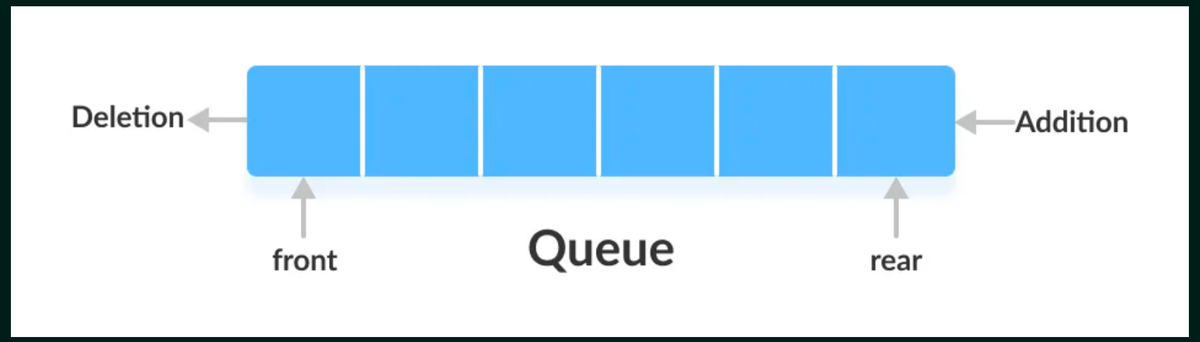
Java Queue and Sets

In This Lecture

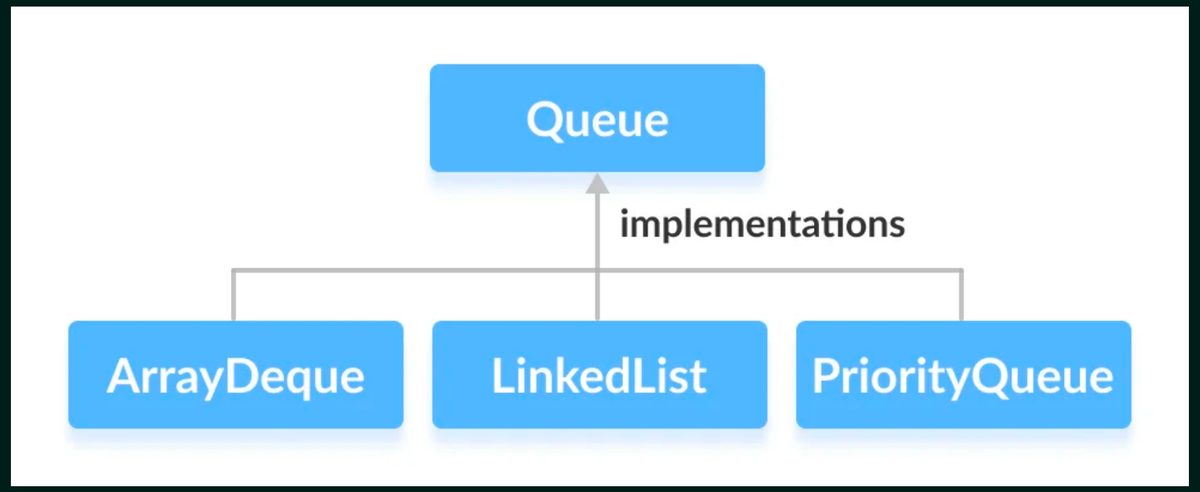
1. Java Queue Interface
2. Java Queue using LinkedList
3. Java PriorityQueue
4. Java ArrayDeque
5. Java Set Interface
6. Java HashSet
7. Java Set of Custom Objects

Java Queue Interface

The Queue interface of the Java collections framework provides the functionality of the queue data structure. It extends the Collection interface.



List, Queue
↑
LinkedList



Java Queue Interface

- **boolean add(E e):** Inserts the specified element into the queue.
- **boolean offer(E e):** Inserts the specified element into the queue. Returns **true** if the element was added successfully, or **false** if the queue is full.
- **E remove():** Removes and returns the element at the front of the queue. Throws an exception if the queue is empty.
- **E poll():** Removes and returns the element at the front of the queue. Returns **null** if the queue is empty.
- **E element():** Retrieves but does not remove the element at the front of the queue. Throws an exception if the queue is empty.
- **E peek():** Retrieves but does not remove the element at the front of the queue. Returns **null** if the queue is empty.

