API Documentation

API Documentation

September 1, 2012

Contents

1 Package pySanaIndeksi

1.1 Modules

- **Support** *(Section 2, p. 6)*
  - **DataHandling** *(Section 3, p. 7)*
  - **DataHandlingUnitTest** *(Section 4, p. 9)*
  - **LinkedList** *(Section 5, p. 11)*
  - **LinkedListUnitTest** *(Section 6, p. 14)*
- **Trees** *(Section 7, p. 16)*
  - **PartialTree** *(Section 8, p. 17)*
  - **PartialTreeUnitTest** *(Section 9, p. 19)*
  - **RedBlack** *(Section 10, p. 21)*
  - **RedBlackUnitTest** *(Section 11, p. 25)*
  - **Tree** *(Section 12, p. 27)*
  - **TreeUnitTest** *(Section 13, p. 29)*
  - **Trie** *(Section 14, p. 31)*
  - **TrieUnitTest** *(Section 15, p. 37)*
- **WordIndex** *(Section 16, p. 39)*
  - **Searcher** *(Section 17, p. 40)*
  - **SearcherUnitTest** *(Section 18, p. 43)*
  - **WordReader** *(Section 19, p. 45)*
  - **WordReaderUnitTest** *(Section 20, p. 49)*
  - **pyUnitTest** *(Section 21, p. 52)*
  - **pysanaindeksi** *(Section 22, p. 53)*
  - **testFindMethod** *(Section 23, p. 55)*
  - **timing** *(Section 24, p. 56)*

1.2 Variables

Name	Description
__package__	**Value:** `None`

2 Package pySanaIndeksi.Support

Date: $Sep 1, 2012 5:42:33 PM$

Author: patrik

2.1 Modules

- **DataHandling** *(Section 3, p. 7)*
- **DataHandlingUnitTest** *(Section 4, p. 9)*
- **LinkedList** *(Section 5, p. 11)*
- **LinkedListUnitTest** *(Section 6, p. 14)*

2.2 Variables

Name	Description
__package__	**Value:** `None`

3 Module pySanaIndeksi.Support.DataHandling

Author: Patrik Ahvenainen

3.1 Functions

openFile(*path*, *io*=`'r'`)

Opens the file in the given path in read mode if param io is not given

Parameters

`io:` string 'r' for read and string 'w' for write

(type=string)

Return Value

file handle

closeFile(*filehandle*)

Exists only to complement openFile syntax

Parameters

`filehandle:` filehandle to be closed

getFileNames(*mypath*, *path*=`True`)

Returns paths to all files in the given path. Returns an empty list on error, does not raise exceptions.

Parameters

`mypath:` The directory whose files are returned

(type=string, name of a directory)

`path:` value False means you don't want filenames listed with the given path

(type=boolean)

Return Value

a list of files in the directory (if any)

printFileList(*mypath*, *noPath*=`True`)

Prints a list containing all files in the given path. File list is obtained with getFileNames. Returns also the list of files returned by getFileNames.

Parameters

`noPath:` value True means you don't want filenames listed with the given path

(type=boolean)

Return Value

a list of the files returned by getFileNames

3.2 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.Support'`

4 Module pySanaIndeksi.Support.DataHandlingUnitTest

Date: $10.8.2012 11:55:42$

Author: Patrik Ahvenainen

4.1 Functions

suite()

4.2 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.Support'`

4.3 Class PyDataHandlingTestCases

object
unittest.TestCase
pySanaIndeksi.Support.DataHandlingUnitTest.PyDataHandlingTestCases

4.3.1 Methods

testRaisesError(*self*)

testOpensFile(*self*)

testFileList(*self*)

Inherited from unittest.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), assertAlmostEqual(), assertAlmostEquals(), assertEqual(), assertEquals(), assertFalse(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertRaises(), assertTrue(), assert_(), countTestCases(), debug(), defaultTestResult(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), setUp(), shortDescription(), tearDown()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

4.3.2 Properties

Name	**Description**
Inherited from object	
__class__	

5 Module pySanaIndeksi.Support.LinkedList

Date: $6.8.2012 13:26:52$

Author: Patrik Ahvenainen

5.1 Variables

Name	Description
__package__	**Value:** `None`

5.2 Class LinkedList

object ┐
pySanaIndeksi.Support.LinkedList.LinkedList

A simple doubly linked list class that provides three public methods; one for adding an item at the end of the list, one for retrieving the last item in the list and one for retrieving a list of all of the values stored in that list. Includes a counter for the number of items in the list. The list can also be cleared. Nodes are instances of class LinkedListNode.

properties: count: Returns the number of items in this list (number of nodes). public methods: clear(): Clears the list addLast(value):Adds the value to the end of the list. removeLast(): Removes the last value in the list and returns its value values(): Retrieve a list of all values in the linked list.

