

CODES User Guide

*October 2013
Version 1.0*

Gerardo Canfora, University of Sannio (Italy)

Massimiliano Di Penta, University of Sannio (Italy)

Sebastiano Panichella, University of Sannio (Italy)

Carmine Vassallo, University of Sannio (Italy)

Table of Contents

1. Introduction	1
2. The context: source code lacks comments	1
3. Start plugin	4
4. The first screen: chosen of classes	6
5. Settings: research's configuration	6
6. Search and load decriptions	10
7. Chosen of descriptions	11
8. Add as <i>Javadoc</i> comments	14

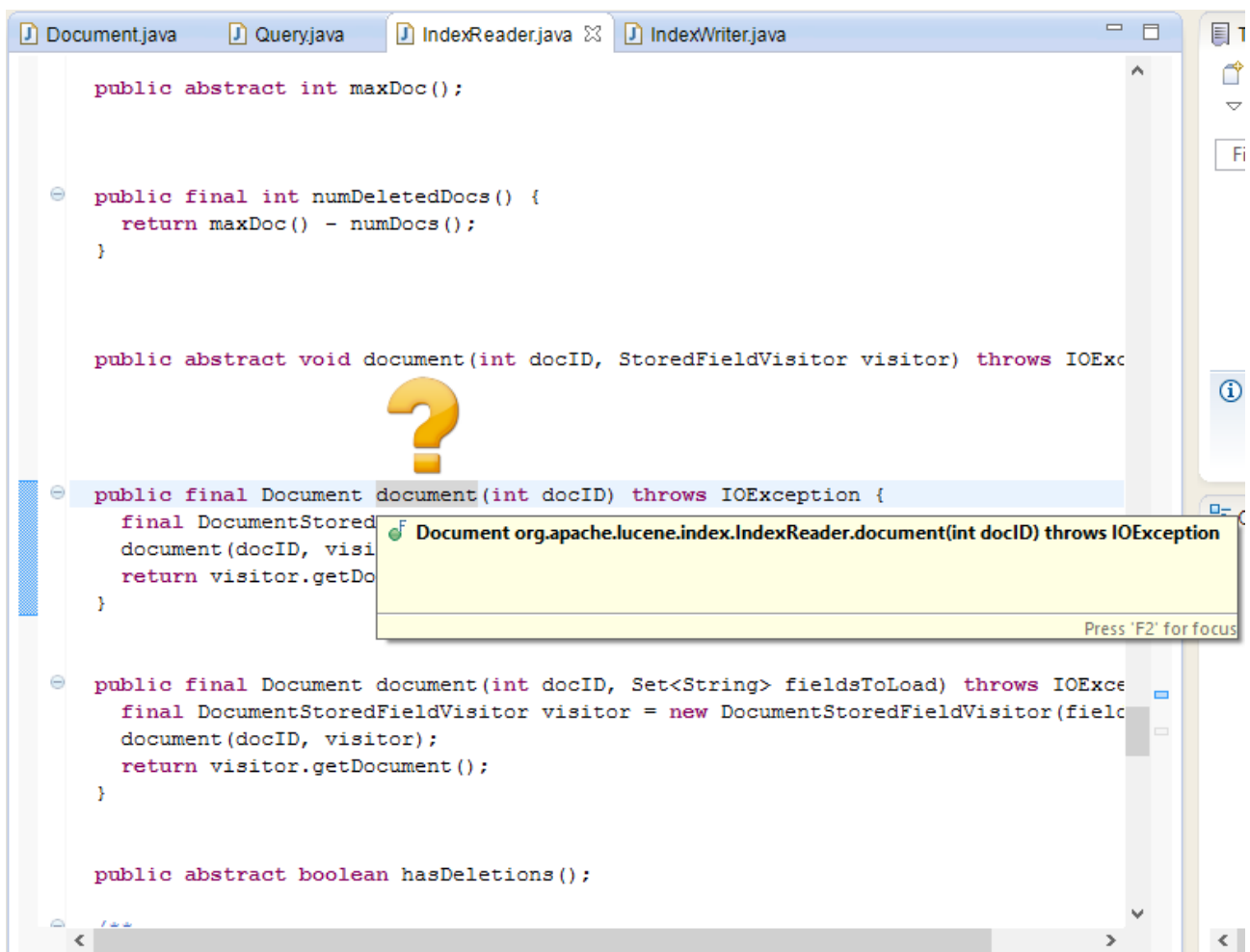
1. Introduction

This document describes how CODES plugin works. You can use it to obtain good descriptions of java class's methods, so build a Javadoc comment. We are going to describe all plugin's features through an example: we choose a method of a particular project and try to find good method's descriptions.

2. The context: source code lacks comments

Consider the following scenario: you're a new-comer of Lucene Project and you want to modify *IndexReader* class in *org.apache.lucene.index* package. You need to understand better some methods: for example *document(int)* method (Figure 1).

