PHP Sorted Collections

Release 1.0.8

Ch. Demko

Apr 08, 2024

CONTENTS

CHAPTER

ONE

PHP SORTED COLLECTIONS

Sorted Collection for PHP. Insertion, search, and removal compute in `log(n)` time where `n` is the number of items present in the collection. It uses AVL threaded tree [see @Knuth97, 1:320, Sect. 2.3.1] as internal structure.

@Knuth97: Donald E. Knuth, The Art of Computer Programming, Addison-Wesley, volumes 1 and 2, 2nd edition, 1997.

This project uses:

- PHP Code Sniffer for checking PHP code style
- PHPUnit for unit test (100% covered)
- Sphinx and Doxygen for the documentation

Instructions

Using composer: either

```
$ composer create-project chdemko/sorted-collections:1.0.*@dev; cd sorted-collections
Creating a "chdemko/sorted-collections:1.0.*@dev" project at "./sorted-collections"
...
```

or create a `composer.json` file containing

```
{
    "require": {
        "chdemko/sorted-collections": "1.0.*@dev"
    }
}
```

and run

```
$ composer install
Loading composer repositories with package information
...
```

Create a `test.php` file containg

```
<?php

require __DIR__ . '/vendor/autoload.php';

use chdemko\SortedCollection\TreeMap;
```

(continues on next page)

(continued from previous page)

```
$tree = TreeMap::create()->put(
    [1=>1, 9=>9, 5=>5, 2=>2, 6=>6, 3=>3, 0=>0, 8=>8, 7=>7, 4=>4]
);
echo $tree . PHP_EOL;
```

And run

```
$ php test.php
[0,1,2,3,4,5,6,7,8,9]
```

See the examples and benchmarks folders for more information.

1.1 Documentation

Run

```
$ sudo apt install doxygen python3-pip python3-virtualenv
$ virtualenv venv
$ venv/bin/activate
(venv) $ pip install -r docs/requirements.txt
(venv) $ sphinx-build -b html docs/ html/
(venv) $ deactivate
$
```

if you want to create local documentation with Sphinx.

1.2 Citation

If you are using this project including publication in research activities, you have to cite it using (BibTeX format). You are also pleased to send me an email to chdemko@gmail.com.

- authors: Christophe Demko
- title: php-sorted-collections: a PHP library for handling sorted collections
- year: 2014
- how published: https://packagist.org/packages/chdemko/sorted-collections

All releases can be found here

CHAPTER

TWO

USAGE

2.1 Creation

The base class for storing sorted maps is the `TreeMap` class.

```
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeMap;

// This will create a map indexed by numbers
// it contains 10 key/value pairs from 0/0 to 9/9
$map = TreeMap::create()->put(
    [1=>1, 9=>9, 5=>5, 2=>2, 6=>6, 3=>3, 0=>0, 8=>8, 7=>7, 4=>4]
);
```

There are two other classes to create maps which are in fact views on another sorted map.

```
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeMap;
use chdemko\SortedCollection\ReversedMap;
use chdemko\SortedCollection\SubMap;

$map = TreeMap::create()->put(
    [1=>1, 9=>9, 5=>5, 2=>2, 6=>6, 3=>3, 0=>0, 8=>8, 7=>7, 4=>4]
);

// This will create a map which is the reverse of $map
$reversed = ReversedMap::create($map);

// This will create a map which is a sub map of $reversed
$sub = SubMap::create($reversed, 7, 3);

// This will display {"7":7,"6":6,"5":5,"4":4}
echo $sub . PHP_EOL;
```

For sub maps there are other methods for creation

```
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeMap;
use chdemko\SortedCollection\SubMap;

$map = TreeMap::create()->put(
```

(continues on next page)

