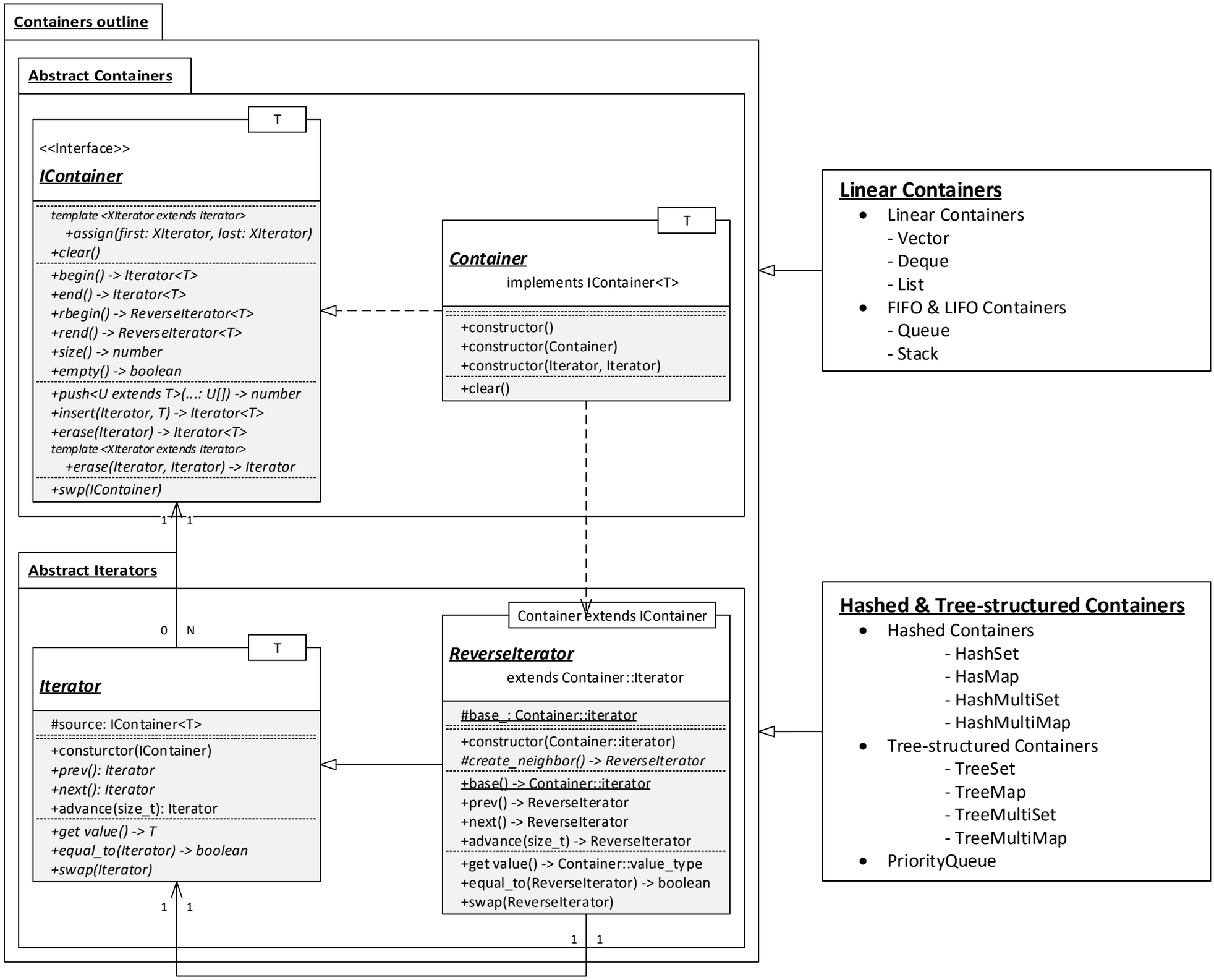
Containers outline
Abstract Containers
T
<<Interface>>
IContainer
template <XIterator extends Iterator>
+assign(first: XIterator, last: XIterator)
+clear()
+begin() -> Iterator<T>
+end() -> Iterator<T>
+rbegin() -> ReverseIterator<T>
+rend() -> ReverseIterator<T>
+size() -> number
+empty() -> boolean
+push<U extends T>(...: U[]) -> number
+insert(Iterator, T) -> Iterator<T>
+erase(Iterator) -> Iterator<T>
template <XIterator extends Iterator>
+erase(Iterator, Iterator) -> Iterator
+swp(IContainer)
T
Container
implements IContainer<T>
+constructor()
+constructor(Container)
+constructor(Iterator, Iterator)
+clear()
1
1
Abstract Iterators
0
N
T
Iterator
#source: IContainer<T>
+consturctor(IContainer)
+prev(): Iterator
+next(): Iterator
+advance(size_t): Iterator
+get value() -> T
+equal_to(Iterator) -> boolean
+swap(Iterator)
1
1
Container extends IContainer
ReverseIterator
extends Container::Iterator
#base_: Container::iterator
+constructor(Container::iterator)
#create_neighbor() -> ReverseIterator
+base() -> Container::iterator
+prev() -> ReverseIterator
+next() -> ReverseIterator
+advance(size_t) -> ReverseIterator
+get value() -> Container::value_type
+equal_to(ReverseIterator) -> boolean
+swap(ReverseIterator)
1
1
Linear Containers
Linear Containers
- Vector
- Deque
- List
FIFO & LIFO Containers
- Queue
- Stack
Hashed & Tree-structured Containers
Hashed Containers
- HashSet
- HasMap
- HashMultiSet
- HashMultiMap
Tree-structured Containers
- TreeSet
- TreeMap
- TreeMultiSet
- TreeMultiMap
PriorityQueue

