

# IntLL.java

```
1 /*
2  * Written by JJ Shepherd
3  */
4 public class IntLL
5 {
6     private class ListNode
7     {
8         int data;
9         ListNode link;
10        public ListNode(int aData, ListNode aLink)
11        {
12            data = aData;
13            link = aLink;
14        }
15    }
16    private ListNode head; //First element
17    private ListNode current; //Current node of interest
18    private ListNode previous; //One node behind current
19
20    public IntLL()
21    {
22        head = current = previous = null;
23    }
24    public void add(int aData)
25    {
26        ListNode newNode = new ListNode(aData, null);
27        if(head == null) //Empty list
28        {
29            head = current = newNode;
30            return;
31        }
32        ListNode temp = head;
33        while(temp.link != null)
34            temp = temp.link;
35        temp.link = newNode;
36    }
37    public void addAfterCurrent(int aData)
38    {
39        if(current == null)
40            return;
41        ListNode newNode = new ListNode(aData, current.link);
42        current.link = newNode;
43    }
44    public void print()
45    {
46        ListNode temp = head;
47        while(temp != null)
48        {
49            System.out.println(temp.data);
50            temp = temp.link;
51        }
52    }
53    public int getCurrent()
54    {
55        if(current == null)
56            return 0;
57        return current.data;
```

```
58     }
59     public void setCurrent(int aData)
60     {
61         if(current == null)
62             return;
63         current.data = aData;
64     }
65     public void gotoNext()
66     {
67         if(current != null)
68         {
69             previous = current;
70             current = current.link;
71         }
72     }
73     public boolean hasMore()
74     {
75         return current != null;
76     }
77     public void reset()
78     {
79         current = head;
80         previous = null;
81     }
82     public void removeCurrent()
83     {
84         if(current != null && previous != null)
85         {
86             previous.link = current.link;
87             current = current.link;
88         }
89         else if(current != null && previous == null)
90         {
91             head = head.link; //current = current.link;
92             current = head;
93         }
94     }
95 }
96
```