Figure 1



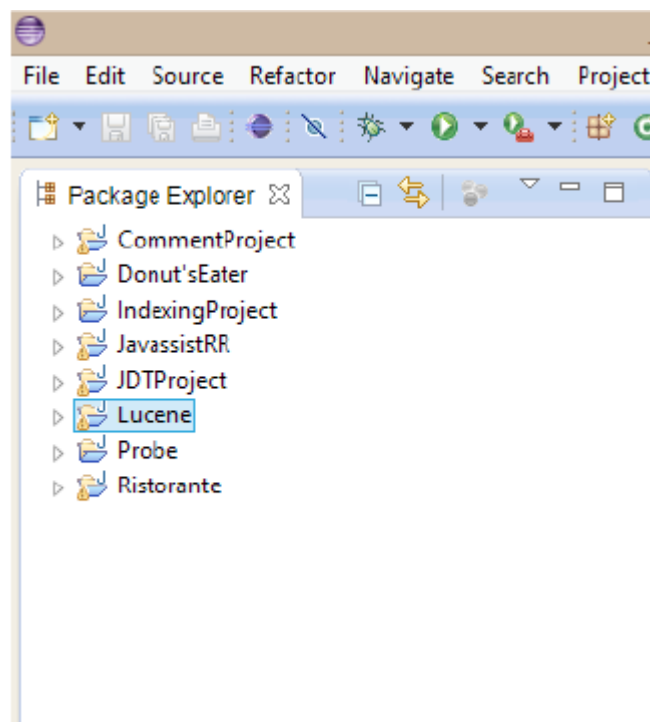
Unfortunately the method lacks a comment that adequately describes its behaviour. In that context, the plugin works: helps you through automatic extraction of method descriptions from StackOverflow.

3. Start plugin

You can start the plugin through two steps:

- 1) You have to push mouse's right button on the Java project¹ for which you hope to find methods' descriptions for some classes (in our example “Lucene”, [Figure 2](#))

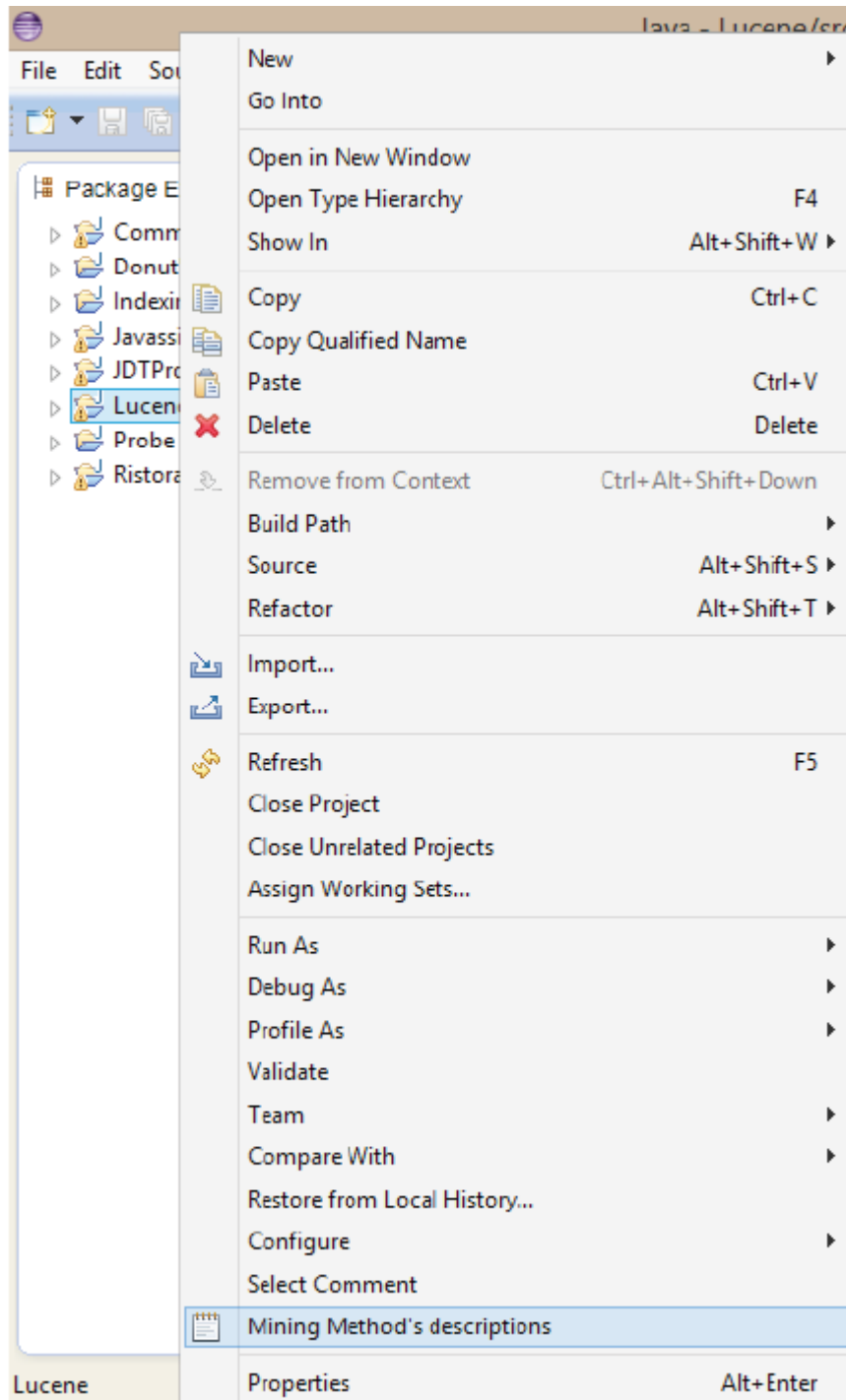
Figure 2



¹ Obviously the plugin is only "visible" in a Java project's context menu and not in others (i.e. C++ project)

2) The project's context menu is opened: you have to choose “Mining Method's Descriptions” (Figure 3)

