NavigableSet methods to get closest matches

All the methods shown on this slide take an element as an argument, and return an element in the set, the closest match to the element passed.

Element passed as the argument	Result From Methods			
	floor(E) (<=)	lower(E) (<)	ceiling(E) (>=)	higher(E) (>)
In Set	Matched Element	Next Element < Element or null if none found	Matched Element	Next Element > Element or null if none found
Not in Set	Next Element < Element	Next Element < Element or null if none found	Next Element > Element	Next Element > Element or null if none found

Getting subsets from a TreeSet.

All three methods, headSet, tailSet and subSet return a subset of elements, backed by the original set.

sub set methods	inclusive	description

`headSet(E toElement)`
 	
`headSet(E toElement, boolean inclusive)`
	toElement is exclusive if not specified	returns all elements less than the passed toElement (unless inclusive is specifically included).

`tailSet(E fromElement)`
 	
`tailSet(E toElement, boolean inclusive)`
	fromElement is inclusive if not specified	returns all elements greater than or equal to the fromElement (unless inclusive is specifically included).

`subSet(E fromElement, E toElement)`
 	
`subSet(E fromElement, boolean fromInclusive, E toElement, boolean toInclusive)`
	fromElement is inclusive if not specified,
	
	toElement is exclusive if not specified
		returns elements greater than or equal to fromElement and less than toElement.

When would you use a TreeSet?

The TreeSet does offer many advantages, in terms of built-in functionality over the other two Set implementations, but it does come at a higher cost.

If your number of elements is not large, or you want a collection that's sorted, and continuously re-sorted as you add and remove elements, and that shouldn't contain duplicate elements, the TreeSet is a good alternative to the ArrayList.