```
::::::::::::::
MyListReferenceBased.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Lab 3 Problem 1
 * Status: Complete and thoroughly tested
 * Last update: 02/26/19
 * Submitted:  02/26/19
 * Comment: test suite and sample run attached, revised
 * @author: Donald DeWitt
 * @version: 2019.02.26
 */
public class MyListReferenceBased implements ListInterface
{
          // reference to linked list of items
          protected Node head;

          public MyListReferenceBased()
          {
            head = null;
          }  // end default constructor

          public boolean isEmpty()
          {
            return head==null;
          }  // end isEmpty

          public int size()
          {
                  int numItems = 0;
                  for(Node curr = head; curr != null; curr = curr.getNext())
                  {
                          numItems+=1;
                  }
                  return numItems;
          }  // end size

          private Node find(int index)
          {
          // --------------------------------------------------
          // Locates a specified node in a linked list.
          // Precondition: index is the number of the desired
          // node. Assumes that 0 <= index <= numItems
          // Postcondition: Returns a reference to the desired
          // node.
          // --------------------------------------------------
            Node curr = head;
            for (int skip = 0; skip < index; skip++)
            {
              curr = curr.getNext();
            } // end for
            return curr;
          } // end find

          public Object get(int index)
                        throws ListIndexOutOfBoundsException
          {
            if (index >= 0 && index < size())
            {
              // get reference to node, then data in node
              Node curr = find(index);
              Object dataItem = curr.getItem();
              return dataItem;
            }
            else
            {
              throw new ListIndexOutOfBoundsException(
                              "List index out of bounds exception on get");
            } // end if
          } // end get

          public void add(int index, Object item)
                          throws ListIndexOutOfBoundsException
          {
            if (index >= 0 && index < size()+1)
            {
              if (index == 0)
              {
                // insert the new node containing item at
                // beginning of list
                Node newNode = new Node(item, head);
                head = newNode;
              }
              else
              {
                Node prev = find(index-1);
                // insert the new node containing item after
                // the node that prev references
                Node newNode = new Node(item, prev.getNext());
                prev.setNext(newNode);
              } // end if
            }
            else
            {
              throw new ListIndexOutOfBoundsException(
                             "List index out of bounds exception on add");
            } // end if
          }  // end add

          public void remove(int index)
                           throws ListIndexOutOfBoundsException
          {
            if (index >= 0 && index < size())
            {
              if (index == 0)
              {
                // delete the first node from the list
                head = head.getNext();
              }
              else
              {
                Node prev = find(index-1);
                // delete the node after the node that prev
                // references, save reference to node
                Node curr = prev.getNext();
                prev.setNext(curr.getNext());
              } // end if
            } // end if
            else
            {
              throw new ListIndexOutOfBoundsException(
                            "List index out of bounds exception on remove");
            } // end if
          }   // end remove
```

```
        public void removeAll()
        {
          // setting head to null causes list to be
          // unreachable and thus marked for garbage
          // collection
          head = null;
        } // end removeAll

        public String toString() //uses StringBuilder to return the items in the
 list as a string
        {
                StringBuilder sb = new StringBuilder("");
                for(Node curr = head; curr != null; curr = curr.getNext())
                {
                        sb.append(curr.getItem() + " ");
                }
                return sb.toString();
        }
}::::::::::::::
Driver.java
::::::::::::::
/*
 * Purpose: Data Structure and Algorithms Lab 3 Problem 2
 * Status: Complete and thoroughly tested
 * Last update: 02/26/19
 * Submitted:  02/26/19
 * Comment: test suite and sample run attached, revised
 * @author: Donald DeWitt
 * @version: 2019.02.26
 */
import java.io.*;

public class Driver {

    private static MyListReferenceBased list = new MyListReferenceBased();
    public static void main(String args[]) throws IOException
    {
        int i = 0; //Integer for controlling the menu
        String item;
        int pos;
        BufferedReader bReader = new BufferedReader(new InputStreamReader(System.i
n));

        System.out.println("1. Insert item to list.\n"
                          + "2. Remove item from list.\n"
                          + "3. Get item from list.\n"
                          + "4. Clear list.\n"
                          + "5. Display size and content of list.\n"
                          + "6. Delete largest item in the list.\n"
                          + "7. Reverse the list.\n"
                          + "8. Exit program.\n");

        while(i != 8)
        {
            System.out.println("Select an option: ");
            i = Integer.parseInt(bReader.readLine().trim());
            System.out.println(i);

            switch(i)
            {
            case 1:
                System.out.println("\t You are now inserting an item into the list
.");
                System.out.println("\t Enter item: ");
                item = bReader.readLine().trim();
                System.out.println(item);
                System.out.println("\t Enter position to insert item in: ");
                pos = Integer.parseInt(bReader.readLine().trim());
                System.out.println(pos);

                if(pos > -1 && pos <= list.size())
                {
                    list.add(pos, item);
                    System.out.println("\t Item " + item + " inserted in position
" + pos + " in the list.");
                }
                else
                {
                    System.out.println("Please enter a number greater than -1 and
less than " + (list.size() + 1));;
                }
                break;

            case 2:
                if(!(list.isEmpty()))
                {
                    System.out.println("\t You are now removing an item from the l
ist.");
                    System.out.println("\t Enter position to remove item from: ");
                    pos = Integer.parseInt(bReader.readLine().trim());
                    System.out.println(pos);


                    if(pos > -1 && pos <= list.size())
                    {
                        list.remove(pos);
                    }
                    else
                    {
                        System.out.println("Please enter a number greater than -1
and less than " + (list.size() + 1));;
                    }
                }//end if
