$$\begin{tikzcd}
 & \text{Collection} \arrow[r] \arrow[dl] \arrow[d] \arrow[dr] & \text{Interface} \\
\text{List} & \text{Set} & \text{Queue}
\end{tikzcd}$$

Sorted Set

Queue

DeQueue

Pairs

Map (Key, Value Pairs

Sorted Map

Interfaces

↳ Multiple Data Structures

Functional Characteristics

Prefer as Variable type

Popular Implementation

Implementation

-> Specific Data Structure

-> Performance characteristics

Concrete & Instantiable

Collection Behaviours:

$$\begin{tikzcd}
\text{Iterable} \arrow[d, dash] \arrow[dr] & \\
& \boxed{\text{Iterator}} \\
\boxed{\text{Collection}} &
\end{tikzcd}$$

Size()
isEmpty()
add(Element)
addAll(Collection)
remove(element)
removeAll(Collection)
retainAll(Collection)
Contains(element)
ContainsAll(Collection)
clear()

List<E> subList(int fromIndex, int toIndex);

	get	add	Contains	next	remove
ArrayList	O(1)	O(N), Ω(1)	O(N)	O(1)	O(N)
Linked list	O(N)	O(1)	O(N)	O(1)	O(1)

result = 31 * ~~object~~ result + obj.hashCode();

Navigable Set $\overset{\text{Extends}}{\&}$ Sorted Set Implemented by TreeSet
↳ Collection with distinct Elements that have Order

Double Ended Queues → DeQue.

ArrayDeque
↳ Less memory, faster
→ No random access

LinkedList
↳ Has random access.
→ Allows null Elements

map → key, Values
↓
Unique

Sorted Maps
Defines an Interface for a map with ordering
Can't override KeySet

remove(key, value)

Views over Maps:
KeySet(); EntrySet()

replace(Key, value)
replaceAll(BiFunction <K, V, V>)
remove(key, value)