(continued from previous page)

```
    [1=>1, 9=>9, 5=>5, 2=>2, 6=>6, 3=>3, 0=>0, 8=>8, 7=>7, 4=>4]
);

// This will create a map which is a sub map of $map from key 3 to the end
$tail = SubMap::tail($map, 3);

$map[10] = 10;

// This will display {"3":3,"4":4,"5":5,"6":6,"7":7,"8":8,"9":9,"10":10}
echo $tail . PHP_EOL;

// This will create a sub map of $map from beginning to key 7 (inclusive)
$head = SubMap::head($map, 7, true);

// This will display [0,1,2,3,4,5,6,7]
echo $head . PHP_EOL;
```

Sets are created using similar functions

```
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeSet;
use chdemko\SortedCollection\ReversedSet;
use chdemko\SortedCollection\SubSet;

$set = TreeSet::create()->put([1, 9, 5, 2, 6, 3, 0, 8, 7, 4]);
$reversed = ReversedSet::create($set);
$sub = SubSet::create($reversed, 7, 3);

// This will display [7,6,5,4]
echo $sub . PHP_EOL;
```

2.2 Iteration

These collections support PHP iteration.

Using maps

```
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeMap;
use chdemko\SortedCollection\ReversedMap;
use chdemko\SortedCollection\SubMap;

$map = TreeMap::create()->put(
    [1=>1, 9=>9, 5=>5, 2=>2, 6=>6, 3=>3, 0=>0, 8=>8, 7=>7, 4=>4]
);
$reversed = ReversedMap::create($map);
$sub = SubMap::create($reversed, 7, 3);

// This will display 7:7;6:6;5:5;4:4;
foreach ($sub as $key => $value)
{
```

(continues on next page)

(continued from previous page)

```
        echo $key . ':' . $value . ';';
}
echo PHP_EOL;
```

Using sets

```
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeSet;
use chdemko\SortedCollection\ReversedSet;
use chdemko\SortedCollection\SubSet;

$set = TreeSet::create()->put([1, 9, 5, 2, 6, 3, 0, 8, 7, 4]);
$reversed = ReversedSet::create($set);
$sub = SubSet::create($reversed, 7, 3);

// This will display 0:7;1:6;2:5;3:4;
foreach ($sub as $key => $value)
{
        echo $key . ':' . $value . ';';
}
echo PHP_EOL;
```

The behavior is unpredictable if the current key of an iterator is removed of the collection.

2.3 Counting

These collections support PHP counting

```
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeMap;
use chdemko\SortedCollection\ReversedMap;
use chdemko\SortedCollection\SubMap;

$map = TreeMap::create()->put(
    [1=>1, 9=>9, 5=>5, 2=>2, 6=>6, 3=>3, 0=>0, 8=>8, 7=>7, 4=>4]
);
$reversed = ReversedMap::create($map);
$sub = SubMap::create($reversed, 7, 3);

// This will display 4
echo count($sub) . PHP_EOL;
```

2.4 Array access

Insertion, modification, access and removal has been designed to work using PHP array access features

Using maps

```php
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeMap;

$map = TreeMap::create();
$map[4] = 4;
$map[2] = 2;
$map[6] = 6;
unset($map[4]);

// This will display 1
echo isset($map[2]) . PHP_EOL;

// This will display 2
echo $map[2] . PHP_EOL;
```

Using sets

```php
require __DIR__ . '/vendor/autoload.php';
use chdemko\SortedCollection\TreeSet;

$set = TreeSet::create();
$set[4] = true;
$set[2] = true;
$set[6] = true;
unset($set[4]);

// This will display 1
echo isset($set[2]) . PHP_EOL;

// This will display 1
echo $set[2] . PHP_EOL;

// This will display nothing
echo $set[4] . PHP_EOL;
```

A lot of methods has been implemented to give access to the minimum element, the lower element...

