Lab 2 data
Red-Black Tree

Names: Ahmed Talaat Noser ID: 05
Salma Ragab Gad 32

Time analysis:

- Red-Black tree has elements n, so the space is O(n).
- Height is O(log n) -the worst case is $O(2 \log n_b)$- where n_b is the number of black nodes.

- Rotation:
  - is O(1) since a constant number of pointers are modified.

- Insertion:
  - To find the place to insert takes the height of the tree, or *O(log n)*.
  - To add the node is *O(1)*.
  - To fix double red , rotation is O(1).
  - Worst case, the double red can cascade all the way to the root. The cascade is proportional to the height of the tree, so the fixing takes *O(logn)*, worst case.
  - Therefore, insertion is *O(log n)*.

- Deletion:

- Finding the node to delete plus finding the left-most right descendant is proportional to the height of the tree, so it is *O(log n)*.
- The swaping and deletion is *O(1)*.
- Each individual fix (rotation, etc.) is *O(1)*.

- In the worst case, a double-black may get passed up to the root. Since each rotation takes constant time, this would be proportional to the height of the tree, and thus is *O(log n)*.
- Therefore, the worst case cost of deletion is *O(log n)*.

- Search & Contains:
- Take the height of the tree. Therefore, they are *O(log n).*

- CeilingEntry: height of the tree O(log n).

- FloorEntry: height of the tree O(log n).

- EntrySet: uses inorder traversal O(n).

- HeadMap: uses inorder traversal, so it's O(n).

- FirstEntry: height of the tree O(log n).

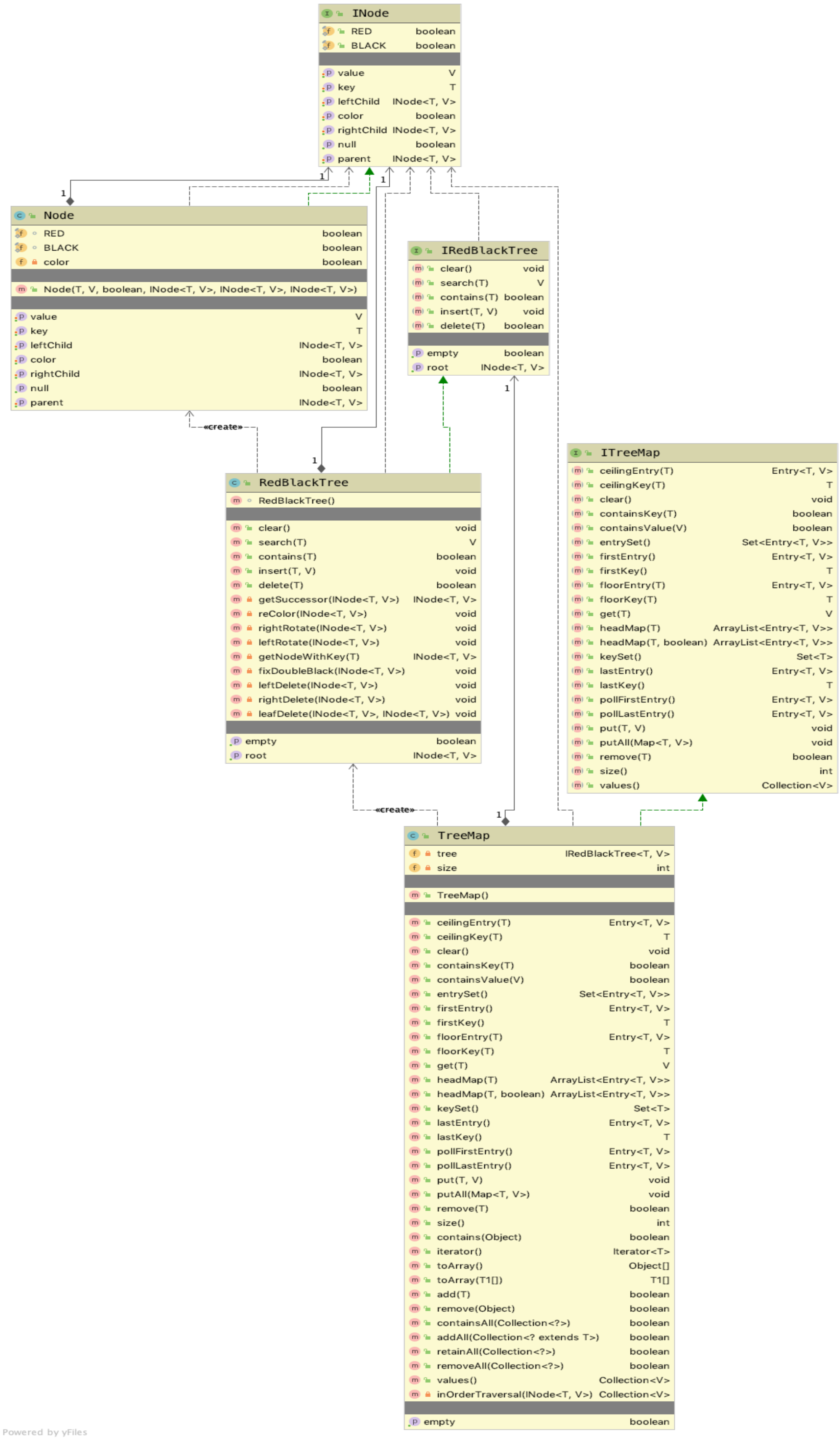
- LastEntry: height of the tree O(log n).