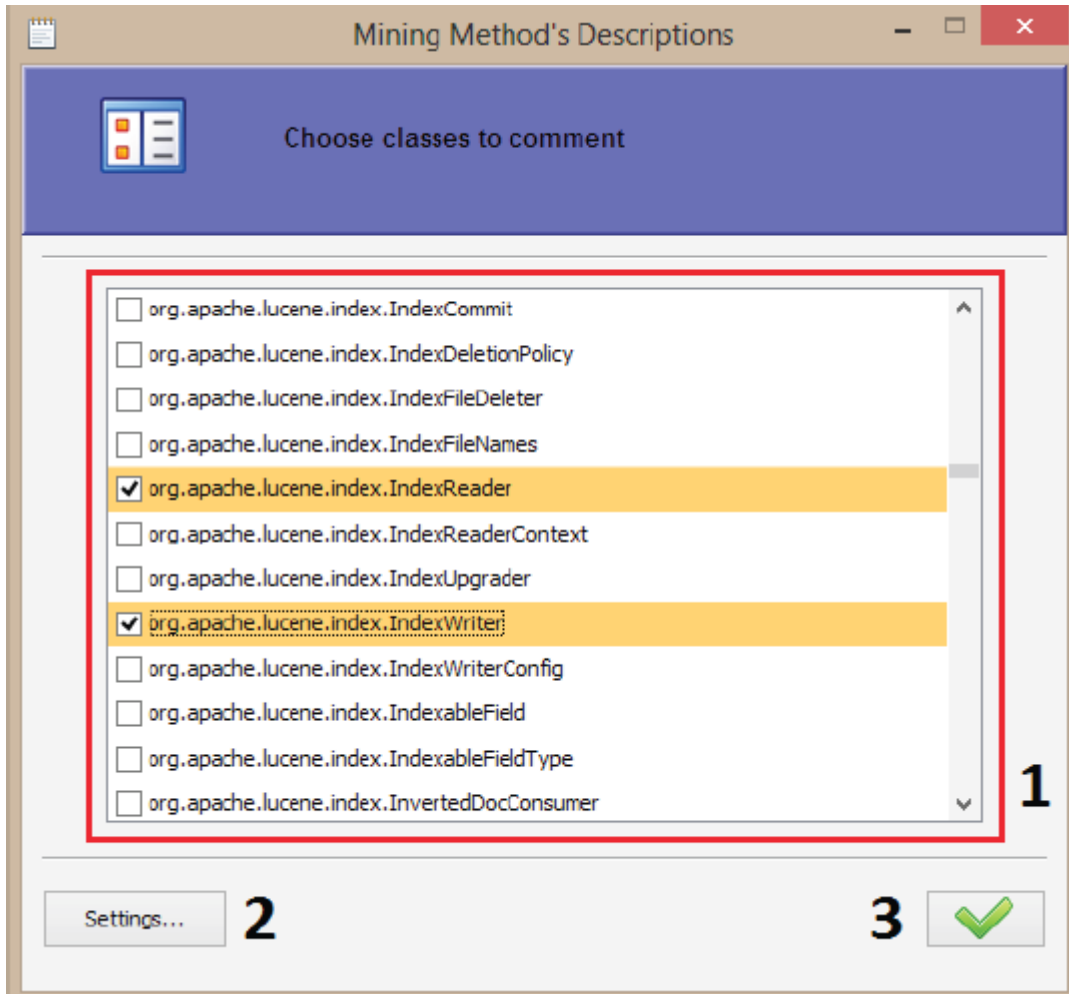
Figure 3



4. The first screen: chosen of classes

As soon as you have chosen “Mining Method's Decriptions” from context menu, the screen in [Figure 4](#) appears.

Figure 4



The screen's main panel (point 1, [Figure 4](#)) allows you to choose some classes for which you want to search descriptions: in particular, there are only classes that have one method at least; for every class you have chosen, the plugin is going to search descriptions about each method (if they exist). In our example we have chosen not only *org.apache.index.IndexReader* (the real class for which we hope to search descriptions) but also other classes to show some features in next paragraphs.

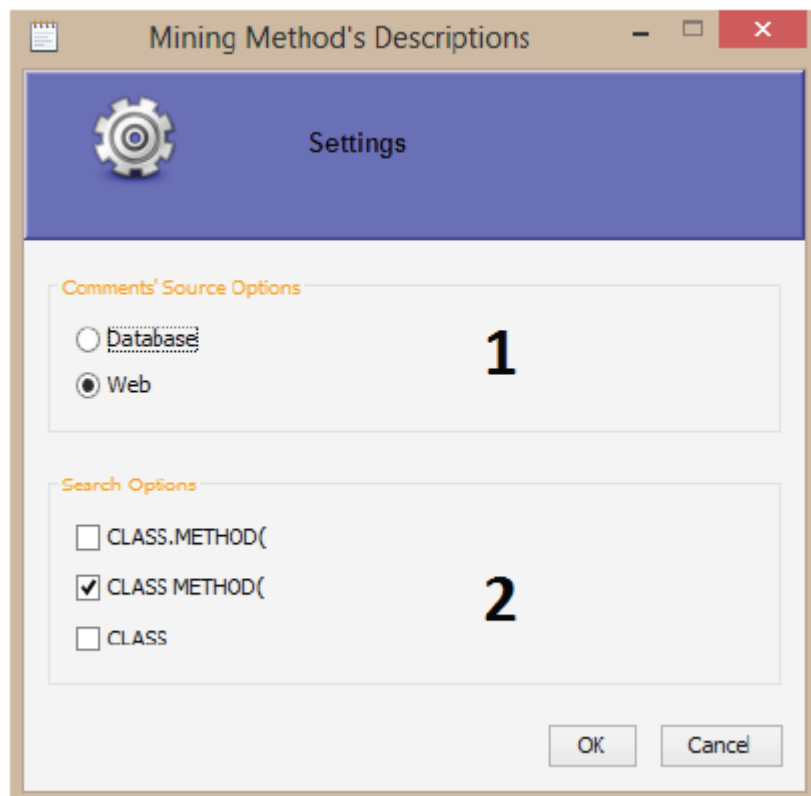
After that, you can go ahead by pushing confirm button (point 3, [Figure 4](#)): obviously, if you don't choose any class or don't configure research correctly, an alert will be shown and you won't be able to go ahead.