CHAPTER

THREE

API DOCUMENTATION

3.1 Interfaces

3.1.1 Sorted Collection

`interface` () → :SortedCollection

Subclassed by chdemko\SortedCollection\SortedMap, chdemko\SortedCollection\SortedSet

Public Functions

`chdemko\SortedCollection\SortedCollection::comparator()`

Get the comparator

Since
1.0.0

return
callable The comparator

`chdemko\SortedCollection\SortedCollection::first()`

Get the first element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The first element

`chdemko\SortedCollection\SortedCollection::last()`

Get the last element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The last element

`chdemko\SortedCollection\SortedCollection::lower( $key)`

Returns the greatest element lesser than the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no lower element

return
mixed The found node

`chdemko\SortedCollection\SortedCollection::floor( $key)`

Returns the greatest element lesser than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no floor element

return
mixed The found node

`chdemko\SortedCollection\SortedCollection::find( $key)`

Returns the element equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no such element

return
mixed The found node

`chdemko\SortedCollection\SortedCollection::ceiling( $key)`

Returns the lowest element greater than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no ceiling element

return
mixed The found node

`chdemko\SortedCollection\SortedCollection::higher( $key)`
Returns the lowest element greater than to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no higher element

return
mixed The found node

3.1.2 Sorted Set

`interface` () → :SortedSet
Subclassed by chdemko\SortedCollection\AbstractSet

3.1.3 Sorted Map

`interface` () → :SortedMap
Subclassed by chdemko\SortedCollection\AbstractMap

Public Functions

`chdemko\SortedCollection\SortedMap::firstKey()`
Get the first key or throw an exception if there is no element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The first key

`chdemko\SortedCollection\SortedMap::lastKey()`

Get the last key or throw an exception if there is no element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The last key

`chdemko\SortedCollection\SortedMap::lowerKey( $key)`

Returns the greatest key lesser than the given key or throw an exception if there is no such key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no lower element

return
mixed The found key

`chdemko\SortedCollection\SortedMap::floorKey( $key)`

Returns the greatest key lesser than or equal to the given key or throw an exception if there is no such key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no floor element

return
mixed The found key

`chdemko\SortedCollection\SortedMap::findKey( $key)`

Returns the key equal to the given key or throw an exception if there is no such key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no such element

return
mixed The found key

`chdemko\SortedCollection\SortedMap::ceilingKey( $key)`

Returns the lowest key greater than or equal to the given key or throw an exception if there is no such key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no ceiling element

return
mixed The found key

`chdemko\SortedCollection\SortedMap::higherKey( $key)`

Returns the lowest key greater than to the given key or throw an exception if there is no such key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no higher element

return
mixed The found key

`chdemko\SortedCollection\SortedMap::predecessor( $node)`

Get the predecessor node

Since
1.0.0

param $node
A tree node member of the underlying TreeMap

return
mixed The predecessor node

`chdemko\SortedCollection\SortedMap::successor( $node)`

Get the successor node

Since
1.0.0

param $node
A tree node member of the underlying TreeMap

return
mixed The successor node

`chdemko\SortedCollection\SortedMap::keys()`

Keys generator

Since
1.0.0

return
mixed The keys generator

`chdemko\SortedCollection\SortedMap::values()`

Values generator

Since
1.0.0

return
mixed The values generator

3.2 Abstract classes

3.2.1 Abstract Set

`SortedCollection()` → :AbstractSet : public chdemko\SortedCollection\SortedSet

Subclassed by chdemko\SortedCollection\ReversedSet, chdemko\SortedCollection\SubSet, chdemko\SortedCollection\TreeSet

Public Functions

`chdemko\SortedCollection\AbstractSet::__get( $property)`

Magic get method

Since
1.0.0

param $property
The property

throws RuntimeException
If the property does not exist

return
mixed The value associated to the property

`chdemko\SortedCollection\AbstractSet::comparator()`

Get the comparator

Since
1.0.0

return
callable The comparator

`chdemko\SortedCollection\AbstractSet::first()`

Get the first element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The first element

`chdemko\SortedCollection\AbstractSet::last()`

Get the last element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The last element

