# The ListNode Class

# Arrays vs. Linked Lists

The data structure that you have used is an *array*. Since an array has a fixed size and items are stored in fixed positions, any piece of data can be accessed in O(1) time. We say that arrays have *random access*. On the other hand, adding or removing a piece of data will take, on average, O(N) time, because many items may need to be shifted. Adding one item of data may also need to resize the array, which takes O(N), on average.

Another data structure is the *linked list,* which is built using one or more list *nodes*. Since you can't jump directly to data in a linked list, the data must be found and accessed in O(N) time. However, an item can be inserted or removed from a linked list simply by rearranging some links, which take O(1) time. Linked lists never have to be resized.

In what situations would it be preferable to use an array instead of a linked list?

When you need to access a single value, use an array.

In what situations would it be preferable to use a linked list instead of an array?

When you need to insert or remove a value, or change the length frequently, use a linked list.

# 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("]");

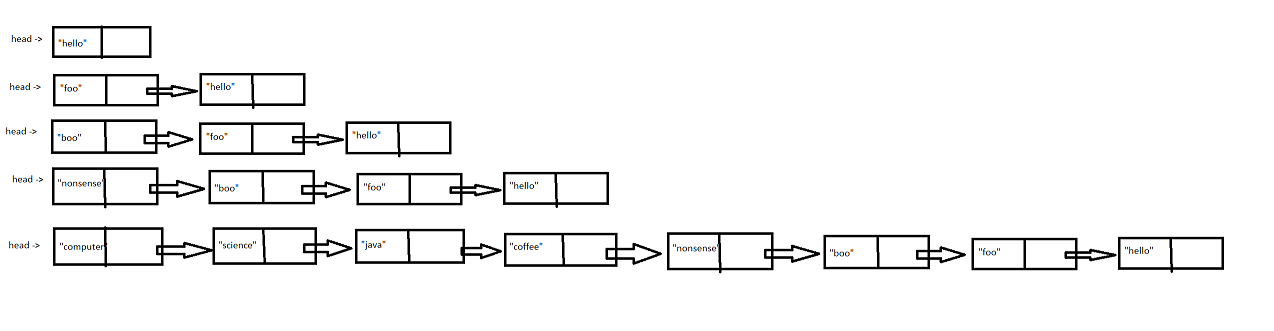
}

}  
**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.

