

# Collections with Uniqueness: Sets

---



**Richard Warburton**  
Java Champion, Author and Programmer

@richardwarburto [www.monotonic.co.uk](http://www.monotonic.co.uk)



Sets are collections of distinct elements. There are no duplicates



# Outline

## **Set Features**

The Why and How of Sets

## **Hashcode and Equals**

Understanding the contract behind HashSet

## **Set**

## **Implementations**

Performance tradeoffs and features



# Set Features

---



# Hashcode and Equals

---



# Signatures

graphs for which the Client grants the  
Photographer the right to use, copy,  
provide, license, publish, exhibit, display  
and otherwise use the Client's name and  
graphs, and hereby reconsents to the  
use of the Client's name and graphs in  
any manner or media now known or  
hereafter devised.

  
Photographer's Signature



`object.equals(other)`



`object.hashCode() == other.hashCode()`

Hashcode / Equals Contract

**One way implication**



# Equality

It can be reference based or value based. Reference based just needs to inherit equals from Object. Value based requires a custom equals method.



```
result = 31 * result +  
obj.hashCode();
```

◀ Combine hashCode information from each field

```
// Arrays
```

```
Arrays.hashCode()
```

◀ IDE Can auto-generate

```
// Primitives (Java 8+)
```

```
Long.hashCode(longValue)
```

◀ Objects.hash() (Java 7+)

```
// Old Primitives
```

```
(int) (l ^ (l >>> 32))  
Float.floatToIntBits(f);
```

◀ ALWAYS use the same fields as equals()



# Set Implementations

---



# HashSet

**Based upon HashMap**

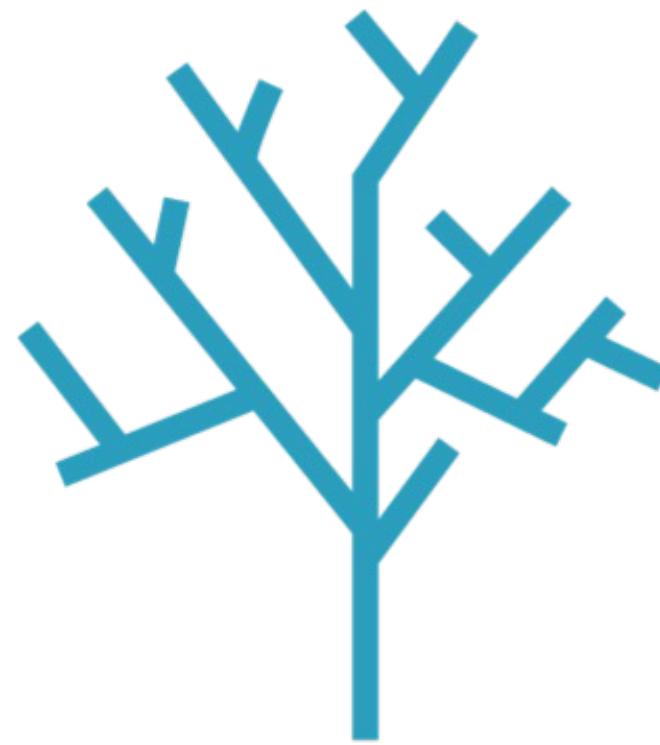
**Uses hashCode() and looks up  
location**