`chdemko\SortedCollection\AbstractSet::lower( $element)`

Returns the greatest element lesser than the given element

Since
1.0.0

param $element
The searched element

throws OutOfBoundsException
If there is no lower element

return
mixed The found element

`chdemko\SortedCollection\AbstractSet::floor( $element)`

Returns the greatest element lesser than or equal to the given element

Since
1.0.0

param $element
The searched element

throws OutOfBoundsException
If there is no floor element

return
mixed The found element

`chdemko\SortedCollection\AbstractSet::find( $element)`

Returns the element equal to the given element

Since
1.0.0

param $element
The searched element

throws OutOfBoundsException
If there is no such element

return
mixed The found element

`chdemko\SortedCollection\AbstractSet::ceiling( $element)`

Returns the lowest element greater than or equal to the given element

Since
1.0.0

param $element
The searched element

throws OutOfBoundsException
If there is no ceiling element

return
mixed The found element

`chdemko\SortedCollection\AbstractSet::higher( $element)`

Returns the lowest element greater than to the given element

Since
1.0.0

param $element
The searched element

throws OutOfBoundsException
If there is no higher element

return
mixed The found element

`chdemko\SortedCollection\AbstractSet::__toString()`

Convert the object to a string

Since
1.0.0

return
string String representation of the object

`chdemko\SortedCollection\AbstractSet::toArray()`

Convert the object to an array

Since
1.0.0

return
array Array representation of the object

`chdemko\SortedCollection\AbstractSet::getIterator()`

Create an iterator

Since
1.0.0

return
Iterator A new iterator

`chdemko\SortedCollection\AbstractSet::offsetGet( $element)`

Get the value for an element

Since
1.0.0

param $element
The element

return
mixed The found value

`chdemko\SortedCollection\AbstractSet::offsetExists( $element)`

Test the existence of an element

Since
1.0.0

param $element
The element

return
boolean TRUE if the element exists, false otherwise

`chdemko\SortedCollection\AbstractSet::offsetSet( $element,  $value)`

Set the value for an element

Since
1.0.0

param $element
The element

param $value
The value

throws RuntimeOperation
The operation is not supported by this class

return
void

`chdemko\SortedCollection\AbstractSet::offsetUnset( $element)`

Unset the existence of an element

Since
1.0.0

param $element
The element

throws RuntimeOperation
The operation is not supported by this class

return
void

`chdemko\SortedCollection\AbstractSet::count()`

Count the number of elements

Since
1.0.0

return
integer

3.2.2 Abstract Map

`SortedCollection()` → :AbstractMap : public chdemko\SortedCollection\SortedMap

Subclassed by chdemko\SortedCollection\ReversedMap, chdemko\SortedCollection\SubMap, chdemko\SortedCollection\TreeMap

Public Functions

`chdemko\SortedCollection\AbstractMap::__get( $property)`

Magic get method

Since
1.0.0

param $property
The property

throws RuntimeException
If the property does not exist

return
mixed The value associated to the property

`chdemko\SortedCollection\AbstractMap::firstKey()`

Get the first key

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The first key

`chdemko\SortedCollection\AbstractMap::firstValue()`

Get the first value

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The first value

`chdemko\SortedCollection\AbstractMap::lastKey()`

Get the last key

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The last key

`chdemko\SortedCollection\AbstractMap::lastValue()`

Get the last value

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The last value

`chdemko\SortedCollection\AbstractMap::lowerKey( $key)`

Returns the greatest key lesser than the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no lower element

return
mixed The found key

`chdemko\SortedCollection\AbstractMap::lowerValue( $key)`

Returns the value whose key is the greatest key lesser than the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no lower element

return
mixed The found value

`chdemko\SortedCollection\AbstractMap::floorKey( $key)`

Returns the greatest key lesser than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no floor element

return
mixed The found key

`chdemko\SortedCollection\AbstractMap::floorValue( $key)`

Returns the value whose key is the greatest key lesser than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no floor element

return
mixed The found value

`chdemko\SortedCollection\AbstractMap::findKey( $key)`

Returns the key equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no such element

