# The ListNode Class

# Big-O in Arrays and. Linked Lists

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | theArray |  | **3** | **3** | **7** | **9** | |  |  | [0] | [1] | [2] | [3] | | 3  7  9  head  3 |
| * Data structure: *array* * O(1) to access any value through its index.(Called *random access*.) * O(N) to insert or remove values because the other values may need to be shifted. * Arrays are fixed-length. If you fill up the array, you need to make a new, larger array and copy the data into it. | * Data structure*: linked list*, built using one or more *list nodes*. * O(N) to access a value because you traverse the list from *head*. * O(1) to insert or remove a value at the front. * Linked lists grow or shrink to hold more or fewer items. Linked lists never have to be resized. |

# The ListNode class

The College Board has specified a ListNode class to be used for constructing linked lists. The complete source code is:

|  |  |
| --- | --- |
|  |  |

**public class** ListNode

{

**private** Object value;

**private** ListNode next;

**public** ListNode(Object initValue, ListNode initNext)

{

value = initValue;

next = initNext;

}

**public** Object getValue()

{

**return** value;

}

**public** ListNode getNext()

{

**return** next;

}

**public** **void** setValue(Object theNewValue)

{

value = theNewValue;

}

**public** **void** setNext(ListNode theNewNext)

{

next = theNewNext;

}

}

Each ListNode object stores one Object in the field value. Technically, ListNodes cannot store primitives. However, Java 1.5 automatically wraps and unwraps the wrapper class around the primitive, so that int automatically becomes Integer, and vice-versa, and the same for double and boolean. You don’t have to worry about that technicality any more.

The field next stores a reference to the node that follows this node in the list.

# Example

An example program for building and displaying a linked list is given by the following program:

**import** ap.ListNode; // Either import it, or have the ListNode class  
 // in your current directory.

**public class** ListLab1

{

**public static void main**(String[] args)

{

ListNode head = **new** ListNode("hello", null);

head = **new** ListNode("foo", head);

head = **new** ListNode("boo", head);

head = **new** ListNode("nonsense", head);

head = **new** ListNode("computer",

**new** ListNode("science",

**new** ListNode("java",

**new** ListNode("coffee", head))));

print(head);

print(head);

}

**public static void print**(ListNode head)

{

System.out.print("[");

**while**(head != null)

{

System.out.print(head.getValue());

head = head.getNext();

**if**(head != **null**)

System.out.print(", ");

}

System.out.println("]");

}

}  
**Sample Run:**

[computer, science, java, coffee, nonsense, boo, foo, hello]   
[computer, science, java, coffee, nonsense, boo, foo, hello]

**Exercises**

1. Step by step, draw a picture of the list as it is being constructed:

2. Step by step, show the movement of the pointer as the list is being printed.