However, before pushing confirm button, you should push settings button (point 2), so access to research's configuration.

5. Settings: research's configuration

The Settings' main screen is in [Figure 5](#).

Figure 5



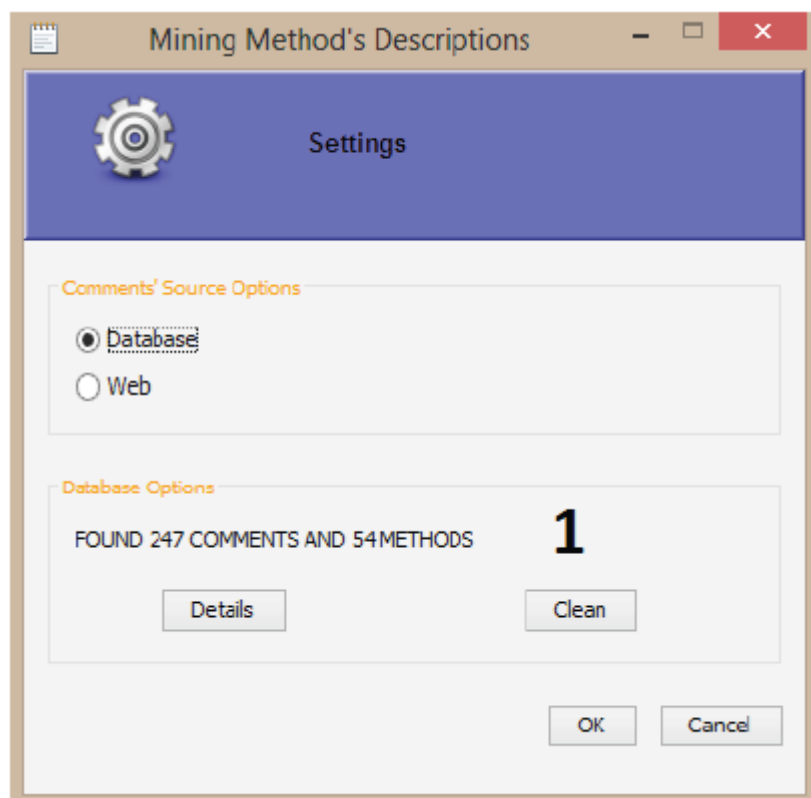
In the first panel (point 1, [Figure 5](#)), you can choose among two possible “sources” of descriptions:

- database (offline research)
- web (online research)

If you choose online research, in the second panel (point 2, [Figure 5](#)) you'll be able to choose among three available search mode (described with a small tooltip): besides, you'll be able to choose more than one search mode at same time.

If you choose offline research, a second panel will be shown in this way (point 1, [Figure 6](#))

