```
::::::::::::::
Lab2Conclusions.txt
::::::::::::::
I found Lab 2 very informative as the method of creating another class for a full
Array/ArrayList is completely new to me and I found it to be a very useful impleme
ntation that I will use in the future. ::::::::::::::
Lab2P1Driver.java
::::::::::::::
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Lab2P1Driver extends ListArrayBasedPlus {
    static BufferedReader stdin = new BufferedReader(new InputStreamReader(System.
in));
    public static void main (String[] args) throws IOException {
        ListArrayBasedPlus list_plus = new ListArrayBasedPlus();
        boolean exit = false;
            System.out.print("Make your menu selection now: \n"
                             + "0. Exit the program \n"
                             + "1. Insert item into the list \n"
                             + "2. Remove item from the list \n"
                             + "3. Get item from the list \n"
                             + "4. Clear the list \n"
                             + "5. Print size and content of the list \n"
                             + "6. Reverse the list \n ");

            while (!exit) {
            System.out.print("You chose: ");
            int input = Integer.parseInt(stdin.readLine());

            // possible cases for initial input
            switch (input) {
                case 0:
                System.out.println("Exiting program... good bye");
                exit = true;
                break;

                case 1:
                System.out.println("You are now inserting an item into the list.")
;
                Object item = stdin.readLine();
                list_plus.add(0, item);
                break;

                case 2:
                System.out.println("Enter posiiton to move item from: ");
                Object item2 = stdin.readLine();
                for(int index = 0; index < list_plus.numItems; index++)
                {
                    if(list_plus.items[index] == item2)
                    {
                        list_plus.remove(index);
                        System.out.println("Item " + item2 + " removed from positi
on " + index + " in the list.");
                    }

                    else
                    {
                        System.out.println("Item " + item2 + " not found in list."
);
                    }
                }
                break;

                case 3:
                System.out.println("Enter position to retrieve item from: ");
                Object item3 = stdin.readLine();
                for (int index = 0; index < list_plus.numItems; index++)
                {
                    if(list_plus.items[index] == item3)
                    {
                        list_plus.get(index);
                    }

                    else
                    {
                        System.out.println("Position specified is out of range!");
                    }
                }
                break;

                case 4:
                System.out.println("Clearing list...");
                list_plus.removeAll();
                System.out.println("List cleared.");
                break;

                case 5:
                System.out.println("List of size " + list_plus.size() + " has the
following items: ");
                for (int index = 0; index < list_plus.numItems; index++)
                {
                    System.out.println(list_plus.get(index) + "\n");
                }
                break;

                case 6:
                System.out.println("Reversing list...");
                list_plus.reverse();
                System.out.println("Reversed list: ");
                for (int index = 0; index < list_plus.numItems; index++)
                {
                    System.out.println(list_plus.get(index) + "\n");
                }
                break;
            }
        }
    }
}
::::::::::::::
Lab2P1Sampleruns.txt
::::::::::::::
::::::::::::::
Lab2P2Driver.java
::::::::::::::
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Lab2P2Driver extends ListArrayListBasedPlus {
    static BufferedReader stdin = new BufferedReader(new InputStreamReader(System.
in));
    public static void main (String[] args) throws IOException
    {
        ListArrayListBasedPlus list = new ListArrayListBasedPlus();
```

```
        boolean exit = false;
            System.out.print("Make your menu selection now: \n"
                             + "0. Exit the program \n"
                             + "1. Insert item into the list \n"
                             + "2. Remove item from the list \n"
                             + "3. Get item from the list \n"
                             + "4. Clear the list \n"
                             + "5. Print size and content of the list \n"
                             + "6. Reverse the list \n ");

            while (!exit) {
            System.out.print("You chose: ");
            int input = Integer.parseInt(stdin.readLine());

            // possible cases for initial input
            switch (input) {
                case 0:
                System.out.println("Exiting program... good bye");
                exit = true;
                break;

                case 1:
                System.out.println("You are now inserting an item into the list.")
;
                Object item = stdin.readLine();
                list.add(0, item);
                break;

                case 2:
                System.out.println("Enter posiiton to remove item from: ");
                int item2 = Integer.parseInt(stdin.readLine());
                if(item2 > list.size())
                {
                    System.out.println("Posiiton specified out of range!");
                }

                else
                {
                    list.remove(item2);
                    System.out.println("Item " + item2 + " removed from the list."
