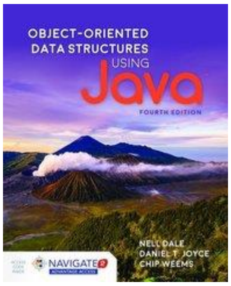
Chapter 6

The List ADT



Section 3

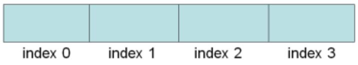
ARRAY-BASED IMPLEMENTATION OF LISTS

Objectives

- Build an array-based implementation of lists
- Note strengths and weaknesses of the array-based implementation
- Test the implementation using generic test, complex numbers, and polynomials

Elementary Facts About Lists

- A list is a collection of objects organized in a sequence
- Each list element is assigned an integer index based on its position in the sequence
- The first element in the sequence has index 0, and each succeeding element has index 1 greater than its predecessor



Contiguous Allocation

- Implementation of lists can be array-based, this is known as *contiguous memory* allocation
- In **contiguous allocation**, list elements occupy consecutive memory locations
- Contiguous allocation is used by array-based list implementations such as `ArrayList` and `Vector`

Random Access

- Contiguous allocation is said to allow **random access** because we can directly jump to any given element without going through its predecessors
- Given any index k, we can compute the memory address of the list element at that position by adding the size (in bytes) of k items to the address of the first element in the list

Advantages of Contiguous Allocation

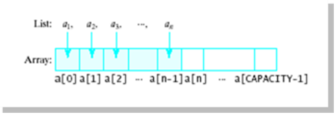
- An array is a viable choice for storing list elements:
  - Elements are sequential
  - It is a commonly available data type
  - Algorithm development is easy
  - Access to a contiguously allocated list element is very fast

Disadvantages of Contiguous Allocation

- Insertion or deletion of elements in the middle of the list involves the laborious relocation of all elements that come after
- For example, if a list has a thousand elements, then to insert a new element at position 5 means all elements from position 5 on up must be moved up.
- This overhead is bad for applications that do a lot of insertions and deletions

Lists and Arrays

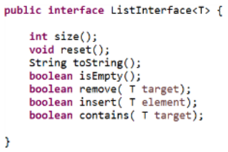
- Normally sequential orderings of list elements match with array elements



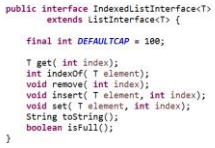
Lists and Arrays

- The array type will be generic so it can be used to implement any type of list in a program
- Must deal with issue of declaration of array capacity
- Stuck with "one size fits all"
  - Could be wasting space
  - Could run out of space
- Possible Solution - If a larger array is needed during program execution:
  - Allocate larger array
  - Copy smaller array to the new one

Implementing the Interfaces

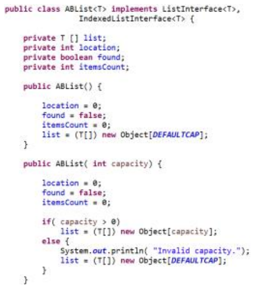


```
public interface ListInterface<T> {

    int size();
    void reset();
    String toString();
    boolean isEmpty();
    boolean remove( T target);
    boolean insert( T element);
    boolean contains( T target);

}
```

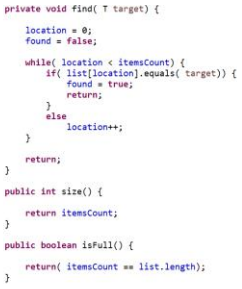


```
public interface IndexedListInterface<T>
        extends ListInterface<T> {

    final int DEFAULTCAP = 100;

    T get( int index);
    int indexOf( T element);
    void remove( int index);
    void insert( T element, int index);
    void set( T element, int index);
    String toString();
    boolean isFull();
}
```

