**Sets**  
\* The last thing we’re going to look at very briefly is Sets.  
\* Sets are an **Abstract Data Type**.  
=> **They can really apply to any data structure**.  
\* All a Set is, is a **data set that doesn’t contain duplicates**.  
\* And so **if something is a Set, it cannot contain duplicate values**.  
**Set** => **A collection that contains no duplicate elements**.  
<https://docs.oracle.com/javase/9/docs/api/java/util/Set.html>  
\* **For Java, that means no pair of elements can exist in the Set such that e1.equals(e2) == true**.  
\* And so if the equals() method is going to return true for 2 instances, then those 2 instances cannot belong to the same Set.  
\* **The Set in JDK can contain 1 null element**.  
**AbstractSet** => just like with the other interfaces, if you want to implement your own customized Set, **rather than implementing the Set interface**, if you **extend AbstractSet**, you’ll get a bit of a **headstart**.  
**HashSet** => implementation of Set that’s **backed by a Hash Table, actually a HashMap**.  
<https://docs.oracle.com/javase/9/docs/api/java/util/HashSet.html> **LinkedHashSet** => a **Hash** **Table** and **Linked** **List** implementation of the Set interface.  
<https://docs.oracle.com/javase/9/docs/api/java/util/LinkedHashSet.html>  
**TreeSet** => a **NavigableSet implementation** based on a **TreeMap**.  
<https://docs.oracle.com/javase/9/docs/api/java/util/TreeSet.html>  
\* And so **if you wanted to build a Tree that has no duplicate elements** in it, you could use TreeSet.  
\* There’s lots of support for Sets in the JDK.  
\* We see the methods for example:  
**add()** => adds the specified element to this set if it is not already present.  
**contains()  
iterator()  
remove()  
toArray()**  
\* So there’s really nothing new here and that’s why I didn’t want to spend a lot of time on it.  
\* Because if you understand the other data structures we’ve looked at, this is sort of an extra layer of a requirement and that requirement is just that you can’t add duplicate items.

**Course Wrap-Up**  
\* We covered quite a lot in this course.  
\* We covered many popular **Sort** **Algorithms**.  
\* We covered **Linear** **Search** and **Binary** **Search**.  
\* We covered a bunch of **Data** **Structures**.  
1. **array**  
2. **Linked List**  
3. **Stack**  
4. **Queue**  
5. **Hash** **Table**  
6. **Tree**  
7. **Heap**  
8. **Set**  
\* Now of course we didn’t cover every data structure, we didn’t cover some of the more advanced Trees, we didn’t cover Graphs, Graphs could be a course in themselves.  
\* But we covered all the basic Data Structures, and a lot of applications will never need a Tree or a Graph, they’ll use arrays or Hash Tables or Sets or Stacks or Queues, and maybe a Linked List.  
\* I hope you have a better understanding of the Data Structures we covered and their strengths and weaknesses.  
\* Now when you need to sort some data, you’ll have a selection of Sort Algorithms to choose from.

**Tell Us What Content You Want Us To Add!**

**Resources**  
Set interface javadoc  
<https://docs.oracle.com/javase/9/docs/api/java/util/Set.html>  
HashSet class javadoc  
<https://docs.oracle.com/javase/9/docs/api/java/util/HashSet.html>  
LinkedHashSet class javadoc  
<https://docs.oracle.com/javase/9/docs/api/java/util/LinkedHashSet.html>  
TreeSet class javadoc  
<https://docs.oracle.com/javase/9/docs/api/java/util/TreeSet.html>