Sets, Maps, and Trees

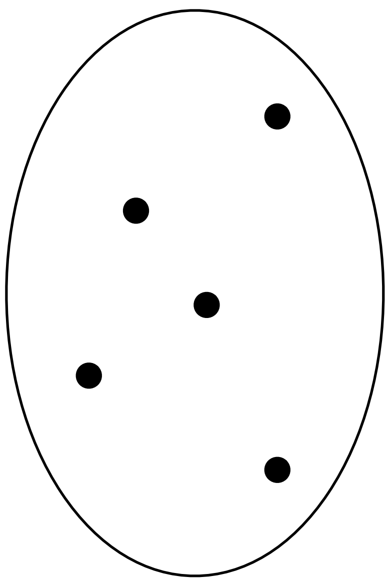
Overview

Sets and maps
  Definitions
  Implementations
Binary trees
  Definitions
  Size
  Traversal
Binary search trees
  Definition
  Search
  Insertion
  Deletion
The Java collections framework

Sets and maps

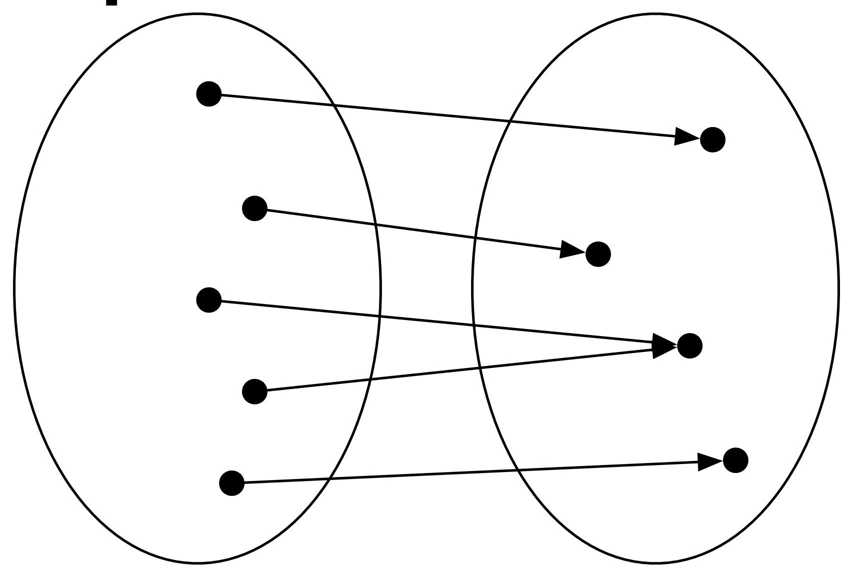
Definitions

Set



```
add(item)
contains(item)
remove(item)
```

Map



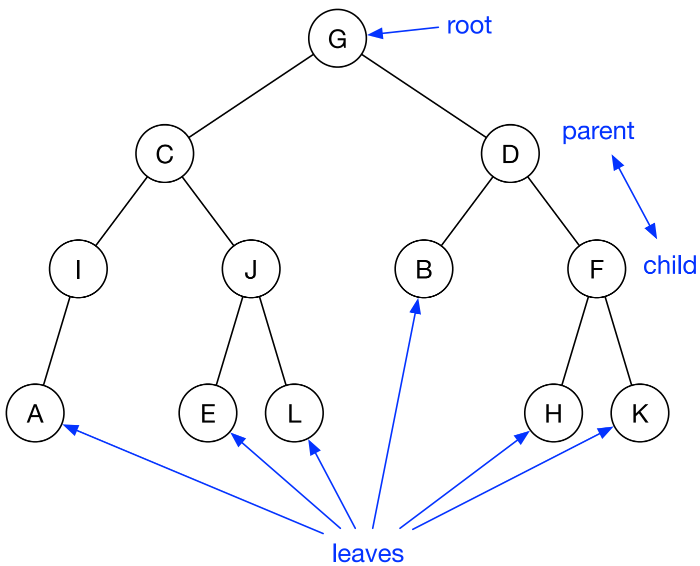
```
get(key)
put(key, value)
remove(key)
```