5.2.1 Methods

__init__(*self*)

Can only be used to create an empty list

Overrides: object.__init__

addLast(*self*, *value*)

Add one value to the end of the list

Parameters

`value`: the value to be added to the list

(type=any valid python object)

clear(*self*)

Clearing removes all the items from the list

removeLast(*self*)

Removes the last value in the list and returns its value

Return Value

returns the last value from the list

values(*self*)

Return a list containing all values in this linked list

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

5.2.2 Properties

Name	Description
count	The number of items in this list (number of nodes)
Inherited from object __class__	

5.3 Class LinkedListNode

```
object ─┐
        │
        pySanaIndeksi.Support.LinkedList.LinkedListNode
```

The nodes are bidirectional: each node has a reference to its parent and to its child. If the item is added to a non-empty list (child or parent given) the corresponding child or parent attribute is modified in the existing node (i.e. giving the node a child will make that child have the new node as its parent).

5.3.1 Methods

__init__(*self*, *value*, *parent*=`None`, *child*=`None`)

A default node has no parent or child, but a value must be given

Overrides: object.__init__

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

5.3.2 Properties

Name	Description
Inherited from object	
__class__	

6 Module pySanaIndeksi.Support.LinkedListUnitTest

Date: $10.8.2012 11:44:18$

Author: Patrik Ahvenainen

6.1 Functions

suite()

6.2 Variables

Name	Description
lastVal	**Value:** `4`
linkedListVals	**Value:** `[1, 2, 3, 4]`
linkedListAfter	**Value:** `[1, 2, 3]`
__package__	**Value:** `'pySanaIndeksi.Support'`

6.3 Class PyLinkedListTestCases

```
object ─┐
        │
unittest.TestCase ─┐
                   │
                   pySanaIndeksi.Support.LinkedListUnitTest.PyLinkedListTestCases
```

6.3.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

tearDown(*self*)

Hook method for deconstructing the test fixture after testing it.

Overrides: unittest.TestCase.tearDown extit(inherited documentation)

testLinkedList(*self*)

Test adding multiple values to a linked list Tests, addLast(), values(), removeLast(), clear() and count

Inherited from unittest.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), assertAlmostEqual(), assertAlmostEquals(), assertEqual(), assertEquals(), assertFalse(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertRaises(), assertTrue(), assert_(), countTestCases(), debug(), defaultTestResult(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), shortDescription()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

6.3.2 Properties

Name	Description
Inherited from object	
__class__	

7 Package pySanaIndeksi.Trees

Date: $Sep 1, 2012 5:39:18 PM$

Author: patrik

7.1 Modules

- **PartialTree** *(Section 8, p. 17)*
- **PartialTreeUnitTest** *(Section 9, p. 19)*
- **RedBlack** *(Section 10, p. 21)*
- **RedBlackUnitTest** *(Section 11, p. 25)*
- **Tree** *(Section 12, p. 27)*
- **TreeUnitTest** *(Section 13, p. 29)*
- **Trie** *(Section 14, p. 31)*
- **TrieUnitTest** *(Section 15, p. 37)*

7.2 Variables

Name	Description
__package__	**Value:** `None`

8 Module pySanaIndeksi.Trees.PartialTree

Date: $13.8.2012 14:01:02$

Author: Patrik Ahvenainen

8.1 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.Trees'`

8.2 Class PartialTree

```
object ─┐
pySanaIndeksi.Trees.Tree.Tree ─┐
                               pySanaIndeksi.Trees.PartialTree.PartialTree
```

Known Subclasses: pySanaIndeksi.Trees.Trie.Trie

8.2.1 Methods

findPartial(*self*, *key*, *output*=`'list'`)

Finds the key from the tree. Return type defined by param output

Parameters

`key`: The key to be found

`output`: the type of output desired

(type=string)

Return Value

- output == 'boolean': 1: A boolean value indicating whether the file was found
- output == 'count': 1: Number of found instances 2: Number of lines where the word was found
- output == 'full': 1: A list of positions where this word was found 2: Number of found instances 3: Number of lines where the word was found
- output == 'list': (default) 1: A list of positions where this word was found

Inherited from pySanaIndeksi.Trees.Tree.Tree(Section 12.2)

add(), addFromReader(), clear(), find()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

8.2.2 Properties

Name	Description
Inherited from pySanaIndeksi.Trees.Tree.Tree (Section 12.2) type, wordCount	
Inherited from object __class__	

8.2.3 Class Variables

Name	Description
__metaclass__	This is an abstract class. Classes that inherit from this class should have implementation for adding an item to a tree and finding an item from a tree. This class is not strictly speaking necessary but it makes sure your tree classes are working properly. Guarantees that the following properties and methods have been defined: properties: wordCount: number of words added to the tree (not number of nodes) type: finds 'exact' or 'partial' matches for words public methods: self.add(key, value): Adds one value with the given key to the tree self.addFromReader(): Adds words from own reader if possible self.clear(): Removes all words from the tree self.find(key): Returns the hits when searching for key **Value:** `ABCMeta`
__abstractmethods__	**Value:** `frozenset(['add', 'addFromReader', 'clear', 'find', 'find...`

