Ordered Sets

If you need an ordered set, you'll want to consider either the **LinkedHashSet** or the **TreeSet**.

A LinkedHashSet maintains the insertion order of the elements.

The TreeSet is a sorted collection, sorted by the natural order of the elements, or by specifying the sort during the creation of the set.

The LinkedHashSet

The LinkedHashSet **extends the HashSet** class.

It maintains relationships between elements with the use of a doubly linked list between entries.

The **iteration order** is therefore the same as the **insertion order** of the elements, meaning the order is **predictable**.

All the methods for the LinkedHashSet are the same as those for the HashSet.

Like HashSet, it provides constant-time performance, O(1), for the add, contains and remove operations.

TreeSet

A TreeSet's class, uses a data structure that's a derivative of what's called a binary search tree, or Btree for short, which is based on the concept and efficiencies of the binary search.

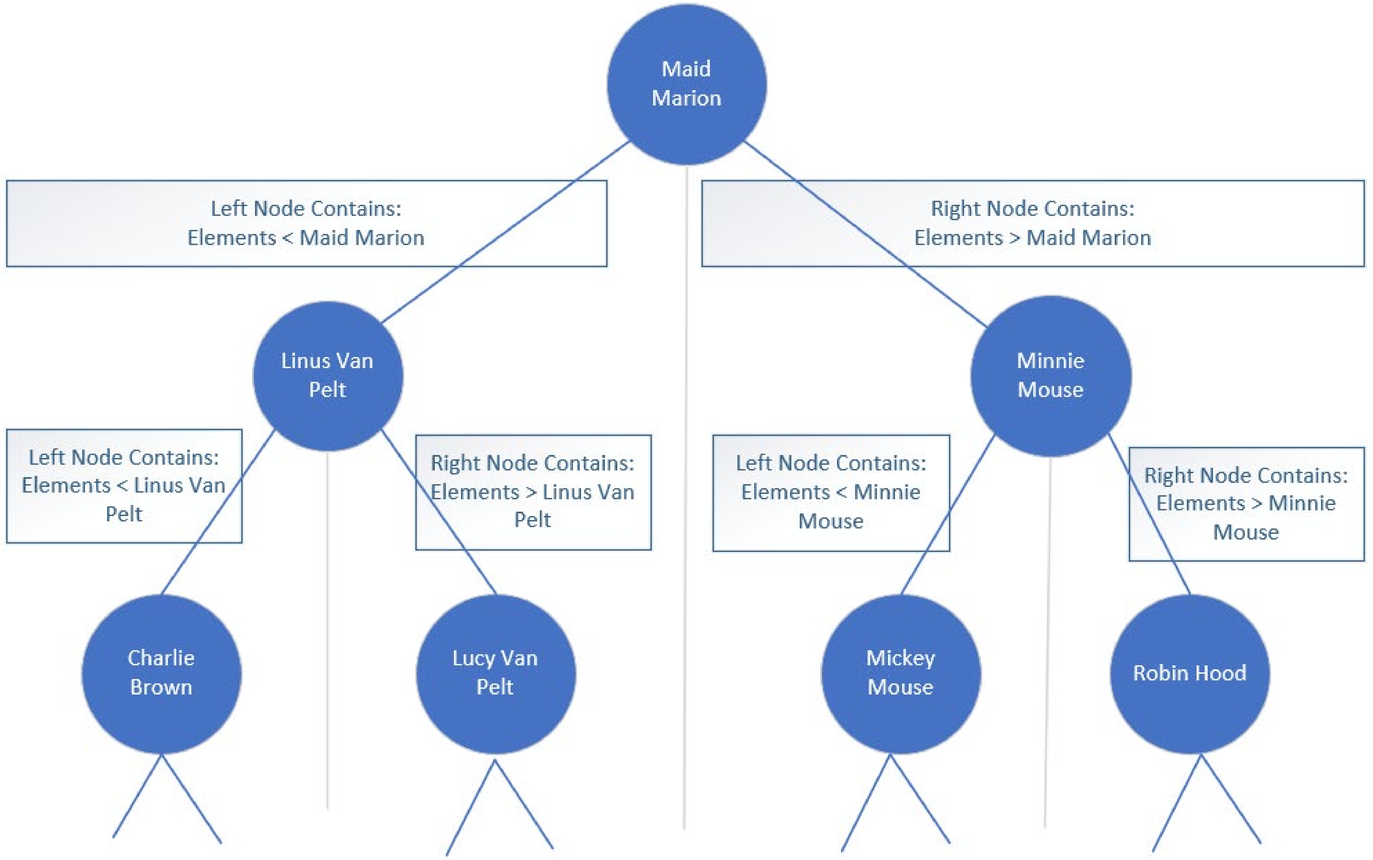
I've discussed the binarySearch method on the List, as well as the java.util.Collections class, and shown that this type of search is very fast, if the elements are sorted.

