

Collection → Interface.

List

Set

Sorted Set

Queue

Deque

Map ^{Pair} (key, value)
Pairs

Sorted Map

Interfaces

↳ Multiple Data Structures

Functional Characteristics

Prefer as Variable type

Popular Implementation

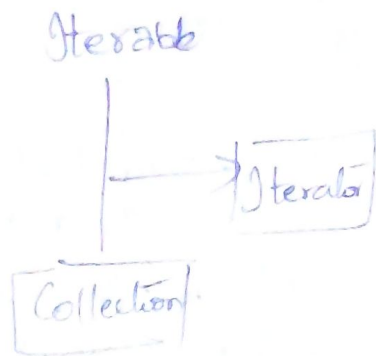
Implementation

→ Specific Data Structure

→ Performance characteristics

Concrete & Instantiable

Collection Behaviours



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	$O(1)$	$O(N), O(1)$	$O(N)$	$O(1)$	$O(N)$
LinkedList	$O(N)$	$O(1)$	$O(N)$	$O(1)$	$O(1)$

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

Navigable Set ^{Extends} Sorted Set Implemented by TreeSet
 ↳ Collection with distinct elements that has order
 Double Ended Queues → Deque.

Array Deque

↳ Less memory, faster
 → No random access

LinkedList
 ↳ Has random access.
 → Allows null elements

map → Key, values
 ↓
 Unique

remove(Key, value)

Views over Maps:

keySet(); entrySet()

replace(Key, value)

replaceAll(BiFunction<K, V, V>)

remove(Key, value)

Sorted Map

Defines an interface
 for a map
 with ordering

Can't override keySet()