CSE143 Cheat Sheet

Linked Lists (16.2)

Below is an example of a method that could be added to the LinkedIntList class to compute the sum of the list:

```
public int sum() {
    int sum = 0;
    ListNode current = front;
    while (current != null) {
        sum += current.data;
        current = current.next;
    }
    return sum;
}
```

`Math` Methods (3.2)   *mathematical operations*

Math.abs(*value*)	absolute value
Math.min(*v1, v2*)	smaller of two values
Math.max(*v1, v2*)	larger of two values
Math.round(*value*)	nearest whole number
Math.pow(*b, e*)	b to the e power
Math.signum(v)	signum of v

`Iterator<E>` Methods (11.1)   *(An object that lets you examine the contents of any collection)*

hasNext()	returns true if there are more elements to be read from collection
next()	reads and returns the next element from the collection
remove()	removes the last element returned by next from the collection

`List<E>` Methods (10.1)   *(An ordered sequence of values)*

add(**value**)	appends value at end of list
add(**index, value**)	inserts given value at given index, shifting subsequent values right
clear()	removes all elements of the list
indexOf(**value**)	returns first index where given value is found in list (-1 if not found)
get(**index**)	returns the value at given index
remove(**index**)	removes/returns value at given index, shifting subsequent values left
set(**index, value**)	replaces value at given index with given value
size()	returns the number of elements in list
isEmpty()	returns true if the list's size is 0
contains(**value**)	returns true if the given value is found somewhere in this list
remove(**value**)	finds and removes the given value from this list if it is present
removeAll(**list**)	removes any elements found in the given collection from this list
iterator()	returns an object used to examine the contents of the list

`Set<E>` Methods (11.2)   *(A fast-searchable set of unique values)*

add(**value**)	adds the given value to the set
contains(**value**)	returns true if the given value is found in the set
remove(**value**)	removes the given value from the set if it is present
clear()	removes all elements of the set
size()	returns the number of elements in the set
isEmpty()	returns true if the set's size is 0
containsAll(**collection**)	returns true if set contains every element from given collection
removeAll(**collection**)	removes any elements found in the given collection from this set
retainAll(**collection**)	removes any elements *not* found in the given collection from this set
iterator()	returns an object used to examine the contents of the set

`Map<K, V>` Methods (11.3)   *(A fast mapping between a set of keys and a set of values)*

put(**key, value**)	adds a mapping from the given key to the given value
get(**key**)	returns the value mapped to the given key (null if none)
containsKey(**key**)	returns true if the map contains a mapping for the given key
remove(**key**)	removes any existing mapping for the given key
clear()	removes all key/value pairs from the map
size()	returns the number of key/value pairs in the map
isEmpty()	returns true if the map's size is 0
keySet()	returns a Set of all keys in the map
values()	returns a Collection of all values in the map
putAll(**map**)	adds all key/value pairs from the given map to this map

`Point` Methods (8.1)   *(an object for storing integer x/y coordinates)*

Point(**x, y**)	constructs a new point with given x/y coordinates
Point()	constructs a new point with coordinates (0, 0)
getX()	returns the x-coordinate of this point
getY()	returns the y-coordinate of this point
equals(**other**)	returns true if this Point stores the same x/y values as the other
translate(**dx, dy**)	translates the coordinates by the given amount

`String` Methods (3.3)   *(An object for storing a sequence of characters)*

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

Collections Implementations

List<E>	ArrayList<E> and LinkedList<E>
Set<E>	HashSet<E> and TreeSet<E> (values ordered)
Map<K, V>	HashMap<K, V> and TreeMap<K, V> (keys ordered)