Set/map implementations

	Search	Insertion	Deletion	Notes
Unsorted array	$\Theta(n)$	$\Theta(1)$ amortized	$\Theta(n)$	
Sorted array	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n)$	
Binary search tree	$\Theta(\log n)$ average, $\Theta(n)$ worst case (common)			
Balanced search tree	$\Theta(\log n)$			
Hash table	$\Theta(1)$ average, $\Theta(n)$ worst case (very unlikely)			No way to traverse in order
Direct addressing table (array)	$\Theta(1)$			Keys must come from small integer range

Binary trees

Definitions



A binary tree is either:

- empty, or
- a node plus left and right subtrees, themselves each binary trees.

Depth of a node is number of *edges* (lines) on path from root.
Height of tree is depth of deepest node.

Size

A binary tree with n nodes has height between $\log_2(n + 1) - 1$ and $n - 1$.

A binary tree of height h has between $h + 1$ and $2^{h+1} - 1$ nodes.

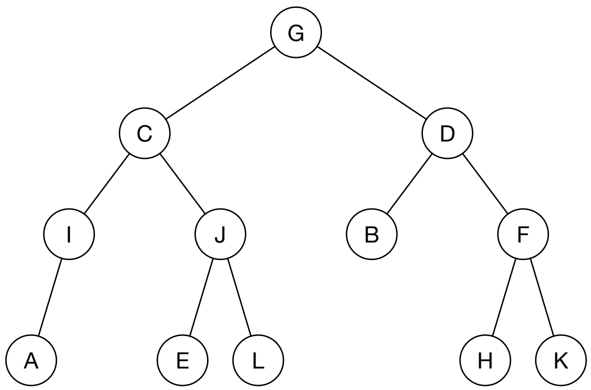
For a perfect tree (all non-leaves have two children, all children at same depth):

$h \in \Theta(\log n)$
$n \in \Theta(2^h)$

For an extremely skewed tree (e.g., all children are right children):

$h \in \Theta(n)$
$n \in \Theta(h)$

Traversal



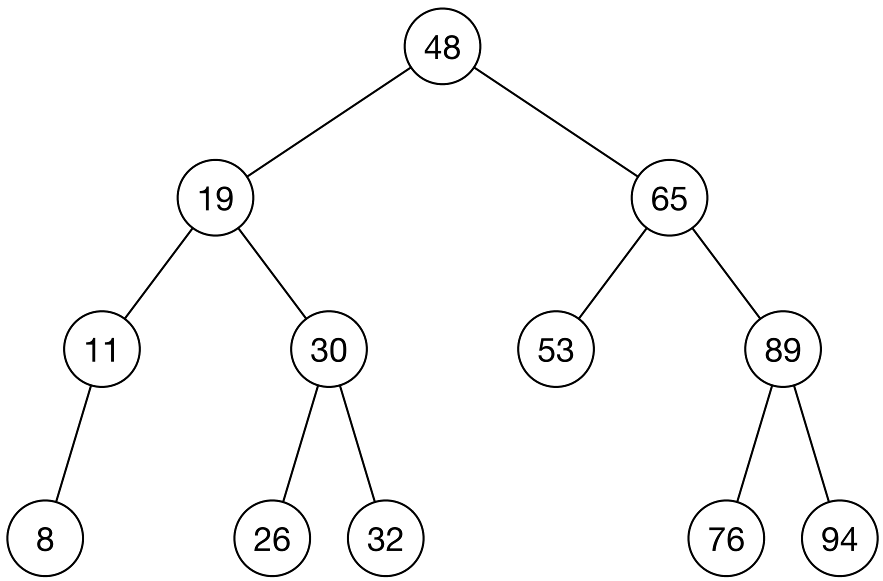
Preorder (root, left, right): G C I A J E L D B F H K
Inorder (left, root, right): A I C E J L G B D H F K
Postorder (left, right, root): A I E L J C B H K F D G
Level order (left to right across rows): G C D I J B F A E L H K

The first three are depth-first and implemented with a stack (or recursion).