                else
                {
                    System.out.println("List is empty");
                }
                break;

            case 3:
                if(!(list.isEmpty()))
                {
                    System.out.println("\t      Enter position to retrieve item fr
om: ");
                    pos = Integer.parseInt(bReader.readLine().trim());
                    System.out.println(pos);
                    if(pos > -1 && pos <= list.size())
                    {
                        list.get(pos);
                    }
                    else
                    {
                        System.out.println("Please enter a number greater than -1
and less than " + (list.size() + 1));;
                    }
                }

                else
                {
                    System.out.println("List is empty");
                }

                break;

            case 4:
                if(!(list.isEmpty()))
                {
                    list.removeAll();
                    System.out.println("\t The list was cleared of all items.");
                }
                else
                {
                    System.out.println("List is empty");
                }
                break;

            case 5:
                if(!(list.isEmpty()))
                {
                    System.out.println("\t List of size " + list.size() + " has th
e following items: " + list.toString());
                }
                else
                {
                    System.out.println("List is empty");
                }
                break;

            case 6:
                if(!(list.isEmpty()))
                {
                    String largestItem = list.get(0).toString();
                    int size = list.size();
                    pos = 0;
                    Node curr = list.head;

                    for (int index = 0; index < size - 1; index++)
                    {
                        if((largestItem.compareTo(curr.getNext().getItem().toStrin
g())) < 0)

                        {
                            largestItem = curr.getNext().getItem().toString();
                            pos = index + 1;
                        }
                        curr = curr.getNext();
                    } // end for
                    list.remove(pos);
                    System.out.println("\t Largest item " + largestItem + " delete
d.");
                }//end if
                else
                {
                    System.out.println("List is empty");
                }
                break;
```

```
                case 7:
                    if(!(list.isEmpty()))
                    {
                        MyListReferenceBased bufferList = new MyListReferenceBased();
                        int size = list.size();
                        Node curr = list.head;
                        for(int index = 0; index < size; index++)//Populate bufferList
                        {
                            bufferList.add(index, list.get(index));
                            curr.setItem(null);
                            curr = curr.getNext();
                        }
                        curr = list.head;
                        for(int lastIndex = size-1; lastIndex > -1; lastIndex--)
                        {
                            curr.setItem(bufferList.get(lastIndex));
                            curr = curr.getNext();
                        }
                    }
                    else
                    {
                        System.out.println("List is empty");
                    }
                    break;

                case 8:
                    System.out.println("Exiting program.");
                    break;

                default:
                    System.out.println("Please select a valid option 1-8.");
                }//end switch
            }//end while
        }//end main()
}
::::::::::::::
output1.output
::::::::::::::
1. Insert item to list.
2. Remove item from list.
3. Get item from list.
4. Clear list.
5. Display size and content of list.
6. Delete largest item in the list.
7. Reverse the list.
8. Exit program.

Select an option:
5
List is empty
Select an option:
6
List is empty
Select an option:
7
List is empty
Select an option:
1
         You are now inserting an item into the list.
         Enter item:
Data
```

```
         Enter position to insert item in:
0
         Item Data inserted in position 0 in the list.
Select an option:
5
         List of size 1 has the following items: Data
Select an option:
7
Select an option:
1
         You are now inserting an item into the list.
         Enter item:
Beverly
         Enter position to insert item in:
0
         Item Beverly inserted in position 0 in the list.
Select an option:
5
         List of size 2 has the following items: Beverly Data
Select an option:
1
         You are now inserting an item into the list.
         Enter item:
Jean-Luc
         Enter position to insert item in:
5
Please enter a number greater than -1 and less than 3
Select an option:
5
         List of size 2 has the following items: Beverly Data
Select an option:
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 in position 2 in the list.
Select an option:
1
         You are now inserting an item into the list.
         Enter item:
Geordi
         Enter position to insert item in:
2
         Item Geordi inserted in position 2 in the list.
Select an option:
1
         You are now inserting an item into the list.
         Enter item:
Worf
         Enter position to insert item in:
3
         Item Worf inserted in position 3 in the list.
Select an option:
5
         List of size 5 has the following items: Beverly Data Geordi Worf Jean-Luc

Select an option:
7
Select an option:
7
Select an option:
6
         Largest item Worf deleted.
Select an option:
5
         List of size 4 has the following items: Beverly Data Geordi Jean-Luc
Select an option:
7
Select an option:
7
Select an option:
2
         You are now removing an item from the list.
         Enter position to remove item from:
9
Please enter a number greater than -1 and less than 5
Select an option:
2
         You are now removing an item from the list.
         Enter position to remove item from:
3
Select an option:
5
         List of size 3 has the following items: Beverly Data Geordi
Select an option:
2
         You are now removing an item from the list.
         Enter position to remove item from:
0
Select an option:
1
         You are now inserting an item into the list.
         Enter item:
Will
         Enter position to insert item in:
0
         Item Will inserted in position 0 in the list.
Select an option:
5
         List of size 3 has the following items: Will Data Geordi
Select an option:
3
                 Enter position to retrieve item from:
2
Select an option:
3
                 Enter position to retrieve item from:
0
Select an option:
3
                 Enter position to retrieve item from:
8
Please enter a number greater than -1 and less than 4
Select an option:
5
         List of size 3 has the following items: Will Data Geordi
Select an option:
6
         Largest item Will deleted.
Select an option:
5
         List of size 2 has the following items: Data Geordi
```

```
Select an option:
6
	Largest item Geordi deleted.
Select an option:
5
	List of size 1 has the following items: Data
Select an option:
4
	The list was cleared of all items.
Select an option:
5
List is empty
Select an option:
7
List is empty
Select an option:
8
Exiting program.
```