9 Module pySanaIndeksi.Trees.PartialTreeUnitTest

Date: $13.8.2012 14:52:35$

Author: Patrik Ahvenainen

9.1 Functions

suite()

9.2 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.Trees'`

9.3 Class PyPartialTreeTestCases

```
object ─┐
        │
unittest.TestCase ─┐
                   │
                   pySanaIndeksi.Trees.PartialTreeUnitTest.PyPartialTreeTestCases
```

9.3.1 Methods

testCannotBeInstantiated(*self*)

This abstract class should not be possible to instantiate

Inherited from unittest.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), assertAlmostEqual(), assertAlmostEquals(), assertEqual(), assertEquals(), assertFalse(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertRaises(), assertTrue(), assert_(), countTestCases(), debug(), defaultTestResult(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), setUp(), shortDescription(), tearDown()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

9.3.2 Properties

Name	Description
Inherited from object	
__class__	

10 Module pySanaIndeksi.Trees.RedBlack

Date: $31.7.2012 17:35:35$

Author: Patrik Ahvenainen

10.1 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.Trees'`

10.2 Class RedBlack

object

pySanaIndeksi.Trees.Tree.Tree

pySanaIndeksi.Trees.RedBlack.RedBlack

A Trie-type tree that can hold words containing characters. The WordReader is optional, it can be passed to facilitate reading words from it.

The tree is initially empty. The leaves are always empty black nodes.

All nodes are from class RedBlackNode.

These properties and methods are defined by Tree class: properties: type: type of searches stored; value is 'exact' wordCount: number of words added to the tree (not number of nodes) public methods: add(key, value): Adds one value with the given key to the tree addFromReader(): Adds words from own reader if possible clear(): Removes all words from the tree find(key): Returns the hits when searching for key private methods: _binaryInsert(self, node): Insert this value and when done, restore RB-tree _internalFind(word): Find the word in the tree, if it exists _leftRotate(node): Do a left-rotate for this node _restoreProperties(node): Starting from node, restore RB-tree _rightRotate(node): Do a right-rotate for this node

10.2.1 Methods

__init__(*self*, *lukija*=`None`)

Only empty trees can be created. WordReader object is optional.

Overrides: object.__init__

add(*self*, *key*, *value*)

Adds a new word to the tree. Adding is done with binaryInsert, which calls for a function that restores the tree to red and black if the addition of this word damaged those properties.

Parameters

- `key:` the key to the value to be added
- `value:` the value to be added

Overrides: pySanaIndeksi.Trees.Tree.Tree.add

addFromReader(*self*, *wordCount*=`None`)

Adds all the words in the WordReader object to this tree

Parameters

- `wordCount:` maximum number of words to be added

Overrides: pySanaIndeksi.Trees.Tree.Tree.addFromReader

clear(*self*)

Gets rid of all the nodes in the tree (if any)

Overrides: pySanaIndeksi.Trees.Tree.Tree.clear

find(*self*, *key*, *output*=`'full'`, *sanitized*=`False`)

Calls an internal function which does the actual finding.

Parameters

- `key:` the key to be found
- `output:` the type of output desired, options defined in Tree class *(type=string)*
- `sanitized:` the word is sanitized unless this param is True *(type=boolean)*

Return Value

hits returned in the format defined by output

Overrides: pySanaIndeksi.Trees.Tree.Tree.find

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

10.2.2 Properties

Name	Description
type	
wordCount	
Inherited from object __class__	

10.2.3 Class Variables

Name	Description
__abstractmethods__	**Value:** `frozenset([])`

10.3 Class RedBlackNode

object ─┐
pySanaIndeksi.Trees.RedBlack.RedBlackNode

Contains the information of one branch. The node knows its children (le[ft] and ri[ght]), its pa[rent], its color, its key and its val[ue]. It also knows if it's empty (end-of-list flag). Note that nodes cannot store empty values.

Nodes know their family through the following methods methods: grandpa(): return the grandpa node or empty node sibling(): return the sibling node uncle(): return the uncle node or empty node updateNode(val): Add the val(ue) to this node

10.3.1 Methods

__init__(*self*, *str*=`''`, *val*=`None`, *pa*=`None`, *red*=`True`)

The node can be empty upon creation (if val is not given)

Overrides: object.__init__

__str__(*self*)

String representation: return key as string

Overrides: object.__str__

__repr__(*self*)

String representation: return key as string

Overrides: object.__repr__

grandpa(*self*)

Return the parent of a parent or empty node

sibling(*self*)

Return the sibling of the node (it must exist)

uncle(*self*)

Return the sibling of the parent or empty node

updateNode(*self*, *value*)

Add the value to this node

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

10.3.2 Properties

Name	Description
Inherited from object	
__class__	

11 Module pySanaIndeksi.Trees.RedBlackUnitTest

Date: $10.8.2012 11:47:25$

Author: Patrik Ahvenainen

11.1 Functions

suite()

11.2 Variables

Name	Description
WordsToAdd	**Value:** `[('ww3fwG', 99, 1), ('Sana', 3, 2), ('ed', 2222, 1), ('Ta...`
MultiWordAdd	**Value:** `[('c', 20, 3), ('a', 1, 1), ('c', 23, 1), ('b', 2, 1), ('...`
MultiWordFindA	**Value:** `[(1, 1), (3, 1), (1, 2), (3, 2)]`
MultiWordFindB	**Value:** `[(2, 1), (2, 2)]`

11.3 Class PyRedBlackTestCases

```
object ─┐
        │
unittest.TestCase ─┐
                   │
                   pySanaIndeksi.Trees.RedBlackUnitTest.PyRedBlackTestCases
```

11.3.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

tearDown(*self*)

Hook method for deconstructing the test fixture after testing it.

