

**COMP 2230 – Data Structures and Algorithm Analysis**

Assignment #5: Lists

## Due Date: October. 17th S01 and October 18th for S02

**Chapter 15**

**Problem 1**: Complete the implementation of the Array List Class.

* **expandCapacity**
* **removeLast**
* **removeFirst**
* **first**
* **last**
* **isEmpty**
* **size**
* **toString**

**Answer:**

**import java.util.Arrays;**

**import java.util.NoSuchElementException;**

**public class ArrayList<T> implements ListADT<T>, Iterable<T>{**

**private final static int DEFAULT\_CAPCITY = 100;**

**private final static int NOT\_FOUND = -1;**

**protected int rear;**

**protected T[] List;**

**protected int modCount;**

**public ArrayList(){**

**this(DEFAULT\_CAPCITY);**

**}**

**public ArrayList(int Capacity){**

**rear = 0;**

**List = (T[]) (new Object[Capacity]);**

**modCount = 0;**

**}**

**public T remove(T element){**

**T result;**

**int index = find(element);**

**if(index == NOT\_FOUND){**

**throw new ElementNotFoundException("ArrayList");**

**}**

**result = List[index];**

**rear--;**

**for( int scan = index; scan< rear ; scan ++){**

**List[scan] = List[scan+1];**

**}**

**List[rear] = null;**

**modCount++;**

**return result;**

**}**

**public boolean Contains(T element){**

**return find(element) == NOT\_FOUND;**

**}**

**public int find(T Target){**

**int scan = 0;**

**int result = NOT\_FOUND;**

**if(!isEmpty()){**

**while(result == NOT\_FOUND && scan < rear){**

**if(Target.equals(List[scan])){**

**result = scan;**

**}**

**else{**

**scan ++;**

**}**

**}**

**}**

**return result;**

**}**

**public void add(T element){**

**if(!(element instanceof Comparable)){**

**throw new NonComparableElementException("OrderedList");**

**}**

**Comparable<T> comparableElement = (Comparable<T>) element;**

**if(size() == List.length){**

**expandCapacity();**

**}**

**int scan =0;**

**while(scan<rear && comparableElement.compareTo(List[scan])>0){**

**scan++;**

**}**

**for(int shift = rear; shift>scan;shift--){**

**List[shift] = List[shift-1];**

**}**

**List[scan] = element;**

**rear++;**

**modCount++;**

**}**

**public void addAfter(T element, T target){**

**if(size() == List.length){**

**expandCapacity();**

**}**

**int scan = 0;**

**while(scan<rear && !target.equals(List[scan])){**

**scan++;**

**}**

**for( int shift = rear; shift > scan; shift++){**

**List[shift] = List[shift -1];**

**}**

**List[scan] = element;**

**rear++;**

**modCount++;**

**}**

**public void expandCapacity(){**

**List = Arrays.copyOf(List, List.length\*2);**

**}**

**public T removeLast() throws NoSuchElementException{**

**if(isEmpty()){**

**throw new NoSuchElementException("ArrayList is Empty");**

**}**

**T result = List[rear-1];**

**List[rear-1] = null;**

**rear--;**

**modCount++;**

**return result;**

**}**

**public T removeFirst() throws NoSuchElementException{**

**if(isEmpty()){**

**throw new NoSuchElementException("ArrayList is Empty");**

**}**

**T result = List[0];**

**for(int scan = 0; scan<rear; scan++){**

**List[scan] = List[scan+1];**

**}**

**List[rear-1] = null;**

**rear--;**

**modCount++;**

**return result;**

**}**

**public T first() throws NoSuchElementException{**

**if(isEmpty()){**

**throw new NoSuchElementException("Array is Empty");**

**}**

**return List[0];**

**}**

**public T last() throws NoSuchElementException{**

**if(isEmpty()){**

**throw new NoSuchElementException("Array is Empty");**

**}**

**return List[rear-1];**

**}**

**public boolean isEmpty(){**

**return rear==0;**

**}**

**public int size(){**

**return rear;**

**}**

**public String toString(){**

**String result = "[";**

**for(int i = 0; i<rear; i++){**

**result += List[i];**

**if(i<rear-1){**

**result += ",";**

**}**

**}**

**result += "]";**

**return result;**

**}**

**@Override**

**public java.util.Iterator<T> iterator() {**

**// TODO Auto-generated method stub**

**throw new UnsupportedOperationException("Unimplemented method 'iterator'");**

**}**

**@Override**

**public boolean contains() {**

**// TODO Auto-generated method stub**

**throw new UnsupportedOperationException("Unimplemented method 'contains'");**

**}**

**@Override**

**public Iterable<T> Iterator() {**

**// TODO Auto-generated method stub**

**throw new UnsupportedOperationException("Unimplemented method 'Iterator'");**

**}**

**}**

**Problem 2**: Complete the implementation of the ArrayUnordered List Class.

* **addToFront**
* **adddToRear**

**Problem 3**: Override the **Find** method in the ArrayOrdered List Class to stop early if an item is not in the list, if encounter a larger item then the target is not in the list.