This search iteratively tests the mid range of a group of elements to be searched, to quickly find its element, in a collection.

TreeSet

As elements are added to a TreeSet, they're organized in the form of a tree, where the top of the tree represents that mid point of the elements.

This slide shows a conceptual example, using some of the character contacts from my last samples of code.

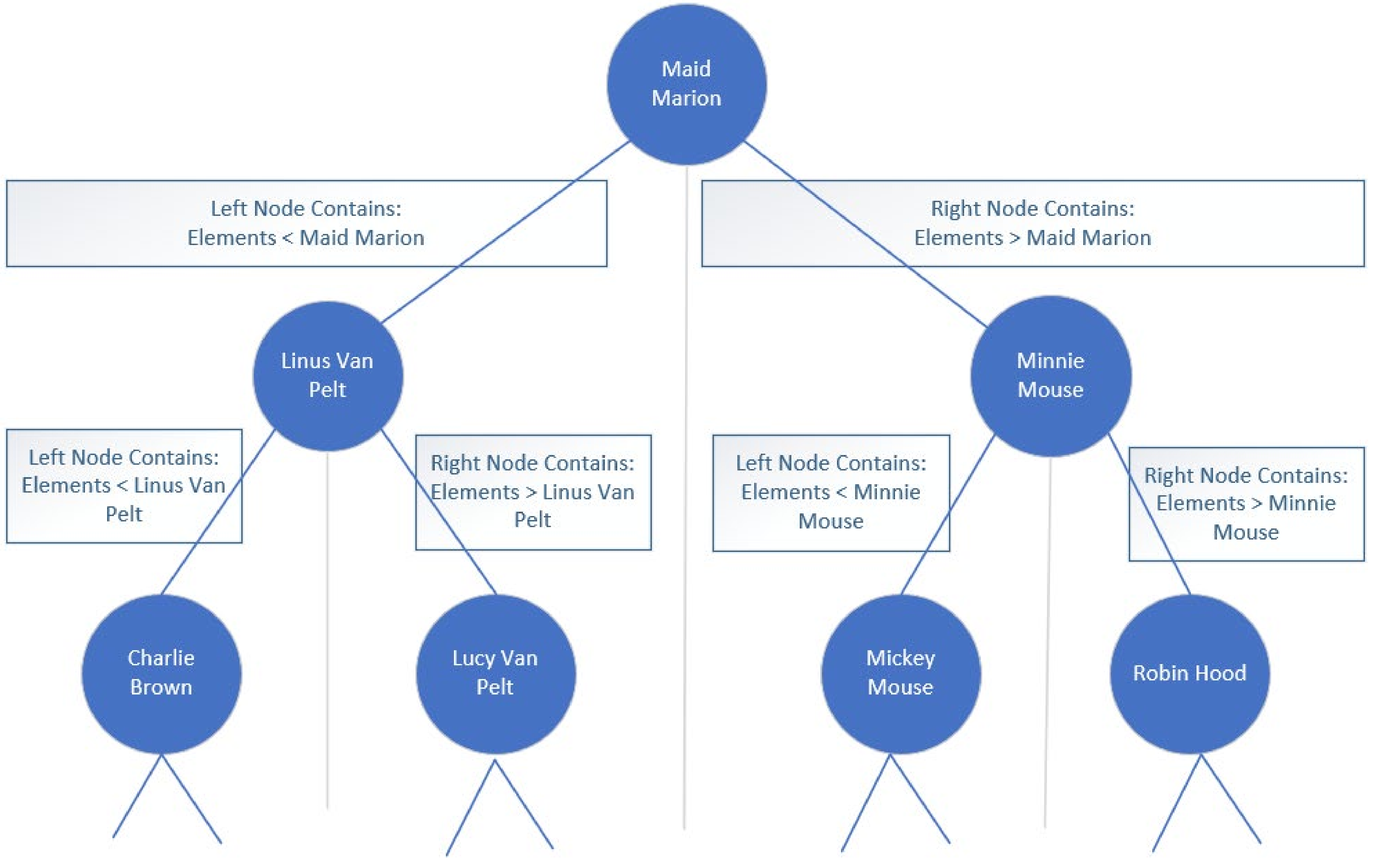


TreeSet

Further binary divisions become nodes under that.

The **left** node and its children are elements that are **less than** the parent node.

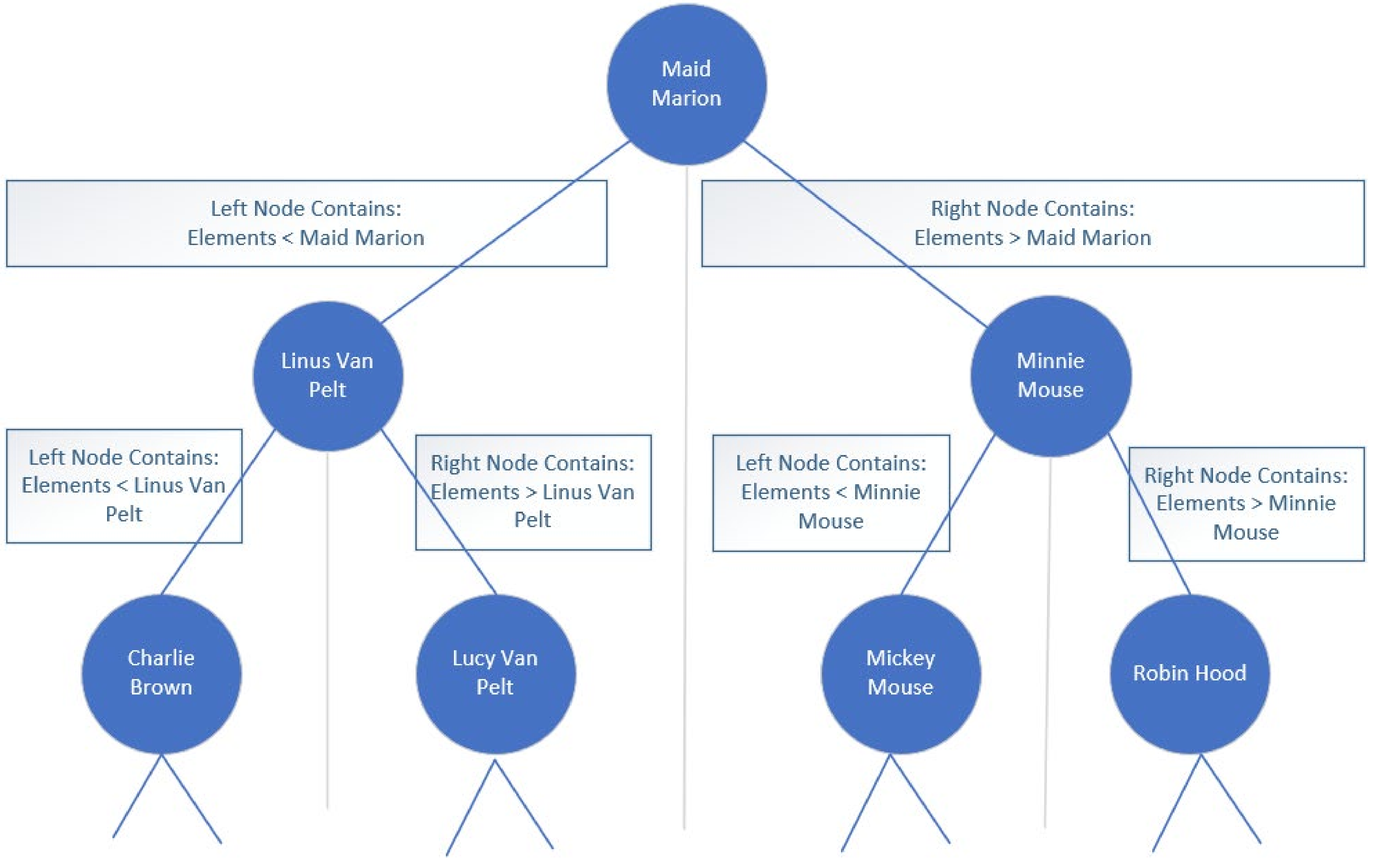
The **right** node and its children are elements that are **greater than** the parent node.