);
                }
                System.out.println("Item " + item2 + " not found in list.");
                break;

                case 3:
                System.out.println("Enter position to retrieve item from: ");
                int item3 = Integer.parseInt(stdin.readLine());
                if(item3 > list.size())
                {
                    System.out.println("Position specified is out of range!");
                }

                else
                {
                    list.get(item3);
                }
                break;

                case 4:
                System.out.println("Clearing list...");
                list.removeAll();
                System.out.println("List cleared.");
                break;

                case 5:
                System.out.println("List of size " + list.size() + " has the follo
wing items: ");
                for (int index = 0; index < list.numItems - 1; index++)
                {
                    System.out.println(list.get(index) + "\n");
                }
                break;

                case 6:
                System.out.println("Reversing list...");
                list.reverse();
                System.out.println("Reversed list: ");
                for (int index = 0; index < list.numItems; index++)
                {
                    System.out.println(list.get(index) + "\n");
                }
                break;
            }
        }
    }
}
::::::::::::::
Lab2P2Sampleruns.txt
::::::::::::::
::::::::::::::
ListArrayBased.java
::::::::::::::
// ******************************************************
// Array-based implementation of the ADT list.
// ******************************************************
public class ListArrayBased implements ListInterface
{

    private static final int MAX_LIST = 3;
    protected static Object []items;  // an array of list items
    protected int numItems;  // number of items in list

    public ListArrayBased()
    {
        items = new Object[MAX_LIST];
        numItems = 0;
    }  // end default constructor

    public boolean isEmpty()
    {
        return (numItems == 0);
    } // end isEmpty

    public int size()
    {
        return numItems;
    }  // end size

    public void removeAll()
    {
        // Creates a new array; marks old array for
        // garbage collection.
        items = new Object[MAX_LIST];
```

```
        numItems = 0;
    } // end removeAll

    public void add(int index, Object item) // fixes implementation/programming st
yle errors
    throws  ListIndexOutOfBoundsException
    {
        if (numItems == items.length)
        {
            throw new ListException("ListException on add");
        }  // end if
        if (index >= 0 && index <= numItems)
        {
            // make room for new element by shifting all items at
            // positions >= index toward the end of the
            // list (no shift if index == numItems+1)
            for (int pos = numItems-1; pos >= index; pos--)  //textbook code modif
ied to eliminate logic error causing ArrayIndexOutOfBoundsException
            {
                items[pos+1] = items[pos];
            } // end for
            // insert new item
            items[index] = item;
            numItems++;
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on add");
        }  // end if
    } //end add

    public Object get(int index)
    throws ListIndexOutOfBoundsException
    {
        if (index >= 0 && index < numItems)
        {
            return items[index];
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on get");
        }  // end if
    } // end get

    public Object remove(int index)
    throws ListIndexOutOfBoundsException
    {   Object result;
        if (index >= 0 && index < numItems)
        {
            // delete item by shifting all items at
            // positions > index toward the beginning of the list
            // (no shift if index == size)
            result = items[index];
            if(numItems == items.length)
            {
                throw new ListException("ListException on remove");
            }
            for (int pos = index+1; pos < numItems; pos++) //textbook code modifie
d to eliminate logic error causing ArrayIndexOutOfBoundsException
            {
                items[pos-1] = items[pos];
            }  // end for
            items[--numItems] = null; // fixes memory leak
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on remove");
        }  // end if
        return result;
    } //end remove
}
::::::::::::::
ListArrayBasedPlus.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Review Programming Assignment
 * Status: Complete and thoroughly tested
 * Last update: 1/30/23
 * Submitted:  1/30/23
 * Comment: test suite and sample run attached
 * Comment: I declare that this is entirely my own work
 * @author: Antonio Rosado
 * @version: 2023.01.30
 */

public class ListArrayBasedPlus extends ListArrayBased
{
    /**
     * Constructor.
     */
    public ListArrayBasedPlus() {
        super();
    }

    /**
     * Adds items to Array
     */
    public void add()
    {
        if(items.length == numItems)
        {
            resize(); // if items reaches max size/num of items, resize array
        }
        super.add(numItems, items); // call superclass
    }

    /**
     * Reverses Array.
     */
    public void reverse()
    {
        for (int index = 0; index < numItems / 2; index++)
        {
            Object temps = items[numItems - index];
            items[numItems - index] = items[index];
            items[index] = temps;
        }
    }
```

```
    /**
     * Resizes array if size cap is reached.
     */
    public void resize()
    {
            if(numItems == items.length)
            {
                int resize = items.length;
                Object[] new_items = new Object[resize * 2]; // new Array == oldAr
ray * 2
                for(int index = 0; index < items.length; index++)
                {
                    new_items[index] = items;
                }
                new_items = items;
            }
            super.add(numItems + 1, items);
    }

    /*
     * Returns a string value of item(s) in Array
     */
    public String toString()
    {
        StringBuilder sb = new StringBuilder();
        for (int index = 0; index < numItems; ++index) // pre-increment
            {
                sb.append(items[index] + " ");
            }
            return sb.toString();
    }
}:::::::::::::::
ListArrayListBased.java
:::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Review Programming Assignment
 * Status: Complete and thoroughly tested
 * Last update: 1/30/23
 * Submitted:  1/30/23
 * Comment: test suite and sample run attached
 * Comment: I declare that this is entirely my own work
 * @author: Antonio Rosado
 * @version: 2023.01.30
 */

import java.util.ArrayList;

public class ListArrayListBased implements ListInterface {
    ArrayList<Object> items;
    private static final int MAX_LIST = 3;
    int numItems;

    public ListArrayListBased()
    {
        items = new ArrayList<Object>(MAX_LIST);
        numItems = 0;
    }

    /**
     *
     * Check if items ArrayList is empty.
