William Nguyen

Ch15 Implementing Collections Class

Contract: A clear specification that tells a client how an object behaves without describing the details of how it is implemented.

An array with size variable and moves used positions to the front. Max size of array is capacity. Add values to size-1 and increase size

You will need an accessor method to examine the value of the size field to work with it. Use accessors for .get as well. indexOf based on size and loop until you find something equal to value, returning loop increment i. For adding, you need index and value, starting i loop at size, >= index + 1 and then decrementing, elementData[i] = elementData[i-1] and size++. Do the reverse for removing.  i = index, < size-1 ++ ,then i = i+1 for elementData;

Make capacity variable as parameter and put it in the constructor. Explain pre and postconditions of what your method does and is capable of. Throw exceptions if preconditions not met. Will satisfy data invariants and won’t corrupt your program.

Convenience Methods. Contains, set, etc.

Resizing array, we have to allocate a brand=new array and copy values of old to new. Instead of checking if you’re over capacity, change it to make a new array and to store the values of the old ones. Arrays.copyOf.

Iterator: hasNext(), next(), remove(). Has Next uses a second list that will be made in the constructor. Also keep a position field set initially to 0, increment for next. Catch with hasNext.

Remove, position --.

When converting to an arraylist, construct with a cast, elmentData = (E[]) new Object(capacity);

Supressed warning. Garbage Collector: process that is part of JRE that frees memory used by objects no longer referenced. You have to set values to null so that they aren’t values anymore.

Inner Class; A class declared inside another class. Objects oinner class have access to the methods and fields of outer class.

Ch16 Linked Lists

Node: single element of a structure such as linked list; each node contains one data value. It is sequential, but it can insert values and delete values without shifting. Easier to change the size.

list.data , then list.next then list.next.data then list.next.next. Next to connect, data to hold value.

To skip first value do list = list.next, so we start from next because first became 2nd. Yet if list = null, it doesn’t destroy the linked list, it just sets the value that we’re holding to null. Imagine two separate nodes list, to connect them set p.next.next to q so after p is done, it leads to q. ListNode current often used to store place in list. I = 0  == ListNode current = list;

i< size == current != null  elementData[i] == current.data i++ == current.next. To get to end, make sure current.next is null, not current.data. Then set current.next to a new value to add to the end. To add in the middle, do current.next = new ListNode(value, current.next). To remove a value, current.next = current.next.next, skipping the removed node. Could make an intList interface that implements methods differently, ArrayIntList and LinkedIntList share methods but use different implementation. Can use a dummy node to represent an empty list. Not longer have to write special code to insert in front of the list. Doubly linked list can store in both directions, so nodes have a next and previous. Iterator will use the same current.next format.