```
::::::::::::::
Driver.java
::::::::::::::
/*

 * Purpose: Data Structure and Algorithms Lab 2 Problem 1

 * Status: Complete and thoroughly tested

 * Last update: 02/4/20

 * Submitted:  02/4/20

 * Comment: test suite and sample run attached

 * @author: Matthew Ryan

 * @version: 2020.02.4

 */

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Driver {

    static BufferedReader stdin = new BufferedReader (new InputStreamReader(System
.in));

    public static void main(String[] args) throws NumberFormatException, IOExcepti
on {

        ListArrayListBasedPlus items = new      ListArrayListBasedPlus();
        boolean switchOn = true;

        System.out.println("\nSelect from the following menu: \n\t0. Exit program\
n\t"
                           + "1. Insert item to list\n\t2. Remove item from list\n
\t"
                           + "3. Get item from list\n\t4. Clear list\n\t5. Print s
ize and content of list"
                           + "\n\t6. Reverse list");

        while(switchOn == true)
        {
            System.out.print("\nMake your selection now: ");
            int selection = Integer.parseInt(stdin.readLine().trim());
            System.out.println(selection);

            switch(selection)
            {
            case 0:
                switchOn = false;
                System.out.print("Exiting program...Good Bye");
                break;
            case 1:
                System.out.print("\nYou are now inserting an item into the list.\n
\tEnter item: ");
                Object item = stdin.readLine().trim();
                System.out.println(item);
                System.out.print("\tEnter position to insert item in: ");
                int index = Integer.parseInt(stdin.readLine().trim());
                System.out.println(index);

                if(index >= items.size()+1)
                {
                    System.out.println("\nPosition specified is out of range!");
                }
                else
                {
                    items.add(index, item);
                    System.out.println("\nItem " + item + " inserted at position "
 + index + " in the list.");
                }
                break;
            case 2:
                System.out.print("\tEnter position to remove item from: ");
                int toRemove = Integer.parseInt(stdin.readLine().trim());
                System.out.println(toRemove);

                if((toRemove >= items.size()) || (toRemove < 0))
                {
                    System.out.println("\nPosition specified is out of range!");
                }
                else
                {
                    System.out.println("\nItem " + items.get(toRemove) + " removed
 from position " + toRemove + " in the list.");
                    items.remove(toRemove);
                }
                break;
            case 3:
                System.out.print("\t\nEnter position to retrieve item from: ");
                int toRetrieve = Integer.parseInt(stdin.readLine().trim());
                System.out.println(toRetrieve);

                if((toRetrieve >= items.size()) || (toRetrieve < 0))
                {
                    System.out.println("\nPosition specified is out of range!");
                }
                else
                {
                    System.out.println("\nItem " + items.get(toRetrieve) + " retri
eved from position " + toRetrieve + " in the list.");
                }
                break;
            case 4:
                items.removeAll();
                break;
            case 5:

                if(items.size() == 0)
                {
                    System.out.println("List is empty.");
                }
                else
                {
                    System.out.print("List of size " + items.size() + " has the fo
llowing items: ");

                    for(int i = 0; i < items.size(); i++)
                    {
                            System.out.print(items.get(i) + " ");
                        }
                    }
                    break;
                case 6:
                    items.reverseArray();
                    System.out.println("List reversed");
                    break;
            }
        }

    }
}
:::::::::::::
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 static 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)
    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 void remove(int index)
    throws ListIndexOutOfBoundsException
    {
        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)
            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-1] = null;
            numItems--;
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on remove");
        }  // end if
    } //end remove
}
:::::::::::::
ListArrayBasedPlus.java
:::::::::::::
/*
```

```
  * Purpose: Data Structure and Algorithms Lab 2 Problem 1

  * Status: Complete and thoroughly tested

  * Last update: 02/4/20

  * Submitted:  02/4/20

  * Comment: test suite and sample run attached

  * @author: Matthew Ryan

  * @version: 2020.02.4

  */

public class ListArrayBasedPlus extends ListArrayBased {

    ListArrayBasedPlus()
    {
        super();
    }

    public void add(int index, Object item) throws  ListIndexOutOfBoundsException
    {
        if(numItems == items.length)
        {
            resizeArray();
        }
        super.add(index, item);
    }

    private static void resizeArray() {
        Object newItems[] = new Object[items.length + 1];

        for(int i = 0; i < items.length; i++)
        {
            newItems[i] = items[i];
        }

        items = newItems;
    }

    public void toString(Object []items)
    {
        for(int i = 0; i < items.length; i++)
        {
            System.out.print(items[i]);
        }
    }

    public void reverseArray()
    {
        Object reverseArray[] = new Object[items.length];
        int tracker = items.length;
        for(int i = 0; i < items.length; i++)
        {
            reverseArray[i] = items[tracker - 1];
            tracker -= 1;
        }

        items = reverseArray;
```

```
    }

}::::::::::::::
ListArrayListBased.java
::::::::::::::
/*

 * Purpose: Data Structure and Algorithms Lab 2 Problem 1

 * Status: Complete and thoroughly tested

 * Last update: 02/4/20

 * Submitted:  02/4/20

 * Comment: test suite and sample run attached

 * @author: Matthew Ryan

 * @version: 2020.02.4