Overrides: unittest.TestCase.tearDown extit(inherited documentation)

testSimpleAddFind(*self*)

Add some objects to Red Black tree and see if you can find them

testMultiWordFind(*self*)

Tests that multiple instances of a word are found correctly

testWordCounter(*self*)

Tests that both the reader and the tree can count the words

Inherited from unittest.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), assertAlmostEqual(), assertAlmostEquals(), assertEqual(), assertEquals(), assertFalse(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertRaises(), assertTrue(), assert_(), countTestCases(), debug(), defaultTestResult(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), shortDescription()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

11.3.2 Properties

Name	Description
Inherited from object	
__class__	

12 Module pySanaIndeksi.Trees.Tree

Date: $31.7.2012 18:12:31$

Author: Patrik Ahvenainen

12.1 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.Trees'`

12.2 Class Tree

```
object ─┐
        │
        pySanaIndeksi.Trees.Tree.Tree
```

Known Subclasses: pySanaIndeksi.Trees.PartialTree.PartialTree, pySanaIndeksi.Trees.RedBlack.RedBla

12.2.1 Methods

add(*self*, *key*, *value*)

This method is used to add items to the tree. Each item contains a key which should be a string and an arbitrary value corresponding to that key.

Parameters

`key`: the key to the value to be added

`value`: the value to be added

addFromReader(*self*, *wordCount*=`None`)

This method should take the words from self.lukija and add them to the tree. If wordCount is given this method should not increase the number of additions to the tree over wordCount. Note that the actual number of unique words in the tree does not equal wordCount.

Parameters

`wordCount`: the maximum number of words added to the tree

clear(*self*)

This method is used to clear all the words from the tree.

find(*self*, *key*, *output*=`'list'`, *sanitized*=`False`)

Finds the key from the tree. Word is first sanitized unless sanitized is set to True.

Parameters

`key:`	the key for the value to be found
`output:`	the type of output desired *(type=boolean)*
`sanitized:`	if set to True words are assumed to be sanitized *(type=boolean)*

Return Value

- output == 'boolean': 1: A boolean value indicating whether the file was found
- output == 'count': 1: Number of found instances
- output == 'full': 1: A list of positions where this word was found 2: Number of found instances 3: Number of lines where the word was found
- output == 'list': (default) 1: A list of positions where this word was found

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

12.2.2 Properties

Name	Description
type	
wordCount	
Inherited from object __class__	

12.2.3 Class Variables

Name	Description
__abstractmethods__	**Value:** `frozenset(['add', 'addFromReader', 'clear', 'find', 'type...`

13 Module pySanaIndeksi.Trees.TreeUnitTest

Date: $10.8.2012 11:46:28$

Author: Patrik Ahvenainen

13.1 Functions

suite()

13.2 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.Trees'`

13.3 Class PyTreeTestCases

```
object ─┐
        │
unittest.TestCase ─┐
                   │
                   pySanaIndeksi.Trees.TreeUnitTest.PyTreeTestCases
```

13.3.1 Methods

testCannotBeInstantiated(*self*)

This abstract class should not be possible to instantiate

Inherited from unittest.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), assertAlmostEqual(), assertAlmostEquals(), assertEqual(), assertEquals(), assertFalse(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertRaises(), assertTrue(), assert_(), countTestCases(), debug(), defaultTestResult(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), setUp(), shortDescription(), tearDown()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

13.3.2 Properties

Name	**Description**
Inherited from object	
__class__	

14 Module pySanaIndeksi.Trees.Trie

Date: $31.7.2012 12:16:17$

Author: Patrik Ahvenainen

14.1 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.Trees'`

14.2 Class Trie

```
object ─┐
        │
pySanaIndeksi.Trees.Tree.Tree ─┐
                               │
pySanaIndeksi.Trees.PartialTree.PartialTree ─┐
                                             │
                                  pySanaIndeksi.Trees.Trie.Trie
```

```
A Trie-type tree that can hold words containing alphanumerals, hyphens and
aposthrophes. The WordReader module handles in reality the the word
sanitizing.

A trie instance is tied to its WordReader object: it gets the size of the
child node list from it and the mapping of characters to indices in that
list.

The value of the trie node is determined by traversing from root to the
node. Each trie node holds two lists: one for positions where an exact match
is found (if any) and one for all words that start with those letters. That
it way it is a PartialTree.

properties:
wordCount:  number of words added to the tree (not number of nodes)
type:       finds 'exact' or 'partial' matches for words according to type
public methods:
add(key, value):    Adds one value with the given key to the tree
clear():            Removes all words from the tree
find(key):          Returns the hits when searching for key
findPartial(key):   Finds hits for keywords starting with key
```

```
addFromReader():    Adds words from own reader if possible
printRandomRoute(): Prints a random route from root to leaf.
private methods:
_addNode(key, value): Recursively adds each letter in the key to the tree
                      (in correct place) and adds value to each node
_find(word):          Handles mainly the type of output and calls _findRe...
_findRecursive(word, charNo, type, node):
                      Traverses the tree to find (if possible) the word
```