return
mixed The found key

`chdemko\SortedCollection\AbstractMap::findValue( $key)`

Returns the value whose key equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no such element

return
mixed The found value

`chdemko\SortedCollection\AbstractMap::ceilingKey( $key)`

Returns the lowest key greater than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no ceiling element

return
mixed The found key

`chdemko\SortedCollection\AbstractMap::ceilingValue( $key)`

Returns the value whose key is the lowest key greater than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no ceiling element

return
mixed The found value

`chdemko\SortedCollection\AbstractMap::higherKey( $key)`

Returns the lowest key greater than to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no higher element

return
mixed The found key

`chdemko\SortedCollection\AbstractMap::higherValue( $key)`

Returns the value whose key is the lowest key greater than to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no higher element

return
mixed The found value

`chdemko\SortedCollection\AbstractMap::keys()`

Keys iterator

Since
1.0.0

return
Iterator The keys iterator

`chdemko\SortedCollection\AbstractMap::values()`

Values iterator

Since
1.0.0

return
Iterator The values iterator

`chdemko\SortedCollection\AbstractMap::__toString()`

Convert the object to a string

Since
1.0.0

return
string String representation of the object

`chdemko\SortedCollection\AbstractMap::toArray()`

Convert the object to an array

Since
1.0.0

return
array Array representation of the object

`chdemko\SortedCollection\AbstractMap::getIterator()`

Create an iterator

Since
1.0.0

return
Iterator A new iterator

`chdemko\SortedCollection\AbstractMap::offsetGet( $key)`

Get the value for a key

Since
1.0.0

param $key
The key

throws OutOfRangeException
If there is no such element

return
mixed The found value

`chdemko\SortedCollection\AbstractMap::offsetExists( $key)`

Test the existence of a key

Since
1.0.0

param $key
The key

return
boolean TRUE if the key exists, false otherwise

`chdemko\SortedCollection\AbstractMap::offsetSet( $key,  $value)`

Set the value for a key

Since
1.0.0

param $key
The key

param $value
The value

throws RuntimeOperation
The operation is not supported by this class

return
void

`chdemko\SortedCollection\AbstractMap::offsetUnset( $key)`

Unset the existence of a key

Since
1.0.0

param $key
The key

throws RuntimeOperation
The operation is not supported by this class

return
void

3.3 Concrete classes

3.3.1 Tree Set

`SortedCollection()` → :TreeSet : public chdemko\SortedCollection\AbstractSet

Public Functions

`chdemko\SortedCollection\TreeSet::put( $traversable = array())`

Put values in the set

Since
1.0.0

param $traversable
Values to put in the set

return
TreeSet $this for chaining

`chdemko\SortedCollection\TreeSet::clear()`

Clear the set

Since
1.0.0

return
TreeSet $this for chaining

`chdemko\SortedCollection\TreeSet::initialise( $traversable = array())`

Initialise the set

Since
1.0.0

param $traversable
Values to initialise the set

return
TreeSet $this for chaining

`chdemko\SortedCollection\TreeSet::__clone()`

Clone the set

Since
1.0.0

return
void

`chdemko\SortedCollection\TreeSet::offsetSet( $element,  $value)`

Set the value for an element

Since
1.0.0

param $element
The element

param $value
The value

return
void

`chdemko\SortedCollection\TreeSet::jsonSerialize()`

Serialize the object

Since
1.0.0

return
array Array of values

`chdemko\SortedCollection\TreeSet::offsetUnset( $element)`

Unset the existence of an element

Since
1.0.0

param $element
The element

return
void

Public Static Functions

`static chdemko\SortedCollection\TreeSet::create( $comparator = null)`

Create

Since
1.0.0

param $comparator
Comparison function

return
TreeSet A new TreeSet

3.3.2 Sub Set

`SortedCollection()` → :SubSet : public chdemko\SortedCollection\AbstractSet

Public Functions

`chdemko\SortedCollection\SubSet::__get( $property)`

Magic get method

Since
1.0.0

param $property
The property

return