Java ArrayDeque Class

Adding Elements:

- **addFirst(E e)** or **offerFirst(E e)**: Adds an element to the front of the deque.
- **addLast(E e)** or **offerLast(E e)**: Adds an element to the end of the deque.

Removing Elements:

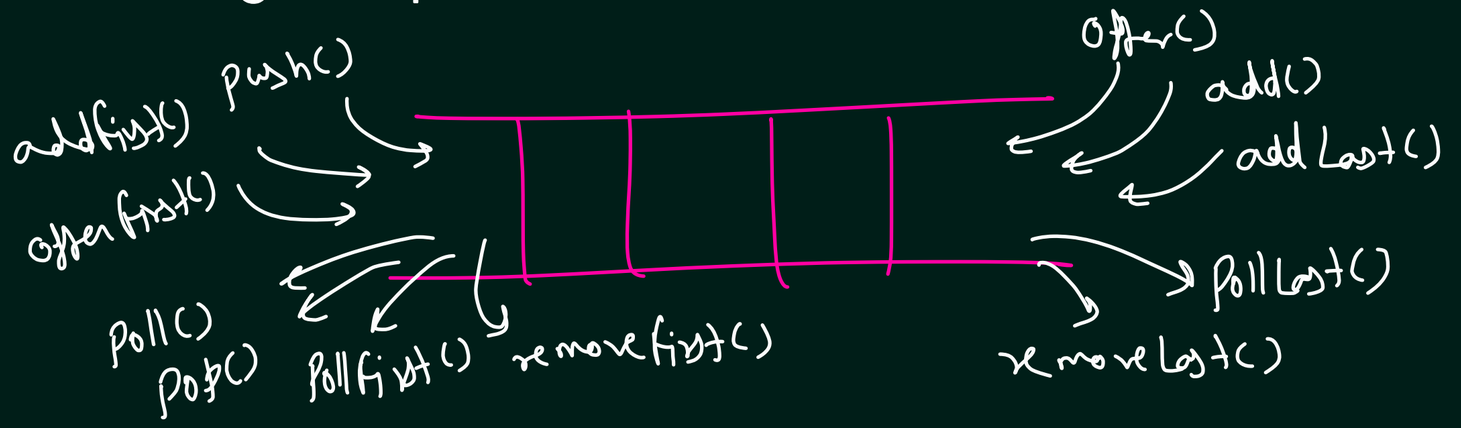
- **removeFirst()** or **pollFirst()**: Removes and returns the element at the front of deque.
- **removeLast()** or **pollLast()**: Removes and returns the element at the end of deque.

Accessing Elements:

- **getFirst()** or **peekFirst()**: Returns the element at the front of the deque without removing it.
- **getLast()** or **peekLast()**: Returns the element at the end of the deque without removing it.

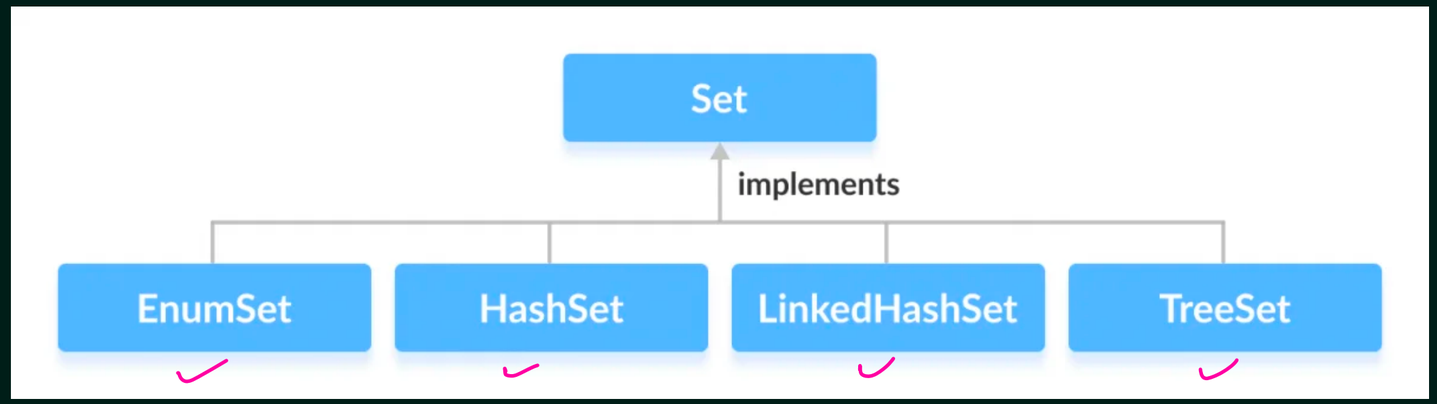
Stack and Queue Operations Using ArrayDeque