14.2.1 Methods

__init__(*self*, *wordreader*)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

---

add(*self*, *key*, *value*)

Adds a new value to the tree. The addition is done via recursive addNode function. The value is stored using the key.

Parameters

`key`: the key to the value to be added

`value`: the value to be added

Overrides: pySanaIndeksi.Trees.Tree.Tree.add

---

addFromReader(*self*, *wordCount*=`None`)

Adds all the words in the WordReader object to this tree.

Parameters

`wordCount`: maximum number of words read from reader

(type=integer)

Overrides: pySanaIndeksi.Trees.Tree.Tree.addFromReader

---

clear(*self*)

Clears all words from the tree

Overrides: pySanaIndeksi.Trees.Tree.Tree.clear

find(*self*, *word*, *output*=`'full'`, *sanitized*=`False`)

Finds the word in this tree using intenal self._find method. Output options are listed in super class Tree in method find. The optional sanitized flag can be used if input is known to be sanitized already. Do not use it for non-sanitized input.

Parameters

`word:`	the word to be found *(type=string)*
`output:`	the type of output desired *(type=boolean)*
`sanitized:`	if set to True words are assumed to be sanitized *(type=boolean)*

Return Value

- output == 'boolean': 1: A boolean value indicating whether the file was found
- output == 'count': 1: Number of found instances
- output == 'full': 1: A list of positions where this word was found 2: Number of found instances 3: Number of lines where the word was found
- output == 'list': (default) 1: A list of positions where this word was found

Overrides: pySanaIndeksi.Trees.Tree.Tree.find

findPartial(*self*, *word*, *output*=`'full'`, *sanitized*=`False`)

Finds the keys beginning with word in this tree using intenal self._find method. Output options are listed in super class Tree in method find. The optional sanitized flag can be used if input is known to be sanitized already. Do not use it for non-sanitized input.

Parameters

`word:`	the word to be found *(type=string)*
`output:`	the type of output desired *(type=boolean)*
`sanitized:`	if set to True words are assumed to be sanitized *(type=boolean)*

Return Value

- output == 'boolean': 1: A boolean value indicating whether the file was found
- output == 'count': 1: Number of found instances 2: Number of lines where the word was found
- output == 'full': 1: A list of positions where this word was found 2: Number of found instances 3: Number of lines where the word was found
- output == 'list': (default) 1: A list of positions where this word was found

Overrides: pySanaIndeksi.Trees.PartialTree.PartialTree.findPartial

printRandomRoute(*self*)

Prints a random route from root to leaf.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

14.2.2 Properties

Name	Description
wordCount	number of words added to the tree (not number of nodes)
type	finds 'exact' or 'partial' matches for words according to type

continued on next page

Name	Description
Inherited from object __class__	

14.2.3 Class Variables

Name	Description
__abstractmethods__	**Value:** `frozenset([])`
Inherited from pySanaIndeksi.Trees.PartialTree.PartialTree (Section 8.2) __metaclass__	

14.3 Class TrieNode

```
object ─┐
        pySanaIndeksi.Trees.Trie.TrieNode
```

```
Contains the information of one node. It contains two lists of the positions
where the string corresponding to that node is found. One stores only the
positions of exact matches and the other all words that start with that
string (match).

Child maintenance is handled with a minimum list. A new TrieNode is given
the maximum amount of children (number of acceptable characters).

self.updateNode(value, exact):
    Adds the value to the value lists of that node. Exact-flag determines
    whether the value is added also to the list of exact matches.
```

14.3.1 Methods

__init__(*self*, *charMapSize*, *value*=`''`, *exact*=`False`)

Trie node contains two linked lists for its values: one is intended for exact matches (it is updated only with exact-flag raised). Upon creation, the minimum list type child list is created for that node.