mixed The value associated to the property

`chdemko\SortedCollection\SubSet::__set( $property,  $value)`

Magic set method

Since
1.0.0

param $property
The property

param $value
The new value

throws RuntimeException
If the property does not exist

return
void

`chdemko\SortedCollection\SubSet::__unset( $property)`

Magic unset method

Since
1.0.0

param $property
The property

throws RuntimeException
If the property does not exist

return
void

`chdemko\SortedCollection\SubSet::__isset( $property)`

Magic isset method

Since
1.0.0

param $property
The property

return
boolean

`chdemko\SortedCollection\SubSet::jsonSerialize()`

Serialize the object

Since
1.0.0

return
array Array of values

Public Static Functions

`static chdemko\SortedCollection\SubSet::create(SortedSet $set, $from, $to, $fromInclusive = true, $toInclusive = false)`

Create

Since
1.0.0

param $set
Internal set

param $from
The from element

param $to
The to element

param $fromInclusive
The inclusive flag for from

param $toInclusive
The inclusive flag for to

return
SubSet A new sub set

`static chdemko\SortedCollection\SubSet::head(SortedSet $set, $to, $toInclusive = false)`

Head

Since
1.0.0

param $set
Internal set

param $to
The to element

param $toInclusive
The inclusive flag for to

return
SubSet A new head set

`static chdemko\SortedCollection\SubSet::tail(SortedSet $set, $from, $fromInclusive = true)`

Tail

Since
1.0.0

param $set
Internal set

param $from
The from element

param $fromInclusive
The inclusive flag for from

return
SubSet A new tail set

`static chdemko\SortedCollection\SubSet::view(SortedSet $set)`

View

Since
1.0.0

param $set
Internal set

return
SubSet A new sub set

3.3.3 Reversed Set

`SortedCollection()` → :ReversedSet : public chdemko\SortedCollection\AbstractSet

Public Functions

`chdemko\SortedCollection\ReversedSet::__get( $property)`

Magic get method

Since
1.0.0

param $property
The property

return
mixed The value associated to the property

`chdemko\SortedCollection\ReversedSet::jsonSerialize()`

Serialize the object

Since
1.0.0

return
array Array of values

Public Static Functions

`static chdemko\SortedCollection\ReversedSet::create(SortedSet $set)`

Create

Since
1.0.0

param $set
Internal set

return
ReversedSet A new reversed set

3.3.4 Tree Map

`SortedCollection()` → :TreeMap : public chdemko\SortedCollection\AbstractMap

Public Functions

`chdemko\SortedCollection\TreeMap::comparator()`

Get the comparator

Since
1.0.0

return
callable The comparator

`chdemko\SortedCollection\TreeMap::first()`

Get the first element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The first element

`chdemko\SortedCollection\TreeMap::last()`

Get the last element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The last element

`chdemko\SortedCollection\TreeMap::predecessor( $element)`

Get the predecessor element

Since
1.0.0

param $element
A tree node member of the underlying TreeMap

throws OutOfBoundsException
If there is no predecessor

return
mixed The predecessor element

`chdemko\SortedCollection\TreeMap::successor( $element)`

Get the successor element

Since
1.0.0

param $element
A tree node member of the underlying TreeMap

throws OutOfBoundsException
If there is no successor

return
mixed The successor element

`chdemko\SortedCollection\TreeMap::lower( $key)`

Returns the element whose key is the greatest key lesser than the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no lower element

return
mixed The found element

`chdemko\SortedCollection\TreeMap::floor( $key)`

Returns the element whose key is the greatest key lesser than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no floor element

return
mixed The found element

`chdemko\SortedCollection\TreeMap::find( $key)`

Returns the element whose key is equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no such element

return
mixed The found element

`chdemko\SortedCollection\TreeMap::ceiling( $key)`

Returns the element whose key is the lowest key greater than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no ceiling element