Figure 6

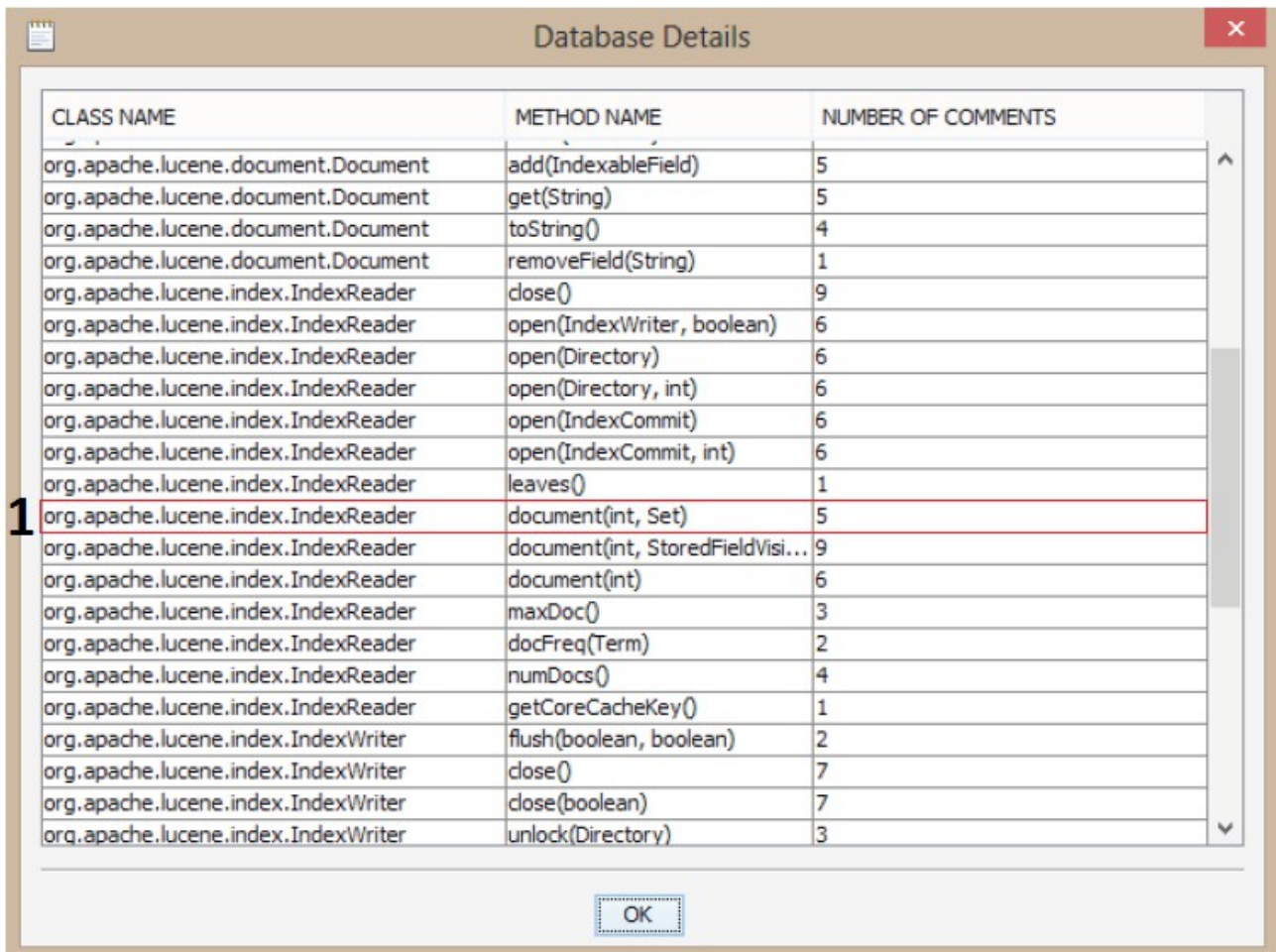


There are:

- number of available descriptions (or comments) in database
- number of project's methods for which descriptions exist in database
- “Clean” button to clean database
- “Details” button to show database's details

If you push “Details” button, the dialog in [Figure 7](#) will be opened.

Figure 7



CLASS NAME	METHOD NAME	NUMBER OF COMMENTS
org.apache.lucene.document.Document	add(IndexableField)	5
org.apache.lucene.document.Document	get(String)	5
org.apache.lucene.document.Document	toString()	4
org.apache.lucene.document.Document	removeField(String)	1
org.apache.lucene.index.IndexReader	close()	9
org.apache.lucene.index.IndexReader	open(IndexWriter, boolean)	6
org.apache.lucene.index.IndexReader	open(Directory)	6
org.apache.lucene.index.IndexReader	open(Directory, int)	6
org.apache.lucene.index.IndexReader	open(IndexCommit)	6
org.apache.lucene.index.IndexReader	open(IndexCommit, int)	6
org.apache.lucene.index.IndexReader	leaves()	1
1 org.apache.lucene.index.IndexReader	document(int, Set)	5
org.apache.lucene.index.IndexReader	document(int, StoredFieldVisi...	9
org.apache.lucene.index.IndexReader	document(int)	6
org.apache.lucene.index.IndexReader	maxDoc()	3
org.apache.lucene.index.IndexReader	docFreq(Term)	2
org.apache.lucene.index.IndexReader	numDocs()	4
org.apache.lucene.index.IndexReader	getCoreCacheKey()	1
org.apache.lucene.index.IndexWriter	flush(boolean, boolean)	2
org.apache.lucene.index.IndexWriter	close()	7
org.apache.lucene.index.IndexWriter	close(boolean)	7
org.apache.lucene.index.IndexWriter	unlock(Directory)	3

OK

In the dialog, there's a table where you can see a summary of database content: number of stored descriptions for each method. Obviously, in database there are only descriptions of methods for which the plugin has already made one online