Overrides: object.__init__

updateNode(*self*, *position*, *exact*)

Add position info for this object

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

14.3.2 Properties

Name	Description
Inherited from object	
__class__	

15 Module pySanaIndeksi.Trees.TrieUnitTest

Date: $10.8.2012 11:50:51$

Author: Patrik Ahvenainen

15.1 Functions

suite()

15.2 Variables

Name	Description
WordsToAdd	**Value:** `[('ww3fwG', 99, 1), ('Sana', 3, 2), ('ed', 2222, 1), ('Ta...`
MultiWordAdd	**Value:** `[('c', 20, 3), ('a', 1, 1), ('c', 23, 1), ('b', 2, 1), ('...`
MultiWordFindA	**Value:** `[(1, 1), (3, 1), (1, 2), (3, 2)]`
MultiWordFindB	**Value:** `[(2, 1), (2, 2)]`

15.3 Class PyTrieTestCases

```
object ─┐
        │
unittest.TestCase ─┐
                   │
                   pySanaIndeksi.Trees.TrieUnitTest.PyTrieTestCases
```

15.3.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

tearDown(*self*)

Hook method for deconstructing the test fixture after testing it.

Overrides: unittest.TestCase.tearDown extit(inherited documentation)

testSimpleAddFind(*self*)

Add some objects to Trie and see if you can find them

testMultiWordFind(*self*)

testWordCounter(*self*)

Tests that both the reader and the tree can count the words

Inherited from unittest.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), assertAlmostEqual(), assertAlmostEquals(), assertEqual(), assertEquals(), assertFalse(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertRaises(), assertTrue(), assert_(), countTestCases(), debug(), defaultTestResult(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), shortDescription()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

15.3.2 Properties

Name	Description
Inherited from object __class__	

16 Package pySanaIndeksi.WordIndex

Date: $Sep 1, 2012 5:40:41 PM$

Author: patrik

16.1 Modules

- **Searcher** *(Section 17, p. 40)*
- **SearcherUnitTest** *(Section 18, p. 43)*
- **WordReader** *(Section 19, p. 45)*
- **WordReaderUnitTest** *(Section 20, p. 49)*
- **pyUnitTest** *(Section 21, p. 52)*
- **pysanaindeksi** *(Section 22, p. 53)*
- **testFindMethod** *(Section 23, p. 55)*
- **timing** *(Section 24, p. 56)*

16.2 Variables

Name	Description
__package__	**Value:** `None`

17 Module pySanaIndeksi.WordIndex.Searcher

Date: $8.8.2012 12:32:57$

Author: Patrik Ahvenainen

17.1 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.WordIndex'`

17.2 Class Searcher

```
object ─┐
        pySanaIndeksi.WordIndex.Searcher.Searcher
```

```
This class provides a simple set-based searching methods. Searches comprise
of set operations between individual find-operations for each given word.

The individual word searches are done with the input object finder using its
find method for each word. Before word searches, the individual words are
sanitized with sanitizer or finder's sanitizer using the sanitize method. If
only a finder is given, its sanitizer holds the filenames of all texts that
have been indexed. In that case the search function can be used to print the
search results using file names and to show each line where a hit was found.

The search supports keywords AND, NOT, OR and XOR corresponding to set
operations intersection, difference, union and symmetric difference.

Basic operation:
searcher = Searcher(finderObject) # initialize
results = searcher.search("word1 AND word2") # Get the hits from search term

properties:
status:     Is 'ok' (string) if current search phrase is ok
maxHitPrint:Tells how many hits need to be found for lines not to be printed
public methods:
search():                   Completes the search and returns the hits
randomWord():               Fetches a random English word from Internet

private methods:
```

```
_categorize(words):        Categorizes words according to type
_checkString():            Checks that the seach phrase is good
_linesByFiles(results):    Returns the search results sorted by file
_prettyResults(results):   Prints the results using pretty formatting
_recursiveSearch(index):   Starting from given index goes through the search
                           string until it finds a closing paranthesis or
                           runs out of search string
_setOperate(left, right, operation):
                           Does the given set operation to the two sets
```

17.2.1 Methods

__init__(*self*, *finder*, *searchString*, *sanitizer*=`None`, *maxHitPrint*=`10`)

x.__init__(...) initializes x; see x.__class__.__doc__ for signature

Overrides: object.__init__ extit(inherited documentation)

search(*self*, *searchPhrase*=`None`, *printPretty*=`False`, *maxHitPrint*=`None`, *returnCount*=`False`)

Search the given search phrase using recursive search. If not given uses pre-existing search term, if it exists.

Parameters

`searchPhrase`:	Use this string as a search phrase
`printPretty`:	Print out hits as pretty as one can
`maxHitPrint`:	If you find many terms, print them out less pretty. This value tells you how many is 'many'.
`returnCount`:	If set to True, return only the number of hits

Return Value

Returns hits as a set of tuples

randomWord(*self*, *count*=`1`)

Returns a random English word fetched from randomword.setgetgo.com

Parameters

`count`: Set to larger than 1 to get more words

Return Value

One random word or a list of random words

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

17.2.2 Properties

Name	Description
maxHitPrint	Returns the maximum number of lines printed per file when search term is found
status	Is 'ok' (string) if current search phrase is ok
Inherited from object __class__	

18 Module pySanaIndeksi.WordIndex.SearcherUnitTest

Date: \$10.8.2012 11:49:09\$

Author: Patrik Ahvenainen

18.1 Functions

suite()

18.2 Variables

Name	Description
MaterialFilePath	**Value:** `'../../Material/Grimm\'s Fairy Tales.txt'`
operations	**Value:** `{'AND': 'grimm* AND brothers', 'Basic': 'brothers', 'NOT'...`
binaryOperationsSearch	**Value:** `{'Grimm*': 10, 'Grimm* OR brothers': 44, 'brothers': 40, ...`
__package__	**Value:** `'pySanaIndeksi.WordIndex'`

18.3 Class PySearcherTestCases

```
object ─┐
        │
unittest.TestCase ─┐
                   │
                   pySanaIndeksi.WordIndex.SearcherUnitTest.PySearcherTestCases
```

18.3.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

tearDown(*self*)

Hook method for deconstructing the test fixture after testing it.

