JAVA COLLECTIONS CHEATSHEET

Array

int[] arr = new **int[10]**; // {1,2,3,4,5}

arr.length;	prints length of array (10)
Arrays.binarySearch(arr, 3)	binary search (2)
Arrays.equals(arr, arr2)	true if arr and arr2 are equal
Arrays.fill(arr, 0, arr.length, 10)	fills arr from 0 to arr.length with value 10
Arrays.sort(arr)	sorts arr
Arrays.toString(arr)	converts array to string

String/StringBuilder

String str = "Hello World";
StringBuilder sb = new **StringBuilder**("Hello World");

str.charAt(0)	Returns char at position 0	sb
str.equals(s2)	Checks equality of strings	
str.indexOf('o')	Get index of character o	sb
str.length()	Length of string	sb
str.replaceAll("H", "X")	Replace all instances of character	
str.substring(0, str.length())	Substring	sb
str.toUpperCase()/toLowerCase()	Convert to upper/lower case	
str.trim()	Trim whitespaces from both ends	

sb.append("abc")	appends to stringbuilder
sb.delete(1,2)	delete character at index 1
sb.insert(0,"xyz")	insert xyz at index 0
sb.replace(0,2, "pq")	replace char at 0,1 with string pq
sb.reverse()	reverse
sb.toString()	convert stringbuilder to string

JAVA UTIL COLLECTIONS

List <Integer> list = new **ArrayList** <Integer>();
Queue <Double> queue = new **LinkedList** <Double>();
Stack <String> stack = new **Stack** <String>();
Set <String> words = new **HashSet** <String>();
Map <String, Integer> counts = new **HashMap** <String, Integer>();

[Note: Methods on next page]

Methods Found in ALL collections (Lists, Stacks, Queues, Sets, Maps)

`clear()`	removes all elements of the collection
`equals(`**collection**`)`	returns `true` if the given other collection contains the same elements
`isEmpty()`	returns `true` if the collection has no elements
`size()`	returns the number of elements in the collection
`toString()`	returns a string representation such as `"[10, -2, 43]"`

Methods Found in both Lists and Sets (ArrayList, LinkedList, HashSet, TreeSet)

`add(`**value**`)`	adds value to collection (appends at end of list)
`contains(`**value**`)`	returns `true` if the given value is found somewhere in this collection
`remove(`**value**`)`	finds and removes the given value from this collection
`removeAll(`**collection**`)`	removes any elements found in the given collection from this one
`retainAll(`**collection**`)`	removes any elements *not* found in the given collection from this one

`List<E>` Methods (10.1)

`add(`**index, value**`)`	inserts given value at given index, shifting subsequent values right
`indexOf(`**value**`)`	returns first index where given value is found in list (-1 if not found)
`get(`**index**`)`	returns the value at given index
`lastIndexOf(`**value**`)`	returns last index where given value is found in list (-1 if not found)
`remove(`**index**`)`	removes/returns value at given index, shifting subsequent values left
`set(`**index, value**`)`	replaces value at given index with given value
`subList(`**from, to**`)`	returns sub-portion at indexes **from** (inclusive) and **to** (exclusive)

`Stack<E>` Methods

`peek()`	returns the top value from the stack without removing it
`pop()`	removes the top value from the stack and returns it; `peek`/`pop` throw an `EmptyStackException` if the stack is empty
`push(`**value**`)`	places the given value on top of the stack

`Queue<E>` Methods

`add(`**value**`)`	places the given value at the back of the queue
`peek()`	returns the front value from the queue without removing it; returns `null` if the queue is empty
`remove()`	removes the value from the front of the queue and returns it; throws a `NoSuchElementException` if the queue is empty

`Map<K, V>` Methods (11.3)

`containsKey(`**key**`)`	`true` if the map contains a mapping for the given key
`get(`**key**`)`	the value mapped to the given key (`null` if none)
`keySet()`	returns a `Set` of all keys in the map
`put(`**key, value**`)`	adds a mapping from the given key to the given value
`putAll(`**map**`)`	adds all key/value pairs from the given map to this map
`remove(`**key**`)`	removes any existing mapping for the given key
`toString()`	returns a string such as `"{a=90, d=60, c=70}"`
`values()`	returns a `Collection` of all values in the map