TreeSet

Instead of looking through all the elements in the collection to locate a match, this allows the tree to be quickly traversed, each node a simple decision point.

The main point is the tree remains balanced as elements are added.



TreeSet O Notation

You'll remember that O(1) is constant time, meaning the time or cost of an operation doesn't change, regardless of how many elements are processed.

O(n) is linear time, meaning it grows in line with the way the collection grows.

Another notation, is O(log(n)), which means the cost falls somewhere in between constant and linear time.

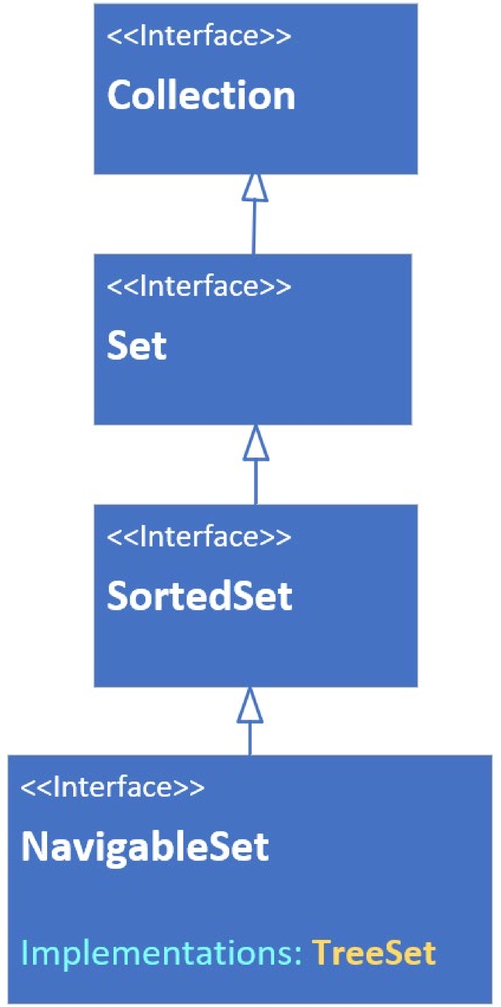
The TreeSet promises O(log(n)) for the add, remove, and contains operations, compared to the HashSet which has constant time O(1) for those same operations.

The TreeSet interface hierarchy

A TreeSet can be declared or passed to arguments typed with any of the interfaces shown on this slide.

This class is sorted, and implements the SortedSet interface, which has such methods as first, last, headSet and tailSet, and comparator.

This set also implements the NavigableSet Interface, so it has methods such as ceiling, floor, higher, lower, descendingSet and others.



The TreeSet relies on Comparable or Comparator methods

Elements which implement Comparable (said to have a natural order sort, like Strings and numbers) can be elements of a TreeSet.

If your elements don't implement Comparable, you must pass a Comparator to the constructor.