return
mixed The found element

`chdemko\SortedCollection\TreeMap::higher( $key)`

Returns the element whose key is the lowest key greater than to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no higher element

return
mixed The found element

`chdemko\SortedCollection\TreeMap::put( $traversable = array())`

Put values in the map

Since
1.0.0

param $traversable
Values to put in the map

return
TreeMap $this for chaining

`chdemko\SortedCollection\TreeMap::clear()`

Clear the map

Since
1.0.0

return
TreeMap $this for chaining

`chdemko\SortedCollection\TreeMap::initialise( $traversable = array())`

Initialise the map

Since
1.0.0

param $traversable
Values to initialise the map

return
TreeMap $this for chaining

`chdemko\SortedCollection\TreeMap::__clone()`

Clone the map

Since
1.0.0

return
void

`chdemko\SortedCollection\TreeMap::jsonSerialize()`

Serialize the object

Since
1.0.0

return
array Array of values

`chdemko\SortedCollection\TreeMap::offsetSet( $key,  $value)`

Set the value for a key

Since
1.0.0

param $key
The key

param $value
The value

return
void

`chdemko\SortedCollection\TreeMap::offsetUnset( $key)`

Unset the existence of a key

Since
1.0.0

param $key
The key

return
void

`chdemko\SortedCollection\TreeMap::count()`

Count the number of key/value pairs

Since
1.0.0

return
integer

Public Static Functions

`static chdemko\SortedCollection\TreeMap::create( $comparator = null)`

Create

Since
1.0.0

param $comparator
Comparison function

return
TreeMap A new TreeMap

3.3.5 Sub Map

`SortedCollection`() → :SubMap : public chdemko\SortedCollection\AbstractMap

Public Functions

`chdemko\SortedCollection\SubMap::__get( $property)`

Magic get method

Since
1.0.0

param $property
The property

throws RuntimeException
If the property does not exist

return
mixed The value associated to the property

`chdemko\SortedCollection\SubMap::__set( $property,  $value)`

Magic set method

Since
1.0.0

param $property
The property

param $value
The new value

throws RuntimeException
If the property does not exist

return
void

`chdemko\SortedCollection\SubMap::__unset( $property)`

Magic unset method

Since
1.0.0

param $property
The property

throws RuntimeException
If the property does not exist

return
void

`chdemko\SortedCollection\SubMap::__isset( $property)`

Magic isset method

Since
1.0.0

param $property
The property

return
boolean

`chdemko\SortedCollection\SubMap::comparator()`

Get the comparator

Since
1.0.0

return
callable The comparator

`chdemko\SortedCollection\SubMap::first()`

Get the first element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The first element

`chdemko\SortedCollection\SubMap::last()`

Get the last element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The last element

`chdemko\SortedCollection\SubMap::predecessor( $element)`

Get the predecessor element

Since
1.0.0

param $element
A tree node member of the underlying TreeMap

throws OutOfBoundsException
If there is no predecessor

return
mixed The predecessor element

`chdemko\SortedCollection\SubMap::successor( $element)`

Get the successor element

Since
1.0.0

param $element
A tree node member of the underlying TreeMap

throws OutOfBoundsException
If there is no successor

return
mixed The successor element

`chdemko\SortedCollection\SubMap::lower( $key)`

Returns the element whose key is the greatest key lesser than the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no lower element

return
mixed The found element

`chdemko\SortedCollection\SubMap::floor( $key)`

Returns the element whose key is the greatest key lesser than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no floor element

return
mixed The found element

`chdemko\SortedCollection\SubMap::find( $key)`

Returns the element whose key is equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no such element

return
mixed The found element

chdemko\SortedCollection\SubMap::ceiling($key)

Returns the element whose key is the lowest key greater than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no ceiling element

return
mixed The found element

chdemko\SortedCollection\SubMap::higher($key)

Returns the element whose key is the lowest key greater than to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no higher element

return
mixed The found element

chdemko\SortedCollection\SubMap::jsonSerialize()