- pollFirstElement:
- Find least key in O(log n).
- Then remove it in O(log n).
- So it takes O(log n).

- PollLastEntry:
- Find greatest key in O(log n).
- Then remove it in O(log n).
- So it takes O(log n).

- Values: uses inorder traversal, so it’s O(n).

UML diagram:



INode
RED boolean
BLACK boolean
value V
key T
leftChild INode<T, V>
color boolean
rightChild INode<T, V>
null boolean
parent INode<T, V>
1
1
1
Node
RED boolean
BLACK boolean
color boolean
Node(T, V, boolean, INode<T, V>, INode<T, V>, INode<T, V>)
value V
key T
leftChild INode<T, V>
color boolean
rightChild INode<T, V>
null boolean
parent INode<T, V>
IRedBlackTree
clear() void
search(T) V
contains(T) boolean
insert(T, V) void
delete(T) boolean
empty boolean
root INode<T, V>
1
«create»
1
RedBlackTree
RedBlackTree()
clear() void
search(T) V
contains(T) boolean
insert(T, V) void
delete(T) boolean
getSuccessor(INode<T, V>) INode<T, V>
reColor(INode<T, V>) void
rightRotate(INode<T, V>) void
leftRotate(INode<T, V>) void
getNodeWithKey(T) INode<T, V>
fixDoubleBlack(INode<T, V>) void
leftDelete(INode<T, V>) void
rightDelete(INode<T, V>) void
leafDelete(INode<T, V>, INode<T, V>) void
empty boolean
root INode<T, V>
ITreeMap
ceilingEntry(T) Entry<T, V>
ceilingKey(T) T
clear() void
containsKey(T) boolean
containsValue(V) boolean
entrySet() Set<Entry<T, V>>
firstEntry() Entry<T, V>
firstKey() T
floorEntry(T) Entry<T, V>
floorKey(T) T
get(T) V
headMap(T) ArrayList<Entry<T, V>>
headMap(T, boolean) ArrayList<Entry<T, V>>
keySet() Set<T>
lastEntry() Entry<T, V>
lastKey() T
pollFirstEntry() Entry<T, V>
pollLastEntry() Entry<T, V>
put(T, V) void
putAll(Map<T, V>) void
remove(T) boolean
size() int
values() Collection<V>
«create»
1
TreeMap
tree IRedBlackTree<T, V>
size int
TreeMap()
ceilingEntry(T) Entry<T, V>
ceilingKey(T) T
clear() void
containsKey(T) boolean
containsValue(V) boolean
entrySet() Set<Entry<T, V>>
firstEntry() Entry<T, V>
firstKey() T
floorEntry(T) Entry<T, V>
floorKey(T) T
get(T) V
headMap(T) ArrayList<Entry<T, V>>
headMap(T, boolean) ArrayList<Entry<T, V>>
keySet() Set<T>
lastEntry() Entry<T, V>
lastKey() T
pollFirstEntry() Entry<T, V>
pollLastEntry() Entry<T, V>
put(T, V) void
putAll(Map<T, V>) void
remove(T) boolean
size() int
contains(Object) boolean
iterator() Iterator<T>
toArray() Object[]
toArray(T1[]) T1[]
add(T) boolean
remove(Object) boolean
containsAll(Collection<?>) boolean
addAll(Collection<? extends T>) boolean
retainAll(Collection<?>) boolean
removeAll(Collection<?>) boolean
values() Collection<V>
inOrderTraversal(INode<T, V>) Collection<V>
empty boolean
Powered by yFiles