Linear Containers

FIFO and LIFO Containers

Stack

T

#container_: ILinearContainer<T>

+Stack()
+Stack(Stack)

+top() -> T
+size() -> size_t
+empty() -> boolean

+push(T)
+pop()

Stack

T

#container_: ILinearContainer<T>

+Stack()
+Stack(Stack)

+front() -> T
+back() -> T
+size() -> size_t
+empty() -> boolean

+push(T)
+pop()

1 1 1 1

Interfaces for linear containers

1 1 1 1

<<Interface>>

ILinearContainer

extends IContainer<T>

T

+using super::assign
+assign(n: size_t, val: T)

+front(): T
+back(): T
+push_back(T)
+pop_back()

+using super::insert
+insert(Iterator, size_t, T) -> Iterator
template <XIterator extends Iterator>
+ insert(Iterator, XIterator, XIterator) -> Iterator

<<Interface>>

IDequeContainer

extends ILinearContainer<T>

T

+push_front(T)
+pop_front()

<<Interface>>

IArrayContainer

extends ILinearContainer<T>

T

+reserve(size_t)
+capacity() -> size_t

+at(size_t) -> T
+set(size_t, T)

List container and its Iterators

List

T

extends Container<T>
implements IDequeContainer<T>

#begin_: ListIterator
#end_: ListIterator
#size_: size_t

+override super::constructor
+constructor(size_t, T)
+override super::assign
+clear()

+push<U extends T>(...: U[]): size_t
+override super::insert
+override super::erase

+{<algorithm> methods}

1 1 2 1 N

ListIterator

T

extends Iterator<T>

#prev_: ListIterator
#next_: ListIterator
#value_: T

+constructor
(List, ListIterator, ListIterator, T)
+prev() -> ListIterator
+next() -> ListIterator
+advance(size_t) -> ListIterator

+get value(): T
+set value(T)
+equal_to(ListIterator) -> boolean

+swap(ListIterator)

1 1 1 1

ListReverseIterator

T

extends ReverseIterator<List<T>>

+constructor(ListIterator)
#create_neighbor()
-> ListReverseIterator

+set value(T)

List container and its Iterators

Deque

T

extends Container<T>
implements IArrayContainer<T>,
IDequeContainer<T>

#matrx_: T[][]
#size_: size_t
#capacity_: size_t

+override super::constructor
+constructor(size_t, T)
+override super::assign
+clear()
+reserve(size_t)

+push<U extends T>(...: U[]): size_t
+at(size_t) -> T
+set(size_t, T)

+override super::insert
+override super::erase
-fetch_index(size_t)
-> Pair<size_t, size_t>

1 1 0 1 N

DequeIterator

T

extends Iterator<T>
implements IArrayIterator<T>

#index_: size_t

+constructor(Deque<T>, size_t)
+prev() -> DequeIterator
+next() -> DequeIterator
+advance(size_t) -> DequeIterator

+get index(): size_t
+get value(): T
+set value(T)
+equal_to(DequeIterator) -> boolean

+swap(DequeIterator)

