

# CSE143 Cheat Sheet

## Math Methods (3.2) *mathematical operations*

<code>Math.abs(value)</code>	absolute value
<code>Math.min(v1, v2)</code>	smaller of two values
<code>Math.max(v1, v2)</code>	larger of two values
<code>Math.round(value)</code>	nearest whole number
<code>Math.pow(b, e)</code>	b to the e power

## Stacks and Queues (14.2)

*(LIFO and FIFO structures)*

Queues should be constructed using the `Queue<E>` interface and the `LinkedList<E>` implementation (you may not pass any arguments to the constructor). For example, to construct a queue of String values, you would say:

```
Queue<String> q = new LinkedList<>();
```

Stacks should be constructed using the `Stack<E>` class (there is no interface):

```
Stack<Integer> s = new Stack<>();
```

For `Stack<E>`, you are limited to the following operations (no iterator or foreach loop):

<code>push(value)</code>	pushes the given value onto the top of the stack
<code>pop()</code>	removes and returns the top of the stack
<code>isEmpty()</code>	returns <code>true</code> if this stack is empty
<code>size()</code>	returns the number of elements in the stack

For `Queue<E>`, you are limited to the following operations (no iterator or foreach loop):

<code>add(value)</code>	adds the given value at the end of the queue
<code>remove()</code>	removes and returns the front of the queue
<code>isEmpty()</code>	returns <code>true</code> if this queue is empty
<code>size()</code>	returns the number of elements in the queue

## String Methods (3.3)

*(An object for storing a sequence of characters)*

<code>length()</code>	returns the number of characters in the string
<code>charAt(index)</code>	returns the character at a specific index
<code>equals(other)</code>	returns true if this string equals the other
<code>toUpperCase()</code>	returns a new string with all uppercase letters
<code>toLowerCase()</code>	returns a new string with all lowercase letters
<code>startsWith(other)</code>	returns true if this string starts with the given text
<code>substring(start, stop)</code>	returns a new string composed of characters from start index (inclusive) to stop index (exclusive)
<code>substring(start)</code>	returns a new string composed of characters from start index (inclusive) to the end of the string