Overrides: unittest.TestCase.tearDown extit(inherited documentation)

testRandomWord(*self*)

Tests that non-empty words are found and they are not the same

testRandomWords(*self*)

Tests that a set of random words do not contain the same words

testBinaryOperationsAreWorking(*self*)

Checks that operations are not identic and that correct number of hits is returned for every known result.

Inherited from unittest.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), assertAlmostEqual(), assertAlmostEquals(), assertEqual(), assertEquals(), assertFalse(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertRaises(), assertTrue(), assert_(), countTestCases(), debug(), defaultTestResult(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), shortDescription()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

18.3.2 Properties

Name	Description
Inherited from object	
__class__	

19 Module pySanaIndeksi.WordIndex.WordReader

Date: $1.8.2012 11:42:53$

Author: Patrik Ahvenainen

19.1 Variables

Name	Description
__package__	**Value:** `'pySanaIndeksi.WordIndex'`

19.2 Class WordReader

```
object ─┐
        │
        pySanaIndeksi.WordIndex.WordReader.WordReader
```

```
This class can be used to read words from one or multiple files. The file
names are given to the object upon initialization.

Words will be stored in tuple self.words which contains the word and the
row number in a file where this word appears and the file number.

Accepted characters are determined when a new WordReader object is created.
The case can be lower, upper or mixed. For lower and upper case only all
letters are converted to lower and upper case, respectively. Numerals are
accepted by default but can be excluded. Letters are from A to Z and any
other letters and non alphanumerical characters can be passed in to the
reader via specialCharacters.

Characters can be mapped to an indexed table via char2ind() function.

public methods:
self.addFileName(filename)      Add a new file by giving its filename.
self.addFileNames(filenames):   Add multiple files by giving a list
                                containing their filenames.
self.char2ind(char)             Returns the index corresponding to character
self.clear()                    Forget any read words
self.clearFileNames()           Empties the filename list
self.getCharMapSize()           Returns the number of accepted characters
self.ind2char(index)            Returns the character corresponding to index
self.readWords()                Reads all words with accepted characters
```

```
                                from all files.
self.sanitize(word)             Removes non-accepted characters from the word

private methods:
self._createChrMap()            Creates the character-index-character
                                mapping.
```

19.2.1 Methods

__init__(*self*, *filenames*=`[]`, *specialCharacters*=`['-', '\'']`, *acceptNumerals*=`True`, *acceptUpperCase*=`True`, *acceptLowerCase*=`False`)

Upon initializing, there are no words in the WordReader. During initialization accepted characters are mapped. They are

- all characters in the specialCharacters list
- numerals if acceptNumerals equals True
- upper case letters if acceptUpperCase equals True
- lower case letters if acceptLowerCase equals True.

If only one case is accepted all letters are considered to be of that case.

Filenames list the names of the files from which the words are read. They must be in a list form. After initialization new files can be added with addFileName(...) and addFileNames(...).

Overrides: object.__init__

addFileName(*self*, *filename*, *readNow*=`False`)

Add the filename to the readers list.

Parameters

`filename`: filename to be added

(type=string of a filename)

`readNow`: set to True to add the words from the given file immediately to the reader

(type=boolean)

addFileNames(*self*, *filenames*, *readNow*=`False`)

Add the filenames to the readers list.

Parameters

`filenames`: filenames to be added

(type=a list of strings (filenames))

`readNow`: set to True to add the words from the given files immediately to the reader

(type=boolean)

char2ind(*self*, *char*)

Maps a character to index of the wordReader

Parameters

`char`: the character to map

(type=character)

Return Value

index of the mapped character

(type=integer)

clear(*self*, *type*=`'empty'`)

Clears any words read so far

clearFileNames(*self*)

Clears the file name list of the WordReader

getCharMapSize(*self*)

Return Value

number of different accepted characters

(type=integer)

ind2char(*self*, *index*)

Maps an index of WordReader to a character

Parameters

`index`: the mapped index

(type=integer)

Return Value

character matching the mapped index

(type=character)

readWords(*self*)

Reads all the words from the file specified in self.filenames. Method sanitize specifies the accepted word formatting.

sanitize(*self*, *word*)

Returns the sanitized word (remove non-allowed characters)

Parameters

`word`: the word to be sanitized

(type=string)