Implementing the `ABList` Class

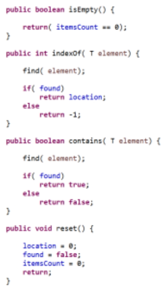


```
public class ABList<T> implements ListInterface<T>,
                IndexedListInterface<T> {

    private T [] list;
    private int location;
    private boolean found;
    private int itemsCount;

    public ABList() {

        location = 0;
        found = false;
        itemsCount = 0;
        list = (T[]) new Object[DEFAULTCAP];
    }

    public ABList( int capacity) {

        location = 0;
        found = false;
        itemsCount = 0;

        if( capacity > 0)
            list = (T[]) new Object[capacity];
        else {
            System.out.println( "Invalid capacity.");
            list = (T[]) new Object[DEFAULTCAP];
        }
    }
```

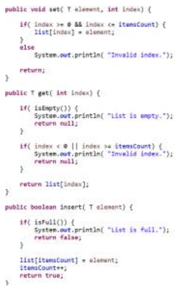


```
private void find( T target) {

    location = 0;
    found = false;

    while( location < itemsCount) {
        if( list[location].equals( target)) {
            found = true;
            return;
        }
        else
            location++;
    }

    return;
}

public int size() {

    return itemsCount;
}

public boolean isFull() {

    return( itemsCount == list.length);
}
```

Implementing the `ABList` Class

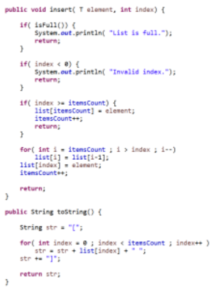


```java
public boolean isEmpty() {

    return( itemsCount == 0);
}

public int indexOf( T element) {

    find( element);

    if( found)
        return location;
    else
        return -1;
}

public boolean contains( T element) {

    find( element);

    if( found)
        return true;
    else
        return false;
}

public void reset() {

    location = 0;
    found = false;
    itemsCount = 0;
    return;
}
```

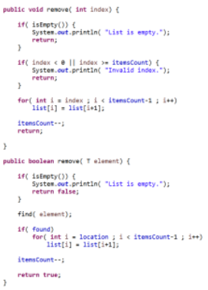


```java
public void set( T element, int index) {

    if( index >= 0 && index <= itemsCount) {
        list[index] = element;
    }
    else
        System.out.println( "Invalid index.");

    return;
}

public T get( int index) {

    if( isEmpty()) {
        System.out.println( "List is empty.");
        return null;
    }

    if( index < 0 || index >= itemsCount) {
        System.out.println( "Invalid index.");
        return null;
    }

    return list[index];
}

public boolean insert( T element) {

    if( isFull()) {
        System.out.println( "List is full.");
        return false;
    }

    list[itemsCount] = element;
    itemsCount++;
    return true;
}
```

Implementing the `ABList` Class

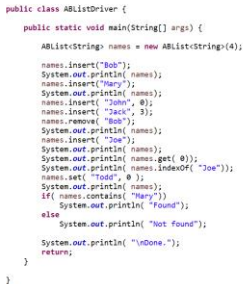


```java
public void insert( T element, int index) {

    if( isFull()) {
        System.out.println( "List is full.");
        return;
    }

    if( index < 0) {
        System.out.println( "Invalid index.");
        return;
    }

    if( index >= itemsCount) {
        list[itemsCount] = element;
        itemsCount++;
        return;
    }

    for( int i = itemsCount ; i > index ; i--)
        list[i] = list[i-1];
    list[index] = element;
    itemsCount++;

    return;
}

public String toString() {

    String str = "[";

    for( int index = 0 ; index < itemsCount ; index++ )
        str = str + list[index] + " ";
    str += "]";

    return str;
}
```