1 1 1 1

DequeReverseIterator

T

extends ReverseIterator<Deque<T>>
implements IArrayIterator<T>

+constructor(DequeIterator)
#create_neighbor()
-> DequeReverseIterator

+get index() -> size_t
+set value(T)

List container and its Iterators

Vector

T

extends Array<T>
implements IArrayContainer<T>

+override super::constructor
+constructor(size_t, T)
+override super::assign
+clear()

+override super::insert
+override super::erase

1 1 0 1 N

VectorIterator

T

extends Iterator<T>

#index_: size_t

+constructor(Vector<T>, size_t)
+prev() -> VectorIterator
+next() -> VectorIterator
+advance(size_t) -> VectorIterator

+getindex(): size_t
+get value(): T
+set value(T)
+equal_to(VectorIterator) -> boolean

+swap(VectorIterator)

1 1 0 1 1

VectorReverseIterator

T

extends ReverseIterator<Vector<T>>
implements IReverseIterator<T>

+constructor(VectorIterator)
#create_neighbor()
-> VectorReverseIterator

+get index() -> size_t
+set value(T)

Set Containers

Abstract Set Containers

SetContainer (T)
extends Container<T>

- **#data_: List<T>**
- +constructor()
- +constructor(Array<T>)
- +constructor(IContainer<T>)
- template <U extends T, InputIterator extends Iterator<T>>
- +constructor(InputIterator, InputIterator)
- +assign(InputIterator, InputIterator)
- #init()
- +clear()
- ***+find(T) -> SetIterator***
- +begin() -> SetIterator
- +end() -> SetIterator
- +rbegin() -> ReverseSetIterator
- +rend() -> ReverseSetIterator
- *+count(T) -> size_t*
- +has(T) -> boolean
- +size() -> size_t
- +push(Pair<L, U>[]): size_t
- *+insert(SetIterator, T) -> SetIterator*
- *template <U extends T, XIterator extends Iterator<U>>*
- *+insert(InputIterator, InputIterator)*
- +erase(T) -> size_t
- template <XIterator extends SetIterator>
- +erase(XIterator, XIterator)
- template <XIterator extends SetIterator<Key, T>>
- #handle_insert(XIterator, XIterator)
- #handle_erase(XIterator, XIterator)

Specified Setss about uniqueness

UniqueSet (T)
extends SetContainer< T>

- +using super::constructor
- +count(Key) -> size_t
- +using super::insert
- *+insert(T) -> Pair<SetIterator, boolean>*
- +swap(UniqueSet)

MultiSet (T)
extends SetContainer<T>

- +using super::constructor
- +using super::insert
- *+insert(T) -> SetIterator*
- +swap(MultiSet)

Iterators

SetIterator (T)
extends Iterator<T>

- #list_it_: ListIterator<T>
- +constructor(SetContainer, ListIterator)
- +prev() -> SetIterator
- +next() -> SetIterator
- +advance(size_t) -> SetIterator
- +get value() -> Pair<Key, T>
- +get first() -> Key
- +get second() -> T
- +set second(T)
- +equal_to(Set_Iterator) -> boolean
- +less(SetIterator) -> boolean
- +hash() -> size_t

SetReverseIterator (T)
extends ReverseIterator <SetContainer<T>>

- +constructor(SetIterator)
- #create_reverse_iterator() -> SetReverseIterator

Final Sets

Hash(Multi)Set (T)
extends SpecifiedSet<T>

- **#hash_buckets_: SetHashBuckets**
- +using super::constructor
- #init()
- template <U extends T, XIterator extends Iterator<U>>
- +assign(InputIterator, InputIterator)
- +clear()
- **+find(T) -> SetIterator**
- +override super::push
- +override super::insert
- template <XIterator extends SetIterator<T>>
- #handle_insert(XIterator, XIterator)
- #handle_erase(XIterator, XIterator(

Tree(Multi)Set (T)
extends SpecifiedSet<T>

- **#tree_: PairTree**
- +using super::constructor
- #init()
- template <U extends T, XIterator extends Iterator<U>>
- +assign(InputIterator, InputIterator)
- +clear()
- **+find(T) -> SetIterator**
- +lower_bound(T) -> SetIterator
- +upper_bound(T) -> SetIterator
- +equal_range(T) -> Pair<SetIterator, SetIterator>
- +override super::insert
- template <XIterator extends SetIterator<T>>
- #handle_insert(XIterator, XIterator)
- #handle_erase(XIterator, XIterator(