The last one is breadth-first and implemented with a queue.

Binary search trees

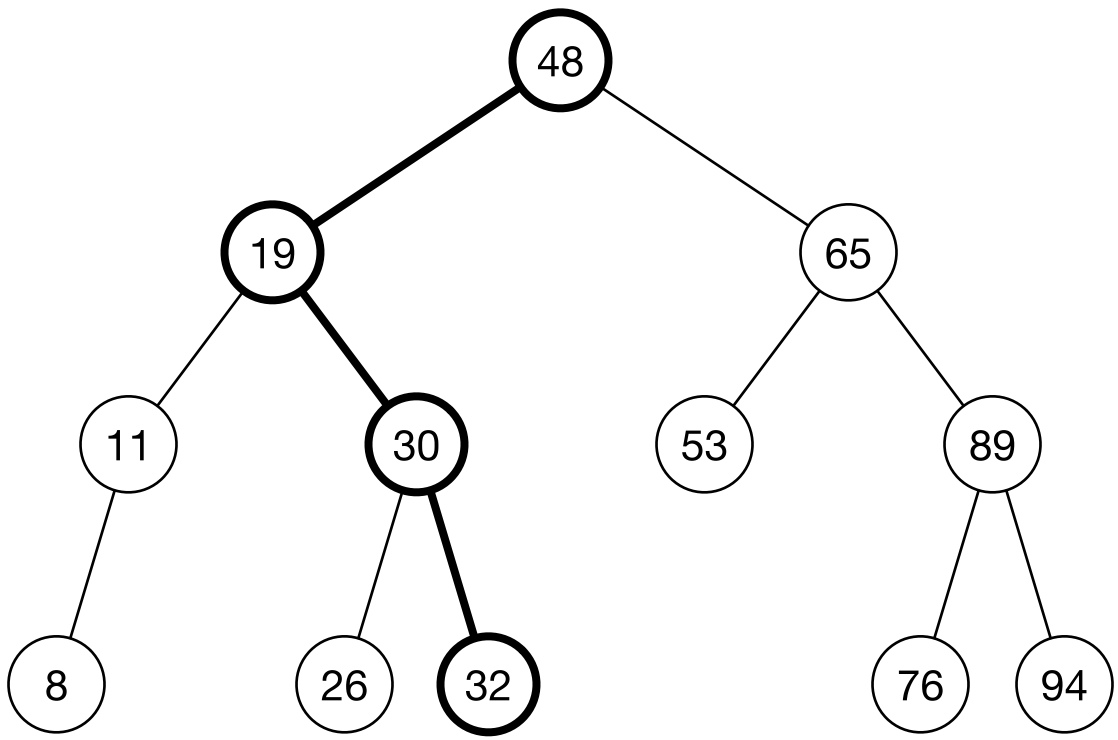
Definition



A binary search tree is a binary tree where:

