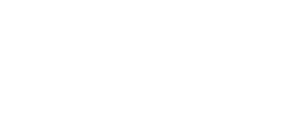


Oct 24

Assignment 6

Comp 2230\_02

Colton isles and kaylee crocker

2024

A close up of a logo

Description automatically generated

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

Assignment #6: Lists

## Due Date: S01 October. 24th S02 October 25th

**Chapter 15**

**Problem 1**: Create a new class that implements an array ordered list:

* **Constructor max size 10 elements, no expand capacity**
* **add**
* **toString**
* **delete**

**Problem 2**: Create a new class that implements an ordered list using linear node class

* **constructor**
* **add**
* **toString**
* **delete**

**Problem #1 Code**

|  |
| --- |
| **OrderedArrayList.java**  package Ass6\_2230;  import Ass6\_2230.Exceptions.\*;  import java.lang.reflect.Array;  import java.util.Arrays;  import java.util.\*;  /\*\*  \* ArrayOrderedList represents an array implementation of an ordered list.  \*  \* @author Colton Isles, Kaylee Crocker  \*/  public class OrderedArrayList<T extends Comparable<T>> {  private static final int MAX\_CAPACITY = 10;  private final static int NOT\_FOUND = -1;  protected T[] list;  protected int rear;  /\*\*  \* Constructs the OrderedListArray with the maximum capacity  \* of 10.  \*/  public OrderedArrayList() {  this(MAX\_CAPACITY);  }  /\*\*  \* Constructs the OrderedListArray with specified capacity  \* sets capacity to 10 if it is over the maximum.  \*  \* @param capacity Capacity of the arrayList with maximum of 10  \*/  OrderedArrayList(int capacity) {  if (capacity >= MAX\_CAPACITY) {  list = (T[]) Array.newInstance(Comparable.class, MAX\_CAPACITY);  } else {  list = (T[]) Array.newInstance(Comparable.class, capacity);  }  rear = 0;  }  /\*\*  \* Adds the element to it's place in the list.  \* @param element element to be added  \*/  public void add(T element) {  int index = 0;  if(rear == MAX\_CAPACITY){  throw new FullCapacityException("list");  }  while (index < rear && element.compareTo(list[index]) > 0) {  index++;  }  for (int i = rear; i > index; i--) {  list[i] = list[i - 1];  }  list[index] = element;  rear++;  }  /\*\*  \* Deletes the element if it is in the list.  \* @param element element to delete  \*/  public void delete(T element) {  int index = find(element);  list[index] = null;  for (int i = index; i < rear - 1; i++) {  list[i] = list[i + 1];  }  rear--;  list[rear] = null;  }  /\*\*  \* Finds the index of the target in the list.  \* @param target element to search for  \* @return result of the search  \*/  private int find(T target) throws ElementNotFoundException {  int result = NOT\_FOUND;  for (int index = 0; index < rear; index++) {  if (list[index].equals(target)) {  result = index;  break;  } else if (index == rear - 1 || list[index].compareTo(target) > 0) {  throw new ElementNotFoundException("list");  }  }  return result;  }  /\*\*  \* Returns a string of items in the list.  \* @return string of list items  \*/  @Override  public String toString() {  return Arrays.toString(list);  }  } |

|  |
| --- |
| **OrderedArrayList.java**  package Ass6\_2230;  import Ass6\_2230.Exceptions.\*;  public class OrderedArrayListTest {  public static void main(String[] args) {  // Create new list  OrderedArrayList<Integer> list = new OrderedArrayList<Integer>();  System.out.println("Testing sorted add method:");  System.out.println("--------------------------");  System.out.println(list.toString());  list.add(2);  System.out.println(list.toString());  list.add(9);  System.out.println(list.toString());  list.add(1);  System.out.println(list.toString());  list.add(6);  System.out.println(list.toString());  list.add(8);  System.out.println(list.toString());  list.add(4);  System.out.println(list.toString());  list.add(3);  System.out.println(list.toString());  list.add(5);  System.out.println(list.toString());  list.add(7);  System.out.println(list.toString());  list.add(0);  System.out.println(list.toString());  /\*list.add(10);  System.out.println(list.toString());\*/  System.out.println("Testing delete method:");  System.out.println("--------------------------");  System.out.println(list.toString());  list.delete(6);  System.out.println(list.toString());  list.delete(1);  System.out.println(list.toString());  list.delete(9);  System.out.println(list.toString());  System.out.println("Trying to delete absent element:");  System.out.println("--------------------------");  try {  list.delete(11);  } catch (ElementNotFoundException e) {  System.out.println("Throws ElementNotFoundException correctly.");  }  }  } |