PriorityQueue (T)
extends TreeMultiSet<T>

- #container_: TreeMultiSet<T>
- +constructor()
- +constructor(T[])
- +constructor(IContainer<T>)
- +constructor(Iterator, Iterator)
- +top() -> T
- +size() -> size_t
- +empty() -> boolean
- +push(T)
- +pop()

Hash functions

MapHashBuckets (T)
extends HashBuckets <SetIterator<T>>

- -set_: SetContainer< T>
- +constructor(SetContainer<T>)
- +find(T) -> SetIterator

HashBuckets (T)

- #buckets_: Array<Array<T>>
- #item_size_: size_t
- +constructor()
- +reserve(size_t)
- +clear()
- +at(size_t) -> Array<T>
- -hash_index(T) -> size_t
- +insert(T)
- +erase(T)

Red-Black Tree

AtomicTree (T)
extends XTree<T>

- -compare_: (Key, Key) => boolean
- +public constructor(Compare = std.less)
- +using super::find
- +find(SetIterator) -> XTreeNode
- template <XIterator extends SetIterator<T>>
- +is_less(XIterator, XIterator) -> boolean
- +is_equal_to(XIterator, XIterator) -> boolean

XTree (T)

- #root_: XTreeNode<T>
- +public constructor()
- +find(T) -> XTreeNode
- #fetch_maximum(XTreeNode) -> XTreeNode
- -fetch_color(XTreeNode) -> Color
- +insert(T)
- +erase(T)
- -insert_case_{1~5}(XTreeNode)
- -erase_case_{1~6}(XTreeNode)
- #rotate_left(XTreeNode)
- #rotate_right(XTreeNode)
- #replace_node(XTreeNode, XTreeNode)
- *+is_less(T, T) -> boolean*
- *+is_equal_to(T, T) -> boolean*

XTreeNode (T)

- parent_: XTreeNode
- left_: XTreeNode
- right_: XTreeNode
- value: T
- color: Color
- +constructor(T, Color)

Map Containers

Abstract Map Containers

MapContainer *(Key, T)*
extends Container<Pair<Key, T>>

- **#data_: List<Pair<Key, T>>**
- +constructor()
- +constructor(Pair<Key, T>[])
- +constructor(IContainer)
- template <L extends Key, U extends T, XIterator extends Iterator<Pair<L, U>>>
- +constructor(InputIterator, InputIterator)
- +assign(InputIterator, InputIterator)
- #init()
- +clear()
- ***+find(Key) -> MapIterator***
- +begin() -> MapIterator
- +end() -> MapIterator
- +rbegin() -> ReverseMapIterator
- +rend() -> ReverseMapIterator
- *+count(Key) -> size_t*
- +has(Key) -> boolean
- +size() -> size_t
- +push(Pair<L, U>[]): size_t
- *+insert(MapIterator, Pair<Key, T>) -> MapIterator*
- *template <L extends Key, U extends T, XIterator extends Iterator<Pair<L, U>>>*
- *+insert(InputIterator, InputIterator)*
- +erase(Key) -> size_t
- template <XIterator extends MapIterator>
- +erase(XIterator, XIterator)
- template <XIterator extends MapIterator<Key, T>>
- #handle_insert(XIterator, XIterator)
- #handle_erase(XIterator, XIterator)

Specified Maps about uniqueness

UniqueMap *(Key, T)*
extends MapContainer<<Key, T>

- +using super::constructor
- +count(Key) -> size_t
- +get(Key) -> T
- +set(Key, T)
- +using super::insert
- *+insert(Pair<Key, T>) -> Pair<MapIterator, boolean>*
- +swap(UniqueMap)

MultiMap *(Key, T)*
extends MapContainer<<Key, T>

- +using super::constructor
- +using super::insert
- *+insert(Pair<Key, T>) -> MapIterator*
- +swap(MultiMap)

Iterators

MapIterator *(Key, T)*
extends Iterator<Pair<Key, T>>

- #list_it_: ListIterator<Pair<Key, T>>
- +constructor(MapContainer, ListIterator)
- +prev() -> MapIterator
- +next() -> MapIterator
- +advance(size_t) -> MapIterator
- +get value() -> Pair<Key, T>
- +get first() -> Key
- +get second() -> T
- +set second(T)
- +equal_to(MapIterator) -> boolean
- +less(MapIterator) -> boolean
- +hash() -> size_t
- +swap(MapIterator)

MapReverseIterator *(Key, T)*
extends ReverseIterator <MapContainer<Key, T>>

- +constructor(MapIterator)
- #create_neighbor() -> MapReverseIterator
- +get first() -> Key
- +get second() -> T
- +set second(T)

