MyArrayList<E> Design Document

Description

The MyArrayList<E> object can store data of data type E, where E is replaced by a specific data type or object. MyArrayList<E> objects can be used for lists of objects where repetition is allowed, order is important, and the number of elements in the list changes, but quick random access of the elements is more important than changing the size of the list.

Services

The constructor MyArrayList<E>() can be used to construct a new MyArrayList<E> object. This runs in O(1) time.

The method size() can be used to return the size of the MyArrayList<E> object. This method runs in O(1) time.

The method get(ind) returns the object in the list at index ind, where the index is based from 0. A precondition for this method is that the parameter ind must be an integer greater than equal to 0 but less than the size of the list. This method runs in O(1) time.

The method set(ind,obj) sets the element at index ind (based from 0) to the given parameter obj. The two preconditions for this method are that the parameter ind must be an integer greater than equal to 0 but less than the size of the list, and the parameter obj must be the same data type or object as the other elements in the list. This method runs in O(1) time.

The method add(obj) appends the element obj to the end of the list. One precondition for this method is that the parameter obj must be the same data type or object as the other elements in the list. This method increases the size of the list by one, and it doubles the size of the underlying array if that array is already full before adding the new element. This method usually runs in O(1) time if the underlying array is not full; otherwise, the method runs in O(n) time.

The method add(ind,obj) appends the element obj to the index ind in the list. The preconditions for this method is that the parameter obj must be the same data type or object as the other elements in the list and that the parameter ind must be an integer greater than equal to 0 but less than or equals to the size of the list. This method increases the size of the list by one, and it doubles the size of the underlying array if that array is already full before adding the new element. This method runs in O(n) time.

The method remove(ind) removes the element at index ind (based from 0) from the list. The precondition for this method is that the parameter ind must be an integer greater than equal to 0 but less than the size of the list. This method decreases the size of the list by one and shifts all the elements from index ind+1 to the end of the list one index to the left (ie index--). This method runs in O(n) time.

The method `toString()` returns the array as a string, with the elements encased in `[]` and separated by commas and spaces. This method runs in `O(n)` time.

The method `getCapacity()` returns the size of the underlying array. This method runs in `O(1)` time.

The method `iterator()` creates an iterator for the `MyArrayList<E>` object. This method runs in `O(1)` time.

The method `listIterator()` creates a list iterator for the `MyArrayList<E>` object. This method runs in `O(1)` time.

Internal Data Structures and State

The only internal data structure used by `MyArrayList<E>` objects is an array of data type Object. This array starts off with a length of `1`. Whenever the `add(obj)` method is called, `obj` is added to the next unused slot in the array. If the array is full, the array doubles in size, with the original elements filling the first half of the new array and the new element coming immediately after the halfway point of the array.

The `MyArrayList<E>` objects also use the private variable `size` to remember the size of the list. The variable increases by one each time `add(obj)` is called and decreases each time `remove(ind)` is called.

Test Plan

The method `toString()` can be tested by printing multiple arraylists and checking that the output matches the expected contents.

The method `size()` can be tested by printing the output of this method for many arraylists and checking that output matches the length of the arraylists.

The method `get(ind)` can be tested by printing the output of this method for many arraylists and checking that output matches the actual element of each arraylist at index `ind`.

The method `set(ind,obj)` can be tested by calling this method on many arraylists and checking that the resulting arraylist matches the expected one.

The method `add(obj)` can be tested by calling this method on many arraylists and checking that the resulting arraylist matches the expected one.

The method `add(ind,obj)` can be tested by calling this method on many arraylists and checking that the resulting arraylist matches the expected one.

The method `remove(ind)` can be tested by calling this method on many arraylists and checking that the resulting arraylist and the returned element are correct

The method `getCapacity()` can be tested by calling this method on many arraylists and checking that the returned value is correct for each arraylist.