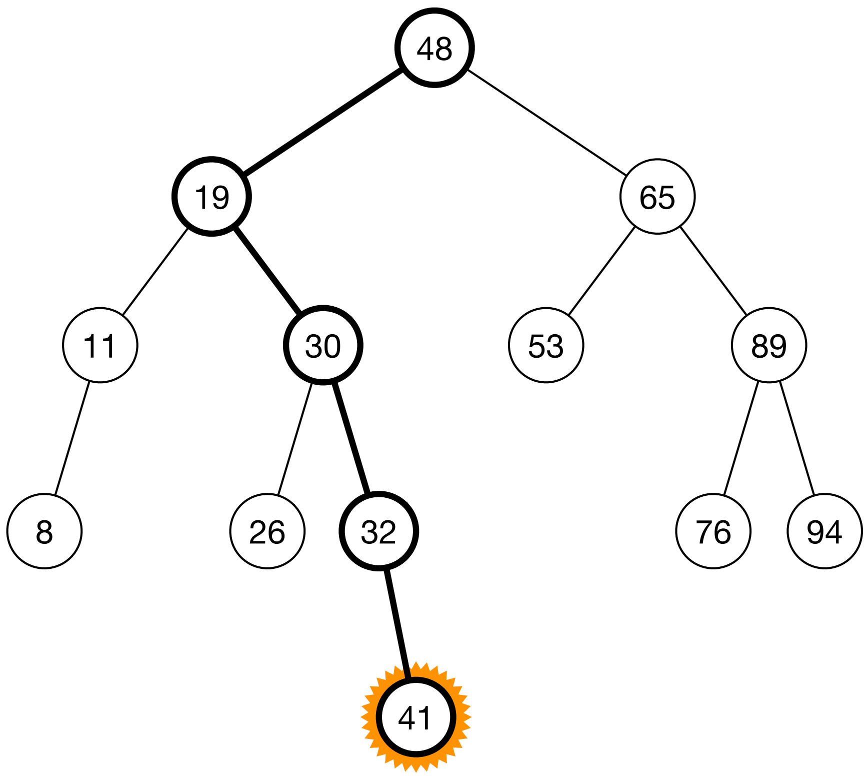
- Everything to the left of the root is less than the root
- Everything to the right of the root is greater than the root
- The subtrees are themselves binary search trees.