**Problem #1 Output**

|  |
| --- |
|  |

**Problem #2 Code**

|  |
| --- |
| **OrderedLinkedList.java**  package Ass6\_2230;  import Ass6\_2230.Exceptions.\*;  public class OrderedLinkedList<T extends Comparable<T>> {  private int count;  private LinearNode<T> head;  public OrderedLinkedList(){  count = 0;  head = null;  }  /\*\*  \* Adds an element to its place in the list.  \* @param element element to add  \*/  public void add(T element){  LinearNode<T> node = new LinearNode<T>(element);  if(head == null){  head = node;  count++;  return;  }  if(element.compareTo(head.getElement()) <= 0){  node.setNext(head);  head = node;  count++;  return;  }  LinearNode<T> current = head;  while(current.getNext() != null && element.compareTo(current.getNext().getElement()) > 0){  current = current.getNext();  }  node.setNext(current.getNext());  current.setNext(node);  count++;  }  /\*\*  \* Deletes an element from the list.  \* @param element element to delete  \* @throws EmptyCollectionException  \* @throws ElementNotFoundException  \*/  public void delete(T element) throws EmptyCollectionException, ElementNotFoundException {  LinearNode<T> current = head;  if(head == null){  throw new EmptyCollectionException("LinkedList");  }  if(head.getElement().equals(element)){  head = head.getNext();  count--;  return;  }  for (int i = 0; i < count; i++) {  if (current.getNext() == null || current.getNext().getElement().compareTo(element) > 0) {  throw new ElementNotFoundException("Arraylist");  } else if (current.getNext().getElement().equals(element)) {  current.setNext(current.getNext().getNext());  count--;  return;  }  current = current.getNext();  }  }  /\*\*  \* Returns a string of items in the list.  \* @return string of list items  \*/  public String toString(){  if(count == 0){  return "Empty List";  }  StringBuilder result = new StringBuilder();  LinearNode<T> current = head;  for (int i = 1; i <= count; i++) {  result.append(current.getElement()).append(", ");  current = current.getNext();  }  result.deleteCharAt(result.length()-2);  return result.toString();  }  } |

|  |
| --- |
| **OrderedLinkedListTest.java**  package Ass6\_2230;  import Ass6\_2230.Exceptions.ElementNotFoundException;  public class OrderedLinkedListTest {  public static void main(String[] args) {  // Create new list  OrderedLinkedList<Integer> list = new OrderedLinkedList<Integer>();  System.out.println("Testing sorted add method:");  System.out.println("--------------------------");  System.out.println(list.toString());  list.add(2);  System.out.println(list.toString());  list.add(9);  System.out.println(list.toString());  list.add(1);  System.out.println(list.toString());  list.add(6);  System.out.println(list.toString());  list.add(8);  System.out.println(list.toString());  list.add(4);  System.out.println(list.toString());  list.add(3);  System.out.println(list.toString());  list.add(5);  System.out.println("Testing delete method:");  System.out.println("--------------------------");  System.out.println(list.toString());  list.delete(6);  System.out.println(list.toString());  list.delete(1);  System.out.println(list.toString());  list.delete(9);  System.out.println(list.toString());  System.out.println("Trying to delete absent element:");  System.out.println("--------------------------");  try {  list.delete(11);  } catch (ElementNotFoundException e) {  System.out.println("Throws ElementNotFoundException correctly.");  }  }  } |

**Problem #2 Output**

|  |
| --- |
|  |