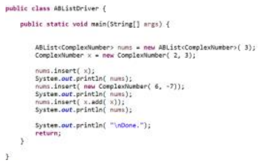


```java
public void remove( int index) {

    if( isEmpty()) {
        System.out.println( "List is empty.");
        return;
    }

    if( index < 0 || index >= itemsCount) {
        System.out.println( "Invalid index.");
        return;
    }

    for( int i = index ; i < itemsCount-1 ; i++)
        list[i] = list[i+1];

    itemsCount--;
    return;

}

public boolean remove( T element) {

    if( isEmpty()) {
        System.out.println( "List is empty.");
        return false;
    }

    find( element);

    if( found)
        for( int i = location ; i < itemsCount-1 ; i++)
            list[i] = list[i+1];

    itemsCount--;

    return true;
}
```

Testing the `ABList` Class

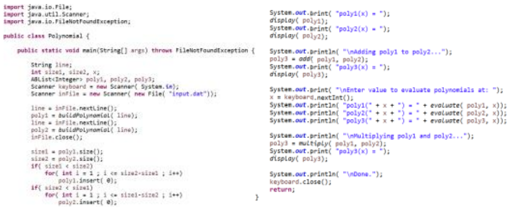


```java
public class ABListDriver {

    public static void main(String[] args) {

        ABList<String> names = new ABList<String>(4);

        names.insert("Bob");
        System.out.println( names);
        names.insert("Mary");
        System.out.println( names);
        names.insert( "John", 0);
        names.insert( "Jack", 3);
        names.remove( "Bob");
        System.out.println( names);
        names.insert( "Joe");
        System.out.println( names);
        System.out.println( names.get( 0));
        System.out.println( names.indexOf( "Joe"));
        names.set( "Todd", 0 );
        System.out.println( names);
        if( names.contains( "Mary"))
            System.out.println( "Found");
        else
            System.out.println( "Not found");

        System.out.println( "\nDone.");
        return;
    }

}
```



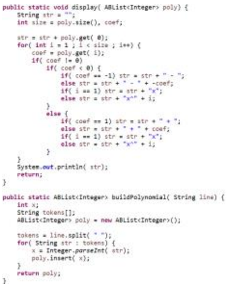
```java
public class ABListDriver {

    public static void main(String[] args) {


        ABList<ComplexNumber> nums = new ABList<ComplexNumber>( 3);
        ComplexNumber x = new ComplexNumber( 2, 3);

        nums.insert( x);
        System.out.println( nums);
        nums.insert( new ComplexNumber( 6, -7));
        System.out.println( nums);
        nums.insert( x.add( x));
        System.out.println( nums);

        System.out.println( "\nDone.");
        return;
    }

}
```

Testing the ABList Class Using Polynomials



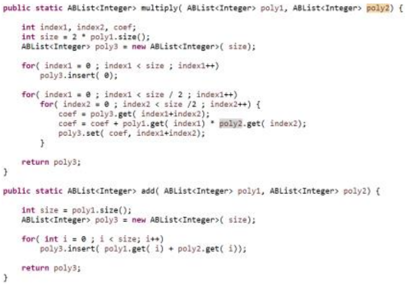
```
import java.io.File;
import java.util.Scanner;
import java.io.FileNotFoundException;

public class Polynomial {

    public static void main(String[] args) throws FileNotFoundException {

        String line;
        int size1, size2, x;
        ABList<Integer> poly1, poly2, poly3;
        Scanner keyboard = new Scanner( System.in);
        Scanner inFile = new Scanner( new File( "input.dat"));

        line = inFile.nextLine();
        poly1 = buildPolynomial( line);
        line = inFile.nextLine();
        poly2 = buildPolynomial( line);
        inFile.close();

        size1 = poly1.size();
        size2 = poly2.size();
        if( size1 < size2)
            for( int i = 1 ; i <= size2-size1 ; i++)
                poly1.insert( 0);
        if( size2 < size1)
            for( int i = 1 ; i <= size1-size2 ; i++)
                poly2.insert( 0);

        System.out.print( "poly1(x) = ");
        display( poly1);
        System.out.print( "poly2(x) = ");
        display( poly2);

        System.out.println( "\nAdding poly1 to poly2...");
        poly3 = add( poly1, poly2);
        System.out.print( "poly3(x) = ");
        display( poly3);

        System.out.print( "\nEnter value to evaluate polynomials at: ");
        x = keyboard.nextInt();
        System.out.println( "poly1(" + x + ") = " + evaluate( poly1, x));
        System.out.println( "poly2(" + x + ") = " + evaluate( poly2, x));
        System.out.println( "poly3(" + x + ") = " + evaluate( poly3, x));

        System.out.println( "\nMultiplying poly1 and poly2...");
        poly3 = multiply( poly1, poly2);
        System.out.print( "poly3(x) = ");
        display( poly3);

        System.out.println( "\nDone.");
        keyboard.close();
        return;
    }
```