Serialize the object

Since
1.0.0

return
array Array of values

chdemko\SortedCollection\SubMap::count()

Count the number of key/value pairs

Since
1.0.0

return
integer

Public Static Functions

`static chdemko\SortedCollection\SubMap::create(SortedMap $map, $fromKey, $toKey, $fromInclusive = true, $toInclusive = false)`

Create

Since
1.0.0

param $map
A sorted map

param $fromKey
The from key

param $toKey
The to key

param $fromInclusive
The inclusive flag for from

param $toInclusive
The inclusive flag for to

return
SubMap A new sub map

`static chdemko\SortedCollection\SubMap::head(SortedMap $map, $toKey, $toInclusive = false)`

Return a head portion of a sorted map

Since
1.0.0

param $map
A sorted map

param $toKey
The to key

param $toInclusive
The inclusive flag for to

return
SubMap A new head map

`static chdemko\SortedCollection\SubMap::tail(SortedMap $map, $fromKey, $fromInclusive = true)`

Return a tail portion of a sorted map

Since
1.0.0

param $map
A sorted map

param $fromKey
The from key

param $fromInclusive
The inclusive flag for from

return
SubMap A new tail map

`static chdemko\SortedCollection\SubMap::view(SortedMap $map)`

Return a view of the map

Since
1.0.0

param $map
A sorted map

return
SubMap A new sub map

3.3.6 Reversed Map

`SortedCollection()` → :ReversedMap : public chdemko\SortedCollection\AbstractMap

Public Functions

`chdemko\SortedCollection\ReversedMap::__get( $property)`

Magic get method

Since
1.0.0

param $property
The property

return
mixed The value associated to the property

`chdemko\SortedCollection\ReversedMap::comparator()`

Get the comparator

Since
1.0.0

return
callable The comparator

`chdemko\SortedCollection\ReversedMap::first()`

Get the first element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The first element

`chdemko\SortedCollection\ReversedMap::last()`

Get the last element

Since
1.0.0

throws OutOfBoundsException
If there is no element

return
mixed The last element

`chdemko\SortedCollection\ReversedMap::predecessor( $element)`

Get the predecessor element

Since
1.0.0

param $element
A tree node member of the underlying TreeMap

throws OutOfBoundsException
If there is no predecessor

return
mixed The predecessor element

`chdemko\SortedCollection\ReversedMap::successor( $element)`

Get the successor element

param $element
A tree node member of the underlying TreeMap

throws OutOfBoundsException
If there is no successor

return
mixed The successor element

`chdemko\SortedCollection\ReversedMap::lower( $key)`

Returns the element whose key is the greatest key lesser than the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no lower element

return
mixed The found element

`chdemko\SortedCollection\ReversedMap::floor( $key)`

Returns the element whose key is the greatest key lesser than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no floor element

return
mixed The found element

`chdemko\SortedCollection\ReversedMap::find( $key)`

Returns the element whose key is equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no such element

return
mixed The found element

`chdemko\SortedCollection\ReversedMap::ceiling( $key)`

Returns the element whose key is the lowest key greater than or equal to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no ceiling element

return
mixed The found element

`chdemko\SortedCollection\ReversedMap::higher( $key)`

Returns the element whose key is the lowest key greater than to the given key

Since
1.0.0

param $key
The searched key

throws OutOfBoundsException
If there is no higher element

return
mixed The found element

`chdemko\SortedCollection\ReversedMap::jsonSerialize()`

Serialize the object

Since
1.0.0

return
array Array of values

`chdemko\SortedCollection\ReversedMap::count()`

Count the number of key/value pairs

Since
1.0.0

return
integer

Public Static Functions

`static chdemko\SortedCollection\ReversedMap::create(SortedMap $map)`

Create

Since
1.0.0

param $map
Internal map

return
ReversedMap A new reversed map

CHAPTER

FOUR

INDICES AND TABLES

- genindex