### CSE 143 Midterm Cheat Sheet

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear()</td>
<td>removes all elements of the collection</td>
</tr>
<tr>
<td>equals(collection)</td>
<td>returns true if the given other collection contains the same elements</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>returns true if the collection has no elements</td>
</tr>
<tr>
<td>size()</td>
<td>returns the number of elements in the collection</td>
</tr>
<tr>
<td>toString()</td>
<td>returns a string representation such as &quot;[10, -2, 43]&quot;</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(value)</td>
<td>adds value to collection (appends at end of list)</td>
</tr>
<tr>
<td>contains(value)</td>
<td>returns true if the given value is found somewhere in this collection</td>
</tr>
<tr>
<td>iterator()</td>
<td>returns an Iterator object to traverse the collection's elements</td>
</tr>
<tr>
<td>remove(value)</td>
<td>finds and removes the given value from this collection</td>
</tr>
<tr>
<td>removeAll(collection)</td>
<td>removes any elements found in the given collection from this one</td>
</tr>
<tr>
<td>retainAll(collection)</td>
<td>removes any elements not found in the given collection from this one</td>
</tr>
</tbody>
</table>

#### List<E> Methods (10.1)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(index, value)</td>
<td>inserts given value at given index, shifting subsequent values right</td>
</tr>
<tr>
<td>indexOf(value)</td>
<td>returns first index where given value is found in list (-1 if not found)</td>
</tr>
<tr>
<td>get(index)</td>
<td>returns the value at given index</td>
</tr>
<tr>
<td>remove(index)</td>
<td>removes/returns value at given index, shifting subsequent values left</td>
</tr>
<tr>
<td>set(index, value)</td>
<td>replaces value at given index with given value</td>
</tr>
</tbody>
</table>

#### Stack<E> Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop()</td>
<td>removes the top value from the stack and returns it; throws an EmptyStackException if the stack is empty</td>
</tr>
<tr>
<td>push(value)</td>
<td>places the given value on top of the stack</td>
</tr>
<tr>
<td>peek()</td>
<td>returns the top value from the stack. Throws EmptyStackException if the stack is empty</td>
</tr>
</tbody>
</table>

#### Queue<E> Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(value)</td>
<td>places the given value at the back of the queue</td>
</tr>
<tr>
<td>remove()</td>
<td>removes the value from the front of the queue and returns it; throws a NoSuchElementException if the queue is empty</td>
</tr>
<tr>
<td>peek()</td>
<td>returns the value from the front of the queue. Returns null if the queue is empty</td>
</tr>
</tbody>
</table>

#### Declaring and Initializing Sets and Maps

- Map<Key, Value> name = new HashMap<Key, Value>();
- Set<Type> name = new HashSet<Type>();

For problems involving stacks or queues, you ARE NOT ALLOWED to use for-each loops, iterators, or any operation other than those specified here for stacks/queues.
Queues should be constructed using the `Queue<E>` interface and the `LinkedList<E>` implementation. Stacks should be constructed using the `Stack<E>` class (there is no interface). For example, to construct a queue and a stack of `String` values, you would say:

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

To transfer from a queue to a stack:

```java
while (!q.isEmpty()) {
    s.push(q.remove());
}
```

To transfer from a stack to a queue:

```java
while (!s.isEmpty()) {
    q.add(s.pop());
}
```

### CSE 143 Midterm Cheat Sheet

**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

**String Methods (3.3, 4.4)**

- `charAt(i)`
  - the character in this String at a given index
- `contains(str)`
  - true if this String contains the other's characters inside it
- `endsWith(str)`
  - true if this String ends with the other's characters
- `equals(str)`
  - true if this String is the same as `str`
- `equalsIgnoreCase(str)`
  - true if this String is the same as `str`, ignoring capitalization
- `indexOf(str)`
  - first index in this String where given String begins (-1 if not found)
- `lastIndexOf(str)`
  - last index in this String where given String begins (-1 if not found)
- `length()`
  - number of characters in this String
- `startsWith(str)`
  - true if this String begins with the other's characters
- `substring(i, j)`
  - characters in this String from index `i` (inclusive) to `j` (exclusive)
- `toLowerCase()`, `toUpperCase()`
  - a new String with all lowercase or uppercase letters

**Random Methods (5.1)**

- `nextBoolean()`
  - random true/false result
- `nextDouble()`
  - random real number between 0.0 and 1.0
- `nextInt()`
  - random integer
- `nextInt(max)`
  - random integer between 0 and `max`