Testing the ABList Class Using Polynomials



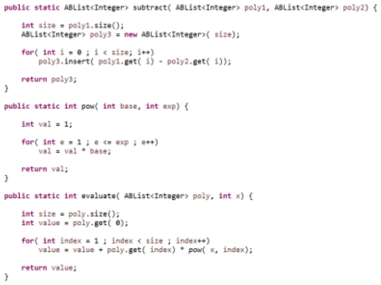
```
public static void display( ABList<Integer> poly) {
    String str = "";
    int size = poly.size(), coef;

    str = str + poly.get( 0);
    for( int i = 1 ; i < size ; i++) {
        coef = poly.get( i);
        if( coef != 0)
            if( coef < 0) {
                if( coef == -1) str = str + " - ";
                else str = str + " - " + -coef;
                if( i == 1) str = str + "x";
                else str = str + "x^" + i;
            }
            else {
                if( coef == 1) str = str + " + ";
                else str = str + " + " + coef;
                if( i == 1) str = str + "x";
                else str = str + "x^" + i;
            }
    }
    System.out.println( str);
    return;
}

public static ABList<Integer> buildPolynomial( String line) {
    int x;
    String tokens[];
    ABList<Integer> poly = new ABList<Integer>();

    tokens = line.split( " ");
    for( String str : tokens) {
        x = Integer.parseInt( str);
        poly.insert( x);
    }
    return poly;
}
```

Testing the ABList Class Using Polynomials



```
public static ABList<Integer> multiply( ABList<Integer> poly1, ABList<Integer> poly2) {

    int index1, index2, coef;
    int size = 2 * poly1.size();
    ABList<Integer> poly3 = new ABList<Integer>( size);

    for( index1 = 0 ; index1 < size ; index1++)
        poly3.insert( 0);

    for( index1 = 0 ; index1 < size / 2 ; index1++)
        for( index2 = 0 ; index2 < size /2 ; index2++) {
            coef = poly3.get( index1+index2);
            coef = coef + poly1.get( index1) * poly2.get( index2);
            poly3.set( coef, index1+index2);
        }

    return poly3;
}

public static ABList<Integer> add( ABList<Integer> poly1, ABList<Integer> poly2) {

    int size = poly1.size();
    ABList<Integer> poly3 = new ABList<Integer>( size);

    for( int i = 0 ; i < size; i++)
        poly3.insert( poly1.get( i) + poly2.get( i));

    return poly3;
}
```

Testing the `ABList` Class Using Polynomials



```
public static ABList<Integer> subtract( ABList<Integer> poly1, ABList<Integer> poly2) {

    int size = poly1.size();
    ABList<Integer> poly3 = new ABList<Integer>( size);

    for( int i = 0 ; i < size; i++)
        poly3.insert( poly1.get( i) - poly2.get( i));

    return poly3;
}

public static int pow( int base, int exp) {

    int val = 1;

    for( int e = 1 ; e <= exp ; e++)
        val = val * base;

    return val;
}

public static int evaluate( ABList<Integer> poly, int x) {

    int size = poly.size();
    int value = poly.get( 0);

    for( int index = 1 ; index < size ; index++)
        value = value + poly.get( index) * pow( x, index);

    return value;
}
```