Return Value

the same word sanitized, or empty string if the word is bad

(type=string)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

19.2.2 Properties

Name	Description
Inherited from object	
__class__	

20 Module pySanaIndeksi.WordIndex.WordReaderUnitTest

Date: $10.8.2012 11:52:12$

Author: Patrik Ahvenainen

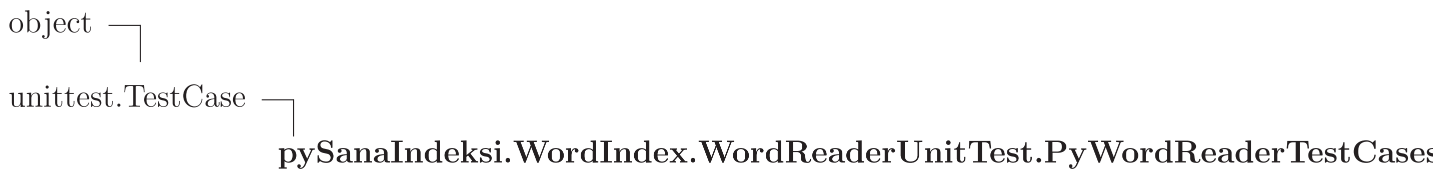
20.1 Functions

suite()

20.2 Variables

Name	Description
unsanitizedWords	**Value:** `[’32\’verreg\xc3\xb6 ’, ’dseFR-fw- ert’, ’ dfg’, ’\xc2\x...`
sanitizedWords	**Value:** `[’32\’VERREG’, ’DSEFR-FW-’, ’DFG’, ’’, ’4EGDFB’]`
properChrMap	**Value:** `[’0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’-’, ’...`
properIdxMap	**Value:** `{’\’’: 37, ’-’: 10, ’0’: 0, ’1’: 1, ’2’: 2, ’3’: 3, ’4’: ...`
linesIn2books	**Value:** `18431`
noOfFiles	**Value:** `2`
wordsInTestFile	**Value:** `50`
__package__	**Value:** `’pySanaIndeksi.WordIndex’`

20.3 Class PyWordReaderTestCases


object
unittest.TestCase
pySanaIndeksi.WordIndex.WordReaderUnitTest.PyWordReaderTestCases

20.3.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.TestCase.setUp extit(inherited documentation)

tearDown(*self*)

Hook method for deconstructing the test fixture after testing it.

Overrides: unittest.TestCase.tearDown extit(inherited documentation)

testSanitize(*self*)

Test whether word sanitizing works

testCreateChrMap(*self*)

Test whether index and character maps are okay

testInd2char(*self*)

Test function ind2char

testChar2ind(*self*)

Test function ind2char

testGetCharMapSize(*self*)

Test whether getCharMapSize returns the correct value

testLineCount(*self*)

Test whether WordReader reads all lines in files

testWordCountAndClear(*self*)

Test if the reader finds the correct number of words

Inherited from unittest.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), assertAlmostEqual(), assertAlmostEquals(), assertEqual(), assertEquals(), assertFalse(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertRaises(), assertTrue(), assert_(), countTestCases(), debug(), defaultTestResult(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), shortDescription()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

20.3.2 Properties

Name	Description
Inherited from object	
__class__	

21 Module pySanaIndeksi.WordIndex.pyUnitTest

22 Module pySanaIndeksi.WordIndex.pysanaindeksi

Date: $31.7.2012 12:14:26$

Author: Patrik Ahvenainen

22.1 Functions

testRun()

indexFile(*tree*, *materialPath*)

Gives the user an option to choose which file to index to the tree

unIndexAllFiles(*tree*)

Removes all indexed words from the tree

doSearch(*tree*)

Searches the tree for the search phrase requested from the user

selectTree(*tree*)

Gives the user the option to choose which tree to use

printOperationOptions(*status*)

Prints out the options for the user

operator(*operation*, *tree*, *materialPath*, *status*)

Calls appropriate function according to the user's choice

printStarred(*word*)

Prints the word and stars above and below it

prompt(*request*=`''`, *limits*=`[]`, *type*=`'int'`)

Asks the user for input, checks integer input for validity

22.2 Variables

Name	**Description**
name	**Value:** `'pySanaIndeksi'`
operationOptions	**Value:** `{0: 'Exit', 1: 'Index file', 2: 'Choose indexer', 3: 'Uni...`
trees	**Value:** `{0: {'RedBlack': RedBlack}, 1: {'Trie': Trie}}`

23 Module pySanaIndeksi.WordIndex.testFindMethod

Date: \$22.8.2012 14:52:54\$

Author: Patrik Ahvenainen

24 Module pySanaIndeksi.WordIndex.timing

Date: $14.8.2012 12:02:36$

Author: Patrik Ahvenainen

24.1 Functions

timeThis(*func*)

average(*values*)

repeat(*repeats*)

addWordsToEmptyList(*tree*, *words*, *repeats*, *string*, *printout*=`True`)

findWords(*tree*, *words*, *repeats*, *string*=`''`, *printout*=`True`)

Index