     * @return  items.isEmpty()
     */
    public boolean isEmpty()
    {
        return items.isEmpty();
    }

    /**
     *
     * Return size of ArrayList.
     * @return  items.size()
     */
    public int size()
    {
        return items.size();
    }

    /**
     *
     * Add item to ArrayList.
     * @param int index     index of item
     * @param Object item   item Object
     */
    public void add(int index, Object item)
    {
        items.add(index, item);
    }

    /**
     *
     * Retrieve item in ArrayList by index.
     * @param int index     index of item
     * @return              item index
     * @throw               ListIndexOutOfBoundsException
     */
    public Object get(int index)
    {
        if(index >= 0 && index < numItems)
        {
            return items.get(index);
        }

        else
        {
            throw new ListIndexOutOfBoundsException("ListIndexOutOfBoundsException
on get");
        }
    }

    /**
     *
     * Remove item in ArrayList by index.
     * @param int index     index of item
     * @return                 items.remove(index)
     * @throws              ListIndexOutOfBoundsException
     */
    public Object remove(int index) throws ListIndexOutOfBoundsException
    {
        return items.remove(index);
    }

    /**
```

```
     *
     * Retrieve all items in ArrayList.
     * @param int index     index of item
     * @throws              ListIndexOutOfBoundsException
     */
    public void removeAll(int index) throws ListIndexOutOfBoundsException
    {
        items.removeAll(items);
    }

    @Override
    public void removeAll() {
        // TODO Auto-generated method stub

    }
}
::::::::::::::
ListArrayListBasedPlus.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Review Programming Assignment
 * Status: Complete and thoroughly tested
 * Last update: 1/30/23
 * Submitted:  1/30/23
 * Comment: test suite and sample run attached
 * Comment: I declare that this is entirely my own work
 * @author: Antonio Rosado
 * @version: 2023.01.30
 */

import java.util.ArrayList;

public class ListArrayListBasedPlus extends ListArrayListBased {

    /**
     * Resizes ArrayList.
     */
    public void resize()
    {
        if(numItems == items.size())
        {
            ArrayList<Object> new_items = new ArrayList<Object> (numItems * 2);
            for(int index = 0; index < items.size(); index++)
            {
                new_items.add(items.get(index));
            }
            items = new_items;
        }
    }

    /**
    * Adds items to ArrayList
    */
    public void add()
    {
        if(items.size() == numItems)
        {
            resize();
        }
        super.add(numItems, items);
    }

    /**
     * Reverses ArrayList if size cap is reached.
     */
    public void reverse()
    {
        ArrayList<Object> temp = new ArrayList<Object> (numItems);
        for(int index = 0; index < items.size() / 2; index++)
        {
            temp.add(items.size() - 1);
        }
        items = temp;
    }

    /*
     * Returns a string value of item(s) in ArrayList
     */
    public String toString()
    {
        return items.toString();
    }
}
::::::::::::::
ListException.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Review Programming Assignment
 * Status: Complete and thoroughly tested
 * Last update: 1/30/23
 * Submitted:  1/30/23
 * Comment: test suite and sample run attached
 * Comment: I declare that this is entirely my own work
 * @author: Antonio Rosado
 * @version: 2023.01.30
 */

public class ListException
    extends RuntimeException
{
    public ListException(String s)
    {
        super(s);
    }  // end constructor
}  // end ListIndexOutOfBoundsException

::::::::::::::
ListInterface.java
::::::::::::::
// *******************************************************
// Interface ListInterface for the ADT list.
// *******************************************************
public interface ListInterface
{
    boolean isEmpty();
    int size();
    void add(int index, Object item) throws ListIndexOutOfBoundsException;
    Object get(int index) throws ListIndexOutOfBoundsException;
    Object remove(int index) throws ListIndexOutOfBoundsException;
    void removeAll();
    String toString();
}  // end ListInterface

/* Source code for ArrayList
```

```
 *
 *
 *
 public boolean isEmpty()
 {
  return size == 0;
 }

public int size()
{
  return size;
}

public void add(int index, E e)
 {
    checkBoundInclusive(index);
    modCount++;
    if (size == data.length)
      ensureCapacity(size + 1);
    if (index != size)
      System.arraycopy(data, index, data, index + 1, size - index);
    data[index] = e;
    size++;
 }

public E get(int index)
{
  checkBoundExclusive(index);
  return data[index];
}


public E remove(int index)
{
  checkBoundExclusive(index);
  E r = data[index];
  modCount++;
  if (index != --size)
    System.arraycopy(data, index + 1, data, index, size - index);
  // Aid for garbage collection by releasing this pointer.
  data[size] = null;
  return r;
}
 *
 *
 */
```