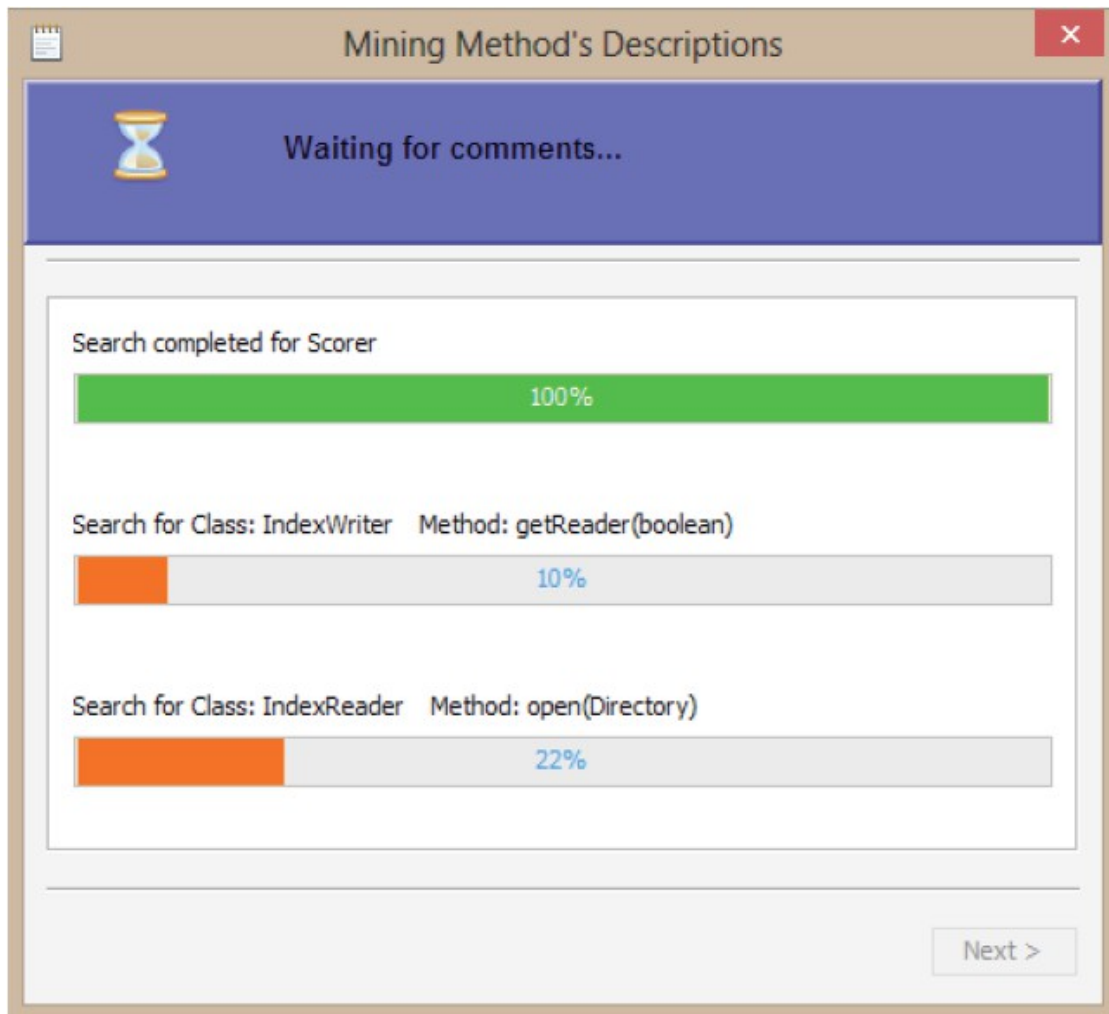
research at least.

The purpose of that table is: if you see the method, which you are interested in (in our example *document(int)*, point 1, [Figure 7](#)), in table, you'll be able to choose offline research and consequently will reduce research time. Otherwise you must choose online research.

6. Search and load decriptions

As soon as you finish configuration, you back to “Confirm” button (point 3, [Figure 4](#)) and start research. The frame in [Figure 8](#) appears.

