Java Methods Cheat Sheet for DSA Problems

String Methods

Basic Operations

- `length()` - Returns the length of the string
- `charAt(int index)` - Returns the character at the specified index
- `substring(int beginIndex)` - Returns substring from beginIndex to end
- `substring(int beginIndex, int endIndex)` - Returns substring from beginIndex to endIndex-1
- `isEmpty()` - Checks if string length is 0
- `toCharArray()` - Converts string to character array

String Comparison

- `equals(Object obj)` - Case-sensitive comparison with another string
- `equalsIgnoreCase(String str)` - Case-insensitive comparison
- `compareTo(String str)` - Lexicographical comparison (returns int)
- `startsWith(String prefix)` - Checks if string starts with prefix
- `endsWith(String suffix)` - Checks if string ends with suffix
- `contains(CharSequence seq)` - Checks if string contains sequence

String Modification

- `toLowerCase()` - Converts to lowercase
- `toUpperCase()` - Converts to uppercase
- `trim()` - Removes whitespace from both ends
- `replace(char oldChar, char newChar)` - Replaces all occurrences of a character
- `replace(CharSequence target, CharSequence replacement)` - Replaces sequence
- `replaceAll(String regex, String replacement)` - Replaces by regex pattern

Searching

- `indexOf(String str)` - First occurrence of specified string
- `indexOf(String str, int fromIndex)` - First occurrence from index

- `lastIndexOf(String str)` - Last occurrence of specified string
- `lastIndexOf(String str, int fromIndex)` - Last occurrence before index
- `matches(String regex)` - Tells if string matches regex pattern

Splitting and Joining

- `split(String regex)` - Splits string based on regex, returns array
- `split(String regex, int limit)` - Splits with limit on array size
- `join(CharSequence delimiter, CharSequence... elements)` - Joins elements with delimiter (static method)
- `concat(String str)` - Concatenates another string

StringBuilder Methods

Basic Operations

- `StringBuilder()` - Constructor creates empty builder with capacity 16
- `StringBuilder(String str)` - Constructor with initial string
- `length()` - Returns length (character count)
- `capacity()` - Returns current capacity
- `charAt(int index)` - Returns char at specified index
- `substring(int start)` - Returns substring from start to end
- `substring(int start, int end)` - Returns substring from start to end-1

Modification

- `append(X x)` - Appends string representation of X (many overloads)
- `insert(int offset, X x)` - Inserts string representation of X at position
- `delete(int start, int end)` - Removes chars from start to end-1
- `deleteCharAt(int index)` - Removes char at specified position
- `replace(int start, int end, String str)` - Replaces substring
- `setCharAt(int index, char ch)` - Sets char at specified position
- `reverse()` - Reverses the sequence
- `setLength(int newLength)` - Sets the length (truncates or adds null chars)
- `toString()` - Converts to String

Arrays Methods (java.util.Arrays)

Basic Operations

- `Arrays.toString(array)` - Returns string representation of array
- `Arrays.deepToString(Object[][] array)` - For multi-dimensional arrays

- `Arrays.equals(array1, array2)` - Compares arrays for equality
- `Arrays.deepEquals(Object[][] a1, Object[][] a2)` - For multi-dimensional arrays
- `Arrays.hashCode(array)` - Returns hash code based on contents
- `Arrays.deepHashCode(Object[][] array)` - For multi-dimensional arrays

Searching and Sorting

- `Arrays.sort(array)` - Sorts array in ascending order
- `Arrays.sort(array, int fromIndex, int toIndex)` - Sorts specified range
- `Arrays.sort(array, Comparator<? super T> c)` - Sorts with custom comparator
- `Arrays.binarySearch(array, key)` - Binary search on sorted array
- `Arrays.binarySearch(array, fromIndex, toIndex, key)` - In specific range
- `Arrays.fill(array, val)` - Fills entire array with specified value
- `Arrays.fill(array, fromIndex, toIndex, val)` - Fills range with value

Array Conversion

- `Arrays.asList(T... a)` - Converts array to fixed-size List
- `Arrays.copyOf(original, newLength)` - Copies and possibly resizes
- `Arrays.copyOfRange(original, from, to)` - Copies specified range

Java 8+ Enhancements

- `Arrays.stream(array)` - Returns a sequential stream
- `Arrays.parallelSort(array)` - Sorts using multiple threads

ArrayList (and List) Methods

Basic Operations

- `ArrayList<E>()` - Constructor creates empty list
- `ArrayList<E>(Collection<? extends E> c)` - Constructor with collection
- `size()` - Returns number of elements
- `isEmpty()` - Checks if list contains no elements
- `get(int index)` - Returns element at position
- `set(int index, E element)` - Replaces element at position
- `add(E e)` - Appends element to end
- `add(int index, E element)` - Inserts element at position
- `remove(int index)` - Removes element at position

- `remove(Object o)` - Removes first occurrence of element
- `clear()` - Removes all elements