Search

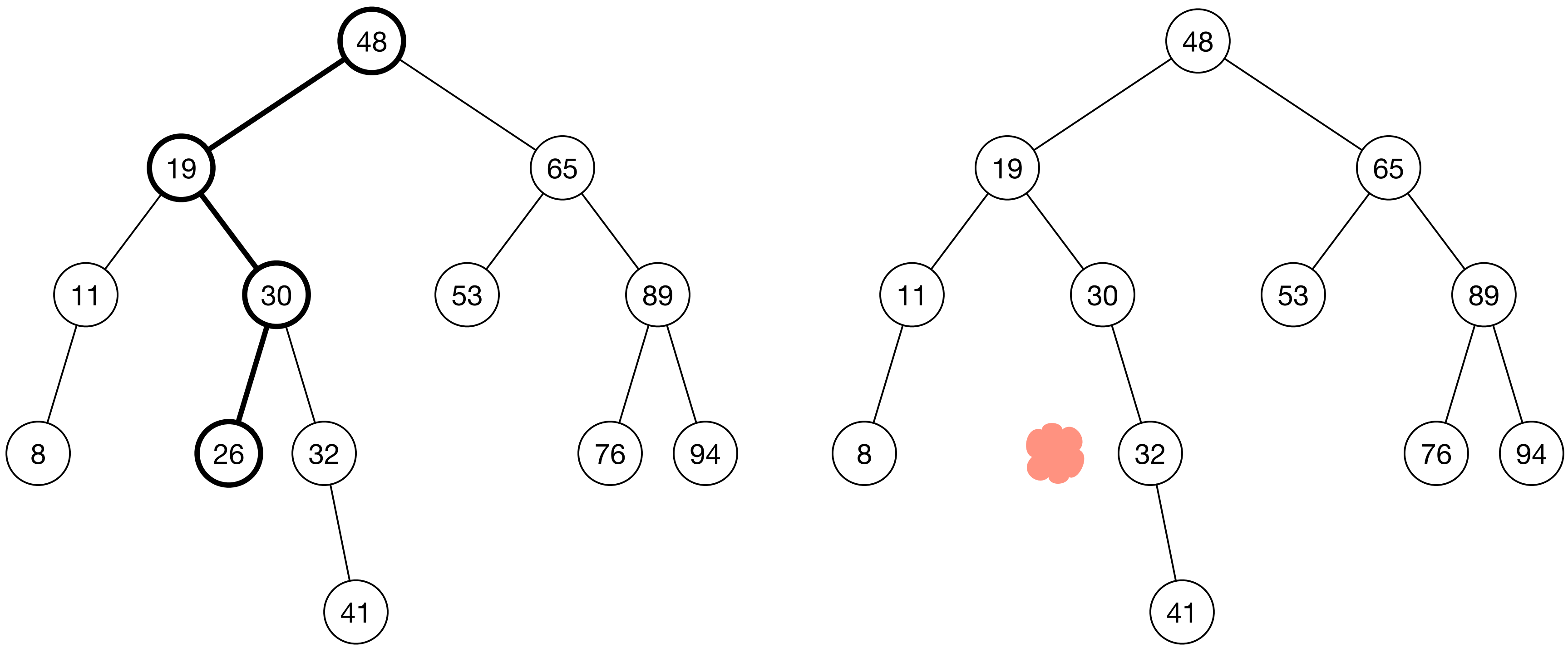


This takes logarithmic time on average, linear time in the worst case.