Figure 8

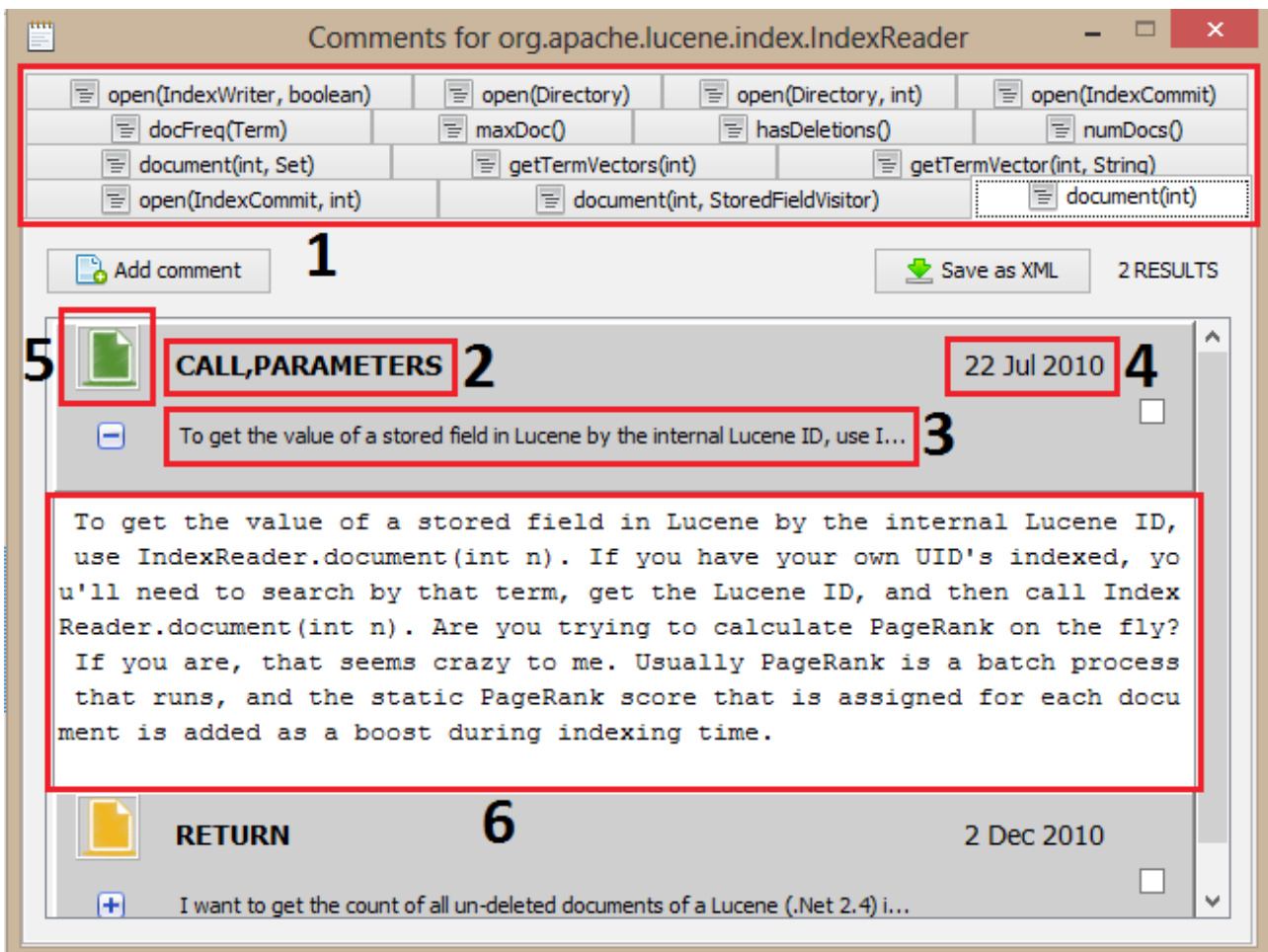


A progress bar that shows how far the research has progressed (as number of methods analyzed on number of total methods) is generated for each chosen class. The end of each research is signaled by acoustic signal and by different bar colour (becomes green). When all research have been finished, the “Next >” button will be activated and you'll be able to push it and view descriptions.

7. Chosen of descriptions

A frame (like one in [Figure 9](#)) is generated for each class for which one description has been found at least.

Figure 9



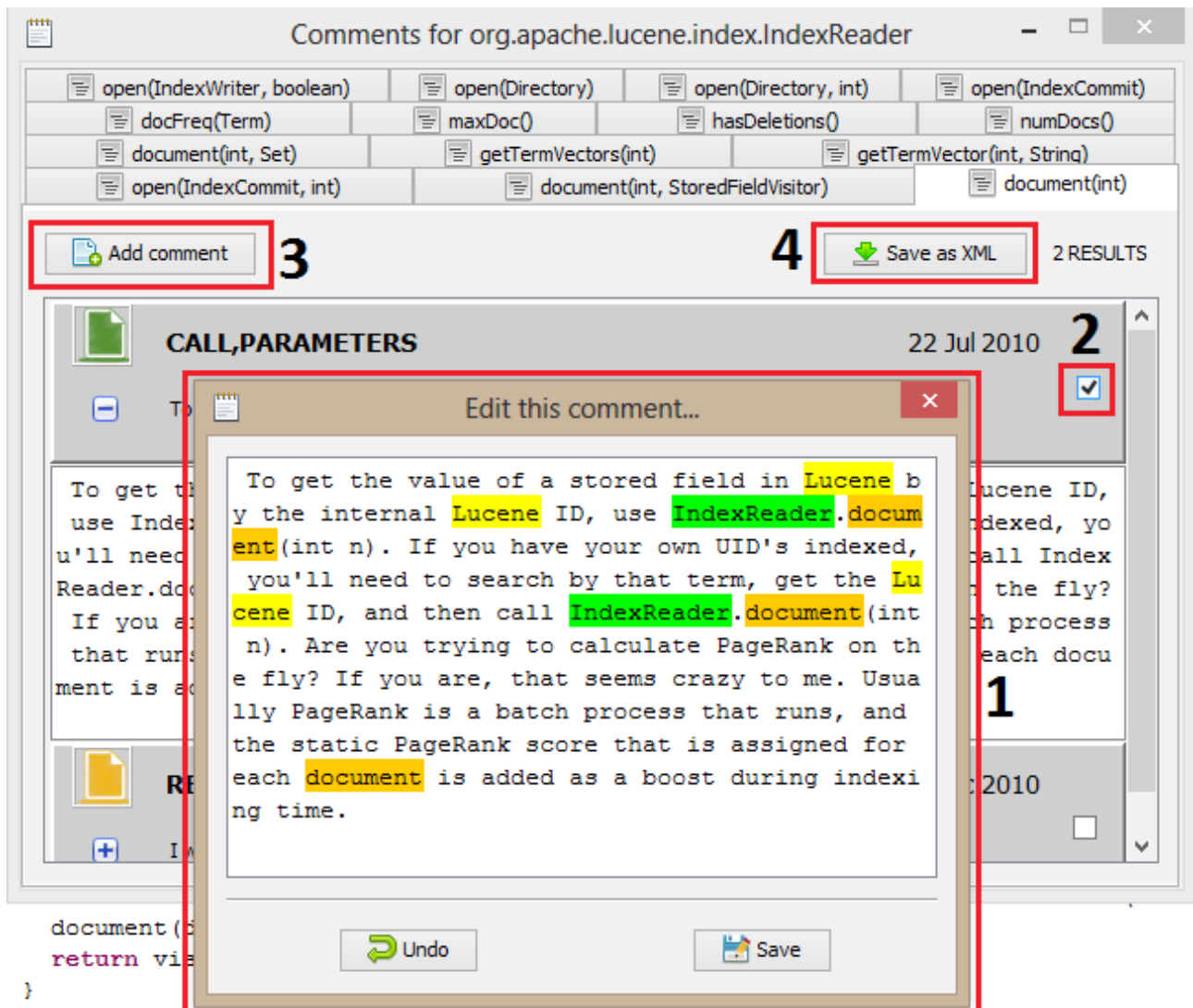
There's a tab (point 1, [Figure 9](#)) dedicated to each method of class for which the number of found descriptions is greater than zero. In a tab there are expandable panels: they are as many as descriptions found for the method. Every panel is

composed by description types (point 2, [Figure 9](#)), a description's preview (point 3, [Figure 9](#)), the date of description (point 4, [Figure 9](#)) and an icon (point 5, [Figure 9](#)) green if the description has totaled a score greater than one, yellow otherwise. If you expand the panel, you'll be able to show the whole found description (point 6, [Figure 9](#)).

