doublylinkedlist/DoublyLinkedList.java

```java
package doublylinkedlist;

public class DoublyLinkedList {
    private Node head;
    private Node tail;
    private int length;

    class Node {
        int value;
        Node next;
        Node prev;

        Node(int value) {
            this.value = value;
        }
    }

    public DoublyLinkedList(int value) {
        Node newNode = new Node(value);
        head = newNode;
        tail = newNode;
        length = 1;
    }

    public void printList() {
        Node temp = head;
        while (temp != null) {
            System.out.println(temp.value);
            temp = temp.next;
        }
    }

    public void getHead() {
        System.out.println("Head: " + head.value);
    }

    public void getTail() {
        System.out.println("Tail: " + tail.value);
    }

    public void getLength() {
        System.out.println("Length: " + length);
    }

    public void append(int value) {
        Node newNode = new Node(value);
        if (length == 0) {
            head = newNode;
            tail = newNode;
        } else {
            tail.next = newNode;
```

```
52              newNode.prev = tail;
53              tail = newNode;
54          }
55          length++;
56      }
57
58      public Node removeLast() {
59          if (length == 0)
60              return null;
61          Node temp = tail;
62          if (length == 1) {
63              head = null;
64              tail = null;
65          } else {
66              tail = tail.prev;
67              tail.next = null;
68              temp.prev = null;
69          }
70          length--;
71          return temp;
72      }
73
74      public void prepend(int value) {
75          Node newNode = new Node(value);
76          if (length == 0) {
77              head = newNode;
78              tail = newNode;
79          } else {
80              newNode.next = head;
81              head.prev = newNode;
82              head = newNode;
83          }
84          length++;
85      }
86
87      public Node removeFirst() {
88          if (length == 0)
89              return null;
90          Node temp = head;
91          if (length == 1) {
92              head = null;
93              tail = null;
94          } else {
95              head = head.next;
96              head.prev = null;
97              temp.next = null;
98          }
99          length--;
100         return temp;
101     }
102
103     public Node get(int index) {
104         if (index < 0 || index >= length)
105             return null;
```

```
106         Node temp = head;
107         if (index < length / 2) {
108             for (int i = 0; i < index; i++) {
109                 temp = temp.next;
110             }
111         } else {
112             temp = tail;
113             for (int i = length - 1; i > index; i--) {
114                 temp = temp.prev;
115             }
116         }
117         return temp;
118     }
119 
120     public boolean set(int index, int value) {
121         Node temp = get(index);
122         if (temp != null) {
123             temp.value = value;
124             return true;
125         }
126         return false;
127     }
128 
129     public boolean insert(int index, int value) {
130         if (index < 0 || index > length)
131             return false;
132         if (length == 0) {
133             prepend(value);
134             return true;
135         }
136         if (index == length) {
137             append(value);
138             return true;
139         }
140         Node newNode = new Node(value);
141         Node before = get(index - 1);
142         Node after = before.next;
143         newNode.prev = before;
144         newNode.next = after;
145         before.next = newNode;
146         after.prev = newNode;
147         length++;
148         return true;
149     }
150 
151     public Node remove(int index) {
152         if (index < 0 || index >= length)
153             return null;
154         if (index == 0) {
155             return removeFirst();
156         }
157         if (index == length - 1) {
158             return removeLast();
159         }
```

```
160
161         Node temp = get(index);
162         temp.next.prev = temp.prev;
163         temp.prev.next = temp.next;
164         temp.next = null;
165         temp.prev = null;
166         length--;
167         return temp;
168     }
169 }
170
```