Final Maps

Hash(Multi)Map *(Key, T)*
extends SpecifiedMap<Key, T>

- **#hash_buckets_: MapHashBuckets**
- +override super::constructor
- #init()
- template <L extends Key, U extends T, XIterator extends Iterator<Pair<L, U>>>
- +assign(InputIterator, InputIterator)
- +clear()
- **+find(Key) -> MapIterator**
- +override super::push
- +override super::insert
- template <XIterator extends MapIterator<Key, T>>
- #handle_insert(XIterator, XIterator)
- #handle_erase(XIterator, XIterator(

Tree(Multi)Map *(Key, T)*
extends SpecifiedMap<Key, T>

- **#tree_: PairTree**
- +override super::constructor
- #init()
- template <L extends Key, U extends T, XIterator extends Iterator<Pair<L, U>>>
- +assign(InputIterator, InputIterator)
- +clear()
- **+find(Key) -> MapIterator**
- +lower_bound(Key) -> MapIterator
- +upper_bound(Key) -> MapIterator
- +equal_range(Key) -> Pair<MapIterator, MapIterator>
- +override super::insert
- template <XIterator extends MapIterator<Key, T>>
- #handle_insert(XIterator, XIterator)
- #handle_erase(XIterator, XIterator(

Hash functions

MapHashBuckets *(Key, T)*
extends HashBuckets <MapIterator<Key, T>>

- -map_: MapContainer<Key, T>
- +constructor(MapContainer<Key, T>)
- +find(Key) -> MapIterator

HashBuckets *(T)*

- #buckets_: Array<Array<T>>
- #item_size_: size_t
- +constructor()
- +reserve(size_t)
- +clear()
- +at(size_t) -> Array<T>
- -hash_index(T) -> size_t
- +insert(T)
- +erase(T)

Red-Black Tree

PairTree *(Key, T)*
extends XTree<Pair<Key, T>>

- -compare_: (Key, Key) => boolean
- +public constructor(Compare = std.less)
- +using super::find
- +find(MapIterator) -> XTreeNode
- template <XIterator extends MapIterator<Key, T>>
- +is_less(XIterator, XIterator) -> boolean
- +is_equal_to(XIterator, XIterator) -> boolean

XTree *(T)*

- #root_: XTreeNode<T>
- +public constructor()
- +find(T) -> XTreeNode
- #fetch_maximum(XTreeNode) -> XTreeNode
- -fetch_color(XTreeNode) -> Color
- +insert(T)
- +erase(T)
- -insert_case_{1~5}(XTreeNode)
- -erase_case_{1~6}(XTreeNode)
- #rotate_left(XTreeNode)
- #rotate_right(XTreeNode)
- #replace_node(XTreeNode, XTreeNode)
- *+is_less(T, T) -> boolean*
- *+is_equal_to(T, T) -> boolean*

XTreeNode *(T)*

- parent_: XTreeNode
- left_: XTreeNode
- right_: XTreeNode
- value: T
- color: Color
- +constructor(T, Color)

Exceptions

Exception

#what_: string

+constructor()
+constructor(string)

+what(): string

LoginError

extends Exception

+constructor(string)

DomainError

extends LogicError

+constructor(string)

InvalidArgument

extends LogicError

+constructor(string)

LengthError

extends LogicError

+constructor(string)

OutOfRange

extends LogicError

+constructor(string)

RuntimeError

extends Exception

+constructor(string)

OverflowError

extends RuntimeError

+constructor(string)

RangeError

extends RuntimeError

+constructor(string)

SystemError

extends RuntimeError

+constructor(ErrorCode)
+constructor(ErrorCode, string)
+constructor(number, ErrorCategory)
+constructor
(number, ErrorCategory, string)

+code() -> ErrorCode

UnderflowError

extends RuntimeError

+constructor(string)

ErrorCategory

+constructor()
+default_error_condition(number)
-> ErrorCondition

+name() -> string
+equivalent(number, ErrorCodition)
+equivalent(ErrorCode, number)

ErrorCondition

extends ErrorInstance

ErrorCode

extends ErrorInstance

ErrorInstance

#category_: ErrorCategory
#value_: number

+constructor()
+constructor(number, ErrorCategory)
+assign(number, category)
+clear()

+category() -> ErrorCategory
+value() -> number
+message() -> string

+default_error_condition()
-> ErrorCondition
+to_bool() -> boolean