**DATA STRUCTURE & COLLECTIONS IN JAVA**

Jargons: i = index, e = element, k = key, v = value

**ARRAY**

|  |  |  |  |
| --- | --- | --- | --- |
| operator [ ] |  |  | length |

**Dynamic ARRAY:** import java.util.**ArrayList**;

|  |  |  |  |
| --- | --- | --- | --- |
| add(e) / add(i, e)  set(i, e)  get(i) | contains(e) | remove(i) | size()  isEmpty() |

**STACK:** import java.util.**Stack**;

|  |  |  |  |
| --- | --- | --- | --- |
| push(e)  add(e) / add(i, e) | contains(e)  search(e) | peek()  poll()  remove(e) | size()  isEmpty() |

**QUEUE:** import java.util.**LinkedList**;

|  |  |  |  |
| --- | --- | --- | --- |
| push(e)  add(e) / add(i, e) | contains(e) | peek()  poll()  remove(e) | size()  isEmpty() |

**SET:** import java.util.**HashSet**;

|  |  |  |  |
| --- | --- | --- | --- |
| add(e) | contains(e) | remove(e) | size()  isEmpty() |

**MAP:** import java.util.**HashMap**;

|  |  |  |  |
| --- | --- | --- | --- |
| put(k, v)  get(k) | containsKey(k)  containsValue(v) | remove(k) | size()  isEmpty() |

**MIN HEAP:** import java.util.**PriorityQueue**;

|  |  |  |  |
| --- | --- | --- | --- |
| add(e)  offer(e) | contains(e) | peek()  poll()  remove(e) | size()  isEmpty() |

|  |  |
| --- | --- |
| Type | Time Complexity |
| Array | **set, access : o(1)** |
| ArrayList | add: amortized o(1)  remove: o(n)  contains: o(n) |
| Linked List | add: o(1), if done at the head, o(n)  remove: o(1), if done at the head, o(n) if anywhere else  search: o(n) |
| Doubly-Linked List | add: o(1), if done at the head or tail, o(n) if anywhere else  remove: o(1), if done at the head or tail, o(n) if anywhere  search: o(n) |
| Stack | push: o(1)  pop: o(1)  peek: o(1)  search/contains : o(n) |
| Queue/Deque/Circular Queue | add: o(1)  remove: o(1) |
| Binary Search Tree | add, remove and search: average case: o(log n), worst case: o(n) |
| Red-Black Tree | add, remove and search: average case: o(log n), worst case: o(log n) |
| PriorityQueue | add: o(log n)  peek: o(1)  poll: o(log n)  contains, remove : o(n), |
| HashMap /HashSet: | add/remove: o(1) amortized  re-size/hash: o(n)  contains: o(1) |

|  |  |  |
| --- | --- | --- |
| Method | Return Type | Method and Description |
| add | boolean | Inserts the specified element into the collection if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available. |
| offer | boolean | Inserts the specified element into the collection if it is possible to do so immediately without violating capacity restrictions. |
| peek | [e](https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html) | Retrieves, but does not remove, the head of the collection, or returns null if the collection is empty. |
| poll | [e](https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html) | Retrieves and removes the head of the collection, or returns null if the collection is empty. |
| remove | e | Retrieves and removes the head of the collection. |