*\*

\* Represents a card with a Suit and a CardValue.

\*/

public class Card

{

## Method Header Comment Block

A method comment block should be placed before the method declaration. It contains the following information:

* Method description
* Input parameters (if any)
* Return (if any)
* Exceptions (if any)
* Pre-conditions (if any)
* Post-conditions (if any)

Java Example:

/\*\*

\* Returns true if either the suit OR the value of the Card is the same as

\* the specified card.

\*

\* @param card The card to compare with this card.

\*

\* @return true if either the suit or the value matches.

\*

\* @throw NullPointerException if the card is null.

\*/

public boolean matches ( Card card ) throws NullPointerException{

C# Example:

/// <summary>

/// Connects to the database and attempts to apply

/// all adds, updates and deletes

/// </summary>

/// <param name="data">a dataset, passed by reference,

/// that contains all the

/// data for updating>/param>

public void SaveData(ref DataSet data)

XML commenting style is used for the C# example. For more details, please see <http://www.codeproject.com/Articles/3009/C-Documenting-and-Commenting>.

# Inline Comments

## Variables

Variable declarations must be commented briefly (in one line) describing their use. These types of variables must be commented:

* Class data members and constants
* Instance data members and constants
* Global variables and constants
* Local variables and constants (if the variable uses are not straightforward)

These types of variables are not necessarily to be commented:

* Temporary variables
* Loop counters

## Body of Method

Please place comments in the body of a method to

* Highlight the major steps of the algorithm
* Explain long calculations and complex logics
* Clarify convoluted or unusual code
* Mark locations where a bug may exist
* Mark locations where improvements or enhancements are planned

Beware:

* Do not use comments to explain things that are very obvious, e.g. “assign 17 to variable count”.
* In some cases, explain why you are writing this code instead of simply describing what this code is doing, e.g. “Wait for 5 minutes for the transition to complete” instead of “Wait for 5 minutes”
* Avoid one-liner comment (unless it is necessary). Instead, group several lines of the code together and write a comment for the group.
* Pay attention to indentation.

In short, commenting the code behavior is a routine practice, and explanation to the implementation should be added when necessary.

# Other Resources

## Metrics

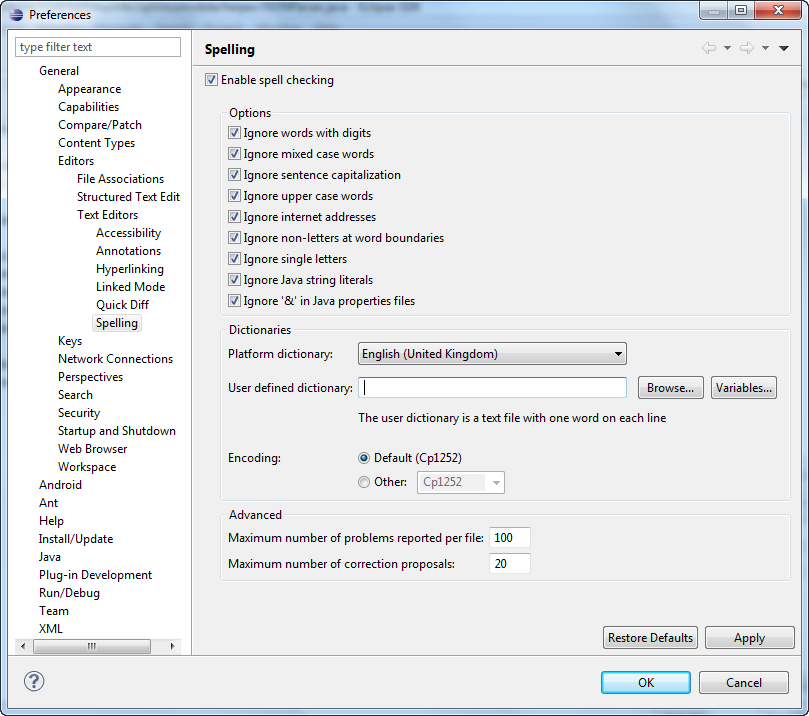
Typically speaking, a comment percentage less than 10% is considered insufficient. Some people suggest the percentage should go up to 20%.

## Spell Check Helps

A lot of the popular IDEs include spell checkers, or spell checker plug-ins are available. Ways to enable spell checkers for several common IDEs are listed here

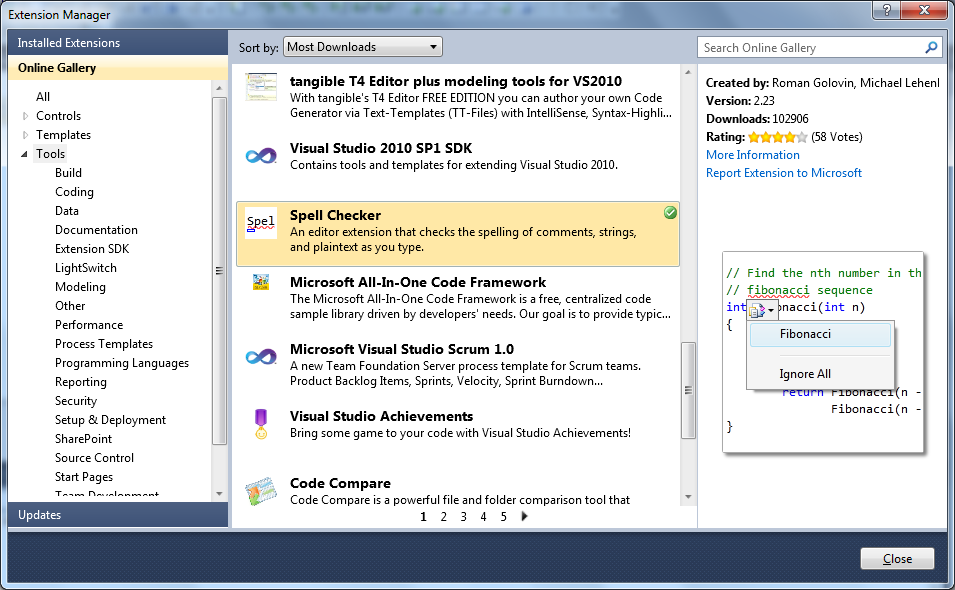
* Eclipse

The spell checker can be enabled in Eclipse. Just go to Windows -> Preferences -> General -> Editors -> Text Editors -> Spelling, and check Enable Spell Checking. User is allowed to select the dictionaries; user can define a Platform dictionary by choosing from the drop-down list, and a User-defined dictionary by providing a text file.



* Visual Studio

A spell checking tool can be downloaded from the Visual Studio Extensions. To download a spell checking extension, go to Tools -> Extension Manager, then select Online Gallery -> Tools. There are several spell checking extensions available. Search for one and download it.



* Notepad++

1. Download and install Aspell at <http://aspell.net/win32/>
2. Download and install a dictionary at <http://aspell.net/win32/>
3. Configure the path to Aspell in Notepad++

Click Plugins -> Spell-Checker -> Spell-Checker, provide the location of Aspell, and restart Notepad++