 */

import java.util.ArrayList;

public class ListArrayListBased {

    private static final int MAX_LIST = 3;
    static ArrayList<Object> items;
    int numItems = 0;

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

    public void removeAll()
    {
        for(int i = items.size()-1; i >= 0; i--)
        {
            items.remove(i);
        }
    }
    public void add(int index, Object item)
    throws  ListIndexOutOfBoundsException
    {
        items.add(index, item);
        numItems++;
    }

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

    public void remove(int index)
    throws ListIndexOutOfBoundsException
    {
        if (index >= 0 && index < numItems)
        {
            items.remove(index);
        }
        else
        {
            // index out of range
            throw new ListIndexOutOfBoundsException(
                "ListIndexOutOfBoundsException on remove");
        }  // end if
    } //end remove


}
::::::::::::::
ListArrayListBasedPlus.java
::::::::::::::
/*

 * Purpose: Data Structure and Algorithms Lab 2 Problem 1

 * Status: Complete and thoroughly tested

 * Last update: 02/4/20

 * Submitted:  02/4/20

 * Comment: test suite and sample run attached

 * @author: Matthew Ryan

 * @version: 2020.02.4

 */

import java.util.Collections;

public class ListArrayListBasedPlus extends ListArrayListBased {

    public ListArrayListBasedPlus()
    {
        super();
    }

    public void toString(Object []items)
    {
        for(int i = 0; i < items.length; i++)
        {
            System.out.print(items[i]);
        }
    }

    public void reverseArray()
    {
        Collections.reverse(items);
    }
```

```
    public int size() {
        // TODO Auto-generated method stub
        return items.size();
    }

}

::::::::::::::
ListException.java
::::::::::::::
/*

 * Purpose: Data Structure and Algorithms Lab 2 Problem 1

 * Status: Complete and thoroughly tested

 * Last update: 02/4/20

 * Submitted:  02/4/20

 * Comment: test suite and sample run attached

 * @author: Matthew Ryan

 * @version: 2020.02.4

 */

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

 * Purpose: Data Structure and Algorithms Lab 2 Problem 1

 * Status: Complete and thoroughly tested

 * Last update: 02/4/20

 * Submitted:  02/4/20

 * Comment: test suite and sample run attached

 * @author: Matthew Ryan

 * @version: 2020.02.4

 */

public class ListIndexOutOfBoundsException
    extends IndexOutOfBoundsException
{
    public ListIndexOutOfBoundsException(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;
    void remove(int index) throws ListIndexOutOfBoundsException;
    void removeAll();
    String toString();
}  // end ListInterface:::::::::::::
output.txt
:::::::::::::

Select from the following menu:
        0. Exit program
        1. Insert item to list
        2. Remove item from list
        3. Get item from list
        4. Clear list
        5. Print size and content of list
        6. Reverse list

Make your selection now: 5
List is empty.

Make your selection now: 1

You are now inserting an item into the list.
        Enter item: Data
        Enter position to insert item in: 0

Item Data inserted at position 0 in the list.

Make your selection now: 5
List of size 1 has the following items: Data
Make your selection now: 1

You are now inserting an item into the list.
        Enter item: Beverly
        Enter position to insert item in: 0

Item Beverly inserted at position 0 in the list.

Make your selection now: 5
List of size 2 has the following items: Beverly Data
Make your selection now: 1

You are now inserting an item into the list.
        Enter item: Jean-Luc
        Enter position to insert item in: 4

Position specified is out of range!

Make your selection now: 5
List of size 2 has the following items: Beverly Data
Make your selection now: 1

You are now inserting an item into the list.
        Enter item: Jean-Luc
        Enter position to insert item in: 2

Item Jean-Luc inserted at position 2 in the list.

Make your selection now: 1

You are now inserting an item into the list.
        Enter item: Geordi
        Enter position to insert item in: 1

Item Geordi inserted at position 1 in the list.

Make your selection now: 1

You are now inserting an item into the list.
        Enter item: Worf
        Enter position to insert item in: 3

Item Worf inserted at position 3 in the list.

Make your selection now: 5
List of size 5 has the following items: Beverly Geordi Data Worf Jean-Luc
Make your selection now: 6
List reversed

Make your selection now: 5
List of size 5 has the following items: Jean-Luc Worf Data Geordi Beverly
Make your selection now: 2
        Enter position to remove item from: 5

Position specified is out of range!

Make your selection now: 2
        Enter position to remove item from: 3

Item Geordi removed from position 3 in the list.

Make your selection now: 5
List of size 4 has the following items: Jean-Luc Worf Data Beverly
Make your selection now: 2
        Enter position to remove item from: 0

Item Jean-Luc removed from position 0 in the list.

Make your selection now: 1

You are now inserting an item into the list.
        Enter item: Will
        Enter position to insert item in: 1

Item Will inserted at position 1 in the list.

Make your selection now: 5
List of size 4 has the following items: Worf Will Data Beverly
Make your selection now: 3

Enter position to retrieve item from: 2
```

```
Item Data retrieved from position 2 in the list.

Make your selection now: 3

Enter position to retrieve item from: 0

Item Worf retrieved from position 0 in the list.

Make your selection now: 3

Enter position to retrieve item from: 7

Position specified is out of range!

Make your selection now: 5
List of size 4 has the following items: Worf Will Data Beverly
Make your selection now: 6
List reversed

Make your selection now: 5
List of size 4 has the following items: Beverly Data Will Worf
Make your selection now: 4

Make your selection now: 5
List is empty.

Make your selection now: 0
Exiting program...Good Bye:::::::::::::
Lab2Conclusions.txt
::::::::::::::

This lab was a doozy, mostly on how to start it.

It gave me a really good reminder of when to use certain collection methods; you w
ouldn’t be making Battleship with an ArrayList, for example.

Overall, a fun little lab, if you ignore the trubleshooting I had to do towards th
e end.
```