**Good General Purpose  
Implementation**

**Use by default**



# TreeSet



**Based Upon TreeMap**  
**Red/Black binary tree with defined**  
**sort order**



**Provides Extra Features**  
**Implements SortedSet and**  
**NavigableSet**

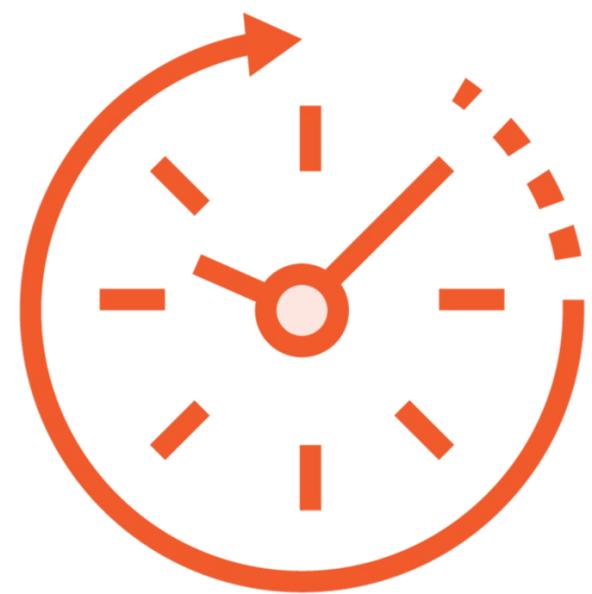


# Performance Comparison

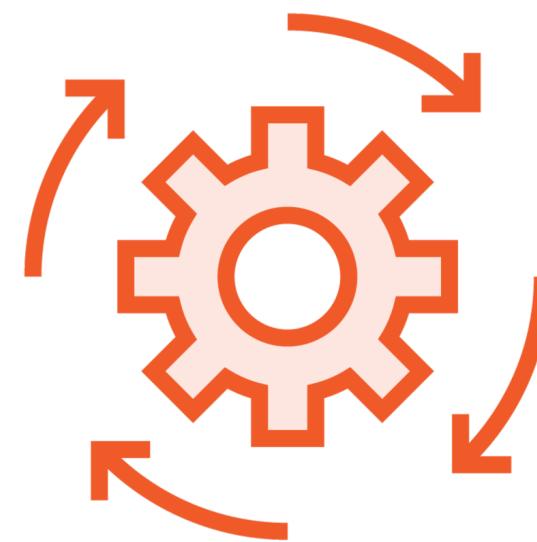
	<b>add</b>	<b>contains</b>	<b>next</b>
HashSet	$O(N)$ , $\Omega(1)$	$O(\log(N))$ , $\Omega(1)$	$O(\text{Capacity}/N)$
TreeSet	$O(\log(N))$	$O(\log(N))$	$O(\log(N))$



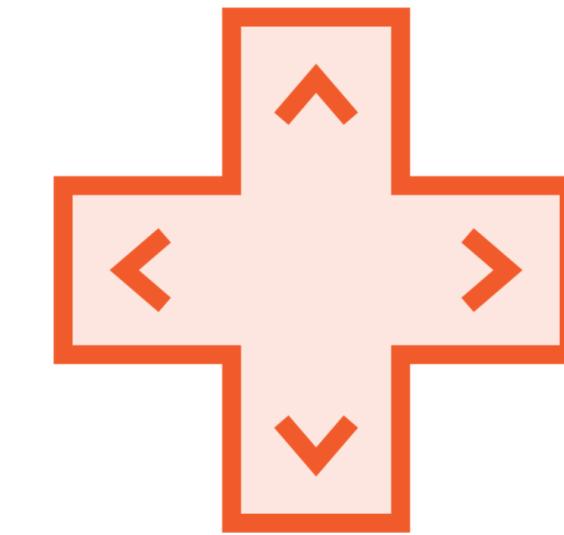
# LinkedHashSet



**When**  
**Copying Set to modify**  
**Deduping List or Queue**



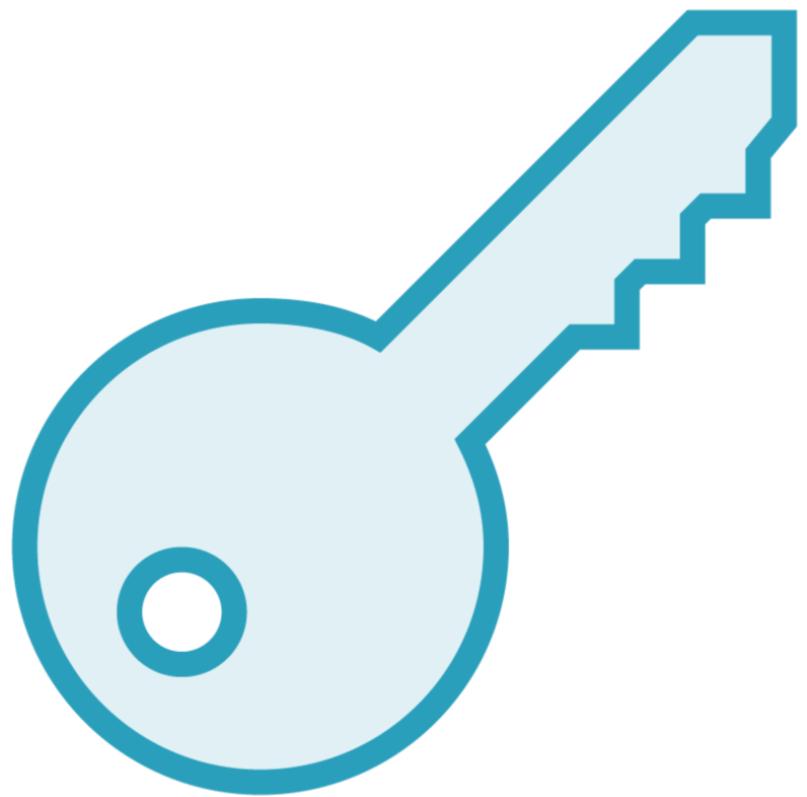
**Maintains Order**  
**Only Insertion**



**Overhead**  
**Slower than HashSet,**  
**less memory than**  
**TreeSet**



# EnumSet



**Keys are Enums**  
**Faster and Low memory usage**



**Bitset Implementation**  
**Only a single long if < 64 elements**



# SortedSet and NavigableSet

---



```
E first();
```

```
E last();
```

```
SortedSet tailSet(E fromElement);
```

```
SortedSet headSet(E toElement);
```

```
SortedSet subSet(E fromElement, E toElement);
```

SortedSet

**Defines an order**

**No indexes, but subset views possible.**

```
E lower(E e);
```

```
E higher(E e);
```

```
E floor(E e);
```

```
E ceiling(E e);
```

```
E pollFirst();
```

```
E pollLast();
```

## NavigableSet

**Extends SortedSet**

**Provides ways to move through the order**

**Implemented by TreeSet**

# Conclusion

---



## Summary



**Sets are a commonly used collection**

**Different implementations for different purposes**

**Remember to get the hashCode/equals contract correct**







