Current Topics in Software Engineering

Software Evolution

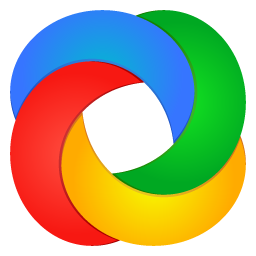
*Testing and Refactoring*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ShareX**

*Sharing has never been easier*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Karl-Heinz Pirolt, 0660277

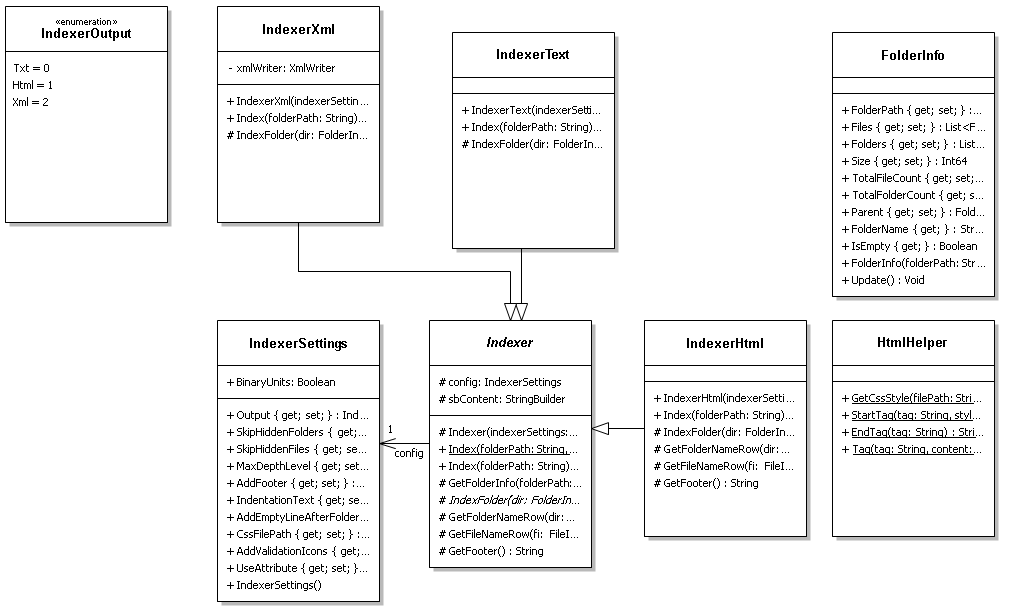
Jennifer Simonjan, 0960452

**Testing and refactoring of the ShareX.IndexerLib**

First of all, we wrote UnitTests for the core functionality of the IndexerLib:

* TestTotalFileCountOfIndexer
* TestTotalFolderCountOfIndexer
* Todo: test printindexer

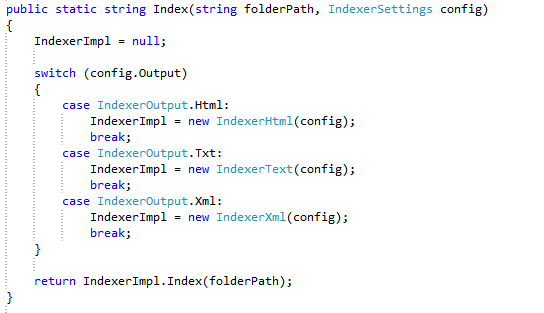
Afterwards we started with refactoring, where we restructured the whole IndexerLib project and also introduced some patterns. Figure 1 shows the class diagram of the IndexerLib before we did any refactoring. As the class diagram shows, there is neither an interface nor a clear structure.



**Figure 1: Class diagram of the ShareX.IndexerLib before refactoring**

After finishing the unit tests, we started refactoring with a code clean-up of the HTML helper class. Then, we started to refactor Indexer.cs, whereby we separated the indexer from the output format generation. Afterwards, a directory structure was introduced which subdivides the project into Helpers, Indexer, Models and Resources. The directory structure noticeably eases the understandability of the project.

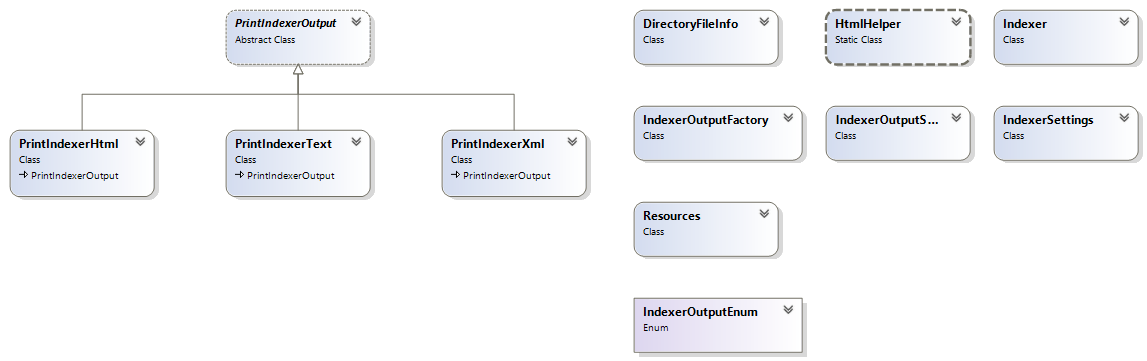
The SOLID principle is violated in various ways by the IndexerLib. An example is the violation of the open/close principle whenever a new Indexer is created:



We solved this problem by introducing the abstract class PrintIndexerOutput, which methods are implemented by PrintIndexerHtml, PrintIndexerText and PrintIndexerXml (see figure 2). This enables extension of the PrintIndexers to further types without any changes to the code. Further, the factory class IndexerOutputFactory was introduced to enable the creation of a PrintIndexerOutput instance depending on the IndexerOutputEnum. We then added a singleton to IndexerOutputFactory.cs to ensure better accessability.

Todo: Doku a bisl ausführlicher (jenny is zu doof um de zu schreiben weil de zu doof is um alle patterns zu kennen u zu beschreiben eieiei)

Figure 2 shows the class diagram of the IndexerLib after our refactorings.



**Figure 2: Class diagram of the ShareX.IndexerLib after refactoring**

**Testing and refactoring of the ShareX main project**

First of all we introduced a directory structure in the main project. We extracted the Hotkey, the Task, the Upload and the WatchFolder components to separate folders.

TODO: write tests