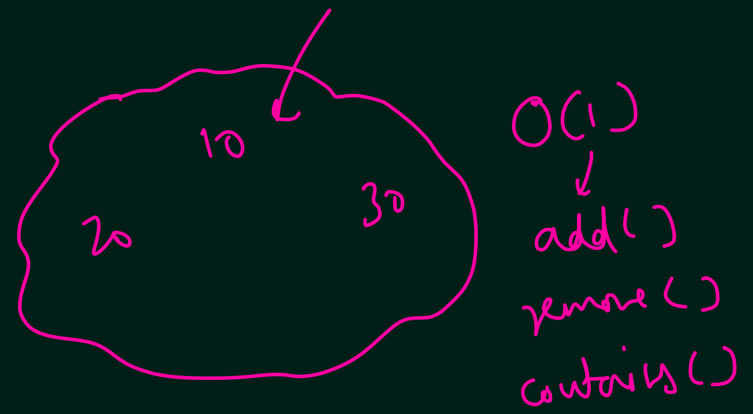
1. **Stack Operations:**
   - **push(E e)**: Pushes an element onto the stack represented by the deque.
   - **pop()**: Pops an element from the stack represented by the deque.
2. **Queue Operations:**
   - **add(E e)** or **offer(E e)**: Adds an element to the end of the deque, effectively makin it a queue.
   - **remove()** or **poll()**: Removes and returns the element at the front of the deque, making it a queue.



Java Set Interface

The Set interface of the Java Collections framework provides the features of the mathematical set in Java. It extends the Collection interface.
Unlike the List interface, sets cannot contain duplicate elements.





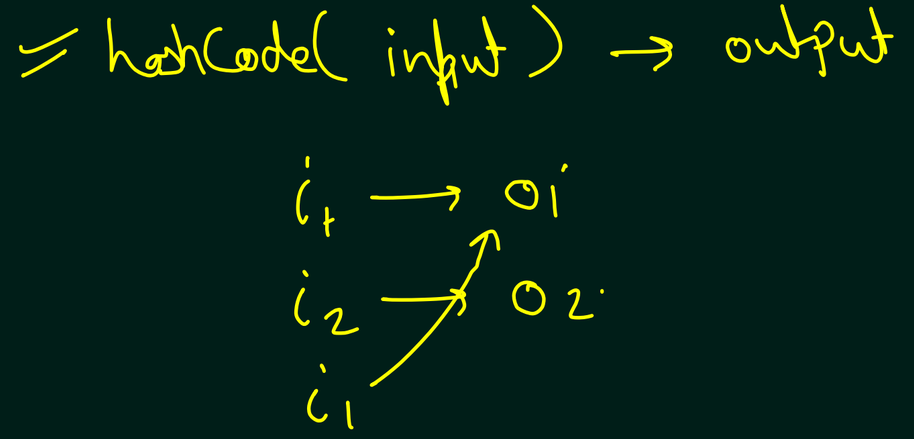
$O(n^2) \longrightarrow O(n)$

Java Set Interface

- add() - adds the specified element to the set
- addAll() - adds all the elements of the specified collection to the set
- remove() - removes the specified element from the set
- removeAll() - removes all the elements from the set that is present in another specified set
- retainAll() - retains all the elements in the set that are also present in another specified set
- clear() - removes all the elements from the set
- size() - returns the length (number of elements) of the set
- contains() - returns true if the set contains the specified element

Java HashSet

- In Java, HashSet is commonly used if we have to access elements randomly. It is because elements in a hash table are accessed using hash codes.
- The hashcode of an element is a unique identity that helps to identify the element in a hash table.
- HashSet cannot contain duplicate elements. Hence, each hash set element has a unique hashcode.



Java HashSet of Custom Objects

When using Set and HashSet in Java, for the primitive types we can just use it without worry about how to implement the hashcode and the comparison logic. But when you want to use the Set with a custom class by putting custom objects into the set, that custom class has to implement the **hashCode() and equals()** methods in order for the HashSet to work.