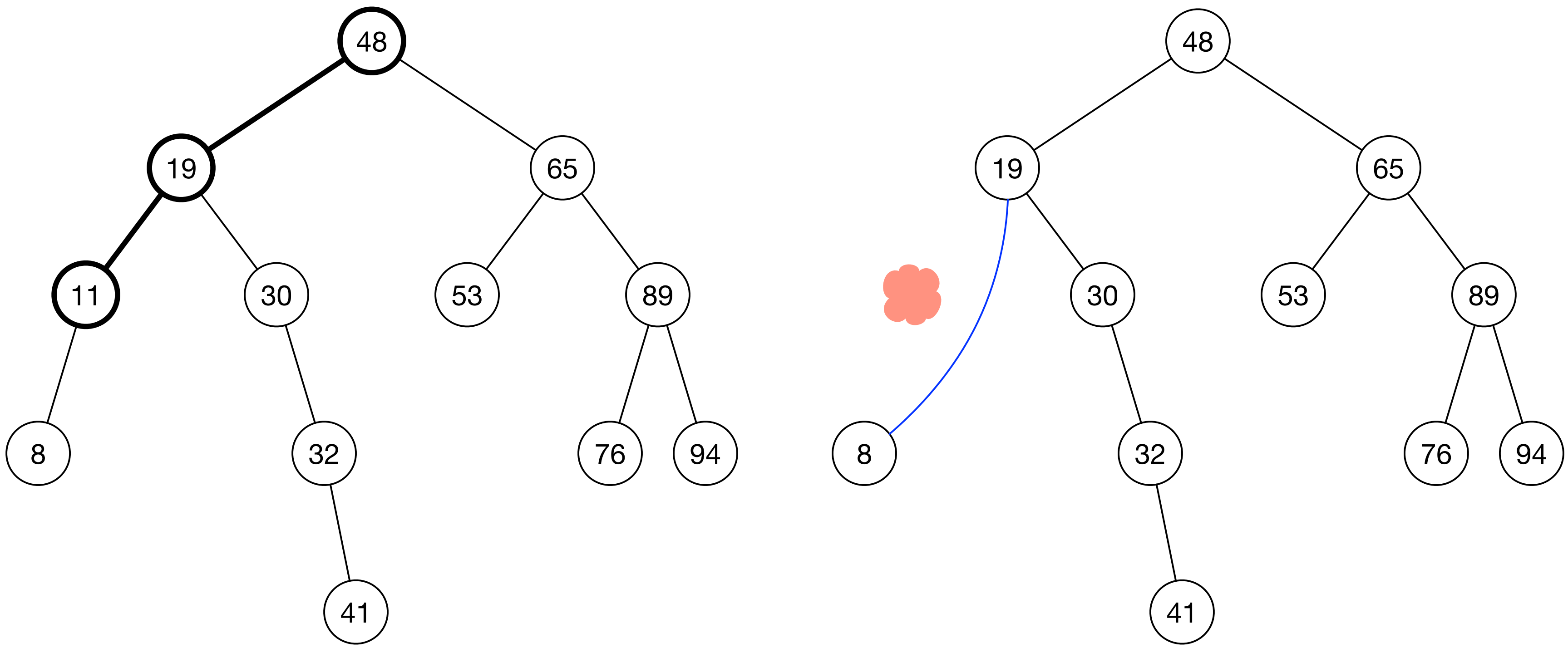
Insertion



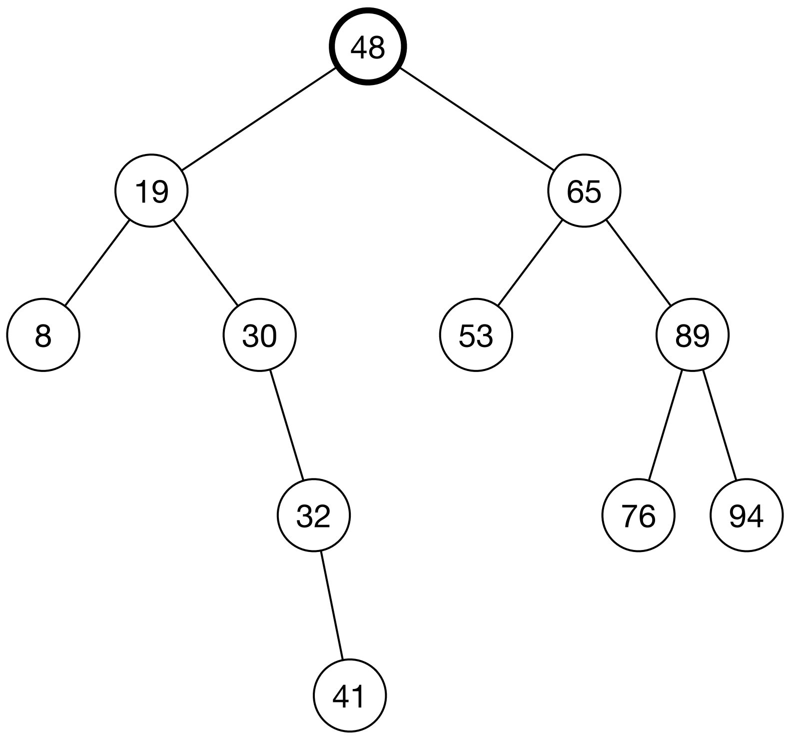
Deletion: leaf

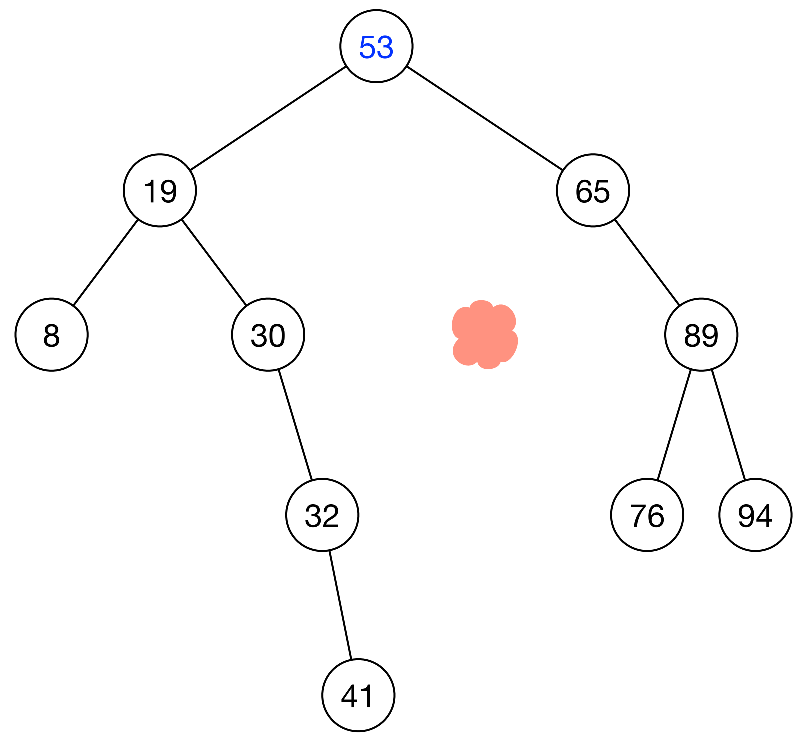


Deletion: one child

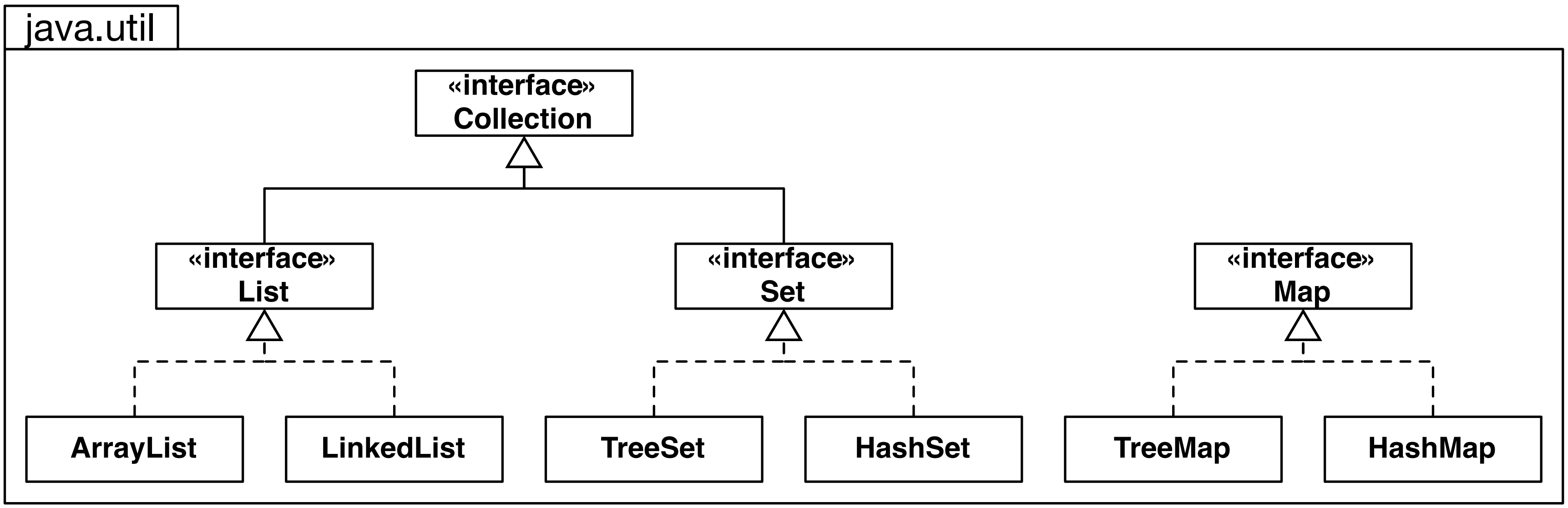


Deletion: two children





The Java collections framework



Review

Sets are collections of elements without duplicates.

Maps associate keys with values.

There are many implementations of each.

Binary trees are recursive data structures.

Binary search trees perform set/map operations in logarithmic time.

The Java collections framework contains many useful classes.