theProject.CollectionInterface\<T\>

- add(element: T): boolean
- get(target: T): T
- contains(target: T): boolean
- remove(target: T): boolean
- isFull(): boolean
- isEmpty(): boolean
- size(): int

theProject.BSTInterface\<T\>

- min(): T
- max(): T
- getIterator(orderType: Traversal): Iterator\<T\>

theProject.BinarySearchTree\<T\>

- root: BSTNode\<T\>
- comp: Comparator\<T\>
- found: boolean
- start: double
- end: double

- BinarySearchTree(): void
- BinarySearchTree(comp: Comparator\<T\>): void
- isFull(): boolean
- isEmpty(): boolean
- min(): T
- max(): T
- recSize(node: BSTNode\<T\>): int
- size(): int
- size2(): int
- recContains(target: T, node: BSTNode\<T\>): boolean
- contains(target: T): boolean
- recGet(target: T, node: BSTNode\<T\>): T
- get(target: T): T
- recAdd(element: T, node: BSTNode\<T\>): BSTNode\<T\>
- add(element: T): boolean
- getPredecessor(subtree: BSTNode\<T\>): T
- removeNode(node: BSTNode\<T\>): BSTNode\<T\>
- recRemove(target: T, node: BSTNode\<T\>): BSTNode\<T\>
- remove(target: T): boolean
- getIterator(orderType: BSTInterface.Traversal): Iterator\<T\>
- preOrder(node: BSTNode\<T\>, q: LinkedQueue\<T\>): void
- inOrder(node: BSTNode\<T\>, q: LinkedQueue\<T\>): void
- postOrder(node: BSTNode\<T\>, q: LinkedQueue\<T\>): void
- iterator(): Iterator\<T\>

theProject.BSTNode\<T\>

- info: T
- left: BSTNode\<T\>
- right: BSTNode\<T\>

- BSTNode(info: T): void
- setInfo(info: T): void
- getInfo(): T
- setLeft(link: BSTNode\<T\>): void
- setRight(link: BSTNode\<T\>): void
- getLeft(): BSTNode\<T\>
- getRight(): BSTNode\<T\>

theProject.StackInterface\<T\>

- push(element: T): void
- pop(): void
- top(): T
- isEmpty(): boolean
- isFull(): boolean

theProject.QueueInterface\<T\>

- enqueue(element: T): void
- dequeue(): T
- isFull(): boolean
- isEmpty(): boolean
- size(): int

theProject.ALCollection<E>

- elements: E[]
- num_elements: int
- capacity: int
- start: double
- end: double

- ALCollection(size: int): void
- ALCollection(): void
- is_full(): boolean
- is_empty(): boolean
- size(): int
- add(e: E): boolean
- remove(i: int): E
- contains(e: E): boolean
- add(where: Where, index: int, e: E): boolean
- add(where: Where, e: E): boolean
- get(i: int): E
- reset(i: int, e: E): E
- grow(): boolean
- show(): void

theProject.myInterface<E>

- is_full(): boolean
- is_empty(): boolean
- size(): int
- add(e: E): boolean
- remove(i: int): E
- contains(e: E): boolean
- reset(i: int, e: E): E
- show(): void