Besides if you click on the whole description, a new frame (point 1, [Figure 10](#)), with the text of the description, will be shown and you'll be able to edit the description and save changes. In the text some key-words are highlighted:

- method name (orange)
- system (or project) name (yellow)
- class name (green)

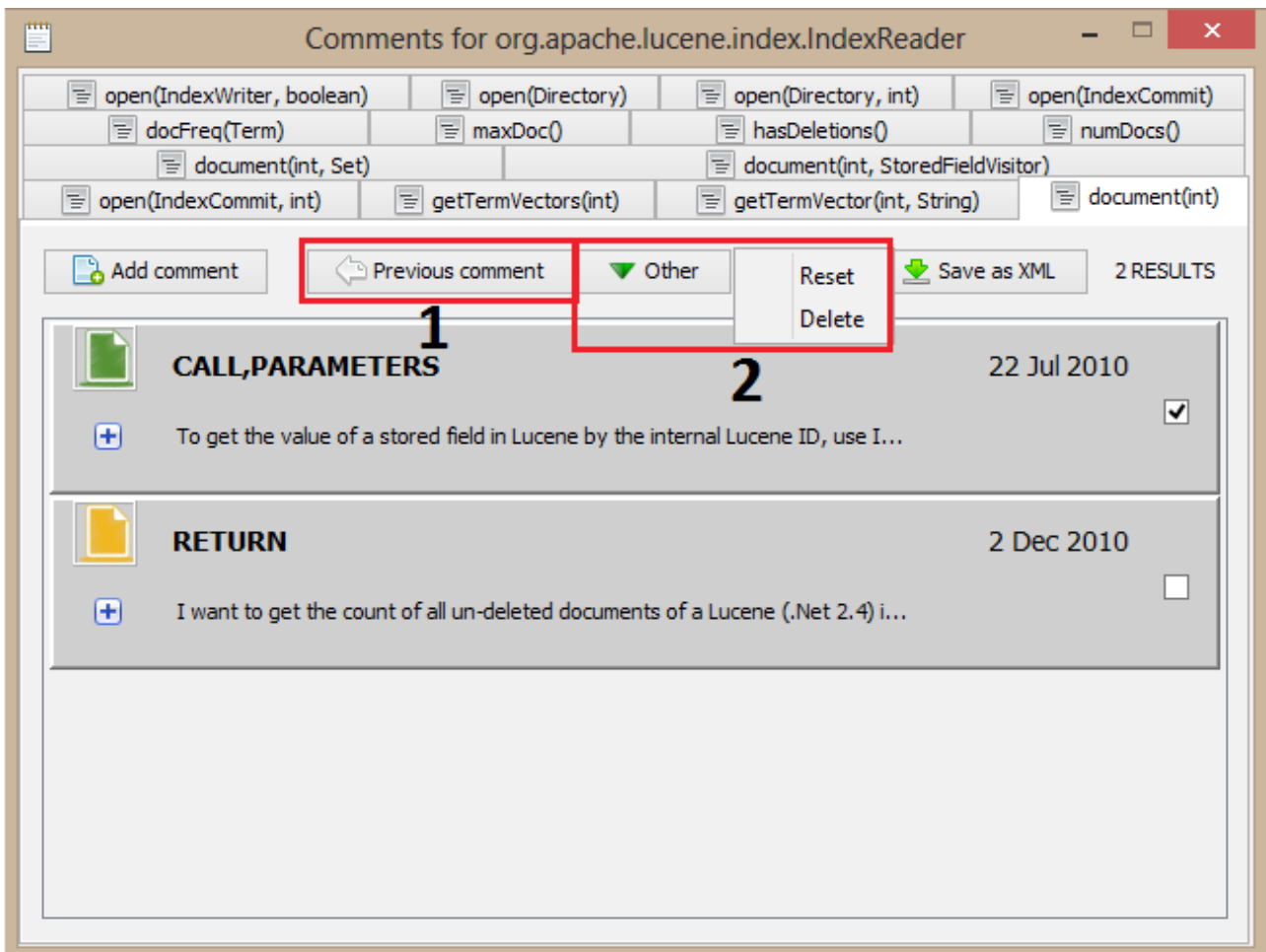
Figure 10



You can select (point 2, [Figure 10](#)) one or more description to build a Javadoc comment for a specific method: after selecting them, you have to push “Add Comment” button (point 3, [Figure 10](#)) to add a new comment (the comment is really added when you close the main frame). Besides there's a particular button (point 4, [Figure 10](#)), “Save as XML”, that allows you to save the selected descriptions in an XML file (useful for validation, etc.): this file is available from “XMLPlugin” directory, automatically created within the Java Project.

As soon as a comment has been added, “Previous comment” button appears (point 1, [Figure 11](#)): instead “Other”(point 2, [Figure 11](#)) button is shown when you select a description.

Figure 11



By pushing “Previous comment”, you can modify or delete previous added comment. By pushing “Other” you can delete (by “Delete”) or reset (from modified to original text, by “Reset”) the selected descriptions.

8. Add as Javadoc comments

In the previous paragraph, we build a comment for *document(int)* method of *IndexReader* class (Figures 9-11), during explanation of plugin's features. When you close the frame related to *IndexReader* class, the built Javadoc comment is added to source code (point 1, Figure 12). If you have chosen comments for other methods of *IndexReader*, they'll be added at the same time.

Figure 12

