public interface ListInterface {
    public boolean isEmpty();
    public int size();
    public void add(int index, Object item)
        throws ListIndexOutOfBoundsException;
    public Object get(int index)
        throws ListIndexOutOfBoundsException;
    public void remove (int index)
        throws ListIndexOutOfBoundsException;
    public void removeAll();
} // end ListInterface
Exceptions

readFile {
    open the file;
    determine its size;
    allocate that much memory;
    read the file into memory;
    close the file;
}

What happens if the file can’t be opened?

What happens if the length of the file can’t be determined?

What happens if enough memory can’t be allocated?

What happens if the read fails?

What happens if the file can’t be closed?
Error Checking

errorCodeType readFile {
    initialize errorCode = 0;
    open the file;
    if (theFileisOpen) {
        determine the length of the file;
        if (gotTheFileLength) {
            allocate that much memory;
            if (gotEnoughMemory) {
                read the file into memory;
                if (readFailed) {
                    errorCode = -1;
                }
            } else {
                errorCode = -2;
            }
        } else {
            errorCode = -3;
        }
    } else {
        close the file;
        if (theFileDidntClose && errorCode == 0) {
            errorCode = -4;
        } else {
            errorCode = errorCode and -4;
        }
    }
} else {
    errorCode = -5;
}
return errorCode; }
Exceptions

```plaintext
readFile {
    try {
        open the file;
        determine its size;
        allocate that much memory;
        read the file into memory;
        close the file;
    } catch (fileOpenFailed) {
        doSomething;
    } catch (sizeDeterminationFailed) {
        doSomething;
    } catch (memoryAllocationFailed) {
        doSomething;
    } catch (readFailed) {
        doSomething;
    } catch (fileCloseFailed) {
        doSomething;
    }
}
```
Copy Example

```java
import java.io.*;

public class Copy {
    public static void main(String[] args)
            throws IOException {
        File inputFile = new File("farrago.txt");
        File outputFile = new File("outagain.txt");

        FileReader in = new FileReader(inputFile);
        FileWriter out = new FileWriter(outputFile);
        int c;

        while ((c = in.read()) != -1)
            out.write(c);

        in.close();
        out.close();
    }

    catch (IOException
        do something here
```
import java.io.*;

public class CopyBytes {
    public static void main(String[] args)
        throws IOException {
        File inputFile = new File("far-rago.txt");
        File outputFile = new File("outagain.txt");

        FileInputStream in = new FileInputStream(inputFile);
        FileOutputStream out = new FileOutputStream(outputFile);
        int c;

        while ((c = in.read()) != -1)
            out.write(c);

        in.close();
        out.close();
    }
}
public class IntegerNode {
    public int item;
    public IntegerNode next;
}

head

4 adds
add 4

delete(3)
traverse the list
3rd item

1 2 3
array

\[\begin{array}{c}
1 & 2 & 3 & 4 & 5 & 6 \\
\hline
\end{array}\]

object

Max-length

num_elements

add \((3 + 1)\)

\(3\)

delete \((1)\)
Lists

add an element to list at a position
remove an element at a position
print
count
reset

size() count
removeAll() reset
add(pos, Object)
delete(position)

interface class

p 142. public interface ListInterface {
    public boolean isEmpty();
    public int size();
    public void add(int index, Object item) throws
        ListIndexOutOfBoundsException ...
    public void remove(int index) throws ...
    public Object get(int index) throws ...
    public void removeAll();
```java
class Box {
    int value;
    int next;
}

Box b(5);
Box b;
Box a = new Box(5);
Box b = new Box(5);

a = b
```
public class ListArrayBased implements ListInterface {
    private final int MAX_LIST = 50;
    private Object items[];
    private int numItems;

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

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

    public int size() {
        return numItems;
    }

    public void removeAll() {
        items = new Object[MAX_LIST];
        numItems = 0;
    }
}
public void add(int index, Object item)
        throws ListIndexOutOfBoundsException {
    if (numItems > MAX_LIST) {
        throw new ListException("ListException on add");
    }
    if (index >= 1 && index <= numItems+1) {
        for(int pos = numItems; pos >= index; pos--){
            items[translate(pos+1)] = items[translate(pos)];
        }
        items[translate(index)] = item;
        numItems++;
    } else {
        throw new ListIndexOutOfBoundsException(
            "ListIndexOutOfBoundsException on add");
    }
}

if (numItems > MAX_LIST) {
    double array
    allocate new array (twice size)
    maxList = maxList * 2
    copy old array to new
public Object get(int index) throws ListIndexOutOfBoundsException {
    if (index >= 1 && index <= numItems) {
        return items[translate(index)];
    } else {
        throw new ListIndexOutOfBoundsException("ListIndexOutOfBoundsException on get");
    }
}

public void remove(int index) throws ListIndexOutOfBoundsException {
    if (index >= 1 && index <= numItems) {
        for (int pos = index + 1; pos <= size(); pos++) {
            items[translate(pos - 1)] = items[translate(pos)];
        }
        numItems--;
    } else {
        throw new ListIndexOutOfBoundsException("ListIndexOutOfBoundsException on remove");
    }
}

private int translate(int position) {
    return position - 1;
}