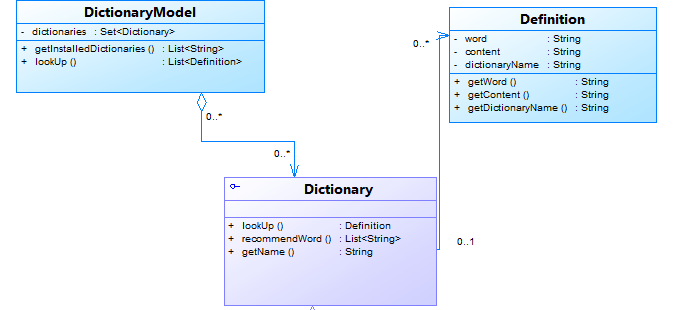
yMEGADICT

# DICTIONARY MODEL

Package com.megadict.model



**Dictionary Interface** – common interface for all format dictionaries (currently we are only supporting DICT format, but StarDict is considering).

* **lookUp(String word)** : look up a word and retrieve it’s Definition.
* **recommendWord(String word)**: return a list of words (include itself if possible) that looks similar to the argument.
* **getName()**: returns the name of the Dictionary.

**DictionaryModel** – a planned model for Megadict.

* **getInstalledDictionaries()**: return a list of installed dicitonaries (only returns name).
* **lookUp(String word):** returns a list of Definition(s) after calling all lookUp methods on installed dictionaries.

**Definition** – definition of a word

* **getWord()**: returns the (head)word of the definition.
* **getContent():** returns the content of the definition. Currently it returns the raw text retreive from .dict file.
* **getDictionaryName():** returns the name of dictionary it belongs to.

# DICT Implementation

Usage:

Code snippet below demonstrates how to use DICTDictionary:

String indexFilePath = "C:/test/dict.index";

String dictFilePath = "C:/test/dict.dict";

File indexRawFile = **new** File(indexFilePath);

File dictRawFile = **new** File(dictFilePath);

IndexFile indexFile = **new** IndexFile(indexRawFile);

// Or

// IndexFile indexFile = new IndexFile(indexFilePath);

DictionaryFile dictionaryFile = **new** BufferedDictionaryFile(dictRawFile);

// Or

// DictionaryFile dictionaryFile = new BufferedDictionaryFile(dictFilePath);

Dictionary dictionary = **new** DICTDictionary(indexFile, dictionaryFile);

// Get the name

System.out.println(dictionary.getName());

// Look up a word

Definition def = dictionary.lookUp("hello");

System.out.println(def.getContent());

// Get similar words

List<String> similarWords = dictionary.recommendWord("hello w");

System.out.println(similarWords);

## New benefits from changes:

- By using IndexFile and DictionaryFile, it is now impossible to specify wrong file to constructors.

String indexFilePath = "C:/test/dict.index";

String dictFilePath = "C:/test/dict.dict";

// WRONG!!!

// Prior revison #44.

// The constructor is ambiguous. So instead of using the right one

Dictionary rightWay = **new** DICTDictionary(indexFilePath, dictFilePath);

// Client can mistakenly end up with this one below (which I've made

// several times)

Dictionary ambiguous = **new** DICTDictionary(dictFilePath, indexFilePath);

// Since revision #45

IndexFile indexFile = IndexFile.*makeFile*(indexFilePath);

DictionaryFile dictionaryFile = DictionaryFile.*makeBufferedFile*(dictFilePath);

// GOOD :)

Dictionary better = **new** DICTDictionary(indexFile, dictionaryFile);

// YEILDS ERROR AT COMPILE TIME

Dictionary yeildError = **new** DICTDictionary(dictionaryFile, indexFile);

Dictionary dictionary =

**new** DICTDictionary.Builder(indexFile, dictFile).enableSplittingIndexFile().build();

- Constructing a DICTDictionary is more configurable. Client can specify which implementation of reading file. Good for unit testing and runtime switching.

IndexFile bufferedIndex = IndexFile.*makeBufferedFile*(indexRawFile);

IndexFile randomIndex = IndexFile.*makeRandomAccessFile*(indexRawFile);

IndexFile cachedIndex = IndexFile.*makeCachedFile*(indexRawFile);

DictionaryFile bufferedDict = DictionaryFile.*makeBufferedFile*(dictRawFile);

DictionaryFile randomAccessDict =

DictionaryFile.*makeRandomAccessFile*(dictRawFile);

DictionaryFile gzipDict = DictionaryFile.*makeGZipFile*(dictRawFile);

// Combo #1

Dictionary bufferedAndRandom =

**new** DICTDictionary(bufferedIndex, randomAccessDict);

// Combo #2

Dictionary cachedAndGzip = **new** DICTDictionary(cachedIndex, gzipDict);

- Adding new implementation of reading file will not impact DICTDictionary implementation.

## Design