Searching

- `contains(Object o)` - Returns true if list contains element
- `indexOf(Object o)` - Index of first occurrence or -1
- `lastIndexOf(Object o)` - Index of last occurrence or -1

Bulk Operations

- `addAll(Collection<? extends E> c)` - Appends all elements
- `addAll(int index, Collection<? extends E> c)` - Inserts all at index
- `removeAll(Collection<?> c)` - Removes all elements in c
- `retainAll(Collection<?> c)` - Retains only elements in c
- `containsAll(Collection<?> c)` - True if contains all elements

List Views

- `subList(int fromIndex, int toIndex)` - Returns view of portion

Java 8+ Enhancements

- `stream()` - Returns a sequential Stream
- `forEach(Consumer<? super E> action)` - Performs action for each element
- `removeIf(Predicate<? super E> filter)` - Removes all elements matching predicate
- `sort(Comparator<? super E> c)` - Sorts list with provided comparator

HashSet (and Set) Methods

Basic Operations

- `HashSet<E>()` - Constructor creates empty set
- `HashSet<E>(Collection<? extends E> c)` - Constructor with collection
- `size()` - Returns number of elements
- `isEmpty()` - Checks if set contains no elements
- `add(E e)` - Adds element if not present (returns boolean)
- `remove(Object o)` - Removes element if present
- `clear()` - Removes all elements
- `contains(Object o)` - Returns true if set contains element

Bulk Operations

- `addAll(Collection<? extends E> c)` - Adds all elements
- `removeAll(Collection<?> c)` - Removes all elements in c
- `retainAll(Collection<?> c)` - Retains only elements in c
- `containsAll(Collection<?> c)` - True if contains all elements

Iteration

- `iterator()` - Returns iterator over elements

Java 8+ Enhancements

- `stream()` - Returns a sequential Stream
- `forEach(Consumer<? super E> action)` - Performs action for each element
- `removeIf(Predicate<? super E> filter)` - Removes all elements matching predicate

HashMap (and Map) Methods

Basic Operations

- `HashMap<K, V>()` - Constructor creates empty map
- `HashMap<K, V>(Map<? extends K, ? extends V> m)` - Constructor with map
- `size()` - Returns number of key-value mappings
- `isEmpty()` - Checks if map contains no mappings
- `put(K key, V value)` - Associates key with value
- `get(Object key)` - Returns value for key or null
- `getOrDefault(Object key, V defaultValue)` - Returns value or default
- `remove(Object key)` - Removes mapping for key
- `clear()` - Removes all mappings

Checking Map Contents

- `containsKey(Object key)` - True if map contains key
- `containsValue(Object value)` - True if map maps to value

Bulk Operations

- `putAll(Map<? extends K, ? extends V> m)` - Copies all mappings
- `putIfAbsent(K key, V value)` - Adds mapping if key not present

Map Views

- `keySet()` - Returns Set view of keys

- `values()` - Returns Collection view of values
- `entrySet()` - Returns Set view of mappings

Java 8+ Enhancements

- `forEach(BiConsumer<? super K, ? super V> action)` - Performs action for each entry
- `replaceAll(BiFunction<? super K, ? super V, ? extends V> function)` - Replaces all values
- `compute(K key, BiFunction<? super K, ? super V, ? extends V> remappingFunction)` - Computes value
- `computeIfAbsent(K key, Function<? super K, ? extends V> mappingFunction)` - If key absent
- `computeIfPresent(K key, BiFunction<? super K, ? super V, ? extends V> remappingFunction)` - If key present
- `merge(K key, V value, BiFunction<? super V, ? super V, ? extends V> remappingFunction)` - Merges values

Queue and PriorityQueue Methods

Basic Operations

- `add(E e)` / `offer(E e)` - Adds element to queue (returns boolean)
- `remove()` / `poll()` - Removes and returns head (poll returns null if empty)
- `element()` / `peek()` - Retrieves head without removing (peek returns null if empty)
- `size()` - Returns number of elements
- `isEmpty()` - Checks if queue is empty

PriorityQueue Specific

- `PriorityQueue<E>()` - Default min heap
- `PriorityQueue<E>(Comparator<? super E> comparator)` - Custom ordering

Stack Methods

Basic Operations

- `push(E item)` - Pushes item onto top of stack
- `pop()` - Removes and returns top item
- `peek()` - Returns top item without removing
- `empty()` - Tests if stack is empty
- `search(Object o)` - Returns position of object (1-based)

Deque Methods (LinkedList, ArrayDeque)

Basic Operations

- `addFirst(E e)` / `offerFirst(E e)` - Inserts at front
- `addLast(E e)` / `offerLast(E e)` - Inserts at end
- `removeFirst()` / `pollFirst()` - Removes first element
- `removeLast()` / `pollLast()` - Removes last element
- `getFirst()` / `peekFirst()` - Retrieves first element
- `getLast()` / `peekLast()` - Retrieves last element